Alfred Nischwitz
Max Fischer
Peter Haberäcker
Gudrun Socher

LEHRBUCH

Bildverarbeitung

Band II des Standardwerks Computergrafik und Bildverarbeitung

4. Auflage

EXTRAS ONLINE

Springer Vieweg

Bildverarbeitung

Alfred Nischwitz · Max Fischer ·
Peter Haberäcker · Gudrun Socher

Bildverarbeitung

Band II des Standardwerks Computergrafik und Bildverarbeitung

4. Auflage

Springer Vieweg

Alfred Nischwitz
Fakultät für Informatik und Mathematik,
Hochschule München
München, Deutschland

Max Fischer
Fakultät für Informatik und Mathematik,
Hochschule München
München, Deutschland

Peter Haberäcker
Rottenbuch, Deutschland

Gudrun Socher
Fakultät für Informatik und Mathematik,
Hochschule München
München, Deutschland

Ergänzendes Material zu diesem Buch finden Sie auf http://www.springer.com/978-3-658-28705-4.

ISBN 978-3-658-28704-7 ISBN 978-3-658-28705-4 (eBook)
https://doi.org/10.1007/978-3-658-28705-4

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Die 1. Auflage erschien unter dem Titel „Masterkurs Computergrafik und Bildverarbeitung".
Seit der 3. Auflage erscheint das Werk in zwei Bänden sowie als Set. Die 3. Auflage erschien unter dem Titel „Computergrafik und Bildverarbeitung – Band II: Bildverarbeitung".

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Vorwort zur 4. Auflage von Band II

Mit der 4. Auflage wurde der Titel des Werkes „Computergrafik und Bildverarbeitung“ angepasst. Zur besseren Unterscheidbarkeit auf dem Buchrücken und nachdem die beiden Bände nicht mehr zeitgleich erscheinen, heißt es jetzt:
„Bildverarbeitung: Band II des Standardwerks Computergrafik und Bildverarbeitung“.

Die Grundidee dieses Werkes, nämlich dem engen Zusammenhang zwischen Computergrafik und Bildverarbeitung (siehe Kapitel 2) dadurch Rechnung zu tragen, dass man beide Bereiche in einem Werk darstellt, tritt durch die Namensgebung der zwei Bände zwar noch etwas weiter in den Hintergrund. Um diese Grundidee jedoch im Bewusstsein zu halten, bleibt der Zusammenhang im Untertitel des Werkes *„Band II des Standardwerks Computergrafik und Bildverarbeitung“* für beide Bände erhalten. Es gibt weiterhin Referenzen zwischen beiden Bänden (gekennzeichnet durch ein vorangestelltes Band I oder II vor der jeweiligen Referenz). Es empfiehlt sich in jedem Fall die Anschaffung beider Bände, da sich nach unserer Erfahrung Kenntnisse in der Bildverarbeitung positiv auf das Verständnis der Computergrafik auswirken und umgekehrt.

Es konnten Fehler und Ungenauigkeiten aus den vorherigen Auflagen korrigiert werden. Dafür danken wir den kritischen Lesern ganz herzlich. Außerdem möchten wir uns ganz besonders bei unserem Korrekturleser Paul Obermeier bedanken, der auch inhaltliche Anstöße gab. Das Kapitel 3.5 zu Farbbildern wurde grundlegend überarbeitet. Insbesondere die Darstellung der Farbmodelle HSI, HSV, HSB und HSL wurde korrigiert und verbessert. Das Kapitel 16 wurde wesentlich erweitert. Neben der Darstellung von SIFT enthält es jetzt auch die Operatoren FAST, SURF und HOG. Weiterhin wurden eigene Arbeiten zu lernbasierten Korrelationstrackern mit aufgenommen.

Alfred Nischwitz, Max Fischer, Gudrun Socher, Peter Haberäcker
15. Oktober 2019

Vorwort zur 3. Auflage

Wir freuen uns, dass wir als vierte Co-Autorin für die Weiterentwicklung dieses Buchs unsere Kollegin Prof. Dr. Gudrun Socher gewinnen konnten. Sie hat sich in erster Linie im Bildverarbeitungsteil eingebracht und dort für frischen Wind gesorgt.

Mit der 3. Auflage wurde das Format dieses Werkes grundlegend geändert: es wurde in zwei Bände aufgeteilt. Dies ist der Tatsache geschuldet, dass das Buch ansonsten deutlich über 1000 Seiten stark geworden wäre. Somit hätte der Verkaufspreis in Regionen vordringen müssen, bei dem der Verlag davon ausging, dass er für Kunden eine zu hohe Schwelle für eine Kaufentscheidung darstellen würde. Außerdem sind Bücher mit über 1000 Seiten auch nicht mehr besonders handlich. Die Grundidee dieses Werkes, nämlich dem engen Zusammenhang zwischen Computergrafik und Bildverarbeitung (siehe Kapitel 2) dadurch Rechnung zu tragen, dass man beide Bereiche in einem Werk darstellt, tritt durch die Aufteilung in zwei Bände zwar wieder etwas in den Hintergrund. Um diese Grundidee jedoch im Bewusstsein zu halten, bleibt der Titel des Werkes *„Computergrafik und Bildverarbeitung“* für beide Bände erhalten. Es gibt weiterhin Referenzen zwischen beiden Bänden (gekennzeichnet durch ein vorangestelltes Band I oder II vor der jeweiligen Referenz) und die Inhaltsangabe des jeweils anderen Bandes findet sich am Ende jeden Bandes wieder, so dass man zumindest einen Überblick über das gesamte Werk in jedem Band erhält. Es empfiehlt sich in jedem Fall die Anschaffung beider Bände, da sich nach unserer Erfahrung Kenntnisse in der Bildverarbeitung positiv auf das Verständnis der Computergrafik auswirken und umgekehrt.

Inhaltlich ergaben sich die größten Änderungen in Band I *„Computergrafik“*. Der Grund dafür ist ein Kulturbruch bei der offenen Programmierschnittstelle OpenGL, die in diesem Werk als Grundlage für die Einführung in die Computergrafik gewählt wurde. Im Jahr 2009 wurde das *„alte“* OpenGL als *„deprecated“* erklärt (das bedeutet so viel wie *„abgekündigt“*) und die damit verbundenen Befehle stehen nur noch in einem sogenannten *„Compatibility Profile“* zur Verfügung. Gleichzeitig wurde das *„neue“* OpenGL in Form eines *„Core Profiles“* eingeführt. In diesem Buch werden die beiden Profile nebeneinander dargestellt, so dass auch Leser, die mit dem alten OpenGL vertraut sind, einen leichten Übergang zum neuen OpenGL finden. Außerdem bietet das alte OpenGL häufig einen leichteren Zugang zu bestimmten Themen der Computergrafik, so dass auch OpenGL-Einsteiger von den didaktischen Vorteilen einer parallelen Darstellung von *„Compatibility“* und *„Core Profile“* profitieren. In diesem Sinne wurden alle Kapitel aus dem Computergrafik-Teil vollständig überarbeitet. Neu dazugekommen ist ein Kapitel über das komplexe Thema Schatten, so-

wie ein Kapitel über die allgemeine Programmierung von Grafikkarten mit CUDA und OpenCL.

Die inhaltlichen Änderungen im Band II *„Bildverarbeitung“* waren dagegen nicht so gravierend. Das Kapitel Grundlagen wurde um einen Abschnitt über den CIELab-Farbraum ergänzt. Neu ist ein Kapitel über die Rekonstruktion fehlender Farbwerte bei Digitalkameras (Demosaicing), sowie ein Kapitel über Korrespondenzen in Bildern. Die Literaturhinweise wurden aktualisiert und durch neue Zitate erweitert.

Weiterhin konnten einige Fehler aus der 2. Auflage korrigiert werden. Dafür danken wir den kritischen Lesern ganz herzlich. Außerdem möchten wir uns ganz besonders bei unserem Korrekturleser Paul Obermeier bedanken, der auch wichtige inhaltliche Anstöße gab.

Alfred Nischwitz, Max Fischer, Gudrun Socher, Peter Haberäcker, 02. März 2011

Vorwort zur 2. Auflage

Aller guten Dinge sind Drei. In diesem Sinne freuen wir uns, dass wir als dritten Co-Autor für die Weiterentwicklung dieses Buchs unseren Kollegen Prof. Dr. Max Fischer gewinnen konnten. Damit wurde der Tatsache Rechnung getragen, dass die sehr dynamischen und immer enger zusammenwachsenden Gebiete der Computergrafik und Bildverarbeitung eines weiteren Autors bedurften, um adäquat abgedeckt zu werden.

Inhaltlich wurde der Charakter der 1. Auflage beibehalten. An einer Reihe von Stellen wurden jedoch Ergänzungen und Aktualisierungen vorgenommen. So ist im Kapitel 2 ein neuer Abschnitt über „Bildverarbeitung auf programmierbarer Grafikhardware“ hinzugekommen, in Band II Kapitel 7 wurde der neuerdings häufig verwendete „Canny-Kantendetektor“ eingefügt, und in Band II Kapitel 23 wurde eine Anwendung des „Run-Length-Coding“ im Umfeld der Objektverfolgung in Echtzeitsystemen ergänzt. Die Literaturhinweise wurden aktualisiert und durch neue Zitate erweitert.

Weiterhin konnten zahlreiche Fehler aus der 1. Auflage korrigiert werden. Dafür sei den kritischen Lesern ganz herzlich gedankt, die sich die Mühe gemacht haben, uns zu schreiben. Ganz besonders möchten wir uns an dieser Stelle bei Fr. Dipl.-Math. Beate Mielke bedanken, die alleine für ca. $\frac{2}{3}$ aller Fehlermeldungen zuständig war.

Alfred Nischwitz, Max Fischer, Peter Haberäcker, 22. Juli 2006

Vorwort

Moderne Computer sind in unserer Zeit weit verbreitet. Sie werden kommerziell und privat zur Bewältigung vielschichtiger Aufgaben eingesetzt. Als Schnittstelle zur Hardware dienen grafische Betriebssysteme, mit denen der Benutzer bequem mit dem System kommunizieren kann. Die meisten Anwendungen präsentieren sich ebenfalls in grafischer Form und begleiten so den Anwender bei der Erledigung seiner Aufgaben.

Viele benützen dabei das Computersystem wie ein praktisches Werkzeug und freuen sich, wenn alles gut funktioniert, oder ärgern sich, wenn es nicht funktioniert. Mancher wird sich aber doch fragen, was hinter der grafischen Oberfläche steckt, wie z.B. ein Computerspiel oder ein Bildbearbeitungspaket zu einer Digitalkamera gemacht wird, welche Verfahren dabei ablaufen und welche Programmiersysteme dazu verwendet werden. An diesen Personenkreis wendet sich das vorliegende Buch.

Bei der Implementierung zeitgemäßer Anwendungen nimmt die Programmierung der Computergrafik und Bildverarbeitung einen wesentlichen Teil ein. In den letzten Jahren sind diese beide Bereiche immer stärker zusammengewachsen. Dieser Entwicklung versucht das vorliegende Buch gerecht zu werden.

Der erste Teil des Buches ist der Computergrafik gewidmet. Es werden die wichtigsten Verfahren und Vorgehensweisen erläutert und an Hand von Programmausschnitten dargestellt. Als grafisches System wurde OpenGL verwendet, da es alle Bereiche abdeckt, sich als weltweiter Standard etabliert hat, plattformunabhängig und kostenlos ist, und mit modernen Grafikkarten bestens zusammenarbeitet.

Der zweite Teil befasst sich mit digitaler Bildverarbeitung, die mit ihren Verfahren die Grundlage für praktische Anwendungen von bildauswertenden und bildgenerierenden Systemen in vielen Bereichen bildet. Der Bildverarbeitungsteil ist als Zusammenfassung von zwei Büchern entstanden, die von einem der beiden Autoren vorlagen. Zunächst war geplant, diese Zusammenfassung ausschließlich über das Internet und auf CD-ROM anzubieten. Die permanente Nachfrage ließ es aber doch sinnvoll erscheinen, sie in dieses Buch zu integrieren. Das Buch wendet sich also an Interessenten, die sich in dieses Gebiet einarbeiten und praktische Erfahrungen sammeln möchten. Deshalb wurde, soweit möglich, auf die Darstellung der oft komplizierten mathematischen Hintergründe verzichtet und häufig eine eher pragmatische Vorgehensweise gewählt. Zur Vertiefung wird das Buch durch ein Internetangebot ergänzt. Hier findet der Leser Übungsaufgaben, vertiefende Kapitel und interaktive Kurse, wie sie auch an Hochschulen angeboten werden. Außerdem wird der Internetauftritt für Korrekturen und die Versionsverwaltung verwendet.

Alfred Nischwitz, Peter Haberäcker, 3. Juni 2004

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die elektronische Datenverarbeitung hat in den letzten sechs Jahrzehnten eine atemberaubende Entwicklung durchgemacht. Sie wurde ermöglicht durch neue Erkenntnisse in der Hardwaretechnologie, die Miniaturisierung der Bauteile, die Erhöhung der Rechengeschwindigkeit und der Speicherkapazität, die Parallelisierung von Verarbeitungsabläufen und nicht zuletzt die enorm sinkenden Kosten. Ende der 60er Jahre des letzten Jahrhunderts wurde z.B. der Preis für ein Bit Halbleiterspeicher mit etwa 0.50 Euro (damals noch 1.- DM) angegeben. Demnach hätte 1 GByte Hauptspeicher für einen Rechner über 4.000.000.000.- Euro gekostet.

Nachdem ursprünglich die elektronischen Rechenanlagen fast ausschließlich zur Lösung numerischer Problemstellungen eingesetzt wurden, drangen sie, parallel zu ihrer Hardwareentwicklung, in viele Gebiete unseres täglichen Lebens ein. Beispiele hierzu sind moderne Bürokommunikationssysteme, Multimedia-Anwendungen und nicht zuletzt die allgegenw"artigen Smartphones. Aus den elektronischen Rechenanlagen entwickelten sich Datenverarbeitungssysteme, die in kommerziellen und wissenschaftlichen Bereichen erfolgreich eingesetzt werden. Aber auch in den meisten privaten Haushalten sind Smartphones, Laptops oder PCs zu finden, die dort eine immer wichtigere Rolle spielen und, gerade im Multimedia-Bereich, angestammte Geräte wie Telefon, Fernseher, Stereoanlagen, DVD-Player/Recorder oder Spielekonsolen verdrängen.

Die Verarbeitung von visuellen Informationen ist ein wichtiges Merkmal höherer Lebensformen. So ist es nicht verwunderlich, dass schon frühzeitig versucht wurde, auch in diesem Bereich Computer einzusetzen, um z.B. bei einfachen, sich immer wiederholenden Arbeitsvorgängen eine Entlastung des menschlichen Bearbeiters zu erreichen. Ein gutes Beispiel ist die automatische Verarbeitung von Belegen im bargeldlosen Zahlungsverkehr. Hier wurde durch den Einsatz der modernen Datenverarbeitung nicht nur eine Befreiung des Menschen von eintöniger Arbeit erreicht, sondern auch geringere Fehlerhäufigkeit und wesentlich höhere Verarbeitungsgeschwindigkeit erzielt.

Die Benutzer von Smartphones und PC-Systemen werden heute nur mehr mit grafischen Betriebssystemen und grafisch aufbereiteter Anwendungssoftware konfrontiert. Für die Entwickler dieser Software heißt das, dass sie leistungsfähige Programmiersysteme für *Computergrafik* benötigen. OpenGL und Vulkan sind derartige Programmiersysteme für

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_1

grafische Computeranwendungen. Sie haben sich als ein weltweiter Standard etabliert und sind weitgehend unabhängig von Hard- und Softwareplattformen. Außerdem sind sie kostenlos verfügbar.

Damit der Anwender die Computergrafik sinnvoll verwenden kann, benötigt er einen Rechner mit schnellem Prozessor, ausreichendem Hauptspeicher und eine leistungsfähige Grafikkarte. Diese Forderungen erfüllen die meisten Smartphones und PC-Systeme, die überall angeboten werden. Der interessierte Computeraspirant kann sich sogar beim Einkauf neben Butter, Brot und Kopfsalat ein passendes System beschaffen, oder er lässt es sich gleich direkt nach Hause liefern.

Wenn es die Hardware der Grafikkarte erlaubt, verlagert OpenGL bzw. Vulkan die grafischen Berechnungen vom Prozessor (CPU, d.h. Central Processing Unit) des Computers auf die Grafikkarte (GPU, d.h. Graphics Processing Unit). Diese Verlagerung geschieht ebenso bei Smartphones, bei denen in der Regel CPU und GPU in einem Chip vereinigt sind (sog. SoCs, d.h. System-on-Chip). Das hat zur Folge, dass grafische Anwendungen die CPU nicht belasten und auf der eigens dafür optimierten GPU optimal ablaufen. Moderne Grafikkarten haben dabei eine Leistungsfähigkeit, die sich mit der von Großrechenanlagen messen kann. Dies hat umgekehrt dazu geführt, dass die Spitzenleistung heutiger Supercomputer zu einem wesentlichen Teil durch Grafikkarten bestimmt wird.

Bei grafischen Anwendungen, etwa bei Simulationen oder bei Computerspielen, wird angestrebt, dass auf dem Bildschirm dem Benutzer ein möglichst realistisches Szenario angeboten wird. Die Bilder und Bildfolgen werden hier ausgehend von einfachen grafischen Elementen, wie Geradenstücke, Dreiecke oder Polygonnetze, aufgebaut. Mit geometrischen Transformationen werden dreidimensionale Effekte erzielt, Beleuchtung, Oberflächengestaltung und Modellierung von Bewegungsabläufen sind weitere Schritte in Richtung realistisches Szenario. Angestrebt wird eine Darstellung, bei der der Betrachter nicht mehr unterscheiden kann, ob es sich z.B. um eine Videoaufzeichnung oder um eine computergrafisch generierte Szene handelt. Der große Vorteil ist dabei, dass der Benutzer interaktiv in das Geschehen eingreifen kann, was bei reinen Videoaufzeichnungen nur eingeschränkt möglich ist. Der Weg der Computergrafik ist also die Synthese, vom einfachen grafischen Objekt zur natürlich wirkenden Szene.

In der *digitalen Bildverarbeitung* wird ein analytischer Weg beschritten: Ausgehend von aufgezeichneten Bildern oder Bildfolgen, die aus einzelnen Bildpunkten aufgebaut sind, wird versucht, logisch zusammengehörige Bildinhalte zu erkennen, zu extrahieren und auf einer höheren Abstraktionsebene zu beschreiben.

Um das zu erreichen, werden die Originalbilddaten in rechnerkonforme Datenformate transformiert. Sie stehen dann als zwei- oder mehrdimensionale, diskrete Funktionen für die Bearbeitung zur Verfügung. Die Verfahren, die auf die digitalisierten Bilddaten angewendet werden, haben letztlich alle die Zielsetzung, den Bildinhalt für den Anwender passend aufzubereiten. Der Begriff „Bildinhalt" ist dabei rein subjektiv: Dasselbe Bild kann zwei Beobachtern mit unterschiedlicher Interessenlage grundsätzlich verschiedene Informationen mitteilen. Aus diesem Grund werden auch die Transformationen, die beide Beobachter auf das Bild anwenden, ganz verschieden sein. Das Ergebnis kann, muss aber nicht in bildlicher oder grafischer Form vorliegen. Es kann z.B. auch eine Kommandofolge zur Steuerung eines

autonomen Fahrzeugs oder Roboters sein.

Mit der digitalen Bildverarbeitung verwandte Gebiete sind die *Mustererkennung* und das *maschinelle Lernen*, bzw. deren Oberbegriff die *künstliche Intelligenz*. Die Mustererkennung bzw. das maschinelle Lernen ist im Gegensatz zur digitalen Bildverarbeitung nicht auf bildhafte Informationen beschränkt. Die Verarbeitung von akustischen Sprachsignalen mit der Zielsetzung der Sprach- oder Sprechererkennung ist z.B. ein wichtiger Anwendungsbereich der Mustererkennung. Im Bereich bildhafter Informationen wird mit den Verfahren der Mustererkennung versucht, logisch zusammengehörige Bildinhalte zu entdecken, zu gruppieren und so letztlich abgebildete Objekte (z.B. Buchstaben, Bauteile, Fahrzeuge) zu erkennen. Um hier zufriedenstellende Ergebnisse zu erzielen, sind in der Regel umfangreiche Bildvorverarbeitungsschritte durchzuführen.

Künstliche Intelligenz ist ein Oberbegriff, der für viele rechnerunterstützte Problemlösungen verwendet wird (z.B. natürlich-sprachliche Systeme, Robotik, Expertensysteme, automatisches Beweisen, Bildverstehen, kognitive Psychologie, Spiele). Im Rahmen der Verarbeitung von bildhafter Information wird die Ableitung eines Sinnzusammenhangs aus einem Bild oder einer Bildfolge versucht. Eine Beschreibung der Art: ,,Ein Bauteil liegt mit einer bestimmten Orientierung im Sichtbereich", kann dann in eine Aktion umgesetzt werden, etwa das Greifen und Drehen des Bauteils mit einer industriellen Handhabungsmaschine. Hier werden also Systeme angestrebt, die im Rahmen eines wohldefinierten ,,Modells der Welt" mehr oder weniger unabhängig agieren und reagieren. Diese Selbstständigkeit ist meistens erst nach einer langen Anwendungskette von Bildverarbeitungs- und Mustererkennungsalgorithmen möglich.

Wenn ein Bild oder eine Bildfolge analytisch aufbereitet ist, kann der Informationsgehalt symbolisch beschrieben sein. Bei einer Bildfolge einer Straßenszene könnte das etwa so aussehen:

- Die Bildfolge zeigt eine Straße, die von rechts vorne nach links hinten verläuft.
- Auf der Straße bewegen sich zwei Fahrzeuge in entgegengesetzter Richtung und etwa gleicher Geschwindigkeit.
- Bei dem Fahrzeug, das von rechts vorne nach links hinten fährt, handelt es sich um ein rotes Cabriolet vom Typ X des Herstellers Y.
- Bei dem Fahrzeug, das von links hinten nach rechts vorne fährt, handelt es sich um einen weißen Minitransporter vom Typ XX des Herstellers YY.
- Links neben der Straße ist ein Wiesengelände, rechts ein Nadelwald.
- Den Horizont bildet eine Bergkette, die zum Teil mit Schnee bedeckt ist.
- usw.

Im Rahmen der Beschreibung könnten auch die Nummernschilder der beiden Fahrzeuge vorliegen. Dann wäre es als Reaktion z.B. möglich, beiden Fahrzeughaltern einen Bußgeldbescheid zuzusenden, da ihre Fahrzeuge mit überhöhter Geschwindigkeit unterwegs waren.

Eine andere interessante Möglichkeit wäre es, aus der symbolischen Beschreibung mit Computergrafik eine Szene zu generieren und zu vergleichen, wie realistisch sie die ursprüngliche Straßenszene wiedergibt.

Mit diesem Beispiel wurde gezeigt, wie eng Computergrafik und Bildverarbeitung heute miteinander verknüpft sind. Dieser Tatsache versucht das vorliegende Werk gerecht zu werden. Es gliedert sich in zwei Bände, die über zahlreiche Referenzen untereinander verbunden sind.

Der Band I befasst sich mit der Thematik „Computergrafik". Nach einem Kapitel, in dem ausführlich auf den Zusammenhang zwischen Computergrafik, Bildverarbeitung und Mustererkennung eingegangen wird, folgen Kapitel über interaktive 3D-Computergrafik und ihre Anwendungen. Ab Kapitel 6 werden einzelne Bestandteile der Computergrafik, wie Grundobjekte, Koordinatentransformationen, Verdeckung, Farbverarbeitung, Anti-Aliasing, Beleuchtung, Schatten und Texturen beschrieben. An Hand von zahlreichen Programmfragmenten wird gezeigt, wie OpenGL bzw. Vulkan die jeweiligen Problemstellungen unterstützt. Den Abschluss des Computergrafik-Teils bildet ein Kapitel über „General Purpose Computing on Graphics Processing Units (GPGPU)", d.h. die Nutzung von Grafikkarten für allgemeine, aber parallelisierbare Programmieraufgaben mit Hilfe der Programmierschnittstelle OpenCL (Open Compute Language von der Khronos Group) bzw. CUDA (Compute Unified Device Architecture von NVIDIA).

Der Band II des Werkes ist der digitalen Bildverarbeitung gewidmet. Zunächst wird die Digitalisierung von Bilddaten untersucht und, ausgehend vom Binärbild (Zweipegelbild) über das Grauwertbild, das Farbbild bis zur Bildfolge verallgemeinert. Anschließend werden Maßzahlen zur Beschreibung digitalisierter Bilddaten vorgestellt und verschiedene mathematische Modelle für Bilder behandelt. Bei der Diskussion der Speicherung von digitalisierten Bilddaten in Datenverarbeitungssystemen werden Lösungsansätze zur Datenreduktion und Datenkompression vorgestellt. Einem Abschnitt über die bildliche Reproduktion von digitalisierten gespeicherten Bildern schließen sich Verfahren zur Modifikation der Grauwertverteilung an. Weiter folgt die Untersuchung von Operationen im Orts- und Frequenzbereich, von morphologischen Operationen und von Kanten und Linien.

Die weiteren Kapitel sind mehr in Richtung Mustererkennung bei bildhaften Daten orientiert. Nach der grundlegenden Darstellung der Szenenanalyse werden unterschiedliche Techniken zur Merkmalsgewinnung besprochen. Stichworte hierzu sind: Grauwert und Farbe, Merkmale aus mehrkanaligen Bildern und aus Bildfolgen. Der Beschreibung von einfachen Texturmerkmalen schließen sich aufwändigere Verfahren, wie Gauß- und Laplace-Pyramiden, Scale Space Filtering und Baumstrukturen an. In den anschließenden Kapiteln wird die Segmentierung mit klassischen Methoden, mit neuronalen Netzen und mit Fuzzy Logic beschrieben. Nach dem Übergang von der bildpunkt- zur datenstrukturorientierten Bildverarbeitung werden unterschiedliche Verfahren zur Segmentbeschreibung erläutert. Den Abschluss bildet ein Kapitel über Kalman Filter und der Synthese von Objekten aus Segmenten.

Zu diesem Buch liegt auch ein Internetangebot unter folgender Adresse vor:

https://w3-o.cs.hm.edu/users/nischwit/public_html/cgbv-buch/index.html

Der Zugang zu dieser Webseite ist passwort-geschützt. Den Benutzernamen und das Passwort erhält man, wenn man auf dieser Webseite dem Link „Passwort“ folgt. Das Passwort wird regelmäßig geändert.

Der Online-Service umfasst folgende Angebote:

- interaktive Vorlesungen zu den Themen:
  - Computergrafik
  - Bildverarbeitung und Mustererkennung
  - Bilddatencodierung für die Übertragung und Kommunikation
- Übungsaufgaben
- Das Bildverarbeitungswerkzeug IGARIS als GIMP-Plugin
- Bilder bzw. Texturen
- Zusatzkapitel zum Buch

Kapitel 2

Zusammenhang zwischen Computergrafik und Bildverarbeitung

Warum fasst man die Gebiete Computergrafik und Bildverarbeitung in einem Buch zusammen? Weil sie die zwei Seiten einer Medaille sind: während man in der Computergrafik aus einer abstrakten Objektbeschreibung ein Bild generiert, versucht man in der Bildverarbeitung nach der Extraktion von charakteristischen Merkmalen die im Bild enthaltenen Objekte zu erkennen und so zu einer abstrakten Objektbeschreibung zu kommen (Bild 2.1). Oder anders ausgedrückt: Computergrafik ist die Synthese von Bildern und Bildverarbeitung ist die Analyse von Bildern. In diesem Sinne ist die Computergrafik die inverse Operation zur Bildverarbeitung.

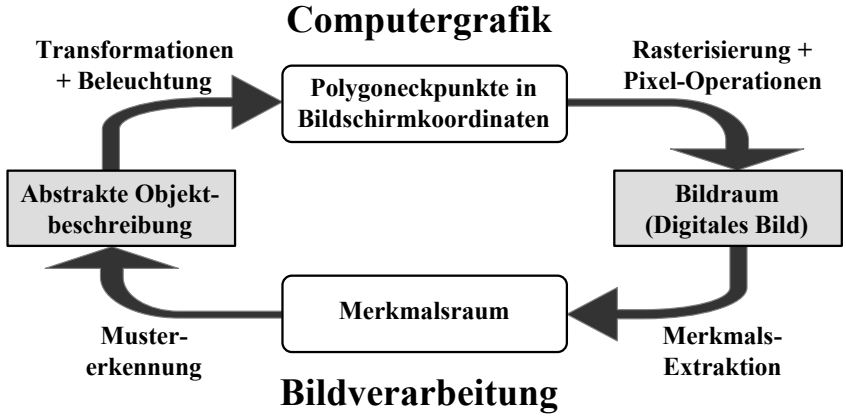


Bild 2.1: Die zwei Seiten einer Medaille: Computergrafik ist die Synthese von Bildern aus einer abstrakten Objektbeschreibung und Bildverarbeitung ist die Analyse von Bildern mit dem Ziel, zu einer abstrakten Objektbeschreibung zu gelangen.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_2

Für Computergrafik und Bildverarbeitung benötigt man in vielen Teilen die gleichen Methoden und das gleiche Wissen. Dies beginnt beim Verständnis für den Orts- und Frequenzbereich, das Abtasttheorem, das Anti-Aliasing und die verschiedenen Farbräume, setzt sich fort bei den Matrizen-Operationen der linearen Algebra (Geometrie-Transformationen wie Translation, Rotation, Skalierung und perspektivische Projektion, Texturkoordinaten-Transformation, Transformationen zwischen den Farbräumen etc.), geht über die Nutzung von Faltungs- und Morphologischen Operatoren bis hin zu Datenstrukturen, wie z.B. Gauß-Pyramiden (MipMaps), Quad- bzw. Octrees, sowie Graphen zur Szenenbeschreibung.

Nachdem Computergrafik und Bildverarbeitung häufig die gleichen Algorithmen einsetzen und die Grafikhardware in den letzten 10 Jahren einen gigantischen Leistungssprung um den Faktor 100 geschafft hat, liegt es Nahe, teure Spezialhardware zur Bildverarbeitung durch billige PC-Grafikkarten zu ersetzen, wie im folgenden Abschnitt erläutert wird. Bei einer zunehmenden Anzahl von Anwendungen wird Computergrafik und Bildverarbeitung gleichzeitig eingesetzt, wie am Beispiel der Simulation von kameragesteuerten Geräten erläutert wird. Im Multimedia-Bereich entsteht eine immer engere Verzahnung von Computergrafik und Bildverarbeitung. Dies wird anhand moderner Bilddatencodierungsmethoden erklärt. Im Rahmen des im letzten Abschnitt vorgestellten „bildbasierten Renderings“ löst sich die Trennung zwischen klassischer Computergrafik und Bildverarbeitung immer mehr auf.

2.1 Bildverarbeitung auf programmierbarer Grafikhardware

Der mit weitem Abstand größte Leistungszuwachs in der Computerhardware hat in den letzten 15 Jahren im Bereich der Grafikhardware stattgefunden. Eine aktuelle nVIDIA GeForce GTX 1080 Ti Grafikkarte hat etwa die 5000-fache Rechenleistung wie die vor 15 Jahren aktuelle GeForce 2 MX, und etwa die 100-fache Floating-Point-Rechenleistung wie die derzeit schnellsten „Core i7“-Prozessoren. Zu verdanken ist dieser enorme Leistungszuwachs in der Grafikhardware vor allem den Millionen Kindern, die von den Möglichkeiten interaktiver Computerspiele fasziniert wurden. Seit Anfang 2002 sind diese Grafikkarten auch noch relativ gut durch Hochsprachen (Band I Abschnitt 5.1.3.3) programmierbar, so dass sie auch für andere Zwecke als nur Computergrafik genutzt werden können. Allerdings sind programmierbare Grafikkarten nicht bei allen Rechenaufgabe schneller als gewöhnliche Prozessoren, sondern nur bei solchen, für die die Grafikhardware optimiert ist, wie z.B. Vektor- und Matrizen-Operationen. Genau diese Operationen werden aber sowohl in der Computergrafik als auch in der Bildverarbeitung sehr häufig benötigt. Ein weiterer wichtiger Grund für die extrem hohe Rechenleistung von Grafikkarten ist der hohe Parallelisierungsgrad in der Hardware. So arbeiten in der bereits erwähnten GeForce GTX 1080 Ti Grafikkarte beispielsweise 3584 Shaderprozessoren parallel. Um diese 3584 Shaderprozessoren gleichmäßig auszulasten, benötigt man eine Rechenaufgabe, die trivial

parallelisierbar ist, wie z.B. die Texturierung aller Pixel eines Polygons in der Computergrafik (Band I Kapitel 13), oder auch die Faltung eines Bildes mit einem Filterkern in der Bildverarbeitung (Band II Abschnitt 5.2).

Die Grundidee besteht nun darin, sich die riesige Rechenleistung heutiger Grafikkarten für eine schnelle Echtzeit-Bildverarbeitung zu Nutze zu machen und somit teure Spezialhardware (FPGAs = Field Programmable Gate Arrays) zur Bildverarbeitung durch billige und hochsprachen-programmierbare Grafikkarten bzw. eingebettete Systeme, wie z.B. nVIDIA's Jetson TX2, zu ersetzen. Da FPGAs nie zu einem richtigen Massenprodukt geworden sind, das in millionenfacher Stückzahl hergestellt worden wäre, resultiert ein erheblich höherer Stückpreis als bei PC-Grafikkarten. Außerdem wurden für FPGAs nie wirklich gute Hochsprachen zur Programmierung entwickelt[1], so dass sie wie Computer in ihrer Frühzeit durch Assembler-Code gesteuert werden müssen. Da jedes FPGA seine eigene ganz spezifische Assembler-Sprache besitzt, muss für jede neue Generation an FPGAs eine teure Anpassentwicklung durchgeführt werden. All diese Probleme entfallen bei den billigen und hochsprachen-programmierbaren Grafikkarten.

Um die Umsetzbarkeit dieser Grundidee in die Praxis zu überprüfen, wurden im Labor für Computergrafik und Bildverarbeitung an der Hochschule München entsprechende Untersuchungen durchgeführt. Dabei wurde zunächst die Implementierbarkeit verschiedener Bildverarbeitungs-Algorithmen auf programmierbaren Grafikkarten sehr erfolgreich getestet. Dazu zählten Faltungsoperatoren (Band II Kapitel 5), wie z.B.

- der gleitende Mittelwert (Band II 5.2),
- der Gauß-Tiefpassfilter (Band II 5.4 und Bild 2.2-b)
  mit unterschiedlich großen Faltungskernen ($3 \times 3, 7 \times 7, 11 \times 11$),
- der Laplace-Operator (Band II 5.30),
- der Sobelbetrags-Operator (Band II 5.28),
- die HOG-Features (Histogramm of Oriented Gradients, Band II 16.3.3),

und morphologische Operatoren (Band II Kapitel 6), wie z.B.

- die Dilatation(Band II 6.3),
- die Erosion(Band II 6.3 und Bild 2.2-c),
- und der Median-Filter (Band II Abschnitt 6.3).

Weitere Beispiele für komplexe Bildverarbeitungsoperationen, die auf programmierbarer Grafikhardware implementiert wurden, sind z.B.

- die Fourier-Transformation [Suma05],

[1] Es gibt zwar mittlerweile sog. High Level Tools, wie z.B. Vivado HLS von der Fa. Xilinx, aber die Erfahrungen damit sind eher gespalten.

- die Hough-Transformation[Strz03],
- der Canny-Kantendetektor [Fung05],
- der KCF-Tracker [Peso16],
- und weitere Tracking-Algorithmen [Fung05].

Häufig werden in der Bildverarbeitung jedoch auch mehrere verschiedene Operatoren hintereinander auf ein Bild angewendet. Dies ist auch auf der programmierbaren Grafikhardware möglich, indem man verschiedene *Shader* in einem sogenannten *Multipass-Rendering-Verfahren* mehrfach über ein Bild laufen lässt (Bild 2.2-d). Die Implementierung kann entweder über die `glCopyTexImage2D`-Funktion (Band I Abschnitt 13.1.1.2) oder über die *Render-to-Texture*-Option (auch *pBuffer*-Funktion genannt) erfolgen, die neuere Grafikkarten bieten.

Ein Leistungsvergleich zwischen Grafikhardware (GPU = *Graphics Processing Unit*) und CPU (*Central Processing Unit*) in Bezug auf einige der oben aufgeführten Bildverarbeitungs-Algorithmen brachte erstaunliche Ergebnisse. So war die Grafikhardware (nVidia GeForce GTX 1080 mobile) bei den Beispielen aus Bild 2.2 um einen Faktor von ca. 100 schneller als die CPU (Intel Core i7-7700K)! Die auf der Grafikhardware erzielbaren Bildraten (*FPS = Frames Per Second*) betrugen gigantische 2522 Bilder/sec bei morphologischen Operatoren (Bild 2.2-c) und immerhin noch 523 Bilder/sec bei der Anwendung eines Gauß-Tiefpassfilters mit einem nicht separiertem $7 \cdot 7$-Filterkern (Bild 2.2-b). Damit ist nicht mehr die Bildverarbeitung der Flaschenhals in Echtzeit-Anwendungen, sondern der Datentransfer von der Kamera in den Computer. Oder anders ausgedrückt: heutzutage kann man sehr aufwändige Bildverarbeitungs-Operationen in Echtzeit auf Grafikkarten durchführen, die sich fast jeder leisten kann.

Das Grundprinzip der Implementierung ist einfach: es muss nur ein bildschirmfüllendes Rechteck gezeichnet werden, auf das das zu bearbeitende Bild mit Hilfe des *Texture-Mappings* (Band I Kapitel 13) aufgebracht wird. Im Fragment-Shader (Band I Abschnitt 5.1.3.2), der den Bildverarbeitungsoperator enthält, werden die Farbwerte des Bildes jetzt nicht nur Pixel für Pixel kopiert, sondern entsprechend dem verwendeten Operator miteinander verknüpft. Programm-Beispiele für verschiedene Punkt-, Faltungs- und Rangordnungsoperatoren werden in Band I Abschnitt 13.6 vorgestellt.

Die vielfältigen Möglichkeiten von programmierbaren Grafikkarten wurden mittlerweile auch für eine Reihe weiterer Anwendungen außerhalb der Computergrafik und Bildverarbeitung erkannt. So werden bereits Software-Pakete für Dynamik-Simulationen angeboten, die nicht mehr wie bisher auf der CPU ausgeführt werden, sondern auf der Grafikkarte (GPU). Für besonders anspruchsvolle Computerspiele werden neuerdings spezielle PCs angeboten, die zwei extrem leistungsfähige Grafikkarten enthalten, eine Karte für die Computergrafik und die zweite Karte für die Physik-Simulationen. Weitere Anwendungen basieren auf der Finiten-Elemente-Methode (FEM) bzw. im allgemeinen auf dem Lösen von großen linearen Gleichungssystemen, wie z.B. Aero- und Fluiddynamik-Simulationen, oder auch virtuelle

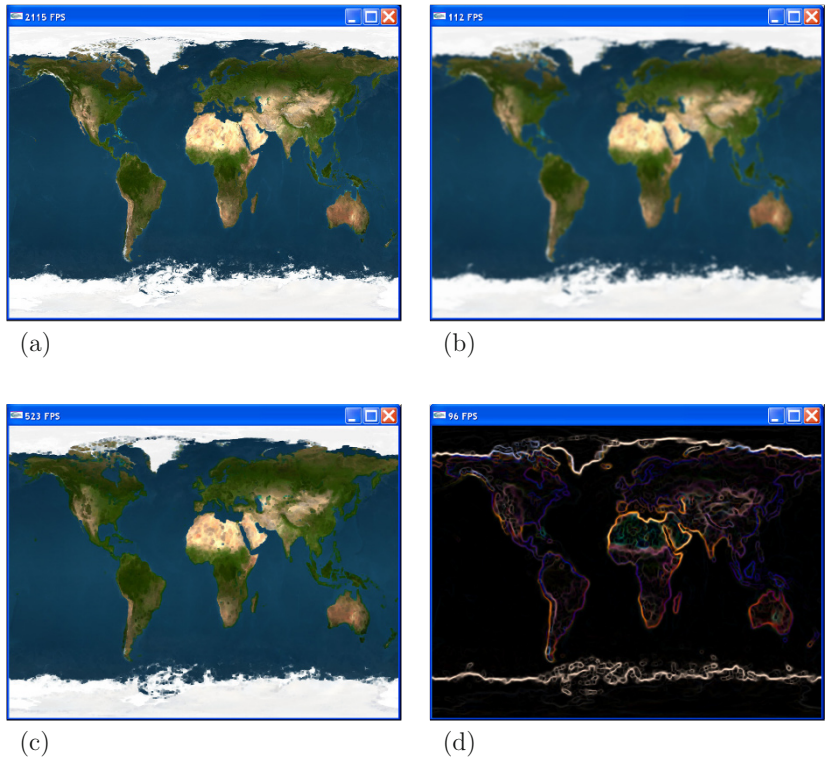


Bild 2.2: Beispiele für Bildverarbeitung auf programmierbarer Grafikhardware: (a) Originalbild. (b) Faltungsoperator: Gauß-Tiefpass mit $7 \cdot 7$-Filterkern und $\sigma = 2$. (c) Morphologischer Operator: Erosion. (d) Kombination aus den drei Operatoren Gauß-Tiefpass $7 \cdot 7$, Sobelbetrag und Erosion.

Crash-Tests. Nachdem die Hardware-Architekten derzeit auch bei CPUs auf Parallelisierung setzen (Stichwort *Dual- und Multi-Core CPUs*), wurden schon Überlegungen angestellt ([Owen05]), ob nicht der Impuls für die Weiterentwicklung von Computerhardware generell durch den rasanten Fortschritt der Grafikhardware gesetzt wurde.

2.2 Simulation von kameragesteuerten Geräten

Ein gutes Beispiel für das Zusammenwirken von Computergrafik und Bildverarbeitung ist die Simulation von kameragesteuerten Geräten. Solche Geräte sind z.B. autonome mobile Roboter oder Lenkflugkörper, die eine Videokamera als wesentlichen Sensor besitzen. Um ein solches Gerät in seiner Interaktion mit der Umwelt simulieren zu können, muss der

Sensor – in diesem Fall die Videokamera – mit geeigneten Stimuli, d.h. Bildern, versorgt werden. Eine Möglichkeit, solche Bilder zur Verfügung zu stellen, besteht darin, einen Videofilm aufzuzeichnen und ihn später wieder in die Simulation einzuspeisen. Ein großer Vorteil dieser Technik ist die absolute Realitätsnähe der Bilder, da sie ja in der realen Umwelt mit dem realen Sensor aufgezeichnet werden können. Der entscheidende Nachteil dieser Technik ist, dass keine Interaktion zwischen dem Gerät und der Umwelt mehr möglich ist, d.h. die Regelschleife der Simulation ist offen (*open loop simulation*) . Um die Regelschleife zu schließen (*closed loop simulation*), ist eine interaktive Bildgenerierung notwendig, d.h. aus der bei jedem Zeitschritt neu bestimmten Position und Lage des Geräts muss ein aktuelles Bild generiert werden. Da es unmöglich ist, für alle denkbaren Positionen und Orientierungen der Kamera reale Bilder aufzuzeichnen, muss das Bild aus einer visuellen 3D-Datenbasis mit Hilfe der Computergrafik interaktiv erzeugt werden. Die Computergrafik generiert in diesem Fall direkt den Input für die Bildverarbeitung.

Aus systemtheoretischer Sicht lässt sich die Simulation solcher Systeme zunächst einmal grob in zwei Bereiche unterteilen: Das kameragesteuerte Gerät auf der einen Seite, das mit der Umwelt auf der anderen Seite interagiert (Bild 2.3). Eine Simulation, in der sowohl die Umwelt, wie auch das kameragesteuerte Gerät durch ein rein digitales Modell ersetzt wird, bezeichnet man als „Mathematisch Digitale Simulation“ (MDS). Die MDS ist also ein Software-Paket, das auf jedem beliebigen Computer ablaufen kann. Das Modell des kameragesteuerten Geräts besteht aus:

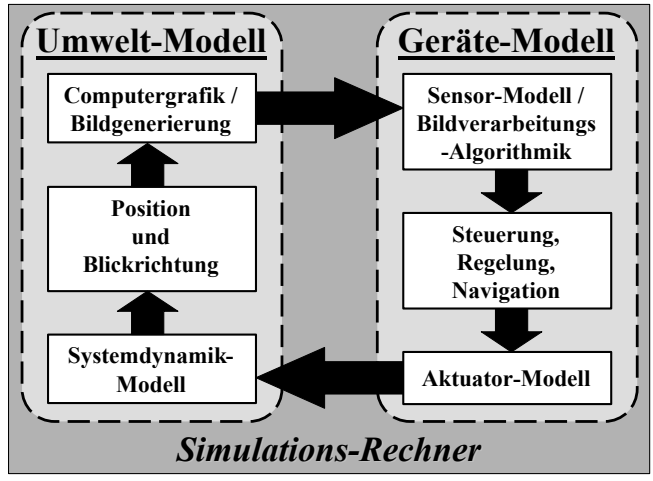


Bild 2.3: Mathematisch Digitale Simulation (MDS) eines kameragesteuerten Geräts. Im Rahmen einer geschlossenen Regelschleife interagieren die Komponenten des Geräts auf der rechten Seite mit den Komponenten der Umwelt auf der linken Seite. Alle Komponenten bestehen aus digitalen Modellen, die gemeinsam auf einem Simulations-Rechner ablaufen.

- Der Sensorik (hier ein Kamera-Modell) zur Erfassung der Umwelt,
- der Bildverarbeitungs-Algorithmik zur Objekterkennung und -verfolgung,
- der Steuer-, Regel- und Navigations-Algorithmik, die auf der Basis der gestellten Aufgabe (Führungsgröße) und der erfassten Situation (Messgröße) eine Reaktion (Stellgröße) des Geräts ableitet,
- der Aktorik (z.B. Lenkrad bzw. Ruder zur Einstellung der Bewegungsrichtung sowie dem Antrieb zur Regelung der Geschwindigkeit).

Das Modell der Umwelt besteht aus:

- Einem Systemdynamik-Modell, das zunächst aus der Stellung der Aktoren die auf das System wirkenden Kräfte und Drehmomente berechnet, und anschließend aus den Bewegungsgleichungen die neue Position und Orientierung des Geräts,
- einer visuellen Datenbasis, die die 3-dimensionale Struktur und die optischen Eigenschaften der Umwelt enthält. Mit Hilfe der Computergrafik wird aus der errechneten Position und Blickrichtung der Kamera der relevante Ausschnitt aus der visuellen Datenbasis in ein Bild gezeichnet. Danach wird das computergenerierte Bild in das Kamera-Modell eingespeist, und der nächste Durchlauf durch die Simulationsschleife startet.

Die MDS ist eines der wichtigsten Werkzeuge für die Entwicklung und den Test kameragesteuerter Systeme und somit auch für die Bildverarbeitungs-Algorithmik. Ein Vorteil der MDS ist, dass in der Anfangsphase eines Projekts, in der die Ziel-Hardware noch nicht zur Verfügung steht, die (Bildverarbeitungs-) Algorithmik bereits entwickelt und in einer geschlossenen Regelschleife getestet werden kann. In späteren Projektphasen wird die Onboard-Bildverarbeitungs-Software ohne Änderungen direkt in die MDS portiert. Deshalb sind Abweichungen vom realen Verhalten, z.B. aufgrund unterschiedlicher Prozessoren, Compiler bzw. des zeitlichen Ablaufs, sehr gering. Im Rahmen der Validation der MDS mit der *Hardware-In-The-Loop* Simulation bzw. mit dem realen System werden evtl. vorhandene Unterschiede minimiert. Da die MDS somit das Verhalten der realen Bildverarbeitungs-Komponente sehr präzise reproduzieren kann, wird sie am Ende der Entwicklungsphase auch zum Nachweis der geforderten Leistungen des Geräts verwendet. Dies spart Kosten und ist wegen des häufig sehr breiten Anforderungsspektrums (unterschiedliche Landschaften kombiniert mit verschiedenen Wetterbedingungen, Tageszeiten, Störungen, etc.) in der Regel nur noch mit einer großen Anzahl virtueller Versuche (d.h. Simulationen) in der MDS realisierbar.

Voraussetzung für eine verlässliche Simulation ist aber nicht nur eine 1:1-Abbildung der Bildverarbeitungs-Software in der MDS, sondern auch die interaktive Generierung möglichst realistätsnaher Bilder mit Hilfe der Computergrafik. Das Ziel der Computergrafik muss sein, dass das Ergebnis der Bildverarbeitung bei computergenerierten Bildern das Gleiche ist wie bei realen Bildern. Und dabei lässt sich ein Algorithmus nicht

so leicht täuschen wie ein menschlicher Beobachter. Schwierigkeiten bei der Generierung möglichst realitätsnaher Bilder bereitet die natürliche Umwelt mit ihrem enormen Detailreichtum und den komplexen Beleuchtungs- und Reflexionsverhältnissen, sowie das zu perfekte Aussehen computergenerierter Bilder (die Kanten sind zu scharf und zu gerade, es gibt keine Verschmutzung und keine Störungen). Dies führt in der Regel dazu, dass die Bildverarbeitungs-Algorithmik in der Simulation bessere Ergebnisse liefert als in der Realität. Um diesen Lerneffekt nach den ersten Feldtests zu vermeiden, ist eine Validation der computergenerierten Bilder anhand realer Videoaufzeichnungen erforderlich. Erst wenn das Ergebnis der Bildverarbeitung im Rahmen bestimmter Genauigkeitsanforderungen zwischen synthetisierten und real aufgenommenen Bildern übereinstimmt, kann der Simulation genügend Vertrauen geschenkt werden, um damit Feldversuche einzusparen bzw. den Leistungsnachweis zu erbringen.

Damit computergenerierte Bilder kaum noch von realen Bildern unterscheidbar sind, muss fast immer ein sehr hoher Aufwand betrieben werden. Um die 3-dimensionale Gestalt der natürlichen Umwelt möglichst genau nachzubilden, ist eine sehr große Anzahl an Polygonen erforderlich (Band I Kapitel 6). Zur realistischen Darstellung komplexer Oberflächen- und Beleuchtungseffekte sind sehr viele Foto-Texturen, Relief-Texturen, Schatten-Texturen usw. notwendig (Band I Kapitel 13), sowie aufwändige Beleuchtungsrechnungen (Band I Kapitel 12). Die Glättung zu scharfer Kanten kann mit Hilfe des Anti-Aliasing, d.h. einer rechenaufwändigen Tiefpass-Filterung erreicht werden (Band I Kapitel 10). Luftverschmutzung und andere atmosphärische Effekte können mit Hilfe von Nebel simuliert werden (Band I Kapitel 11). Die Liste der Maßnahmen zur Steigerung des Realitätsgrades computergenerierter Bilder könnte noch um viele weitere und zunehmend rechenintensivere Punkte ergänzt werden. Wichtig ist aber nicht, dass das computergenerierte Bild in allen Eigenschaften exakt dem realen Bild entspricht, sondern dass diejenigen Merkmale, die die Bildverarbeitung später für die Objekterkennung nutzt, möglichst gut reproduziert werden. Für einen effizienten Einsatz der Ressourcen bei der Computergrafik ist es deshalb unerlässlich, zu verstehen, mit welchen Algorithmen die Bildverarbeitung die erzeugten Bilder analysiert.

In diesem Zusammenhang ist es ein Vorteil der MDS als reinem Software-Paket, dass die Simulation auch langsamer als in Echtzeit erfolgen kann. Wenn eine Komponente, wie z.B. die Bildgenerierung, sehr viel Rechenzeit benötigt, warten die anderen Simulationskomponenten, bis das Bild fertig gezeichnet ist. Dadurch hat man in der MDS die Möglichkeit, auch sehr detailreiche Szenarien in die Simulation zu integrieren, so dass die computergenerierten Bilder auch höchsten Anforderungen genügen. Deshalb ist die MDS ideal geeignet, um den statistischen Leistungsnachweis für Subsysteme (z.B. die Bildverarbeitungs-Komponente) und das Gesamtsystem mit höchster Genauigkeit durchzuführen.

Der Vorteil der MDS, wegen der nicht vorhandenen bzw. „weichen“ Echtzeit-Anforderung (Band I Abschnitt 3.2) beliebig detailreiche Szenarien darstellen zu können, wandelt sich aber im Hinblick auf die Verifikation der Bildverarbeitungs-Software auf dem Zielrechner in einen Nachteil um. Denn auf dem Onboard-Rechner des Geräts muss die Bildverarbeitungs-Software in Echtzeit getestet werden, damit deren Funktionstüchtigkeit und das Zusammenspiel mit anderen Komponenten nachgewiesen werden kann. Deshalb

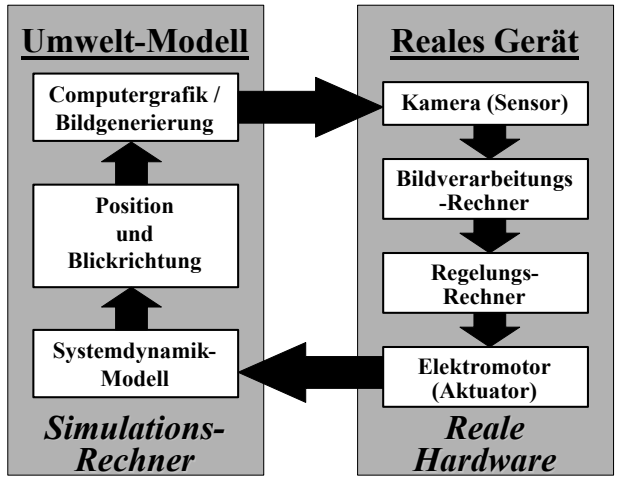


Bild 2.4: Hardware-In-The-Loop (HIL) Simulation eines kameragesteuerten Geräts. Das reale Gerät (oder evtl. nur einzelne Komponenten davon) interagiert in Echtzeit mit den simulierten Komponenten der Umwelt.

wird in späteren Phasen eines Entwicklungsprojekts, in denen die Komponenten des Geräts bereits als reale Prototypen zur Verfügung stehen, eine *Hardware-In-The-Loop* (HIL)-Simulation (Bild 2.4) aufgebaut, mit der die gesamte Regelschleife des Geräts in Echtzeit getestet wird. In einer HIL-Simulation muss die Umwelt weiterhin simuliert werden, allerdings in Echtzeit. Für die Computergrafik bedeutet dies eine hohe Anforderung, da für die Bildgenerierrate in einer HIL-Simulation eine „harte“ Echtzeitanforderung gilt (Band I Abschnitt 3.1). Verarbeitet die Kamera z.B. 50 Bilder pro Sekunde, muss die Computergrafik mindestens mit dieser Rate neue Bilder generieren. Dies bedeutet, dass pro Bild maximal 20 Millisekunden Rechenzeit zur Verfügung stehen und deshalb gewisse Einschränkungen bei der Bildqualität in Kauf genommen werden müssen. Durch den enormen Fortschritt bei der Leistungsfähigkeit von Grafikhardware und neue Beschleunigungsverfahren für Echtzeit-3D-Computergrafik (Band I Kapitel 16) ist zu erwarten, dass bald ein ausreichend hoher Realitätsgrad der Bilder bei interaktiven Generierraten (50 Bilder pro Sekunde und mehr) erzielt werden kann, damit der Leistungsnachweis auch in einer Echtzeit-Simulation erbracht werden kann.

2.3 Bilddatencodierung

Die Videocodierstandards MPEG-1 (Moving Picture Experts Group, ISO/IEC Standard 11172) und MPEG-2 (ISO/IEC Standard 13818) dienen der komprimierten digitalen Repräsentation von audiovisuellen Inhalten, die aus Sequenzen von rechteckigen 2-dimensionalen Bildern und den zugehörigen Audiosequenzen bestehen. Der Erfolg dieser Videoco-

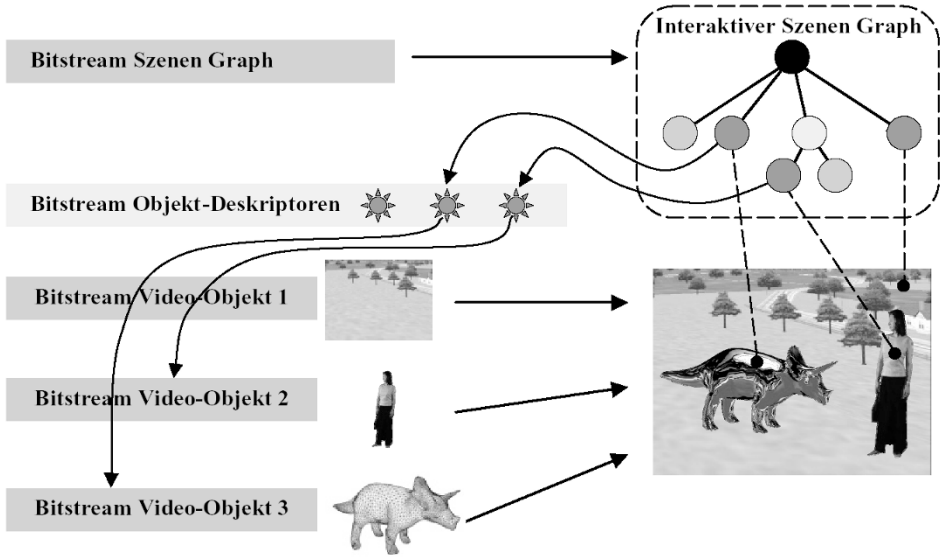


Bild 2.5: Szenen Graph zur Beschreibung der Video-Objekte in MPEG-4.

dierstandards verhalf einigen Produkten, wie Video-CD, DVD-Video, Digitales Fernsehen, Digitales Radio und MP3-Geräten (MPEG-1 Audio layer 3), zum kommerziellen Durchbruch. Methoden der 3D-Computergrafik, der Bildverarbeitung und der Mustererkennung kommen dabei (fast[2]) nicht zum Einsatz. Den Videocodierstandards MPEG-1 und MPEG-2 liegt das Paradigma des passiven Zuschauers bzw. Zuhörers zugrunde, genau wie beim Fernsehen oder Radio. Der Konsument hat in diesem Fall keine Möglichkeit zur Interaktion mit den audiovisuellen Inhalten bzw. mit dem Anbieter dieser Inhalte.

Genau in diese Lücke stößt der neuere Codierstandard MPEG-4 (ISO/IEC Standard 14496). Er stellt ein objekt-orientiertes Konzept zur Verfügung, in dem eine audiovisuelle Szene aus mehreren Video-Objekten zusammengesetzt wird. Die Eigenschaften der Video-Objekte und ihre raum-zeitlichen Beziehungen innerhalb der Szene werden mit Hilfe einer Szenenbeschreibungssprache in einem dynamischen Szenen Graphen definiert (Bild 2.5). Dieses Konzept ermöglicht die Interaktion mit einzelnen Video-Objekten in der Szene, denn der Szenen Graph kann jederzeit verändert werden [Pere02]. Die Knoten des Szenen Graphen enthalten sogenannte „Objekt-Deskriptoren“, die die individuellen Eigenschaften der Objekte beschreiben. Ein Objekt-Deskriptor kann z.B. ausschließlich eine Internet-Adresse (URL = *Uniform Resource Locator*) enthalten. Dadurch ist es möglich, die Inhalte einer audiovisuellen Szene durch Objekte zu ergänzen, die auf Media-Servern rund um die

[2]Bis auf das Blockmatching zur Berechnung von Bewegungsvektorfeldern in Bildsequenzen (Band II Abschnitt 14.5).

Welt verteilt sind. Voraussetzung für die Darstellung solcher Szenen ist dann natürlich ein entsprechender Internet-Zugang. Im Normalfall enthält ein Objekt-Deskriptor das eigentliche Video-Objekt (einen sogenannten „*elementary stream*“), sowie Zusatzinformationen zu dem Video-Objekt (eine Beschreibung des Inhalts und der Nutzungsrechte).

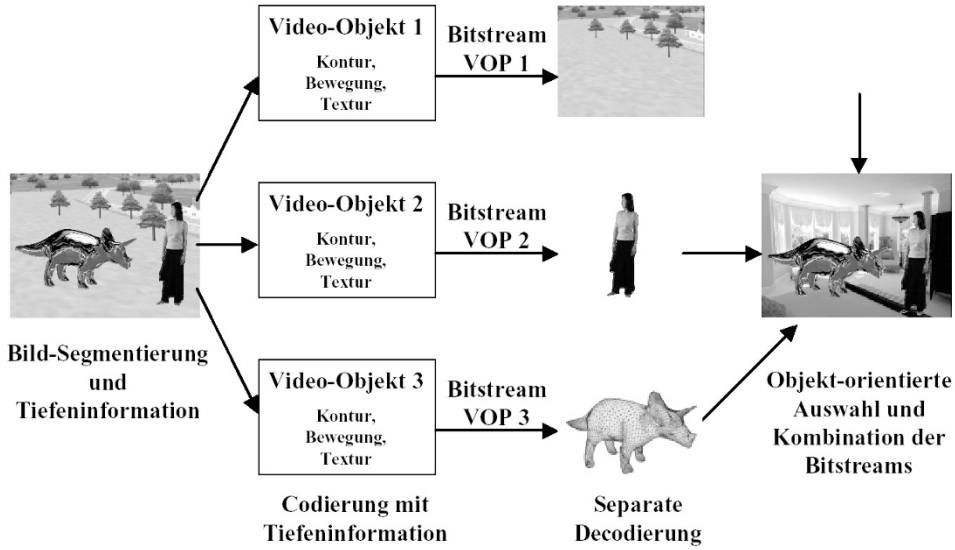


Bild 2.6: Zerlegung einer Bildsequenz in einzelne MPEG-4 Video-Objekte (*Video Object Planes* = VOPs). Nach der Decodierung der einzelnen Video-Objekte besteht die Möglichkeit, die Objekte neu zu kombinieren. Somit kann die Szene interaktiv und objektbezogen verändert werden.

Die äußere Form der Video-Objekte kann, im Unterschied zu MPEG-1/-2, variieren. Während bei MPEG-1/-2 nur ein einziges Video-Objekt codiert werden kann, das aus einer Sequenz von rechteckigen Bildern besteht, erlaubt MPEG-4 die Codierung mehrerer Video-Objekte, die jeweils eine Sequenz von beliebig geformten 2-dimensionalen Bildern enthalten. MPEG-4 erlaubt daher die Zerlegung eines Bildes in einzelne Segmente, die sich unterschiedlich bewegen. Jedes Segment ist ein Video-Objekt, das von Bild zu Bild in seiner Form veränderlich ist. In Bild 2.5 ist eine Szene gezeigt, die in drei Video-Objekte zerlegt werden kann: Einen statischen Hintergrund, eine bewegte Person und ein bewegtes Tier. Für jedes Video-Objekt wird die Kontur, die Bewegung und die Textur in einem eigenen Bitstream codiert. Der Empfänger decodiert die Video-Objekte und setzt sie wieder zur vollständigen Szene zusammen. Da der Empfänger jedes Video-Objekt separat decodiert und er den Szenen Graph interaktiv verändern kann, hat er die Möglichkeit, die einzelnen Video-Objekte nach seinen Vorstellungen zu kombinieren. In Bild 2.6 ist diese Möglichkeit illustriert: Hier werden zwei Video-Objekte aus Bild 2.5 mit einem neuen Video-Objekt für

den Hintergrund kombiniert, so dass die Person und der Dinosaurier nicht mehr im Freien stehen, sondern in einem Zimmer.

Der Inhalt eines Video-Objekts kann eine „natürliche“ Videosequenz sein, d.h. ein sogenanntes „*natural video object*“, das mit einer Kamera in einer „natürlichen“ Umgebung aufgezeichnet wurde (wie die Person in Bild 2.5 bzw. 2.6), oder ein künstliches Objekt (*synthetic video object*), aus dem der Konsument mit Hilfe von interaktiver 3D-Computergrafik eine synthetische Videosequenz generieren kann (der Dinosaurier in Bild 2.5 bzw. 2.6). Ein einfaches synthetisches Video-Objekt kann z.B. ein Text sein, der später einem Bild überlagert wird (*text overlay*). Anspruchsvollere synthetische Video-Objekte sind 3-dimensionale Computergrafik-Modelle, die im Wesentlichen aus Polygonnetzen, Farben, Normalenvektoren und Texturen bestehen. Ein großer Vorteil von synthetischen Video-Objekten gegenüber natürlichen Video-Objekten besteht darin, dass der Blickwinkel, unter dem sie dargestellt werden, interaktiv und frei wählbar ist. Der Grad an Interaktivität, den synthetische Video-Objekte bieten, ist demnach sehr viel höher, als der von natürlichen Video-Objekten.

In MPEG-4 können also nicht nur klassische Videofilme, sondern auch 3D-Computergrafik-Modelle und Kombinationen von beiden effizient codiert werden. Somit ist es z.B. möglich, in eine natürliche 2D-Videosequenz ein computergeneriertes 3D-Modell eines Dinosauriers einzublenden. Diese Technik, die in der Filmproduktion für Trickeffekte mittlerweile Standard ist, kann mit MPEG-4 von jedem Konsumenten genutzt werden. Umgekehrt ist es ebenso machbar, in eine 3-dimensionale virtuelle Szene eine natürliche 2D-Videosequenz einzublenden (dies ist genau die gleiche Idee, wie bei animierten Texturen auf Billboards in der Echtzeit-3D-Computergrafik, Band I Abschnitt 16.4). Die Kombination von synthetischen und natürlichen Bildinhalten im Rahmen einer effizienten Codierung ist ein sehr mächtiges Werkzeug, das im MPEG-Fachjargon „*Synthetic and Natural Hybrid Coding* (SNHC)“ genannt wird ([Pere02]).

Voraussetzung für eine effiziente Bilddatencodierung in MPEG-4 ist die Kombination anspruchsvoller Methoden der 3D-Computergrafik, der Bildverarbeitung und der Mustererkennung. MPEG-4 bietet mehrere Möglichkeiten an, um sehr hohe Kompressionsraten bei vorgegebener Bildqualität für natürliche Videosequenzen zu erreichen:

- Der Einsatz blockbasierter Verfahren zur Codierung von Bewegungsvektoren, Texturen und Objektkonturen.

- Der Einsatz von 2-dimensionalen Polygonnetzen zur Codierung von Objektkonturen und -bewegungen in Verbindung mit einer Abbildung von Texturen auf das Polygonnetz.

- Der Einsatz von 3-dimensionalen Polygonnetzen zur Codierung von Objektkonturen und -bewegungen in Verbindung mit einer Textur-Abbildung.

Der erste Schritt bei allen drei Verfahren ist die Zerlegung eines Bildes in einzelne Segmente. Die dafür nötigen Bildverarbeitungsalgorithmen werden in MPEG-4 bewusst nicht festge-

legt, sondern den Entwicklern eines Codecs[3] überlassen. Typischerweise werden zunächst verschiedene Bildmerkmale extrahiert, wie z.B. Bewegungsvektorfelder (Band II Kapitel 14) und Texturmerkmale (Band II Kapitel 15). Auf dieser Basis erfolgt nun die Bildsegmentierung, für die die Bildverarbeitung eine ganze Reihe von Verfahren zur Verfügung stellt, wie z.B. Minimum-Distance- und Maximum-Likelihood-Klassifikatoren (Band II Kapitel 20), Neuronale Netze (Band II Kapitel 21), oder Fuzzy Logic (Band II Kapitel 22). Da in der Regel nicht nur Einzelbilder segmentiert werden, sondern Bildfolgen, lohnt es sich, Kalman-Filter zur Schätzung der Segmentbewegungen in aufeinander folgenden Bildern einzusetzen (Band II Kapitel 27).

Nach dem Segmentierungsschritt trennen sich die Wege der drei Codierverfahren. Beim blockbasierten Verfahren wird genau wie bei transparenten Texturen in der 3D-Computergrafik (Band I Abschnitt 9.3.4) ein vierter Farbkanal, der sogenannte Alpha- oder Transparenz-Kanal, eingeführt. Jeder Bildpunkt des gesamten rechteckigen Bildfeldes bekommt einen Alpha-Wert zugewiesen, wobei Bildpunkte, die zum segmentierten Video-Objekt gehören, den Alpha-Wert 1 (nicht transparent), und alle anderen Bildpunkte den Alpha-Wert 0 (transparent)[4] bekommen. Die Alpha-Bitmasken der einzelnen Video-Objekte werden nun blockweise arithmetisch codiert.

Beim 2D-netzbasierten Verfahren wird jedem Segment ein 2-dimensionales Netz aus verbundenen Dreiecken (Band I Abschnitt 6.2.3.6) nach bestimmten Optimierungskriterien zugewiesen. Die Topologie des Dreiecksnetzes für ein Segment darf sich innerhalb einer Bildsequenz nicht ändern, nur seine Form. Aus diesem Grund genügt es, beim ersten Bild die Anzahl und die 2D-Positionen der Eckpunkte zu codieren, für alle folgenden Bilder des Video-Objekts muss man nur noch die Verschiebungsvektoren für die Eckpunkte codieren. Die Verfolgung der Netz-Eckpunkte in einer Bildfolge, d.h. die Bestimmung der Verschiebungsvektoren kann wieder sehr gut mit Hilfe eines Kalman-Filters durchgeführt werden. Die Bilddaten des Video-Objekts werden als Textur auf das 2D-Dreiecksnetz abgebildet. Da sich die Textur innerhalb der Bildsequenz eines Video-Objekts kaum ändert, können die geringfügigen Texturänderungen zwischen zwei aufeinander folgenden Bildern sehr gut komprimiert werden. Die Codierung der Objekte erfolgt nun in zwei Abschnitten:

- Für das erste Bild einer Folge muss die Geometrie-Information in Form von Polygonnetzen und die Bildinformation in Form von Texturen codiert werden. Dies erfordert zu Beginn eine hohe Datenrate.
- Für die folgenden Bilder der Folge müssen nur noch die Verschiebungsvektoren der Netzeckpunkte und die geringfügigen Texturänderungen codiert werden, so dass bei den Folgebildern nur noch eine relativ niedrige Datenrate nötig ist.

Auf der Empfängerseite werden die Bilder mit Hilfe von 3D-Computergrafik schließlich wieder in Echtzeit erzeugt.

Die Codierung 3-dimensionaler Polygonnetze in MPEG-4 dient eigentlich nicht dem Zweck, noch höhere Kompressionsraten bei natürlichen Video-Objekten zu erreichen, als

[3] Codec = Software zur *Co*dierung und *Dec*odierung.

[4] Es gibt auch einen Modus in MPEG-4, der 256 (8 bit) verschiedene Transparenzwerte zulässt.

mit 2D-Netzen, sondern dazu, statische 3D-Modelle für interaktive Computergrafik oder hybride Anwendungen (SNHC) effizient zu codieren und so deren Verbreitung über Kommunikationnetze (Internet, Rundfunk, Mobilfunk etc.) zu fördern. Es gibt allerdings über MPEG-4 hinausgehende Ansätze, bei denen versucht wird, die 3D-Geometrie segmentierter Video-Objekte aus der Bildfolge mit Hilfe eines Kalman-Filters zu rekonstruieren und Bewegungen des 3D-Netzes zu verfolgen (Band II Abschnitt 27.2.3 und [Calv00]). MPEG-4 bietet zwar keinen allgemeinen Ansatz für die Verwendung von 3D-Netzen zur Komprimierung natürlicher Video-Objekte, aber für die wirtschaftlich bedeutenden Anwendungsfelder Bildtelefonie und Videokonferenzen können zwei spezielle 3D-Netzmodelle vordefiniert werden: Je ein Prototyp für einen menschlichen Kopf (*face animation*) und einen menschlichen Körper (*body animation*). MPEG-4 lässt zur Bewegung dieser 3D-Netze nur einen eingeschränkten Satz an Animationsparametern zu (68 *face* bzw. 168 *body animation parameters*). Zu Beginn der Übertragung muss deshalb kein komplexes 3D-Netz codiert werden, sondern nur ein Satz von Animationsparametern. Während der laufenden Übertragung müssen nicht mehr die Verschiebungsvektoren aller Eckpunkte des 3D-Netzes codiert werden, sondern nur noch die Änderungen der Animationsparameter. Mit diesem Konzept lassen sich extrem niedrige Bitraten (*very low bitrate coding*) erzielen.

Die geschilderten Codierverfahren bei MPEG-4 und die Entwicklungstendenzen bei den modernen Codierstandards MPEG-7 (ISO/IEC 15938, *Multimedia Content Description Interface*) und MPEG-21 (ISO/IEC 21000, *Multimedia Framework*) zeigen deutlich die immer engere Verquickung zwischen Computergrafik, Bildverarbeitung und Codierung.

2.4 Bildbasiertes Rendering

In der klassischen 3D-Computergrafik werden Objekte durch Polygon-Netze beschrieben (Band I Kapitel 7). Dies ermöglicht die Betrachtung bzw. das Rendering[5] der Objekte aus beliebigen Blickwinkeln, ohne dass sich dabei die polygonale Repräsentation der Objekte ändert. Ein entscheidender Vorteil dieser Methode ist, dass sich der Betrachter interaktiv durch eine 3-dimensionale Szene bewegen kann. Der Nachteil dieser Technik ist, dass eine realitätsgetreue Darstellung von Szenen sehr schwierig bzw. sehr aufwändig ist. Probleme bereiten dabei vor allem komplexe Oberflächen von natürlichen Objekten (Gelände, Pflanzen, Lebewesen etc.). So ist es mit einem rein polygonalen Modell fast unmöglich, z.B. eine echt wirkende Gras- oder Felloberfläche darzustellen.

In der Bildverarbeitung hat man es in der Regel mit 2-dimensionalen Bildern oder Bildfolgen zu tun, die mit einer Kamera aufgenommen wurden. Der Vorteil dabei ist, dass auch komplexeste Szenen unter schwierigsten Lichtverhältnissen exakt eingefangen werden. Ein wesentlicher Nachteil der Fotografie ist, dass die Betrachtung der Szene nur aus einem einzigen Blickwinkel – dem der aufnehmenden Kamera – möglich ist. Eine interaktive Bewegung des Betrachters durch eine 3-dimensionale Szene ist daher nicht realisierbar.

Die Kombination von polygon-orientierter 3D-Computergrafik und Bildverarbeitung ist schon seit langem im Rahmen des „Textur Mappings“ (Band I Kapitel 13), also dem „Auf-

[5]Rendering = engl. Fachbegriff für Bild mit dem Computer generieren oder zeichnen

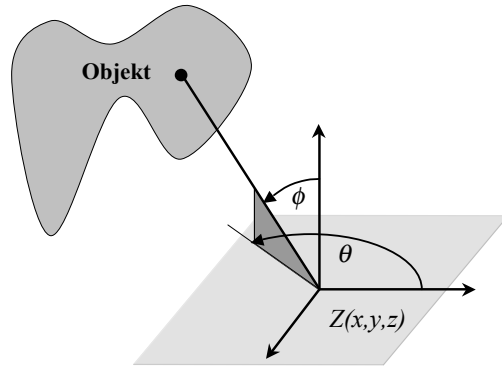


Bild 2.7: Die plenoptische Funktion p definiert für jeden Raumpunkt (x, y, z), jede Blickrichtung (θ, ϕ), jeden Zeitpunkt t und jede Wellenlänge λ eine Lichtintensität I.

kleben“ von Foto-Texturen auf Polygone, etabliert. Das Neue an der Idee des bildbasierten Renderings (*Image Based Rendering* (IBR)) ist der völlige Verzicht auf eine polygonale Repräsentation der Szene. In seiner Reinkultur wird beim bildbasierten Rendering eine Szene, durch die sich der Beobachter beliebig bewegen kann, ausschließlich aus Bilddaten aufgebaut. Dies setzt aber voraus, dass an jeder möglichen Beobachterposition (x, y, z), für jede mögliche Blickrichtung (θ, ϕ) und zu jedem Zeitpunkt (t), der darstellbar sein soll, ein Foto für jede Wellenlänge λ gespeichert werden muss. Man bezeichnet die Summe dieser unendlich vielen Fotos als *plenoptische Funktion p*. Sie ordnet jedem Strahl, der durch das Zentrum $Z = (x, y, z)$ einer virtuellen Kamera geht, eine Intensität I zu (Bild 2.7):

$$I = p(x, y, z, t, \theta, \phi, \lambda) \tag{2.1}$$

Die plenoptische Funktion enthält also sieben Parameter, und entspricht daher einer 7-dimensionalen Textur. Selbst bei einer groben Quantisierung der sieben Dimensionen wäre für die Erzielung einer akzeptablen Bildqualität eine gigantische Speicherkapazität[6] von ca. 35.000.000 $TeraByte$ erforderlich. Der kontinuierliche Bereich der Spektralfarben wird in der Computergrafik und Bildverarbeitung an drei Stellen (Rot, Grün und Blau) abgetastet. Dadurch lässt sich die plenoptische Funktion als Vektor darstellen, dessen drei Komponenten nur noch von sechs Dimensionen abhängen.

[6]Quantisierung des Raums: $x, y, z = 1km$ á $1m = 1000$, der Zeit: $t = 1min$ á $30Hz = 1800$, des Azimutwinkels: $\theta = 360°$ á $0,1° = 3600$, des Elevationswinkels: $\phi = 180°$ á $0,1° = 1800$ und des Wellenlängenspektrums: $\lambda = R, G, B = 3$, d.h. $10^3 \cdot 10^3 \cdot 10^3 \cdot 1,8 \cdot 10^3 \cdot 3,6 \cdot 10^3 \cdot 1,8 \cdot 10^3 \cdot 3 \approx 3,5 \cdot 10^{19}$ Bildpunkte $\approx 35.000.000$ $TeraByte$, bei $1Byte = 256$ Intensitätswerten.

Für statische Szenen reduziert sich die plenoptische Funktion weiter auf fünf Dimensionen:

$$\mathbf{p}(x,y,z,\theta,\phi) = \begin{pmatrix} p_R(x,y,z,\theta,\phi) \\ p_G(x,y,z,\theta,\phi) \\ p_B(x,y,z,\theta,\phi) \end{pmatrix} \tag{2.2}$$

In dem dargestellten Rechenexempel reduziert sich der Speicherplatzbedarf um einen Faktor 1800 (1 Minute bei 30 Hz) auf 20.000 *TeraByte*. Dies sind Größenordnungen, die sicher noch für einige Jahrzehnte eine praktische Realisierung der plenoptischen Funktion verhindern werden. Die plenoptische Funktion ist daher ein idealisierter Grenzfall des bildbasierten Renderings, der die Weiterentwicklung der Computergrafik in eine neue Richtung lenken kann. Von praktischer Relevanz sind daher Spezialfälle der plenoptischen Funktion, die mit der klassischen, polygonalen 3D-Computergrafik kombiniert werden.

Beschränkt man die plenoptische Funktion auf einen festen Standort, lässt aber den Blickwinkel frei wählbar, so erhält man ein Panorama der Szene. Die plenoptische Funktion vereinfacht sich dabei auf zwei Dimensionen:

$$\mathbf{p}(\theta,\phi) = \begin{pmatrix} p_R(\theta,\phi) \\ p_G(\theta,\phi) \\ p_B(\theta,\phi) \end{pmatrix} \tag{2.3}$$

Das von der Firma Apple eingeführte System „Quicktime VR“ legt die Szene in Form eines zylindrischen Panoramafotos ab. Der Beobachter kann sich in (fast[7]) allen Richtungen interaktiv umschauen und er kann zoomen. Allerdings kann er keine translatorischen Bewegungen ausführen. In der 3D-Computergrafik hat sich eine andere Variante des Panoramafotos etabliert: Die „kubische Textur“ (Band I Abschnitt 13.4.2). Sie wird dadurch erzeugt, dass man die Umgebung aus der Position des Beobachter im Zentrum des Kubus' sechs mal mit einem Öffnungswinkel von 90° fotografiert oder rendert, und zwar so, dass die sechs Würfelflächen genau abgedeckt werden. Die sechs Einzeltexturen müssen also an den jeweiligen Rändern übergangslos zusammen passen. In Band I Bild 13.23 ist ein Beispiel für eine kubische Textur dargestellt. Der Speicherplatzbedarf ist mit ca. 10 *MegaByte* moderat (sechs 2D-Texturen) und die Erzeugung verhältnismäßig einfach. Kubische Texturen können als „*Sky Boxes*“ eingesetzt werden, um den komplexen Hintergrund durch ein Panoramafoto zu ersetzen. Davor können konventionell modellierte 3D-Objekte aus Polygon-Netzen platziert und animiert werden. In der Praxis wird die Panorama-Technik auch bei translatorisch bewegten Beobachtern eingesetzt. Solange sich der Beobachter nur in einem eingeschränkten Bereich innerhalb des Kubus' bzw. Zylinders bewegt und die in den Panoramafotos abgebildeten Objekte relativ weit von der Kamera entfernt waren, sind die Bildfehler, die aufgrund von Parallaxenveränderungen entstehen, praktisch vernachlässigbar. Kubische Texturen eignen sich außerdem ausgezeichnet, um Spiegelungs- oder Brechungseffekte zu simulieren (Band I Abschnitt 13.4.2).

Die Umkehrung des Panoramafotos ist das blickpunktabhängige „*Billboarding*“ (Band I Abschnitt 16.4): Der Blickwinkel ist immer fest auf ein Objekt gerichtet, aber der Ort des

[7]Bei einem zylindrischen Panoramafoto kann der Beobachter einen beliebigen Azimutwinkel wählen, aber nur einen eingeschränkten Bereich an Elevationswinkeln.

Beobachters ist (in gewissen Grenzen) frei. Falls die Ausdehnung des betrachteten Objekts klein im Verhältnis zum Abstand Objekt – Beobachter ist, kommt es nur auf den Raumwinkel an, aus dem der Beobachter das Objekt betrachtet. Der Beobachter sitzt gewissermaßen an einer beliebigen Stelle auf einer kugelförmigen Blase, die das Objekt weiträumig umschließt, und blickt in Richtung Objektmittelpunkt. Die Position des Beobachters auf der Kugeloberfläche kann durch die beiden Winkel α und β eines Kugelkoordinatensystems beschrieben werden (anstatt der drei kartesischen Koordinaten x, y, z). Die (eingeschränkte) Blickrichtung wird weiterhin durch die Winkel (θ, ϕ) festgelegt. Damit reduziert sich die statische plenoptische Funktion von fünf auf vier Dimensionen:

$$\mathbf{p}(\alpha, \beta, \theta, \phi) = \begin{pmatrix} p_R(\alpha, \beta, \theta, \phi) \\ p_G(\alpha, \beta, \theta, \phi) \\ p_B(\alpha, \beta, \theta, \phi) \end{pmatrix} \tag{2.4}$$

Das plenoptische Modell des Objekts besteht also aus einer Ansammlung von Fotos des Objekts, die (mit einer gewissen Quantisierung) von allen Punkten der Kugeloberfläche in Richtung des Objekts aufgenommen wurden (Band I Bild 16.15). Dieses plenoptische Objekt kann nun in eine konventionell modellierte 3D-Szene aus Polygon-Netzen integriert werden. Tritt das plenoptische Objekt ins Sichtfeld, wird das Foto ausgewählt, dessen Aufnahmewinkel den Betrachterwinkeln am nächsten liegen. Diese Foto-Textur wird auf ein Rechteck gemappt, das senkrecht auf dem Sichtstrahl steht, und anschließend in den Bildspeicher gerendert. Die Teile der rechteckigen Foto-Textur, die nicht zum Objekt gehören, werden als transparent gekennzeichnet und somit nicht gezeichnet. Damit lässt sich bei komplexen Objekten, die sonst mit aufwändigen Polygon-Netzen modelliert werden müssten, sehr viel Rechenzeit während der interaktiven Simulation einsparen. Allerdings geht dies auf Kosten eines stark vergrößerten Texturspeicherbedarfs in der Größenordnung von *GigaByte*[8]. Mit modernen Codierverfahren (JPEG, MPEG) sind aufgrund der großen Ähnlichkeit benachbarter Bilder hohe Kompressionsfaktoren (ca. 100) möglich, so dass der Texturspeicherbedarf eher akzeptabel wird. Eine weitere Technik zur Reduktion des Speicherplatzbedarfs, bei der Bildverarbeitung und Mustererkennung eine wesentliche Rolle spielt, ist die Interpolation des Blickwinkels [Watt02]. Die Grundidee ist dabei die Gleiche, wie bei der Bewegungskompensation in modernen Codierverfahren (Band II Abschnitt 27.2.3): Man berechnet aus zwei benachbarten Bildern ein Verschiebungsvektorfeld (Band II Abschnitt 14.5), das angibt, durch welchen Verschiebungsvektor jedes Pixel aus dem ersten Bild in das korrespondierende Pixel des zweiten Bildes überführt wird. Jeder Blickwinkel zwischen den Winkeln, unter denen die beiden Bilder aufgenommen wurden, kann jetzt durch eine Interpolation der Verschiebungsvektoren näherungsweise erzeugt werden. Damit lässt sich die Anzahl der Bilder, die für die plenoptische Modellierung eines Objekts notwendig ist, deutlich senken. Im Gegenzug steigt natürlich der Rechenaufwand während der laufenden Simulation wieder an.

[8] Ein plenoptisches Objekt, bei dem die Einzelbilder eine Auflösung von $1280 \cdot 1024$ Pixel zu je 24 bit besitzen und das im Azimutwinkel 64 mal bzw. im Elevationswinkel 32 mal abgetastet wird, benötigt ca. 8 *GigaByte* an Speicherplatz.

Eine ähnliche Technik wie das blickpunktabhängige Billboarding ist das Lichtfeld-Rendering (*Light Fields* bzw. *Lumigraph*, [Watt02]). Im Gegensatz zum Billboarding, bei dem eine bestimmte Anzahl an Fotos von einer Kugeloberfläche in Richtung des Mittelpunkts erzeugt werden, wird die Kamera beim Lichtfeld-Rendering in einer Ebene parallel verschoben. Dabei werden in äquidistanten Abständen Fotos aufgenommen, die zusammen eine 4-dimensionale plenoptische Funktion definieren. Mit gewissen Einschränkungen bzgl. Blickwinkel und Position ist damit eine freie Bewegung einer virtuellen Kamera durch die Szene darstellbar, da für jede zugelassene Position und Orientierung des Beobachters die notwendigen Bilddaten gespeichert sind.

Aufgrund des sehr hohen Speicherplatzbedarfs von blickpunktabhängigen Billboards werden schon seit Längerem bestimmte Spezialfälle eingesetzt, die entweder nur unter gewissen Einschränkungen anwendbar sind, oder bei denen Abstriche bei der Bildqualität hingenommen werden müssen. In vielen Anwendungen kann sich der Beobachter z.B. nur auf einer Ebene bewegen, so dass die Blickwinkelabhängigkeit des Billboards um eine Dimension reduziert werden kann. Die plenoptische Funktion hängt in diesem Fall nur noch von drei Winkeln ab:

$$\mathbf{p}(\alpha,\theta,\phi) = \begin{pmatrix} p_R(\alpha,\theta,\phi) \\ p_G(\alpha,\theta,\phi) \\ p_B(\alpha,\theta,\phi) \end{pmatrix} \tag{2.5}$$

In diesem Fall werden nur Fotos von einem Kreis um das Objekt benötigt (Band I Bild 16.15). Falls das Objekt rotationssymmetrisch ist (wie in guter Näherung z.B. Bäume), oder die Darstellung des Objekts aus einem Blickwinkel ausreicht, genügt zur Modellierung ein einziges Foto des Objekts. Dies ist das klassische Billboard, bei dem eine Foto-Textur mit Transparenz-Komponente (Band I Kapitel 9) auf ein Rechteck gemappt wird, das sich immer zum Beobachter hin ausrichtet (Band I Bild 16.14). Bewegte Objekte mit inneren Freiheitsgraden, wie z.B. Fußgänger, können durch ein Billboard mit animierten Texturen, d.h. einer kurzen Bildsequenz, dargestellt werden. Die plenoptische Funktion eines solchen Modells enthält als Variable die Zeit t: $\mathbf{p} = \mathbf{p}(t,\theta,\phi)$. Einen Mittelweg zwischen vorab gespeicherten blickpunktabhängigen Billboards und zur Laufzeit berechneten 3D-Geometriemodellen stellen *Impostors* dar. Darunter versteht man die Erzeugung der Foto-Textur eines Objekts während der Laufzeit durch klassisches Rendering der Geometrie, sowie das anschließende Mapping auf ein normales Billboard. Solange sich der Beobachter nicht allzu weit von der Aufnahmeposition entfernt hat, kann ein solcher Impostor anstelle des komplexen 3D-Modells für eine gewisse Zeit genutzt werden. Bewegt sich der Beobachter weiter weg, muss ein neuer Impostor gerendert werden.

Wie dargestellt, existiert mittlerweile ein nahezu kontinuierliches Spektrum an Möglichkeiten, was den Anteil an Geometrie bzw. Bilddaten bei der Modellierung von 3D-Szenen betrifft. Inwieweit eher der klassische, polygonbasierte Ansatz oder mehr die plenoptische Funktion für das Rendering genutzt werden, hängt von der Anwendung ab: Sind Geometrie-Daten leicht zugänglich (wie bei CAD-Anwendungen) oder eher Fotos? Welche Hardware steht zur Verfügung? Was sind die Anforderungen an die Darstellungsqualität? Welche Einschränkungen gelten für die Position, den Blickwinkel und die Bewegung des Beobachters?

In jedem Fall verschwimmt die strikte Trennlinie zwischen klassischer Computergrafik und Bildverarbeitung bei der Bildgenerierung immer mehr.

In diesen vier Beispielen für den immer enger werdenden Zusammenhang zwischen Computergrafik und Bildverarbeitung tauchen viele Begriffe auf, die dem Leser möglicherweise (noch) nicht geläufig sind. Dies soll jedoch nicht abschrecken, sondern vielmehr Appetit auf die kommenden Kapitel machen.

Kapitel 3

Digitale Bilddaten

3.1 Prinzipielle Vorgehensweise bei sichtbeeinflussten Anwendungen

An den Anfang des Bildverarbeitungsteils und dieses Grundlagenkapitels ist die Beschreibung einer „prinzipiellen Vorgehensweise“, wie sie bei Anwendungen der digitalen Bildverarbeitung und Mustererkennung beobachtet werden kann, gestellt. Grundsätzlich gibt es zwei Zielsetzungen der digitalen Bildverarbeitung: Die *Bildverbesserung* und die *Extraktion von Information*. Der Schwerpunkt des Bildverarbeitungsteils liegt auf der Informationsextraktion. Die Techniken der Bildverbesserung werden immer dann herangezogen, wenn sie die Informationsextraktion erleichtern oder überhaupt erst ermöglichen.

3.1.1 Sensoren

Die richtige Wahl von geeigneten Sensoren für eine aktuelle Problemstellung ist von grundlegender Bedeutung. Dabei können Kriterien maßgebend sein wie z.B.:

- Welche physikalischen Messgrößen sind für die Charakterisierung der Objekte der aktuellen Problemstellung notwendig?
- Welche Ausschnitte aus dem elektromagnetischen Spektrum liefert ein bestimmter Sensor?
- Welche Datenmenge liefert der Sensor?
- In welcher Zeit muss die Datenmenge verarbeitet werden?
- Ist eine multisensorielle Lösung notwendig?
- usw.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_3

Dabei spielt die richtige Beleuchtung eine wesentliche Rolle. Beispiele dazu sind: Auflicht, Durchlicht, diffuses Licht, Verwendung von Farbfiltern oder Blitz. Ähnliche grundsätzliche Überlegungen müssen auch bei nicht-optischen Verfahren angestellt werden, z.B. bei Nuklear-Scannern, Röntgengeräten, usw.

Ein allgemeiner Grundsatz für diesen Bereich ist: Je mehr Sorgfalt man bei der Sensor- und Lichtauswahl verwendet, desto einfacher werden möglicherweise anschließende Bearbeitungsverfahren. Es ist immer besser, wenn möglich gewisse Probleme schon vor der Digitalisierung zu beseitigen, als dies danach mit hohem Aufwand an Hard- und Software zu versuchen.

3.1.2 Digitalisierung

Digitale Bilder kann man aus unterschiedlichen Quellen erhalten. Digitalkameras liefern bereits digitale Bilder. Bei Verwendung von analogen Kameras ist man entweder auf Framegrabber oder auf digitale Videoserver angewiesen. Scanner kommen meist in Bereichen zum Einsatz, wo man papiergebundene Informationen digital weiterverarbeiten möchte. Man sollte sich hier Gedanken über die benötigte Auflösung machen: Ist sie zu gering, so verliert man eventuell benötigte feine Bildstrukturen. Bei zu hoher Auflösung belastet man das verarbeitende System mit großen Datenmengen und, daraus resultierend, mit langen Rechenzeiten.

Hier sollte man nochmals überprüfen, ob der gewählte Sensor auch die benötigten Daten liefert. Wenn sich z.B. zwei Objekte nur in ihrem Gewicht unterscheiden, so wäre es unsinnig, das Merkmal „Gewicht“ mit einem Videosensor erfassen zu wollen. Wenn somit der Sensor die charakteristischen Merkmale, die zur Unterscheidung der Objekte einer Problemstellung notwendig sind, nicht aufzeichnet, ist er nicht geeignet. Diese Binsenweisheit wird manchmal vergessen! Es wird dann oft versucht, mit riesigem Aufwand aus den digitalisierten Daten Merkmale zur Unterscheidung zu berechnen, die man mit anderen Sensoren problemlos erhalten hätte.

3.1.3 Vorverarbeitung der Rohdaten

In diesem Schritt werden Fehler, die durch die Aufzeichnung in das digitalisierte Datenmaterial eingefügt wurden, aus den Rohdaten entfernt. Beispiele dazu sind: Rauschen durch die Atmosphäre oder die Elektronik, Verzerrungen durch das Linsensystem, Randabschattungen durch die Beleuchtung, Unschärfe durch Bewegung, usw.

Um diese Fehler zu korrigieren, können Verfahren eingesetzt werden wie die Veränderung von Helligkeit und Kontrast, das Angleichen von Farbauszügen, die digitale Filterung im Orts- und Frequenzbereich oder geometrische Korrekturen.

3.1.4 Berechnung von Merkmalen

Nachdem die Rohdaten vorverarbeitet sind, können Merkmale zur Charakterisierung der Objekte berechnet werden. Dabei wird die Vorgehensweise meistens abgestuft sein: Zu-

nächst werden Merkmale für die Segmentierung des Bildes berechnet, dann Merkmale zur Beschreibung der Segmente und schließlich Merkmale, die es erlauben, die erkannten Segmente zu sinnvollen Objekten zusammenzufügen.

Bei Verwendung eines Videosensors können aus den vorverarbeiteten Rohdaten verschiedene Merkmalstypen berechnet werden:

- Rein *bildpunktbezogene Merkmale* wie: Helligkeit, Farbton oder multispektrale Eigenschaften (Signatur).
- *Umgebungsbezogene Merkmale* wie: Kontrast, Gradient oder andere Maßzahlen für die Oberflächenstruktur (Textur) oder die Form.
- *Zeitbezogene Merkmale*, die die Bewegung oder das allgemeine zeitliche Verhalten erfassen.

3.1.5 Segmentierung des Bildes

Bei der Segmentierung wird das Bild nach Maßgabe der berechneten Merkmale in einheitliche Bereiche (*Segmente*) aufgeteilt. Hier kommen Verfahren zum Einsatz wie die Binärbilderzeugung, multivariate statistische Klassifikatoren, geometrische Klassifikatoren, neuronale Netze oder Operatoren auf der Basis der *fuzzy logic*.

Auch hier ist es wichtig, die passende Wahl zu treffen. Wenn z.B. ein Bild durch ein einfaches statisches Schwellwertverfahren in die Klassen „Bildhintergrund“ und „Objekt“ segmentiert werden kann, so wäre der Einsatz eines neuronalen Netzes überdimensioniert.

3.1.6 Kompakte Speicherung der Segmente

Zur Reduzierung der weiteren Datenmengen und damit der Rechenzeiten wird es häufig sinnvoll sein, die Segmente mit geeigneten Datenstrukturen so zu speichern, dass redundante Informationen eliminiert und die jeweiligen Zugriffsmechanismen erleichtert werden. Beispiele hierzu sind Datenstrukturen wie *run-length*-Code, *quadtrees*, *chain*-Code, Laplace-Pyramiden, usw. Bei der Implementierung von Systemen findet an dieser Stelle der Übergang von der *pixelorientierten Verarbeitung* zur *listenorientierten Verarbeitung* statt.

3.1.7 Beschreibung der Segmente

Als Nächstes schließen sich in der Regel Verfahren zur Beschreibung der Segmente an. Hier können einfache Segmenteigenschaften wie der Flächeninhalt, die Länge des Umfangs oder die Lage des Schwerpunktes schon ausreichend sein. Aufwändigere Algorithmen untersuchen die Form des Segments durch die Analyse der relativen Lage der Bildpunkte des Segments. Aber auch kanten- und linienorientierte Verfahren werden hier verwendet.

3.1.8 Synthese von Objekten

Die in den vorhergehenden Verarbeitungsschritten erkannten und beschriebenen Segmente werden jetzt zu sinnvollen Objekten zusammengefügt. Neben heuristischen Verfahren werden hier Methoden der künstlichen Intelligenz, wie z.B. semantische Netze oder Expertensysteme, eingesetzt.

3.1.9 Ableitung einer Reaktion

Als letzter Schritt muss aus den analysierten Bilddaten eine Reaktion abgeleitet werden, z.B. das Auslösen eines Alarms, die Steuerung eines Roboters oder das Nachführen einer Kamera. Diese Komponente bezeichnet man als die *Reaktionskomponente* oder *Exekutive* des Systems.

3.1.10 Schlussbemerkung zu Abschnitt 3.1

Bei vielen Anwendungen müssen nicht alle oben aufgezählten Punkte durchlaufen werden. Auch ist es möglich, dass einzelne Verarbeitungsstufen trivial lösbar sind oder einfach ausgelassen werden können.

In den folgenden Abschnitten und Kapiteln, die grob nach diesem Schema gegliedert sind, steht immer die Anwendung im Vordergrund. Der vorliegende Bildverarbeitungsteil gibt somit keine Abfolge von logisch, mathematisch zusammenpassenden Algorithmen wieder. Vielmehr kann es sein, dass bei einer Anwendung die unterschiedlichsten Verfahren kombiniert werden müssen. Diese Verfahren der Bildverarbeitung sind meist auch u.a. in [Gonz18, Toen05, Jaeh11] beschrieben.

3.2 Unterabtastung und Quantisierung

Um Bilder mit Computersystemen verarbeiten zu können, müssen sie in Datenformate umgesetzt werden, die rechnerkompatibel sind. Diese Umsetzung heißt *Digitalisierung*. Für fotografische Vorlagen oder Strichzeichnungen werden hierzu *Zeilenabtaster* (*Scanner*) mit unterschiedlicher technischer Realisierung der Abtastung eingesetzt. Auch Videokameras, die über Analog/Digital-Wandler (*frame grabber*, *video capture card*) an das Computersystem angeschlossen sind, werden zur Digitalisierung verwendet. Moderne digitale Fotoapparate (Digitalkameras) oder Videokameras führen die Digitalisierung schon im Gerät durch und liefern somit bereits digital aufbereitete Daten. In manchen Bereichen, z.B. in der Fernerkundung (Luft- und Satellitenbildauswertung) wird die Digitalisierung mit flugzeug- oder satellitengetragenen *Multispektralscannern* durchgeführt. Im Folgenden wird nicht auf die technische Realisierung der einzelnen Abtastsysteme, sondern vielmehr auf einige grundlegenden Aspekte der Digitalisierung eingegangen.

Zur Einführung werden zunächst eindimensionale Funktionen $s(t)$ einer Variablen t verwendet. Wenn man eine eindimensionale Funktion $s(t)$ digital aufbereiten will, muss man sie *abtasten* und *quantisieren*. Die Abtastung erfolgt entlang der t-Achse (Abszisse).

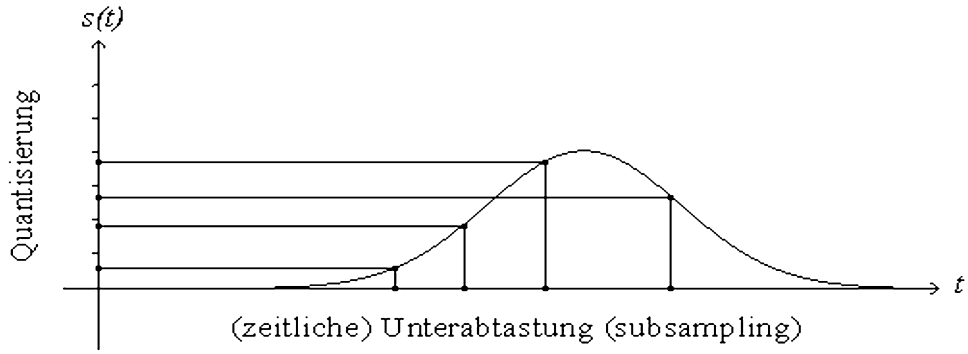


Bild 3.1: Unterabtastung und Quantisierung: Die Unterabtastung erfolgt entlang der Abszisse, die Quantisierung entlang der Ordinate

Da nicht alle Werte von $s(t)$ erfasst werden, spricht man hier auch von *Unterabtastung* (*subsampling*). Die Quantisierung wird senkrecht dazu entlang der Ordinate durchgeführt (Bild 3.1).

Zunächst einige Bemerkungen zur Abtastung. Man kann sich natürlich fragen, wie eng man die Abtastwerte auf der Abszisse legen muss, damit die originale Funktion $s(t)$ möglichst genau erfasst wird. Dazu gibt das *Abtasttheorem von Shannon* [Shan48] eine Hilfestellung. Man kann unter bestimmten nicht sehr restriktiven Bedingungen eine Funktion $s(t)$ als eine Überlagerung von endlich oder unendlich vielen Sinus- und Cosinus-Funktionen mit unterschiedlichen Frequenzen und Amplituden darstellen (endliche oder unendliche Reihe). Wenn es unendliche viele sind, muss man sich für eine maximale Frequenz entscheiden, da man ja nicht unendlich viele Abtastpunkte erfassen kann. Man spricht dann von einem bandbegrenzten Signal. Das Shannon'sche Abtasttheorem besagt nun, dass man die Abtastfrequenz, also das Setzen der Abtastpunkte, so wählen muss, dass sie mindestens doppelt so groß ist, wie die höchste im bandbegrenzten Signal auftretende Frequenz. Das ist auch anschaulich klar, weil man dann sicher sein kann, dass zwischen zwei Abtastpunkten „nicht mehr viel passiert". Dazu ein Beispiel aus der Praxis: Das menschliche Gehör kann Frequenzen bis etwa 20000 *Hz* wahrnehmen (bei zunehmendem Alter sinkt diese Grenze deutlich). Will man nun ein Audiosignal $s(t)$ auf eine Audio-CD bringen, so muss man also Frequenzen bis maximal etwa 20000 *Hz* erfassen, d.h. dass die Abtastfrequenz bei ca. 40000 *Hz* liegen muss (tatsächlich: 44000 *Hz*).

Nun zur Quantisierung. Sie ist in Bild 3.2 schematisch dargestellt. Das Prinzip: Intervalle des Eingabesignals $s_e(t)$ werden im Ausgabesignal $s_a(t)$ festen Quantisierungsstufen zugeordnet. Sind die Quantisierungsstufen gleichabständig, so spricht man von einer gleichmäßigen Quantisierung (*uniform quantiser*). Sollen Eigenheiten der menschlichen Perzeption berücksichtigt werden, so kann man auch eine nicht gleichmäßige Quantisierung

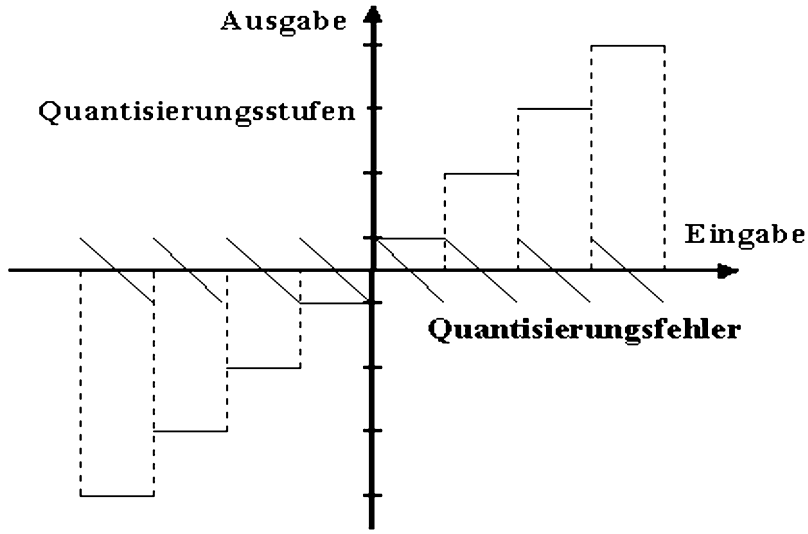


Bild 3.2: Das Prinzip Quantisierung: Intervalle des Eingabesignals werden im Ausgabesignal festen Quantisierungsstufen zugeordnet.

(*nonuniform quantiser*) verwenden. So können z.B. betragsmäßig kleine Werte des Eingabesignals feiner quantisiert werden als betragsmäßig große Werte.

Das digitalisierte Signal $s_a(t)$ ist eine diskrete Funktion der diskreten Variablen t. Da es endlich viele Punkte $(t_i, s_a(t_i)), i = 0, 1, ..., n-1$ sind, kann man sie in einer Tabelle unterbringen:

t	t_0	t_1	...	t_{n-1}
$s_a(t)$	$s_a(t_0)$	$s_a(t_1)$	...	$s_a(t_{n-1})$

Fotografische Bilder kann man als zweidimensionale Funktionen $s(x, y)$ interpretieren: An der Stelle (x, y) tritt die Schwärzung $s(x, y)$ auf. Die digitale Aufbereitung erfolgt im Prinzip wie oben. Bei der (Unter-)Abtastung, die hier oft auch *Rasterung* genannt wird, werden diskrete Positionen (x_i, y_i) festgelegt, an denen der Wert $s_e(x_i, y_i)$ abgelesen wird. Dieser Wert wird dann quantisiert. Da nach der Digitalisierung auch hier endlich viele Werte vorliegen, können sie in einer Tabelle oder besser in einer Matrix gespeichert werden:

	y_0	y_1	y_2	...
x_0	$s_a(x_0, y_0)$	$s_a(x_0, y_1)$	$s_a(x_0, y_2)$	...
x_1	$s_a(x_1, y_0)$	$s_a(x_1, y_1)$	$s_a(x_1, y_2)$	...
x_2	$s_a(x_2, y_0)$	$s_a(x_2, y_1)$	$s_a(x_2, y_2)$	...
...	...	...	...	...

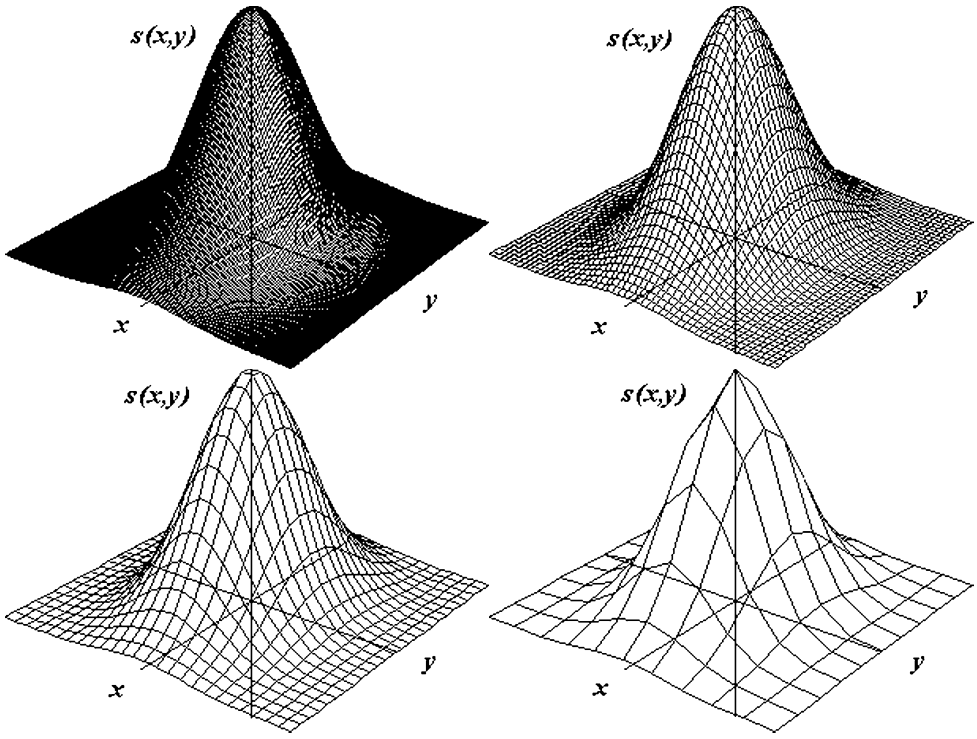


Bild 3.3: Bilder sind zweidimensionale Funktionen, die wie ein eindimensionales Signal gerastert und quantisiert werden. Hier sind vier verschieden große Unterabtastungsbeispiele einer zweidimensionalen Gauß'schen Normalverteilung dargestellt.

Bild 3.3 zeigt ein Beispiel einer zweidimensionalen Funktion (zweidimensionale Gauß'sche Normalverteilung) mit unterschiedlich großer Unterabtastung.

3.3 Digitalisierung von Schwarz/Weiß-Bilddaten

Nachdem in Abschnitt 3.2 das allgemeine Konzept bei der Digitalisierung von Funktionen kurz beispielhaft erläutert wurde, wird in diesem Abschnitt die digitale Aufbereitung einer Schwarz/Weiß-Bildvorlage untersucht. Ein Schwarz/Weiß-Bild ist nur aus zwei Grautönen (schwarz und weiß) zusammengesetzt und könnte z.B. als fotografisches Papierbild oder als Transparent vorliegen. Die Darstellung des Motivs erfolgt durch Schwärzung der fotografischen Schicht: An Stellen, die in der Helligkeit unter einer bestimmten Schwelle liegen, ist die Schicht geschwärzt, an den anderen Stellen ist sie weiß. Zur Digitalisierung sind die zwei Schritte (Unter-)Abtastung (Rasterung) und Quantisierung notwendig.

Bei der Rasterung wird die Bildvorlage durch die Überlagerung eines rechteckigen oder quadratischen Gitters in einzelne Rasterflächenstücke unterteilt. Zur Quantisierung wird jede dieser Rasterflächen entweder nur schwarz oder weiß dargestellt, je nachdem, ob der größere Teil der Rasterfläche ursprünglich weiß oder schwarz war. Wird nun ein weißes Rasterflächenstück symbolisch durch die Zahl 1 und ein schwarzes Rasterflächenstück durch die Zahl 0 dargestellt, so kann das digitalisierte Bild als eine rechteckige Zahlenanordnung (*Bildmatrix*) interpretiert werden. Bild 3.4 veranschaulicht diese grundlegenden Schritte der Digitalisierung.

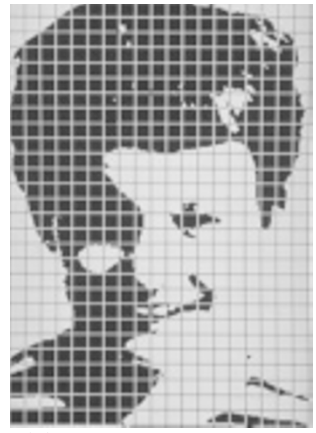
Leider ist in der Praxis die Form der Rasterflächen nicht einheitlich. Wenn z.B. ein Framegrabber oder Scanner die Vorlage mit rechteckigen, nicht quadratischen Rasterflächen verarbeitet, so wird das digitalisierte Bild, dargestellt auf einem Displaysystem mit quadratischem Raster, verzerrt erscheinen. Das ist bei vielen Operationen zwar nur ein Schönheitsfehler. Es gibt aber auch Anwendungen, z.B. bei geometrischen Operationen, wo dieser Effekt störend ist oder sogar zu falschen Ergebnissen führt.

Eine Zeile der Bildmatrix wird als *Bildzeile*, eine Spalte wird als *Bildspalte* und ein Element der Bildmatrix wird als *Bildpunkt* (*Pixel*, *picture element*) bezeichnet. Die Zählung der Bildzeilen erfolgt von oben nach unten, beginnend mit der Bildzeile 0, und die Zählung der Bildspalten erfolgt von links nach rechts, beginnend mit der Bildspalte 0 (Bild 3.5).

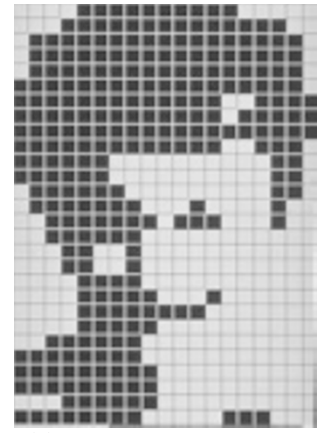
Diese Lage des Koordinatensystems wurde aus zwei Gründen gewählt: Wie bei der Bezeichnung von Matrixelementen in der Mathematik laufen die Indizes von oben nach unten und von links nach rechts. Wenn dieses Koordinatensystem um 90° gegen den Uhrzeigersinn gedreht wird, ist es mit dem in der Mathematik üblichen Koordinatensystem identisch. Das hat den Vorteil, dass Begriffe, wie z.B. die mathematisch positive oder negative Drehrichtung direkt übernommen werden können. Es wird darauf hingewiesen, dass in anderer Bildverarbeitungsliteratur oder in Grafiksystemen das Koordinatensystem anders definiert sein kann. Es wird empfohlen, im jeweiligen Anwendungsfall sich über die konkrete Definition des Koordinatensystem zu informieren.

Die einem Bildpunkt zugeordnete Zahl 0 oder 1 ist der *Grauwert* des Bildpunktes. Da ein digitalisiertes Schwarz/Weiß-Bild nur durch die zwei Grauwerte 0 und 1 dargestellt

(a)



(b)



(c)

(d)

1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1
1	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1
1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1

Bild 3.4: Digitalisierung eines Schwarz/Weiß-Bildes. (a) Original. (b) Rasterung des Originals durch die Überlagerung eines quadratischen Gitters. (c) Quantisierung durch Auffüllen der Rasterflächen mit weiß oder schwarz. (d) Darstellung der schwarzen oder weißen Rasterflächenstücke durch 0 oder 1.

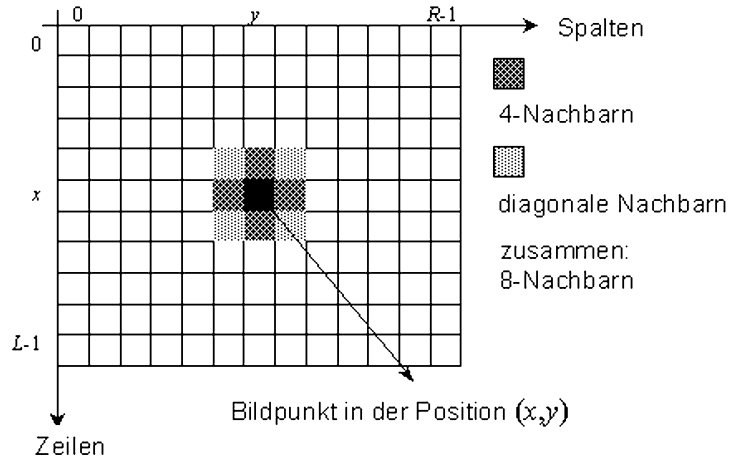


Bild 3.5: Bildmatrix eines digitalisierten Bildes mit Koordinatensystem und 4-Nachbarn und 8-Nachbarn.

wird, bezeichnet man es oft als *Binärbild*. Werden anstatt 0 und 1 beliebige andere Zahlen verwendet, so spricht man von einem *Zweipegelbild*.

Auf Displaysystemen ist es meistens notwendig, Binärbilder mit den Grauwerten 0 (schwarz) und 255 statt 1 (weiß) darzustellen, da man die Graustufen 0 und 1 mit den Augen nicht unterscheiden kann.

Im quadratischen Raster hat jeder Bildpunkt (x, y) zwei Arten von Nachbarn. Die vier Nachbarbildpunkte in den Positionen $(x, y-1)$, $(x, y+1)$, $(x-1, y)$ und $(x+1, y)$ heißen die *4-Nachbarn* des Bildpunktes in der Position (x, y). Zusätzlich hat der Bildpunkt (x, y) noch die vier *diagonalen Nachbarn* in den Positionen $(x-1, y-1)$, $(x-1, y+1)$, $(x+1, y-1)$ und $(x+1, y+1)$. Die 4-Nachbarn und die vier diagonalen Nachbarn zusammen werden als *8-Nachbarn* oder schlicht *Nachbarn* des Bildpunktes (x, y) bezeichnet (Bild 3.5).

In manchen Algorithmen ist es wichtig anzugeben, ob 4- oder 8-Nachbarschaft zugrunde gelegt wird. Bild 3.6 zeigt dazu ein Beispiel: Wenn bei einer Anwendung gezählt werden muss, wie viele Segmente (hier die schwarzen Bildpunkte) sich in einem Bild oder Bildausschnitt befinden, so ist es wichtig, welche Bildpunkte als Nachbarn zählen und somit zum selben Segment gehören. Bei 4-Nachbarschaft enthält Bild 3.6 vier Segmente, bei 8-Nachbarschaft nur zwei.

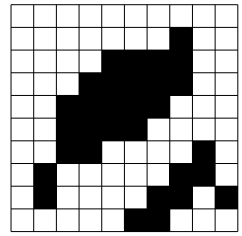


Bild 3.6: Wenn gezählt werden soll, wie viele Segmente (hier die schwarzen Bildpunkte) das Bild enthält, ist es wichtig anzugeben, ob 4- oder 8-Nachbarschaft zugrunde gelegt wird. Bei 4-Nachbarschaft enthält das Bild vier und bei 8-Nachbarschaft zwei Segmente.

Zur Berechnung des *Abstandes* zwischen zwei Bildpunkten (x_1, y_1) und (x_2, y_2) werden unterschiedliche *Distanzmaße* verwendet. Die *euklidische Distanz* berechnet sich gemäß:

$$d_e = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{3.1}$$

Für den Abstand eines Bildpunktes zu seinen 4-Nachbarn ergibt sich $d = 1$, während sich für die vier diagonalen Nachbarn $d = \sqrt{2}$ ergibt. Ein anderes Distanzmaß ist die *Schachbrettdistanz*. Hier haben alle 8-Nachbarn dieselbe Distanz $d = 1$ zum Bildpunkt in der Position (x, y):

$$d_s = \max\{|x_1 - x_2|, |y_1 - y_2|\}. \tag{3.2}$$

Abschließend zu diesem Kapitel wird ein erstes mathematisches Modell zur Beschreibung von digitalisierten Bilddaten formuliert. Eine Möglichkeit ist die Notation als (Bild-) Matrizen. Im einfachsten Fall ist das Bild $\mathbf{S}$ (die *Szene* $\mathbf{S}$) eine rechteckige (Bild-)Matrix $\mathbf{S} = (s(x, y))$ mit Bildzeilen und Bildspalten. Der Zeilenzähler (Zeilenindex) ist x und der Spaltenzähler (Spaltenindex) ist y. Der Bildpunkt an der Stelle (Zeile, Spalte)=(x, y) besitzt den Grauwert $s(x, y)$, hier 0 oder 1. Bei exakter Anlehnung an die mathematische Matrizenschreibweise (z.B. $\mathbf{A} = (a_{ij})$) müsste man den Zeilen- und Spaltenzähler tiefgestellt als Index s_{xy} schreiben. Davon wird hier abgewichen, zum einen, weil bei manchen Verfahren die vielen hoch- und tiefgestellten Indizes verwirrend sein können, und zum anderen, weil die Notation $s(x, y)$ auch ausdrücken kann, dass man den Grauwert an der Stelle (x, y) als Funktionswert $s(x, y)$ auffassen kann.

Somit wird ein digitalisiertes Schwarz/Weiß-Bild mit dem folgenden *mathematischen*

Modell beschrieben:

$G = \{0, 1\}$	Grauwertmenge mit den Grauwerten 0 und 1
$\mathbf{S} = (s(x, y))$	Bildmatrix des Bildes
$x = 0, 1, ..., L - 1$	L Bildzeilen
$y = 0, 1, ..., R - 1$	R Bildspalten
(x, y)	Ortskoordinaten des Bildpunktes
$s(x, y) \in G$	Grauwert des Bildpunktes

(3.3)

3.4 Digitalisierung von Grautonbildern

Bei einem Grautonbild erfolgt die Abbildung des Motivs durch unterschiedliche Schwärzung der Fotoschicht. Es treten somit neben schwarz und weiß auch andere Grautöne auf. In der Umgangssprache werden Grautonbilder oft fälschlich als Schwarz/Weiß-Bilder bezeichnet (z.B. Schwarz/Weiß-Fernsehapparat). Bei der Digitalisierung eines Grautonbildes muss nach der Rasterung jeder Rasterfläche ein Grauwert zugeordnet werden. Die Bestimmung dieses Grauwertes kann punktuell erfolgen (Deltafunktion) oder durch Mittelung über die Rasterfläche. Die Mittelung wird dabei in der Regel nicht gleichgewichtig über die gesamte Rasterfläche durchgeführt. Als Grauwertmenge wird meistens $G = \{0, 1, 2, ..., 255\}$ verwendet, da diese 256 Grauwerte mit einem Byte (1 Byte = 8 Bit und $2^8 = 256$) dargestellt werden können. In byteorientierten Rechenanlagen ist damit der Speicherplatz für digitalisierte Bilddaten optimal ausgenützt und die Adressierung der Bildpunkte ist einfach. Ein digitalisiertes Grautonbild ist somit wieder eine Matrix $\mathbf{S} = (s(x, y))$, nur dass jetzt die Grauwerte aus der erweiterten Grauwertmenge G sind.

Die 256 Grauwerte von 0 bis 255 sind in den meisten Fällen ausreichend. Der Grauwert 0 wird dabei in der Regel als schwarz, der Grauwert 255 als weiß und ein Grauwert um 127 als grau interpretiert. Bild 3.7 zeigt die Quantisierung eines Originals mit 64, 32, 16, und 8 Grauwerten. Die Verallgemeinerung der Grauwertmenge G auf eine allgemeine Grauwertmenge $G' = \{z_1, z_2, ..., z_k\}$ ist jederzeit möglich, im Rahmen dieser Darstellung wird jedoch darauf verzichtet.

Mit diesen Festlegungen lässt sich ein digitalisiertes Grauwertbild wie folgt beschreiben:

$G = \{0, 1, ..., 255\}$	Grauwertmenge
$\mathbf{S} = (s(x, y))$	Bildmatrix des Grauwertbildes
$x = 0, 1, ..., L - 1$	L Bildzeilen
$y = 0, 1, ..., R - 1$	R Bildspalten
(x, y)	Ortskoordinaten des Bildpunktes
$s(x, y) = g \in G$	Grauwert des Bildpunktes

(3.4)

In praktischen Problemlösungen werden im Rahmen eines Algorithmus häufig unterschiedliche Verfahren der Reihe nach angewendet: Nach einer nicht linearen Skalierung der Grauwerte könnte z.B. eine Filterung im Ortsbereich folgen, darauf eine geometrische Korrektur, dann die Berechnung von Texturmerkmalen für Umgebungen von Bildpunkten

(a)

(b)

(c)

(d)

Bild 3.7: Quantisierung eines Testbildes mit (a) 64, (b) 32, (c) 16, (d) 8 Grauwerten. Für viele Anwendungsbeispiele sind weniger als 256 Grauwerte ausreichend.

und schließlich eine Klassifizierung mit einem neuronalen Netz. Es ist nun aber nicht notwendig und oft auch nicht sinnvoll, nach jedem Verarbeitungsschritt den Wertebereich des Ergebnisbildes wieder in die Grauwertmenge $G = \{0, ..., 255\}$ abzubilden. Vielmehr wird z.B. der nichtlineare Skalierungsoperator sein Ergebnisbild als *float-* (*real-*) Werte an den nächsten Operator weitergeben. Die Abbildung in die Grauwertmenge $G = \{0, ..., 255\}$ ist nur notwendig, wenn das Zwischenergebnis auf einem Displaysystem dargestellt werden soll, das diese Grauwertmenge verwendet.

Nun noch einige Bemerkungen zur Rasterung. Diese Thematik wurde in Kapitel 3.2 schon angesprochen. Es ist offensichtlich, dass die gewählte Rasterflächengröße die Qualität des digitalisierten Bildes wesentlich beeinflusst: Wird bei einer gegebenen Vorlage die Rasterfläche zu groß gewählt, so gehen feine Details des Originals verloren. Ist dagegen die Rasterfläche zu klein, so wird das Rechensystem mit zu vielen Daten belastet (Bild 3.8).

Für die Wahl der richtigen Rasterung bei der Digitalisierung werden sowohl pragmatische als auch theoretische Hilfestellungen angeboten. Zunächst kann die richtige Rasterung durch „Augenschein“ festgelegt werden. Soll z.B. eine Strichzeichnung digitalisiert werden, deren minimale Linienbreite etwa $0.1mm$ ist, so könnte man eine Rasterung wählen, bei der man sicher ist, dass auf eine Breite von $0.1mm$ mindestens zwei Bildpunkte kommen. Man müsste also eine Rasterung wählen, bei der ein Quadratmillimeter in $20 \cdot 20 = 400$ oder mehr Bildpunkte zerlegt wird.

Die theoretische Grundlage hierzu bildet das Shannon'sche Abtasttheorem, das in Kapitel 3.2 schon erwähnt wurde. Eine ausführliche Darstellung dazu wird in [Bose04] oder [Grün02] gegeben.

In der Praxis liegen die Bilddaten bei vielen Anwendungsfällen bereits digitalisiert vor, so dass die Wahl der Rastergröße bereits vorweggenommen ist.

In Bild 3.9 wird ein anderer Effekt bei der Digitalisierung von Bilddaten gezeigt. Neben einer Digitalisierung des Testbildes mit $512 \cdot 512$ Bildpunkten, 256 Graustufen, ist eine Digitalisierung mit $32 \cdot 32$ Bildpunkten, 256 Graustufen abgebildet, bei der die Bildpunkte vergrößert wurden, um denselben Maßstab wie bei Bild 3.9-a zu erreichen. Bild 3.9-b ist in der Qualität deutlich schlechter als Bild 3.9-a. Durch die vergrößerte Wiedergabe der Bildpunkte als homogene graue Flächen werden Grauwertkanten in das Bild eingeführt, die im Original nicht vorhanden sind. Das menschliche Auge spricht auf diese Kanten sehr stark an, wodurch die eigentliche Bildstruktur verloren geht. Betrachtet man dagegen Bild 3.9-b aus der Ferne oder kneift man bei der Betrachtung die Augen etwas zusammen (oder nimmt man die Brille ab), so bemerkt man, dass doch mehr Information des Originals enthalten ist als zunächst vermutet wird. Bild 3.9-c, in dem über die Grenzen benachbarter Bildpunkte gemittelt wurde, zeigt einen ähnlichen Effekt.

Ein Sonderfall von Grauwertbildern sind *logische Bilder*. Sie haben gewöhnlich eine Bildmatrix der Form $\mathbf{S} = (s(x, y))$. Die Grauwerte g haben aber hier nicht die Bedeutung von digitalisierten Grautönen, sondern sind Codes für bestimmte „Klassen“, die im Bild enthalten sind. Ein einfaches Beispiel ist ein Binärbild einer digitalisierten Strichzeichnung: Die weißen Bildpunkte mit dem Grauwert 1 werden als Hintergrund interpretiert und die schwarzen Bildpunkte mit dem Grauwert 0 als Bestandteile der Strichzeichnung. Ein anderes Beispiel ist ein digitalisierter Stadtplan, in dem Häuser rot, Straßen gelb, Grünflächen

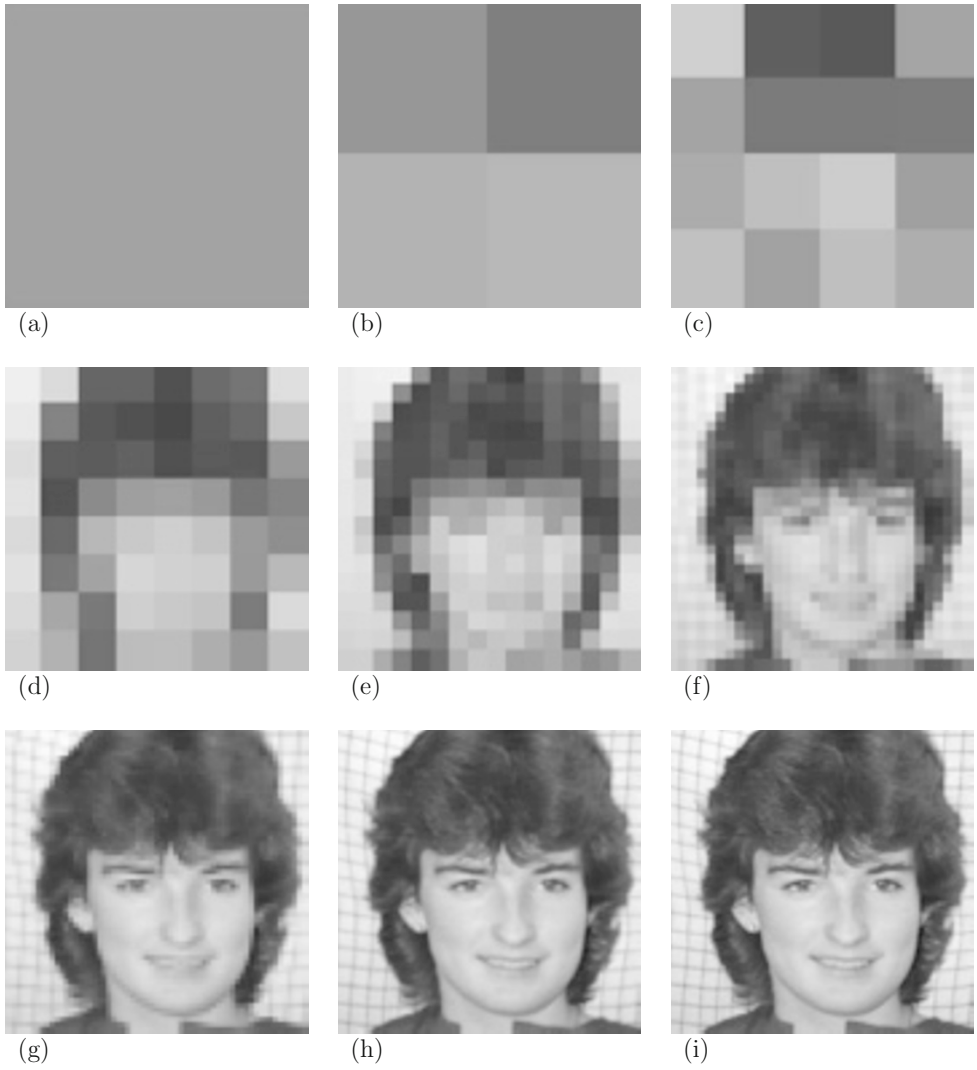


Bild 3.8: Rasterung einer Grautonvorlage mit einem quadratischen Gitter mit (a) $1 \cdot 1$, (b) $2 \cdot 2$, (c) $4 \cdot 4$, (d) $8 \cdot 8$, (e) $16 \cdot 16$, (f) $32 \cdot 32$, (g) $64 \cdot 64$, (h) $128 \cdot 128$ und (i) $256 \cdot 256$ Gitterpunkten. In der Darstellung sind alle Bildbeispiele durch Vergrößerung der Rasterflächen auf denselben Maßstab gebracht. Die neun Teilbilder sind auch ein Beispiel für die Darstellung eines Bildes als Baumstruktur (*quad tree*, Kapitel 19).

(a)

(b)

(c)

Bild 3.9: Quantisierung und Bildqualität. Die Qualität der Digitalisierung hängt nicht nur von der Wahl der Rasterflächengröße und der Quantisierung, sondern auch von der Art der Wiedergabe ab. (b) Wurde mit $32 \cdot 32$ Bildpunkten und 256 Graustufen digitalisiert. Die deutliche Verschlechterung gegenüber (a), $512 \cdot 512$ Bildpunkte, 256 Graustufen, ergibt sich auch aus der vergrößerten Darstellung der Bildpunkte als homogene graue Flächen. Bei (c) wurde über die Ränder der Bildpunkte hinweg gemittelt, und man erkennt, dass in (b) mehr Information des Originals vorhanden ist als zunächst vermutet.

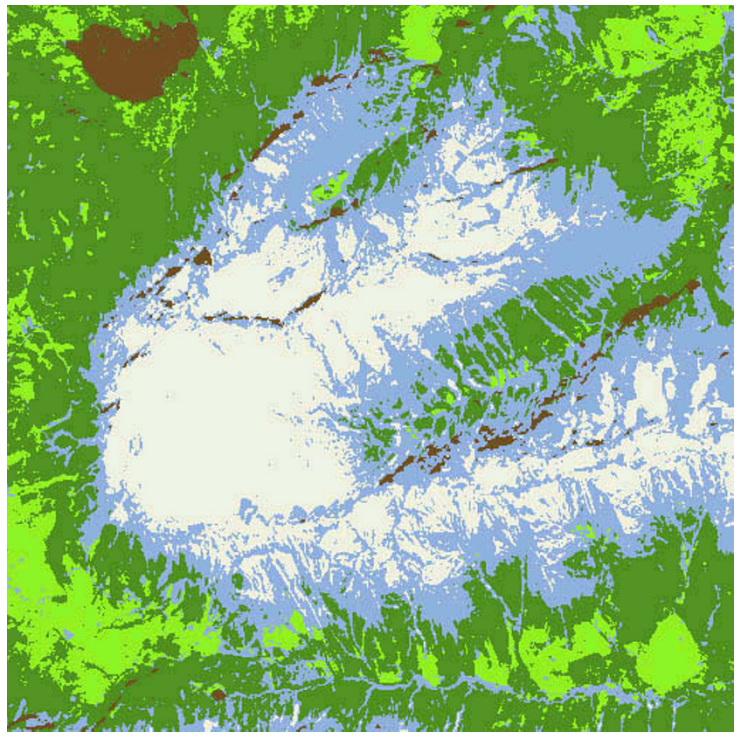


Bild 3.10: Beispiel eines klassifizierten Satellitenbildes. Als Original dienten SPOT-Bilddaten. Es zeigt das Wettersteingebirge mit der Zugspitze. Das Bild wurde mit einem neuronalen Netz klassifiziert. Die Farben bedeuten: weiß - Eis/Schnee, grau - Gestein, dunkelbraun - Gewässer, dunkelgrün - Wald und hellgrün - Grünland.

grün, Eisenbahnlinien schwarz, usw. dargestellt sind. Jeder „Klasse“ wird im digitalisierten Bild ein Grauwert als *Klassencode* zugeordnet. Bei der bildlichen Reproduktion kann dann der Klassencode wieder mit dem zugehörigen Farbton ausgegeben werden. Logische Bilder sind oft Zwischenprodukte auf dem Weg vom Original zu einem interpretierten Bild, z.B. das Ergebnis einer Bildsegmentierung (Bild 3.10 und die Kapitel 11 und 20).

3.5 Farbbilder

In diesem wird Kapitel die Verarbeitung von Farbbildern erläutert. Bevor das mathematische Modell für Farbbilder formuliert wird, werden unterschiedliche Aspekte des Phänomens „Farbe“ betrachtet:

- Die physikalischen Aspekte,
- die physiologischen Aspekte,
- die normativen, technischen Aspekte und
- die darauf aufbauenden Farbmodelle.

3.5.1 Farbe: Physikalische Aspekte

Physikalisch betrachtet wird Farbe durch Licht erzeugt. Licht ist eine elektromagnetische Welle und kann somit durch die Frequenz f und die Wellenlänge λ beschrieben werden. Über die Lichtgeschwindigkeit c sind die Frequenz und die Wellenlänge miteinander verbunden:

$$c = f \cdot \lambda. \tag{3.5}$$

Für den Menschen ist nur ein sehr schmaler Ausschnitt aus dem elektromagnetischen Spektrum sichtbar: Er liegt bei Licht der Wellenlänge 400 nm (Nanometer) bis 700 nm. Trifft Licht mit einer bestimmten Wellenlänge (*monochromatisches Licht*) auf das Auge, so wird eine Farbempfindung hervorgerufen, die durch die Farben des Regenbogenspektrums, von Violett (400 nm) bis Rot (700 nm), beschrieben werden kann.

3.5.2 Farbe: Physiologische Aspekte

Damit wird aber bereits der zweite, der physiologische Aspekt, diskutiert. Ausschlaggebend für das Farbempfinden sind *Fotopigmente* im Auge, die auf unterschiedliche Ausschnitte des sichtbaren Spektrums reagieren. Sie sprechen vor allem auf Rot, Grün und Blau an, reagieren aber auch auf andere Wellenlängen (Bild 3.11). Diese Grafik zeigt, dass das Auge am intensivsten auf Grün und weniger stark auf Rot und Blau reagiert.

Aus diesem Grund entspricht die Lichtstärke (die wahrgenommene Helligkeit) nicht direkt seiner Energie: Um die Helligkeit von grünem Licht zu erhöhen, wird weniger Energie benötigt als für die Erhöhung der Helligkeit von rotem oder blauem Licht. Bild 3.12 zeigt die relative Empfindlichkeit des Auges für die Wellenlängen des sichtbaren Spektrums.

Monochromatisches Licht erzeugt die Empfindung „reiner“ Farben. Dasselbe oder ähnliches Farbempfinden ist aber auch durch die Mischung verschiedener Wellenlängen zu erzeugen. Man kann jedoch monochromatisches Licht so mischen, dass Farbempfindungen hervorgerufen werden, die durch monochromatisches Licht nicht erzeugt werden können. Eine solche Mischfarbe kann in weißes Licht und eine reine Farbe oder ihr Komplement aufgeteilt werden. Die Empfindung „weißes Licht“ entsteht, wenn alle „Wellenlängen in derselben Menge“ vorkommen. Die reine Farbe, z.B. Rot, bestimmt die *Färbung* (*hue*), der Anteil an weißem Licht bestimmt die *Tönung*, z.B. Rosa. Mit zunehmendem Anteil an weißem Licht sinkt die *Sättigung* (*saturation*) der reinen Farbe. Die Gesamtmenge des Lichtes bestimmt die *Intensität* (*Luminanz*, *intensity*).

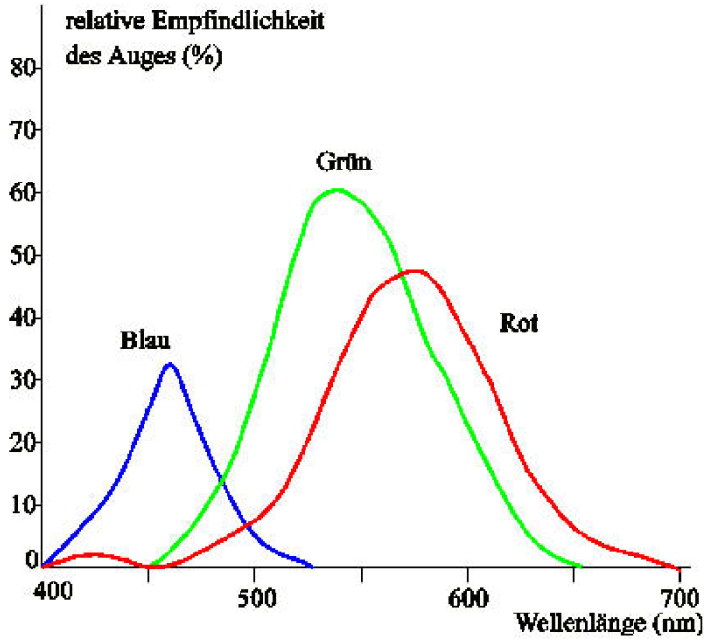


Bild 3.11: Relative Reaktion der Fotopigmente des menschlichen Auges.

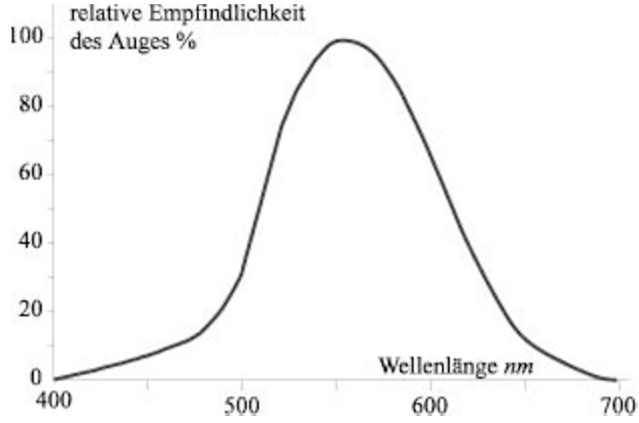


Bild 3.12: Relative Empfindlichkeit des Auges für die Wellenlängen des sichtbaren Spektrums.

3.5.3 Das CIE-Farbdreieck

Nun zum normativen, technischen Aspekt. Auf der Basis der physiologischen Gegebenheiten, dass die Farbreaktion im Auge durch die Mischung der Signale der rot-, grün- und blauempfindlichen Fotopigmente hervorgerufen wird, geht man davon aus, dass sich Farben durch drei Grundfarben zusammensetzen lassen. Allerdings gibt es keine drei Grundfarben, aus denen sich alle Farben erzeugen lassen. Um hier eine Normierung und Vergleichbarkeit zu erreichen, hat die Commission Internationale de l'Eclairage (CIE) 1931 drei künstliche Grundfarben, die *Normfarbwerte* oder *Primärfarben* eingeführt. Sie werden mit X, Y und Z bezeichnet und sind durch Energieverteilungskurven charakterisiert. So wurde z.B. für Y die Lichtstärkenreaktion des menschlichen Auges (Bild 3.12) verwendet. Außerdem wurde von der CIE die *Normfarbtafel* entwickelt (Bild 3.13), die so aufgebaut ist, dass

- jeder Punkt des Diagramms eine Farbe repräsentiert,
- alle Farben auf der Strecke zwischen zwei Farbpunkten durch Mischen der Farben der Endpunkte hergestellt werden können und
- alle Punkte innerhalb eines Dreiecks durch Mischen der Farben der Eckpunkte zu erhalten sind.

Entlang des zungenförmigen Randes liegen die Farben des monochromatischen Lichtes, und im unteren Bereich, nahe der x-Achse, die *Purpurfarben*, die nicht monochromatisch dargestellt werden können. In der Mitte ist ein Bereich, in dem die Farben für das menschliche Auge weiß erscheinen.

Wenn X, Y und Z als Anteile der Primärfarben einer beliebigen Farbe gemessen werden, so kann die Position dieser Farbe im CIE-Farbdreieck leicht ermittelt werden:

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z}, \qquad z = \frac{Z}{X+Y+Z}. \tag{3.6}$$

Im CIE-Diagramm werden die (x, y)-Werte gezeigt, auf den z-Wert wird verzichtet, da er wegen der Beziehung

$$z = 1 - x - y \tag{3.7}$$

leicht berechnet werden kann und daher ein zweidimensionales Diagramm ausreicht.

Wenn nun eine Farbe durch die (x, y)-Koordinaten im CIE-Dreieck ausgewählt wird, so muss, um die Anteile an den Primärfarben berechnen zu können, noch die Helligkeit (Intensität , Luminanz) vorgegeben werden. Dies ist aber gerade der Y-Wert. Die X- und Z-Werte berechnen sich dann gemäß:

$$X = x \cdot \frac{Y}{y} \quad \text{und} \quad Z = z \cdot \frac{Y}{y}. \tag{3.8}$$

Wenn man im CIE-Farbdreieck drei Punkte als Ecken eines Dreiecks festlegt, so sind alle Farben innerhalb des Dreiecks durch Mischen der Eckfarben zu erhalten. Man sagt,

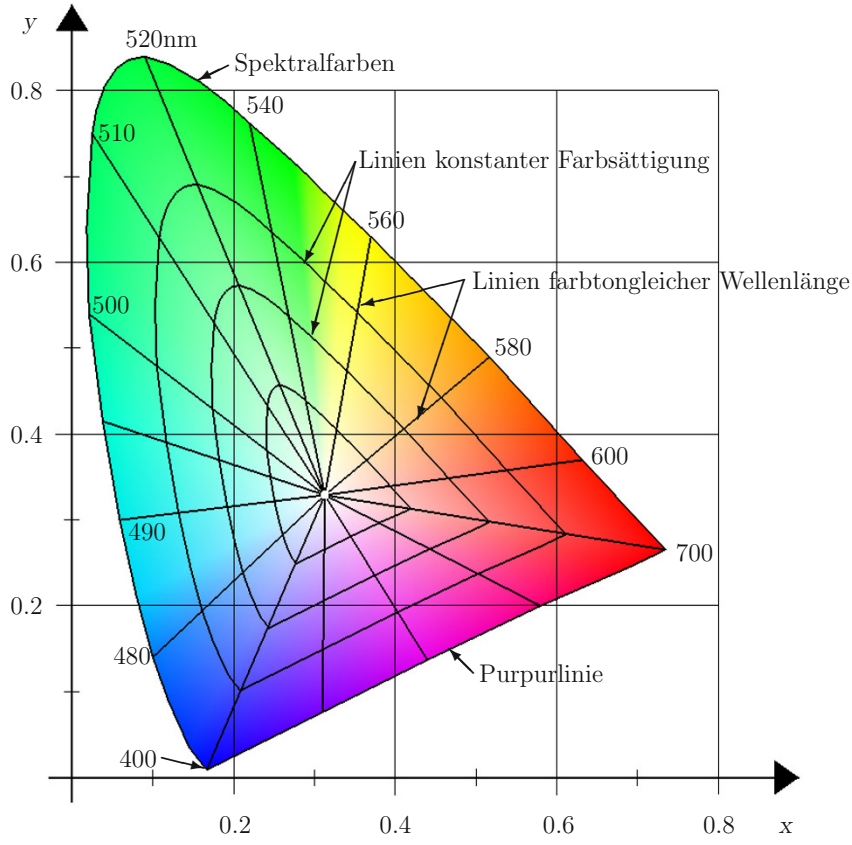


Bild 3.13: Normfarbtafel der Commission Internationale de l'Eclairage (CIE), 1931.

dass durch die drei Eckpunkte eine *Farbskala* definiert wurde. Je nach Festlegung der drei Eckpunkte erhält man eine andere Farbskala. Die CIE hat für Rot, Grün und Blau einen Standard eingeführt, den die folgende Tabelle enthält:

CIE-Farbe	Wellenlänge (nm)	x	y	z
Spektral-Rot	700.0	0.73467	0.26533	0.0
Spektral-Grün	546.1	0.27367	0.71741	0.00892
Spektral-Blau	435.8	0.16658	0.00886	0.82456

Auf Farbmonitoren werden die Farben durch rot, grün und blau leuchtende Phosphore erzeugt, denen Punkte im CIE-Dreieck entsprechen. Für moderne Farbmonitore werden folgende Werte für die Grundfarben angegeben:

Grundfarbe	x	y	z
Monitor-Rot	0.628	0.346	0.026
Monitor-Grün	0.268	0.588	0.144
Monitor-Blau	0.150	0.070	0.780

In den folgenden Abschnitten werden einige Farbmodelle vorgestellt, die auf diesen Grundlagen aufbauen.

3.5.4 Das RGB-Farbmodell

Wenn die drei Punkte für Rot, Grün und Blau im CIE-Farbdreieck festgelegt sind, kann man alle Farben in dem durch die Eckpunkte festgelegten Dreieck mischen. Eine bestimmte Farbe ist somit eine Linearkombination der drei Grundfarben. Man stellt deshalb den RGB-Farbraum als dreidimensionales, kartesisches Koordinatensystem dar (Bild 3.14). Die Größen werden so normiert, dass die $R-$, $G-$ und $B-$Komponenten zwischen 0 und 1 liegen.

Ein Problem ist noch zu lösen: Durch die Wahl der drei Grundfarben im CIE-Farbdreieck wird eine bestimmte Farbskala definiert. Es soll aber die Eigenschaft erhalten bleiben, dass die Mischung der drei Grundfarben mit maximaler Intensität die Farbe Weiß ergibt. Dazu muss man im CIE-Dreieck einen Punkt als Weiß definieren. Häufig wird dazu das Weiß eines auf 6500 *Grad Kelvin* erhitzten schwarzen Körpers verwendet:

$$x = 0.313, \quad y = 0.329 \quad \text{und} \quad z = 0.358. \tag{3.9}$$

Bei einer Luminanz von $Y_w = 1$ ergeben sich die Anteile der CIE-Normfarben an diesem Weiß mit Formel (3.8):

$$X_w = 0.951, \quad Y_w = 1.000 \quad \text{und} \quad Z_w = 1.088. \tag{3.10}$$

Den Grundfarben (z.B. eines Monitors) entsprechen die CIE-Normfarbanteile (X_r, Y_r, Z_r), (X_g, Y_g, Z_g) und (X_b, Y_b, Z_b). Diese zu gleichen Anteilen gemischt sollen den Weißpunkt ergeben:

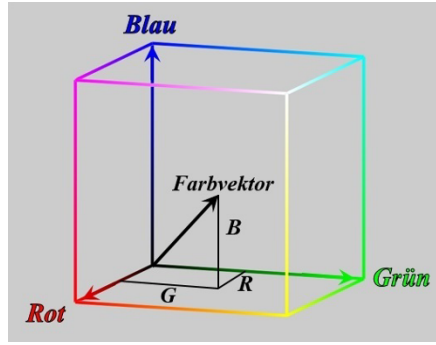


Bild 3.14: Der RGB-Farbraum. Jeder Punkt in diesem kartesischen Koordinatensystem entspricht einer Farbe mit einer bestimmten Helligkeit.

$$\begin{aligned} X_r + X_g + X_b &= X_w, \\ Y_r + Y_g + Y_b &= Y_w, \\ Z_r + Z_g + Z_b &= Z_w. \end{aligned} \tag{3.11}$$

Mit (3.8) erhält man:

$$\begin{aligned} x_r \cdot \frac{Y_r}{y_r} + x_g \cdot \frac{Y_g}{y_g} + x_b \cdot \frac{Y_b}{y_b} &= X_w, \\ y_r \cdot \frac{Y_r}{y_r} + y_g \cdot \frac{Y_g}{y_g} + y_b \cdot \frac{Y_b}{y_b} &= Y_w, \\ z_r \cdot \frac{Y_r}{y_r} + z_g \cdot \frac{Y_g}{y_g} + z_b \cdot \frac{Y_b}{y_b} &= Z_w; \end{aligned} \tag{3.12}$$

Das bedeutet, dass man die Intensitäten Y_r, Y_g und Y_b der drei (Monitor-)Grundfarben so einstellen muss, dass sich das festgelegte Normweiß ergibt. Fasst man die Quotienten zu den Linearfaktoren r, g und b zusammen und setzt man die (x, y, z)-Koordinaten der (Monitor-) Grundfarben ein, so erhält man folgendes lineares Gleichungssystem:

$$\begin{aligned} 0.628r + 0.268g + 0.150b &= 0.951, \\ 0.346r + 0.588g + 0.070b &= 1.000, \\ 0.026r + 0.144g + 0.780b &= 1.088. \end{aligned} \tag{3.13}$$

Als Lösung ergibt sich für r, g und b:

$$r = 0.761, \quad g = 1.114 \quad \text{und} \quad b = 1.164. \tag{3.14}$$

Wenn eine Farbe im RGB-Farbraum durch die Koordinaten (R, G, B) gegeben ist, so kann man z.B. den Anteil X des Normrot an dieser Farbe wie folgt berechnen:

$$\begin{aligned} X &= X_r R + X_g G + X_b B = \\ &= x_r r R + x_g g G + x_b b B = \\ &= 0.628 \cdot 0.761 \cdot R + 0.268 \cdot 1.114 \cdot G + 0.150 \cdot 1.164 \cdot B = \\ &= 0.478 \cdot R + 0.299 \cdot G + 0.175 \cdot B. \end{aligned} \tag{3.15}$$

Analog kann man den Y- und den Z-Anteil berechnen. In Matrizenschreibweise erhält man dann die beiden Gleichungen für die Umrechnung $(R, G, B) \to (X, Y, Z)$ und invers dazu $(X, Y, Z) \to (R, G, B)$:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.478 & 0.299 & 0.175 \\ 0.263 & 0.655 & 0.081 \\ 0.020 & 0.160 & 0.908 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{3.16}$$

und

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 2.741 & -1.147 & -0.426 \\ -1.118 & 2.028 & 0.034 \\ 0.137 & -0.332 & 1.105 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}. \tag{3.17}$$

An dieser Stelle sei darauf hingewiesen, dass diese Umrechnungsformeln nur gelten, wenn die Farbskala der Monitorgrundfarben gemäß obiger Tabelle vorausgesetzt werden und wenn der angegebene Weißpunkt verwendet wird. Sollte man diese Umrechnung für ein anderes Farbdreieck benötigen, so lässt sie sich anhand dieses Beispiels leicht nachvollziehen.

Das RGB-Farbssystem bezeichnet man als *additives Farbmodell*, da sich z.B. bei einem Farbmonitor die Farbempfindung im Auge durch die Überlagerung des von den einzelnen Phosphorpunkten ausgestrahlten Lichts ergibt.

3.5.5 Das CMY-Farbmodell

Im Druckereiwesen wird ein *subtraktives Farbmodell* verwendet. Es verwendet die Grundfarben Zyan (*cyan*), Magenta (*magenta*) und Gelb (*yellow*) und wird deshalb in der Literatur als *CMY-Farbmodell* bezeichnet. Der Farbeindruck im Auge entsteht dadurch, dass die Grundfarben mit unterschiedlicher Intensität auf das Papier aufgebracht werden und diese aus weißem Licht Farben absorbieren. Die gesehene Farbe ist somit die Überlagerung des nicht absorbierten, also reflektierten Lichts. Da Zyan Rot absorbiert, Magenta Grün und Gelb Blau, sagt man auch Rot, Grün und Blau sind die *komplementären Farben* von Zyan, Magenta und Gelb. Das CMY-Farbmodell definiert wie das RGB-Farbmodell einen dreidimensionalen kartesischen Raum. Zusammen mit dem RGB-Raum kann man sich einen Farbwürfel vorstellen, dessen gegenüberliegende Ecken die Komplementärfarben sind (Bild 3.15).

Beim RGB-System liegt im Koordinatenursprung die Farbe Schwarz, beim CMY-System die Farbe Weiß. Die Umrechnung zwischen RGB- und CMY-System ist einfach:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} C \\ M \\ Y \end{pmatrix}, \tag{3.18}$$

und

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{3.19}$$

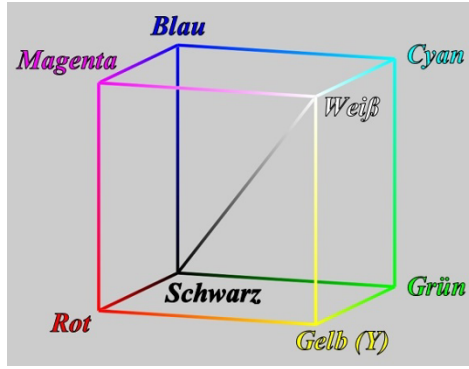


Bild 3.15: Der RGB-CMY-Farbwürfel.

Beim *Vierfarbendruck* wird zusätzlich noch als vierte Grundfarbe Schwarz verwendet, um satte Schwarztöne nicht durch Überdrucken von Zyan, Magenta und Gelb erzeugen zu müssen. Dadurch entsteht das sogenannte *CMYK-Farbmodell* (die vierte Komponente, nämlich Schwarz, wird im Offset-Druck als *Key* bezeichnet).

3.5.6 Das YIQ-Farbmodell

Das *YIQ-Farbmodell* wird beim Farbfernsehen verwendet. Das Problem ist hier die Farbinformation so zu übertragen, dass auch Schwarz-/Weiß-Fernsehgeräte vernünftige Bilder erzeugen können. Dazu verwendet man als Y dasselbe Y wie bei den CIE-Normfarben, das ja die Intensität des Lichts wiedergibt. Schwarz-/Weiß-Fernsehgeräte verwenden nur dieses Y. Die Komponenten I und Q bestimmen die Färbung und die Sättigung. Die Transformation vom RGB- in den YIQ-Farbraum lautet:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.20}$$

Das YIQ-Modell wird in dem in den USA eingeführten *NTSC-System* verwendet. Das europäische *PAL-System* verwendet die Komponenten Y, $R-Y$ und $B-Y$.

3.5.7 Die Farbmodelle HSI, HSV, HSB und HSL

Alle HS∗-Farbmodelle sind einander ähnlich und gehen von einer für Menschen intuitiven Beschreibung von Farben aus. Menschliche Begriffe für Farben, wie z.B. Rot, Orange, Gelb usw. geben einen Farbton (engl. hue) wieder und zwar relativ unabhängig von der Helligkeit. Abstufungen, wie z.B. Hellrot oder Dunkelrot geben an, ob Weiß oder Schwarz zum reinen Farbton Rot gemischt wurde und bestimmen somit die Reinheit bzw. Sättigung der Farbe. Bei Sättigung 0 sieht man keine Farbe, somit je nach Intensität Schwarz, Weiß oder Grautöne.

Die Verwendung des HSI- bzw. der anderen HS∗-Farbmodelle kann z.B. bei Segmentierungsoperationen mit dem Merkmal „Farbe“ Vorteile bringen, da im RGB-Modell für eine bestimmte Farbe immer alle drei Komponenten benötigt werden, während im HSI-Modell der Farbton nur durch die H-Komponente bestimmt wird. Der Farbton ist in diesem Modell also unabhängig von der Intensität. So kann z.B. eine Banane aufgrund ihres gelben Farbtons detektiert werden, wobei es dabei keine Rolle spielt, ob Teile der Banane im Schatten liegen und andere Teile voll beleuchtet werden.

Das HSI-Farbmodell

Als erstes wird das *HSI-Farbmodell* dargestellt, weil es etwas leichter nachvollziehbar aus dem RGB-Farbmodell hergeleitet werden kann, als die anderen. Das *HSI-Farbmodell* wird in der Farbfotografie und in der Videotechnik verwendet. Hier steht H für *hue* (*Farbton*, *Färbung*, *Tönung*), S für *saturation* (*Sättigung*, *Chroma*) und I für *intensity* (*Intensität*,

Helligkeit). H und S haben bei allen HS$*$-Farbmodellen die gleiche Bedeutung, auch wenn sie sich vom Zahlenwert her an einigen Stellen leicht unterscheiden. Bei der dritten Komponente (I für Intensität, V für Value, B für Brightness und L für Lightness) unterscheiden sich die Farbmodelle stärker, obwohl es sich auch hier letztendlich immer um eine Beschreibung der vorhandenen Gesamtlichtmenge handelt.

Anstatt eines kartesischen Koordinatensystems verwendet man beim HSI-Farbmodell eine Darstellung mit Zylinderkoordinaten. Die Höhe des Zylinders stellt die I-Achse (Intensität) dar, der Winkel zwischen der Rot-Achse und dem Farbton ist der $H(hue)$-Wert und der Abstand zur zentralen Zylinderachse entspricht der Sättigung. Die Umrechnung von (R, G, B) nach (H, S, I) erfolgt in zwei Schritten: Zuerst wird das RGB-Koordinatensystem so gedreht, dass eine Achse mit der Raumdiagonalen des Farbwürfels zusammenfällt. Diese Achse wird mit I_1 bezeichnet und stellt die zentrale Zylinderachse dar. Die beiden anderen Achsen werden mit M_1 und M_2 bezeichnet. Da die drei Achsen ein rechtwinkliges Koordinatensystem aufspannen sollen, müssen die Achsen M_1 und M_2 senkrecht auf I_1 stehen. Folglich liegen M_1 und M_2 parallel zur Ebene, die durch die drei Punkte R, G, B aufgespannt wird, weil I_1 der Normalenvektor dieser Ebene ist. Die M_1-Achse ergibt sich aus der Projektion der Rot-Achse auf die RGB-Ebene, oder anders ausgedrückt, aus einem Blickwinkel von oben auf die RGB-Ebene liegt die M_1-Achse deckungsgleich über der Rot-Achse (Bild 3.16-b). Deshalb ist der $H(hue)$-Wert sowohl der Winkel zwischen der Rot-Achse und dem Farbton, als auch der Winkel zwischen der M_1-Achse und dem Farbton.

Zur Veranschaulichung der Drehung des Koordinatensystems kann man sich vorstellen, dass man den RGB-Farbwürfel auf die schwarze Ecke stellt und sich die weiße Ecke senkrecht darüber befindet (Bild 3.16-a). Die Raumdiagonale steht also senkrecht auf dem Boden. Man betrachte nun ein Dreieck mit den Eckpunkten Schwarz, Weiß und einer dritten Farbe, z.B. Rot. Alle Farben innerhalb dieses Dreiecks müssen dann einen roten Farbton haben, da sie eine Linearkombination von Rot mit Schwarz und Weiß sind. Um einen anderen Farbton zu erhalten, dreht man die Ebene des Dreiecks um die senkrecht stehende Intensitätsachse. Der Farbton ist definiert als Drehwinkel einer solchen Rotation zwischen 0 und 360 Grad. Diese Drehung des Koordinatensystems wird mit folgender Gleichung beschrieben:

$$\begin{pmatrix} m_1 \\ m_2 \\ i_1 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{3.21}$$

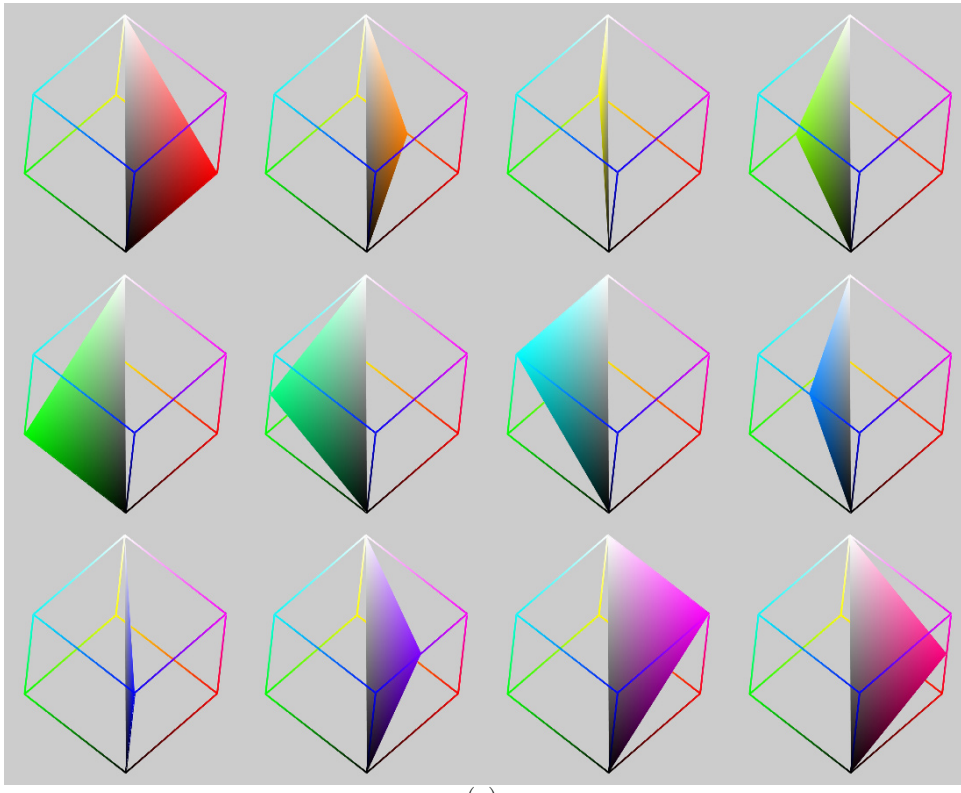


Bild 3.16: Zusammenhänge zwischen dem RGB- und dem HSI-Farbmodell. (a) Der RGB-Farbwürfel wird auf die schwarze Ecke gestellt, so dass sich die weiße Ecke senkrecht darüber befindet. Die Dreiecke mit den Eckpunkten Schwarz, Weiß und einer dritten Farbe besitzen einen einheitlichen Farbton. (b) Drehung des RGB-Koordinatensystems, so dass die I_1-Achse mit der Raumdiagonalen des Farbwürfels zusammenfällt. (c) Umrechnung der kartesischen (m_1, m_2, i_1)-Koordinaten in Zylinderkoordinaten H, S, I. (d) Blick von oben auf das HSI-Farbhexagon. Der Winkel zwischen der Rot-Achse bzw. der M_1-Achse und dem Farbton ist der $H(hue)$-Wert und der Abstand zur zentralen Zylinderachse entspricht der Sättigung S.

Im zweiten Schritt werden die kartesischen (m_1, m_2, i_1)-Koordinaten in Zylinderkoordinaten umgerechnet[1]:

$$H = \arctan\left(\frac{m_2}{m_1}\right) = \arctan\left(\frac{\frac{\sqrt{3}}{2}(G-B)}{R-\frac{G}{2}-\frac{B}{2}}\right),$$

$$\begin{aligned} S &= \sqrt{m_1^2+m_2^2} = \sqrt{(R-\frac{G}{2}-\frac{B}{2})^2+\frac{3}{4}(G-B)^2} \\ &= \sqrt{R^2+G^2+B^2-RG-RB-GB}, \end{aligned} \tag{3.22}$$

$$I = i_1 = \frac{1}{3}(R+G+B).$$

Die Bilder 3.16-b,-c,-d zeigen die Zusammenhänge zwischen dem RGB- und dem HSI-Farbmodell: Einem Farbvektor (R, G, B) im RGB-Koordinatensystem entspricht ein Vektor mit den Komponenten (m_1, m_2, i_1) im gedrehten Koordinatensystem. Die H-Komponente (also der Farbton) ist der Winkel des auf die M_1,M_2-Ebene projizierten Farbvektors mit der M_1-Achse und die S-Komponente (die Sättigung) der Betrag dieses Vektors.

Die Umrechnung des HSI-Systems in das RGB-System erfolgt sinngemäß umgekehrt:

$$\begin{aligned} m_1 &= S\cos H, \\ m_2 &= S\sin H, \\ i_1 &= I \end{aligned} \tag{3.23}$$

und

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{3}} & 0 & 1 \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 1 \\ -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ i_1 \end{pmatrix}. \tag{3.24}$$

[1]Bemerkung 1: bei der Funktion arctan sind einige Fallunterscheidungen zu beachten:
falls $m_1 = m_2 = 0$ handelt sich um einen Grauwert ($R = G = B$) und es gilt: $S = 0$ und $H =$ undefiniert,
falls $m_1 \geq 0 \wedge m_2 \geq 0$ liegt H im Intervall $[0°, 90°]$ und es gilt: $H = \arctan\left(\frac{m_2}{m_1}\right)$,
falls $m_1 < 0$ liegt H im Intervall $]90°, 270°[$ und es gilt: $H = 180° + \arctan\left(\frac{m_2}{m_1}\right)$,
falls $m_1 \geq 0 \wedge m_2 < 0$ liegt H im Intervall $[270°, 360°[$ und es gilt: $H = 360° + \arctan\left(\frac{m_2}{m_1}\right)$,
Bemerkung 2: anstatt des arctan kann man auch den arccos benutzen, da gilt:
$H = \arctan\left(\frac{m_2}{m_1}\right) = \arccos\left(\frac{m_1}{\sqrt{m_1^2+m_2^2}}\right)$

Diese aus mathematischer Sicht konsequente Definition (3.21) bzw. (3.22) der Farbraumtransformation von (R, G, B) nach (H, S, I), bei der ein Farbvektor $(m_1, m_2)^T$ in Betrag S und Richtung H umgerechnet wird, hat einen wesentlichen Nachteil: nur die Grundfarben (R, G, B) und deren Komplementärfarben (C, M, Y) führen zu einem Sättigungswert $S = 1$, alle anderen Farbtöne dazwischen führen zu Sättigungswerten $S < 1$, obwohl sie eigentlich vollkommen reine Farbtöne darstellen. Ein reines Orange z.B. mit den (R, G, B)-Farbwerten $(1, \frac{1}{2}, 0)$ ergibt nach (3.22) eine Sättigung $S = \frac{\sqrt{3}}{2} \approx 0.866$. Der Grund dafür ist aus Bild 3.16-d leicht ersichtlich: der Kreis mit Radius $S = 1$ schneidet das HSI-Farbhexagon nur an den 6 Ecken (R, G, B) und (C, M, Y), bei allen anderen Farbtönen dazwischen ist der Radius $S < 1$, da man sich entlang einer Sekante des Kreises bewegt. Damit alle reinen Farbtöne eine Sättigung $S = 1$ erhalten und somit aus dem HSI-Farbhexagon ein HSI-Farbkreis wird, definiert man die Sättigung im *korrekten HSI-Farbmodell*, abweichend von (3.22), folgendermaßen:

$$S = 1 - \frac{3}{R + G + B} min(R, G, B) \tag{3.25}$$

Durch die neue Definition der Sättigung (3.25) ändert sich auch die Rückrechnung vom HSI- in das RGB-System folgendermaßen:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{cases} \begin{pmatrix} I + I \cdot S(\frac{2}{1+\sqrt{3} \cdot tan(H)}) \\ I + I \cdot S(1 - \frac{2}{1+\sqrt{3} \cdot tan(H)}) \\ I - I \cdot S \end{pmatrix} & \text{falls } 0° \leq H < 120°, \\ \begin{pmatrix} I - I \cdot S \\ I + I \cdot S(1 - \frac{2}{\frac{1}{2} - \frac{\sqrt{3}}{2} tan(H)}) \\ I + I \cdot S(1 - \frac{2}{\frac{1}{2} + \frac{\sqrt{3}}{2} tan(H)}) \end{pmatrix} & \text{falls } 120° \leq H < 240°, \\ \begin{pmatrix} I + I \cdot S(\frac{2}{1-\sqrt{3} \cdot tan(H)}) \\ I - I \cdot S \\ I + I \cdot S(1 - \frac{2}{1-\sqrt{3} \cdot tan(H)}) \end{pmatrix} & \text{falls } 240° \leq H < 360°. \end{cases} \tag{3.26}$$

Dennoch hat das HSI-Farbmodell bestehend aus den Umrechnungsformeln (3.22) und (3.25) einen gravierenden Nachteil: nicht alle Farben aus dem Definitionsbereich $H \in [0°, 360°[$, $S \in [0, 1]$ und $I \in [0, 1]$ lassen sich in den RGB-Farbraum umrechnen, da sie sonst den Definitionsbereich $R, G, B \in [0, 1]$ überschreiten würden (z.B. HSI=(0,1,1) würde zu RGB=(3,0,0) führen). Ursache für dieses Problem ist die Definition der Sättigung als Abstand von der zentralen Grauwertachse. Bei hohen Intensitätswerten müsste die Sättigung eigentlich immer weiter abnehmen, damit man den RGB-Farbwürfel nicht verlässt. Im Extremfall bei $I = 1$ müsste $S = 0$ sein (Weiß). Leider wurde bisher keine entsprechende Zusatzbedingung definiert, die bestimmte HSI-Wertekombinationen ausschließt. Dies soll

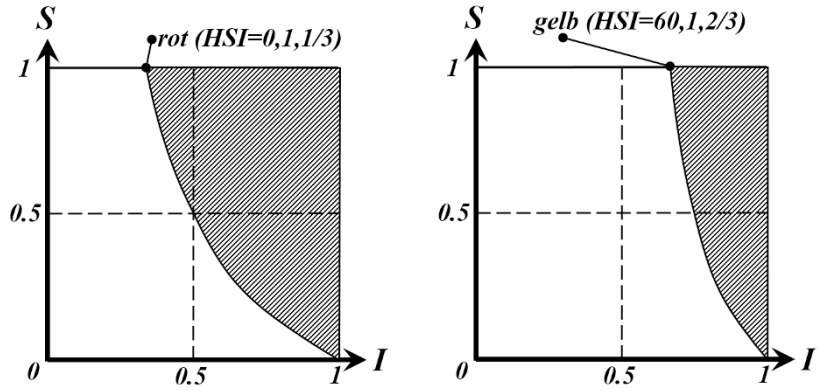


Bild 3.17: Unzulässige Bereiche des HSI-Farbraums sind schraffiert dargestellt für die beiden Farbtöne (a) Rot (RGB=(1,0,0) bzw. HSI=$(0,1,\frac{1}{3})$) und (b) Gelb (RGB=(1,1,0) bzw. HSI=$(60,1,\frac{2}{3})$). Die schraffierten Bereiche erfüllen somit die Ungleichung (3.27) nicht.

mit der folgenden Ungleichung nachgeholt werden, die man durch eine Umrechnung aus (3.22) und (3.25) erhält[2]:

$$S \leq \left(\tfrac{1}{2} + \tfrac{\sqrt{3}}{2} \cdot tan(H)\right) \cdot \left(\tfrac{1}{I} - 1\right), \quad \text{falls } 0° \leq H < 60° \tag{3.27}$$

In Bild 3.17 ist der Bereich des HSI-Farbraums für die zwei Farbtöne Rot und Gelb schraffiert dargestellt, der keine gültigen Werte enthält und somit die Ungleichung (3.27) nicht erfüllt, oder anders ausgedrückt, schraffiert ist der Bereich des HSI-Farbraums, der nicht in den RGB-Farbraum transformiert werden kann. Dieses Problem wurde bei den folgenden Farbmodellen HSV/HSB und HSL vermieden.

Das HSV- bzw. HSB-Farbmodell

Das HSV-Farbmodell [Smit78] (auch HSB-Farbmodell genannt, mit B für Brightness) unterscheidet sich vom HSI-Farbmodell nur leicht bzgl. der Werte, wie aus der Tabelle 3.1 ersichtlich ist. Dies äußerst sich in marginalen Abweichungen beim $H(hue)$-Wert, der beim HSI-Farbmodell mittels der nichtlinearen Arcustangens-Funktion (3.22) und beim HSV-Farbmodell mit einer quasilinearen Funktion (3.30) der normierten Farbdifferenzen berechnet wird. Das führt bei Zwischenfarben wie z.B. Orangerot zu Abweichungen im $H(hue)$-Wert von ca. 1.1° (RGB=(1.0, 0.25, 0.0); H(HSI)=13.9°; H(HSV)=15.0°), während bei

[2] falls $60° \leq H < 120°$ ist H zu ersetzen durch $(120° - H)$,
falls $120° \leq H < 180°$ ist H zu ersetzen durch $(H - 120°)$,
falls $180° \leq H < 240°$ ist H zu ersetzen durch $(240° - H)$,
falls $240° \leq H < 300°$ ist H zu ersetzen durch $(H - 240°)$,
falls $300° \leq H < 360°$ ist H zu ersetzen durch $(360° - H)$.

allen Farbtönen, die Vielfache von 30° sind (wie z.B. Rot, Orange, Gelb usw.), die $H(hue)$-Werte in beiden Farbmodellen die gleichen sind. Beim Sättigungswert ist die Tendenz in beiden Farbmodellen ebenfalls die gleiche, allerdings fallen die Werte im HSI-Farbmodell wegen (3.25) fast generell etwas niedriger aus, als im HSV-Farbmodell mit (3.29). Die größten Unterschiede ergeben sich bei der dritten Komponente, da die Intensität (3.22) das arithmetische Mittel der drei Farbkomponenten R,G,B ist, während der Value-Wert das Maximum der drei Farbkomponenten R,G,B ist (3.28).

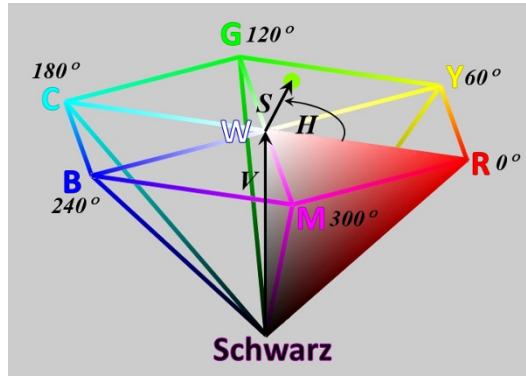


Bild 3.18: Das HSV-Farbmodell dargestellt als hexagonale Pyramide, die auf der Spitze steht. Die zentrale Achse der Pyramide ist die V($value$)-Achse, die von $V = 0$ (Schwarz) bis $V = 1$ (Weiß) reicht und die Grauwerte enthält. Der $H(hue)$-Wert entspricht einem stückweise linearisierten Winkel zwischen der Rot-Ebene und der Ebene, die den Farbwert enthält. Die Sättigung S entspricht dem Abstand zur zentralen Pyramidenachse, wobei der Abstand nicht mit einer euklidischen Metrik, sondern mit (3.29) berechnet wird. Dadurch erhält man am Pyramidenrand immer Sättigungswerte von $S = 1$, obwohl man sich entlang eines Sechsecks und nicht eines Kreises bewegt.

Die Umrechnung des RGB-Farbmodells in das HSV-System geschieht folgendermaßen:

$$V = max(R, G, B). \tag{3.28}$$

$$S = \frac{max(R, G, B) - min(R, G, B)}{max(R, G, B)}, \text{ falls } V \neq 0 \text{ sonst } S = 0. \tag{3.29}$$

$$H = \begin{cases} undefiniert, & \text{falls } S = 0,^{3} \\ 60\frac{G-B}{max(R,G,B)-min(R,G,B)}, & \text{falls } R = max(R,G,B) \\ & \quad \wedge (G-B) \geq 0,^{4} \\ 120 + 60\frac{B-R}{max(R,G,B)-min(R,G,B)}, & \text{falls } G = max(R,G,B),^{5} \\ 240 + 60\frac{R-G}{max(R,G,B)-min(R,G,B)}, & \text{falls } B = max(R,G,B),^{6} \\ 360 + 60\frac{G-B}{max(R,G,B)-min(R,G,B)}, & \text{falls } R = max(R,G,B) \\ & \quad \wedge (G-B) < 0,^{7} \end{cases} \tag{3.30}$$

Die Rückrechnung vom HSV- in das RGB-System geschieht folgendermaßen:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{cases} \begin{pmatrix} V \\ V - S \cdot V + \frac{1}{60} H \cdot S \cdot V \\ V - S \cdot V \end{pmatrix} & \text{falls } 0° \leq H < 60°, \\ \begin{pmatrix} V - S \cdot V - (\frac{1}{60} H - 2) \cdot S \cdot V \\ V \\ V - S \cdot V \end{pmatrix} & \text{falls } 60° \leq H < 120°, \\ \begin{pmatrix} V - S \cdot V \\ V \\ V - S \cdot V + (\frac{1}{60} H - 2) \cdot S \cdot V \end{pmatrix} & \text{falls } 120° \leq H < 180°, \\ \begin{pmatrix} V - S \cdot V \\ V - S \cdot V - (\frac{1}{60} H - 4) \cdot S \cdot V \\ V \end{pmatrix} & \text{falls } 180° \leq H < 240°, \\ \begin{pmatrix} V - S \cdot V + (\frac{1}{60} H - 4) \cdot S \cdot V \\ V - S \cdot V \\ V \end{pmatrix} & \text{falls } 240° \leq H < 300°, \\ \begin{pmatrix} V \\ V - S \cdot V \\ V - S \cdot V - (\frac{1}{60} H - 6) \cdot S \cdot V \end{pmatrix} & \text{falls } 300° \leq H < 360°. \end{cases} \tag{3.31}$$

Der HSV-Farbraum kann als hexagonale Pyramide dargestellt werden, wie in Bild 3.18. Die Pyramide steht auf der Spitze mit $V = 0$ (Schwarz). Die zentrale Achse der Pyramide

[3] (Es handelt sich um einen Grauton)
[4] (Der Farbton liegt zwischen Rot und Gelb, da Rot und Grün dominieren)
[5] (Der Farbton liegt zwischen Gelb und Cyan, da Grün dominiert)
[6] (Der Farbton liegt zwischen Cyan und Magenta, da Blau dominiert)
[7] (Der Farbton liegt zwischen Magenta und Rot, da Rot und Blau dominieren)

ist die V-Achse, die bis zum Maximalwert $V = 1$ (Weiß) reicht und die Grauwerte enthält. Auf dieser Achse ist der $H(hue)$-Wert nicht definiert und es gilt $S = 0$. Der $H(hue)$-Wert entspricht einem stückweise linearisierten Winkel zwischen der Rotebene und der Ebene durch den Farbwert, so dass z.B. Grüntöne bei $H = 120$ und Blautöne bei $H = 240$ liegen. Komplementärfarben liegen im HSV-Farbraum immer um 180° gedreht auf der gegenüberliegenden Seite der V-Achse. Der Sättigungswert S entspricht dem Abstand zur zentralen Pyramidenachse, wobei der Abstand nicht mit einer euklidischen Metrik, sondern mit (3.29) berechnet wird. Dadurch erhält man am Pyramidenrand immer Sättigungswerte von $S = 1$, obwohl man sich entlang eines Sechsecks und nicht eines Kreises bewegt. Die Sättigungswerte S reichen von 0 auf der V-Achse bis zu 1 auf dem Rand der hexagonalen Pyramide.

Farbe	R	G	B	H	S	I	H	S	V	H	S	L
Weiß	1.0	1.0	1.0	n.d.	0	1	n.d.	0	1	n.d.	0	1
Grau	0.5	0.5	0.5	n.d.	0	0.5	n.d.	0	0.5	n.d.	0	0.5
Schwarz	0.0	0.0	0.0	n.d.	0	0.0	n.d.	0	0.0	n.d.	0	0.0
Rot	1.0	0.0	0.0	0	1.0	0.33	0	1.0	1.0	0	1.0	0.5
Korallrot	1.0	0.25	0.25	0	0.5	0.5	0	0.75	1.0	0	1.0	0.62
Rosa	1.0	0.5	0.5	0	0.25	0.66	0	0.5	1.0	0	1.0	0.75
Hellrosa	1.0	0.75	0.75	0	0.1	0.83	0	0.25	1.0	0	1.0	0.88
Signalrot	0.87	0.12	0.12	0	0.66	0.37	0	0.86	0.87	0	0.75	0.5
Altrosa	0.75	0.25	0.25	0	0.4	0.42	0	0.66	0.75	0	0.5	0.5
Pastellrot	0.62	0.37	0.37	0	0.18	0.46	0	0.4	0.62	0	0.25	0.5
Orangerot	1.0	0.25	0.0	13.9	1.0	0.42	15	1.0	1.0	15	1.0	0.5
Korall	1.0	0.5	0.25	19.1	0.57	0.58	20	0.75	1.0	20	1.0	0.62
Orange	1.0	0.5	0.0	30	1.0	0.5	30	1.0	1.0	30	1.0	0.5
Gelborange	1.0	0.75	0.0	46.1	1.0	0.58	45	1.0	1.0	45	1.0	0.5
Gelb	1.0	1.0	0.0	60	1.0	0.66	60	1.0	1.0	60	1.0	0.5
Oliv	0.5	0.5	0.25	60	0.40	0.42	60	0.5	0.5	60	0.33	0.38
Grüngelb	0.5	1.0	0.0	90	1.0	0.5	90	1.0	1.0	90	1.0	0.5
Grün	0.0	1.0	0.0	120	1.0	0.33	120	1.0	1.0	120	1.0	0.5
Cyan	0.0	1.0	1.0	180	1.0	0.66	180	1.0	1.0	180	1.0	0.5
Blau	0.0	0.0	1.0	240	1.0	0.33	240	1.0	1.0	240	1.0	0.5
Magenta	1.0	0.0	1.0	300	1.0	0.66	300	1.0	1.0	300	1.0	0.5

Tabelle 3.1: Beispiele für ausgewählte Farben im RGB-Farbmodell und ihre entsprechenden Werte im HSI-, HSV-, bzw HSL-Farbmodell: die ersten drei 'Farben' sind reine Grautöne, d.h ein H-Wert ist hier nicht definiert (n.d.); die nächsten sieben Farben sind reine Rottöne (H-Wert = 0) mit abnehmender Sättigung; die restlichen Farben steigen im H-Wert (Farbton) an und zwar bei unterschiedlich hohen Sättigungswerten je Farbmodell.

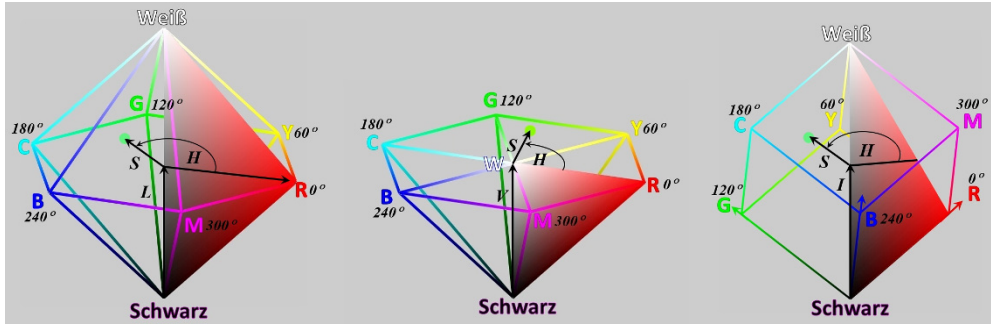
Ausgangspunkt für die Entwicklung des HSV-Farbmodells war das intuitive Vorgehen von Malern bei der Mischung einer Farbe. Ein Maler startet üblicherweise mit einem reinen Farbton ($S = 1$, $V = 1$), d.h. die Farbe enthält ausschließlich Pigmente dieses Farbtons. Wenn er Weiß dazu mischt, reduziert er die Sättigung ohne den V-Wert zu ändern, d.h. er wandert im HSV-Farbraum auf gleicher Höhe (V-Wert) in Richtung der zentralen Achse. Mischt er Schwarz dazu, reduziert er gleichzeitig Sättigung und Value, d.h. er wandert im HSV-Farbraum in Richtung der Pyramidenspitze (Schwarz).

Das HSL-Farbmodell

Den HSL-Farbraum (Hue, Saturation, Lightness) kann man sich als eine Deformation des HSV-Farbraums vorstellen, bei der der Weißpunkt in der obersten Pyramidenebene bei $L = 1$ einfach nach oben gezogen wird, so dass eine doppelte hexagonale Pyramide entsteht (Bild 3.19). Deshalb unterscheidet sich das HSL-Farbmodell nicht in der Berechnung der $H(hue)$-Werte gegenüber HSV, sondern nur bei S und V bzw. L.

Die Umrechnung des RGB-Farbmodells in das HSL-System geschieht folgendermaßen:

$$L = \frac{max(R,G,B) + min(R,G,B)}{2}. \tag{3.32}$$



HSL (a) HSV (b) HSI/RGB (c)

Bild 3.19: (a) Das HSL-Farbmodell dargestellt als hexagonale Doppel-Pyramide, deren untere Spitze Schwarz und deren obere Spitze Weiß ist. Die zentrale Achse der Doppel-Pyramide ist die L(*Lightness*)-Achse, die von $L = 0$ (Schwarz) bis $L = 1$ (Weiß) reicht und die Grauwerte enthält. Der $H(hue)$-Wert entspricht einem stückweise linearisierten Winkel zwischen der Rot-Ebene und der Ebene, die den Farbwert enthält. Die Sättigung S entspricht dem Abstand zur zentralen Pyramidenachse, wobei der Abstand nicht mit einer euklidischen Metrik, sondern mit (3.32) berechnet wird. Dadurch erhält man am Pyramidenrand immer Sättigungswerte von $S = 1$, obwohl man sich entlang eines Sechsecks und nicht eines Kreises bewegt. (b) Zum Vergleich das HSV-Farbmodell dargestellt als einfache hexagonale Pyramide. (c) Zum Vergleich der auf der schwarzen Spitze stehende RGB-Farbwürfel, in den das HSI-Koordinatensystem eingezeichnet ist.

$$S = \begin{cases} 0, & \text{falls } max(R,G,B) = min(R,G,B), \\ \frac{max(R,G,B) - min(R,G,B)}{max(R,G,B) + min(R,G,B)}, & \text{falls } L \leq 0.5 \\ \frac{max(R,G,B) - min(R,G,B)}{2.0 - (max(R,G,B) + min(R,G,B))}, & \text{falls } L > 0.5 \end{cases} \tag{3.33}$$

Die Berechnung des $H(hue)$-Werts erfolgt nach demselben Algorithmus (3.30), wie beim HSV-Farbmodell.

Die Umrechnung vom HSL- in das RGB-System geschieht folgendermaßen, falls $L \leq 0.5$ ist:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{cases} \begin{pmatrix} L + S \cdot L \\ L - S \cdot L + \frac{1}{30} H \cdot S \cdot L \\ L - S \cdot L \end{pmatrix} & \text{falls } 0° \leq H < 60° \\ \begin{pmatrix} L - S \cdot L - (\frac{1}{30} H - 4) \cdot S \cdot L \\ L + S \cdot L \\ L - S \cdot L \end{pmatrix} & \text{falls } 60° \leq H < 120° \\ \begin{pmatrix} L - S \cdot L \\ L + S \cdot L \\ L - S \cdot L + (\frac{1}{30} H - 4) \cdot S \cdot L \end{pmatrix} & \text{falls } 120° \leq H < 180° \\ \begin{pmatrix} L - S \cdot L \\ L - S \cdot L - (\frac{1}{30} H - 8) \cdot S \cdot L \\ L + S \cdot L \end{pmatrix} & \text{falls } 180° \leq H < 240° \\ \begin{pmatrix} L - S \cdot L + (\frac{1}{30} H - 8) \cdot S \cdot L \\ L - S \cdot L \\ L + S \cdot L \end{pmatrix} & \text{falls } 240° \leq H < 300° \\ \begin{pmatrix} L + S \cdot L \\ L - S \cdot L \\ L - S \cdot L - (\frac{1}{30} H - 12) \cdot S \cdot L \end{pmatrix} & \text{falls } 300° \leq H < 360° \end{cases} \tag{3.34}$$

Falls $L > 0.5$ ist, erfolgt die Umrechnung vom HSL- in das RGB-System folgendermaßen:

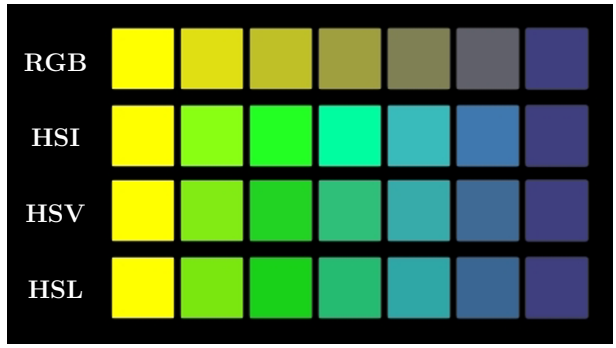
$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{cases} \begin{pmatrix} L + S(1-L) \\ L - S(1-L) + \frac{1}{30} H S(1-L) \\ L - S(1-L) \end{pmatrix} & \text{falls } 0° \leq H < 60° \\ \begin{pmatrix} L - S(1-L) - (\frac{1}{30} H - 4) S(1-L) \\ L + S(1-L) \\ L - S(1-L) \end{pmatrix} & \text{falls } 60° \leq H < 120° \\ \begin{pmatrix} L - S(1-L) \\ L + S(1-L) \\ L - S(1-L) + (\frac{1}{30} H - 4) S(1-L) \end{pmatrix} & \text{falls } 120° \leq H < 180° \\ \begin{pmatrix} L - S(1-L) \\ L - S(1-L) - (\frac{1}{30} H - 8) S(1-L) \\ L + S(1-L) \end{pmatrix} & \text{falls } 180° \leq H < 240° \\ \begin{pmatrix} L - S(1-L) + (\frac{1}{30} H - 8) S(1-L) \\ L - S(1-L) \\ L + S(1-L) \end{pmatrix} & \text{falls } 240° \leq H < 300° \\ \begin{pmatrix} L + S(1-L) \\ L - S(1-L) \\ L - S(1-L) - (\frac{1}{30} H - 12) S(1-L) \end{pmatrix} & \text{falls } 300° \leq H < 360° \end{cases} \tag{3.35}$$

Eine zweite Möglichkeit, wie man sich die Entstehung des HSL-Farbraums vorstellen kann, besteht in einer Verformung des auf der schwarzen Spitze stehenden RGB-Farbwürfels (Bild 3.19-c) und zwar so, dass die drei Eckpunkte R,G,B, die im HSI-Farbraum auf einer Ebene mit $I = 0.33$ liegen, etwas nach oben auf einen Wert von 0.5 gezogen werden und die drei Eckpunkte C,M,Y, die HSI-Farbraum auf einer Ebene mit $I = 0.66$ liegen, etwas nach unten ebenfalls auf einen Wert von 0.5 gezogen werden.

Farbübergänge

Eine wichtige Anwendung von Farbmodellen ist die Interpolation zwischen Farben, d.h. die Berechnung von Farbübergängen. Diese werden nicht nur im künsterlischen Umfeld benötigt, sondern z.B. auch beim Gouraud-Shading (Band I Abschnitt 12.2.2), beim Anti-Aliasing (Band I Kapitel 10) und bei der Farbmischung von semi-transparenten Objekten (Band I Abschnitt 9.3). Dabei ist zu beachten, dass das Ergebnis der Farbmischung wesentlich von der Wahl des Farbmodells abhängt, in dem die Farben interpoliert werden.

Intuitiv geht man davon aus, dass bei der Berechnung eines Überganges zwischen zwei Farben eine lineare Interpolation der Farbwerte durchgeführt wird, die man sich im Far-



(a) (b)

Bild 3.20: Farbübergang zwischen den Farben (a) Gelb (RGB=(1,1,0)) und (b) Dunkelblau (RGB=(0.25, 0.25, 0.5)) dargestellt in fünf Schritten mit gleichen Farbabständen im jeweiligen Farbmodell (RGB, HSI, HSV, HSL). (a) Beim RGB-Farbmodell in der obersten Bildzeile erhält man den Eindruck, dass der gelbe Farbton langsam ausgeblendet wird und in den blauen Farbton übergeht, ohne dass andere Farbtöne beteiligt wären. Bei den HS∗-Farbmodellen entsteht dagegen nicht der Eindruck, dass die beiden äußeren Farben ineinander übergehen, sondern dass man sich entlang eines mehr oder weniger gesättigten Farbkreises bewegt.

braum als gerade Linie zwischen den beiden Farbwerten vorstellen kann. Wenn unter einer Farbraumtransformation eine solche gerade Linie erhalten bleibt, wird der Farbübergang in beiden Farbräumen identisch sein. Dies gilt beispielsweise für die Farbmodelle RGB, CMY, YIQ und CIE-XYZ, die durch affine Transformationen ineinander umgerechnet werden können. Eine gerade Linie im RGB-Farbmodell wird in den HS∗-Farbmodellen üblicherweise nicht in eine gerade Linie übergehen, da es sich dabei um nicht-lineare Farbraumtransformationen handelt. In Bild 3.20 ist der Farbübergang zwischen den Farben Gelb (RGB=(1,1,0)) und Dunkelblau (RGB=(0.25, 0.25, 0.5)) dargestellt. Werden beide Farben im RGB-Farbraum zu 50% gemischt, erhält man als Mischfarbe (RGB=(0.625, 0.625, 0.25)), in Bild 3.20 in der Mitte der obersten Zeile dargestellt. Im HSL-Farbraum dagegen erhält man aus Gelb (HSL=(60,1,0.5)) und Dunkelblau (HSL=(240,0.33,0.375)) als Mischfarbe (HSL=(150,0.66,0.44)), die umgerechnet im RGB-Farbraum einer ganz anderen Mischfarbe RGB=(0.15, 0.73, 0.44)) entspricht, wie in Bild 3.20 in der Mitte der untersten Zeile auch zu erkennen ist. Ähnlich verhält es sich auch in der beiden anderen Farbräumen HSI und HSV. Beim RGB-Farbmodell in der obersten Bildzeile erhält man den Eindruck, dass der gelbe Farbton langsam ausgeblendet wird und in den blauen Farbton übergeht, ohne dass andere Farbtöne beteiligt wären. Bei den HS∗-Farbmodellen entsteht dagegen nicht der Eindruck, dass die beiden äußeren Farben ineinander übergehen, sondern dass man sich entlang eines mehr oder weniger gesättigten Farbkreises bewegt.

3.5.8 Das CIE-Lab-Farbmodell

Alle bisher beschriebenen Farbmodelle besitzen eine gemeinsame Schwachstelle: gleiche Abstände zwischen verschiedenen Farbwerten im jeweiligen Farbraum werden vom Menschen nicht als gleiche Farbunterschiede empfunden. Um diese Unzulänglichkeit zu überwinden, wurde von der CIE (Commission Internationale de l'Eclairage) im Jahre 1976 das *CIE-Lab-Farbmodell* eingeführt (die zugehörige deutsche Norm ist die [DIN6174]). In diesem Farbmodell lassen sich Farbabstände durch eine Maßzahl charakterisieren, die die Größe eines wahrgenommenen Farbunterschiedes zwischen zwei Proben (z.B. Vorlage und Nachfärbung) beschreibt.

Das *CIE-Lab-Farbmodell* wird in der industriellen Praxis sehr häufig eingesetzt. So wird es in der modernen Medientechnik, wie z.B. in der Bildbearbeitungssoftware Adobe Photoshop, verwendet, um Bilder vom RGB-Farbraum eines Monitors in das zum Ausdrucken nötige CMYK-Farbmodell zu bringen. Bei der Produktion von Farben und Lacken werden meist die RAL DESIGN-Farben benutzt, die auf dem CIE-Lab-Farbmodell basieren und am Ende dieses Abschnitts genauer beschrieben werden.

Das *CIE-Lab-Farbmodell* wird durch ein dreidimensionales Koordinatensystem beschrieben, in dem die Koordinaten L (Luminanz / Helligkeit), a (rot-grün Achse) und b (gelb-blau Achse) senkrecht aufeinander stehen. Direkt auf der vertikalen L-Achse befinden sich alle Grautöne mit den Werten von 0 (Schwarz) bis 100 (Weiß). Auf einer durch die a- und b-Achsen aufgespannten Ebene, die senkrecht auf der L-Achse steht, befinden sich alle Farbtöne gleicher Helligkeit (siehe Bild 3.21). Positive a-Werte, die bis zu +100 gehen können, stellen rote Farbtöne dar, und negative a-Werte, die bis zu -150 gehen können, stellen grüne Farbtöne dar. Positive b-Werte, die bis zu +150 gehen können, stellen gelbe Farbtöne dar, und negative b-Werte, die bis zu -100 gehen können, stellen blaue Farbtöne dar. In den Bildern 3.21(a)-(i) sind die a-b-Ebenen gleicher Helligkeit für den RGB-Farbraum der Monitor-Grundfarben dargestellt.

Die Umrechnung vom (R, G, B)-Farbraum in den $(CIE - Lab)$-Farbraum erfolgt in zwei Schritten: Zuerst werden die RGB-Farbwerte gemäß Gleichung 3.36 in die (X, Y, Z)-CIE-Normfarbwerte transformiert (diese Operation wurde im Abschnitt 3.5.4 schon einmal ausführlich dargestellt):

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.478 & 0.299 & 0.175 \\ 0.263 & 0.655 & 0.081 \\ 0.020 & 0.160 & 0.908 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.36}$$

Im zweiten Schritt werden die Farbmaßzahlen (L, a, b) des angenähert gleichförmigen CIE-Lab-Farbmodells nach folgenden Gleichungen berechnet:

$$L = 116 f\left(\frac{Y}{Y_w}\right) - 16 \tag{3.37}$$

$$a = 500\left\{f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right)\right\} \tag{3.38}$$

$$b = 200\left\{f\left(\frac{Y}{Y_w}\right) - f\left(\frac{Z}{Z_w}\right)\right\} \tag{3.39}$$

Dabei gilt

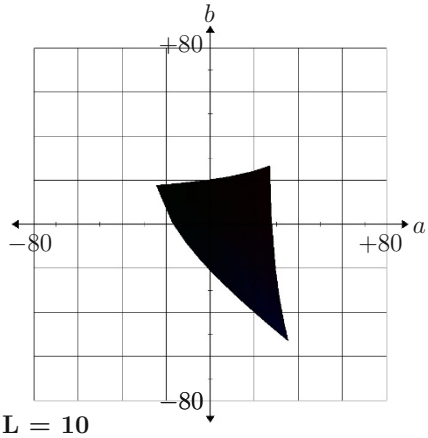
$$f(x) = \begin{cases} \sqrt[3]{x}, & \text{falls } x > \left(\frac{24}{116}\right)^3 \approx 0.00885645, \\ \frac{841}{108}x + \frac{16}{116}, & \text{falls } x \leq \left(\frac{24}{116}\right)^3 \approx 0.00885645. \end{cases} \tag{3.40}$$

Die CIE-Normfarbwerte für die in der Praxis häufig verwendete Normlichtart **D65** (das Weiß eines auf 6500° Kelvin erhitzten schwarzen Strahlers) und den 2°-Normalbeobachter sind:

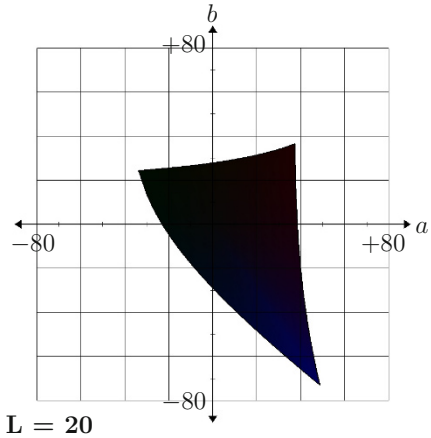
$$X_w = 0.951, \quad Y_w = 1.000 \quad \text{und} \quad Z_w = 1.088. \tag{3.41}$$

Hier einige Beispiele für ausgewählte (Monitor-)Grundfarben im RGB-Farbmodell und ihre entsprechenden Werte im CIE-Lab-Farbmodell:

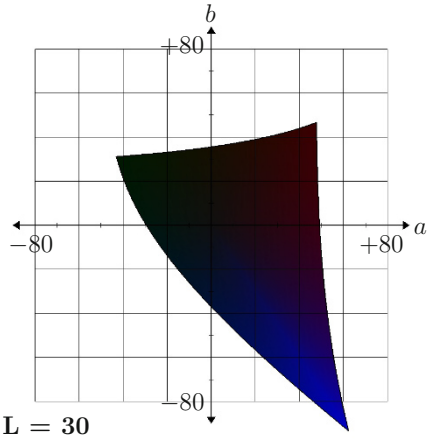
Farbe	R	G	B	L	a	b
Weiß	1.0	1.0	1.0	100	0	0
Grau	0.5	0.5	0.5	76	0	0
Schwarz	0.0	0.0	0.0	0	0	0
Rot	1.0	0.0	0.0	58	77	75
Gelb	1.0	1.0	0.0	97	-19	85
Grün	0.0	1.0	0.0	85	-94	68
Cyan	0.0	1.0	1.0	89	-55	-18
Blau	0.0	0.0	1.0	34	68	-102
Magenta	1.0	0.0	1.0	65	91	-50



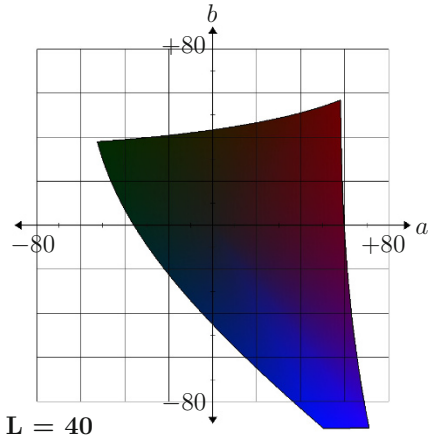
(a) **L = 10**



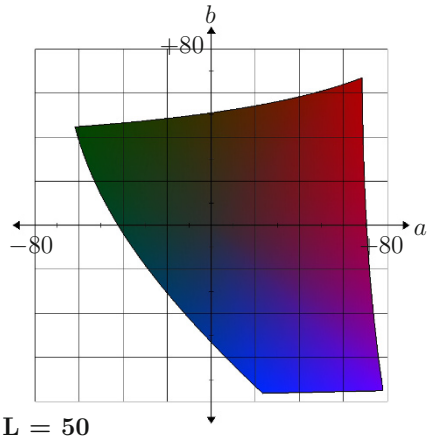
(b) **L = 20**



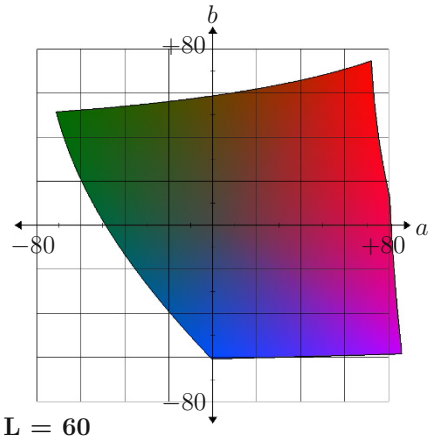
(c) **L = 30**



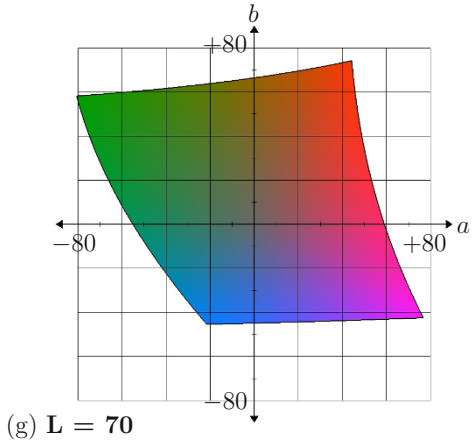
(d) **L = 40**



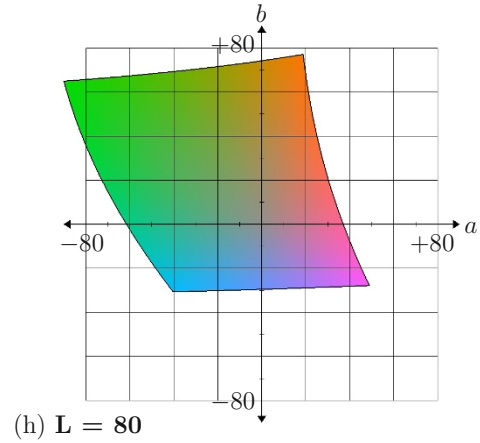
(e) **L = 50**



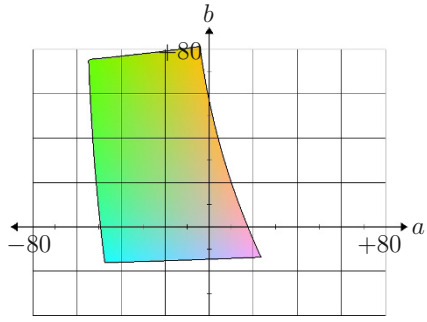
(f) **L = 60**



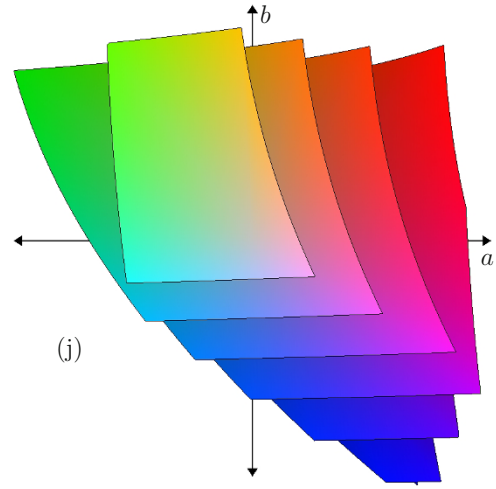
(g) **L = 70**



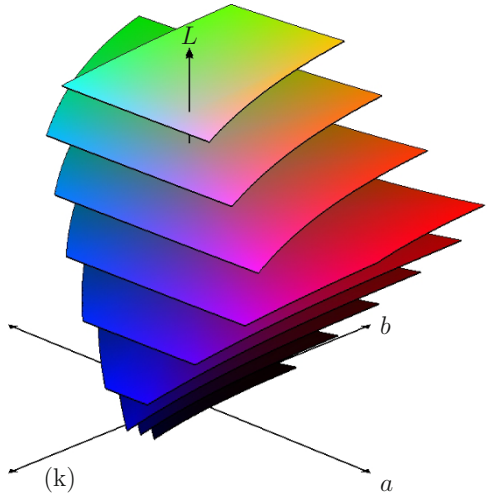
(h) **L = 80**



(i) **L = 90**



(j)



(k)

Bild 3.21: Der CIE-Lab-Farbraum (für das RGB-Farbdreieck der (Monitor)-Grundfarben) (a) - (i) dargestellt ist immer die a-b-Ebene des Farbraums mit konstantem Luminanzwert L. (j) Hier sind alle Ebenen von L = 10 bis L = 90 übereinander gestapelt. (k) Der CIE-Lab-Farbraum in 3D-Darstellung.

Farbabstandsformel

Ausgangspunkt bei der Einführung des *CIE-Lab-Farbmodells* war, dass die euklidische Distanz zwischen Farben, z.B. einer Vorlage (Index V) und einer Nachfärbung (Index N), der vom Menschen empfundenen Größe eines Farbunterschiedes entspricht. Das sehr gebräuchliche Maß für diesen Farbabstand wird mit ΔE_{ab} bezeichnet:

$$\Delta E_{ab} = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2} \tag{3.42}$$

wobei gilt:

$$\Delta L = L_V - L_N, \quad \Delta a = a_V - a_N, \quad \Delta b = b_V - b_N \tag{3.43}$$

Ein Farbabstand von $\Delta E_{ab} < 1$ ist bzw. soll unter Normbeleuchtung und -beobachtung nicht mehr wahrnehmbar sein, ein Farbabstand von $\Delta E_{ab} = 5$ ist dagegen deutlich erkennbar, obwohl sich die beiden Farben immer noch stark ähneln. Leider ist der *CIE-Lab-Farbraum* für kleine Farbabstände ($\Delta E_{ab} < 2$) jedoch ortsabhängig: bei zwei grauen Farbtönen ist ein Farbabstand von $\Delta E_{ab} = 0.5$ gerade noch wahrnehmbar, bei zwei orangen Farbtönen ist ein $\Delta E_{ab} = 2$ nicht mehr unterscheidbar. Aus diesem Grund wurde eine neue Norm entwickelt, die [DIN6176]:*„Farbmetrische Bestimmung von Farbabständen bei Körperfarben nach der DIN99-Formel“*, die auch für kleine Farbabstände ortsunabhängige ΔE_{ab}-Werte liefert.

Umrechnung von *Lab*- in *HLC*-Koordinaten

Analog zum *HSI-Farbmodell* (siehe Abschnitt 3.5.7) kann man auch aus den kartesischen (L,a,b)-Koordinaten durch eine Umrechnung in (H,L,C)-Zylinderkoordinaten empfindungsgemäße Größen für Hue (H = Farbton / Buntton), Luminanz (L = Helligkeit / Intensität / Lightness) und Chroma (C = Sättigung / Saturation / Buntheit) berechnen:

$$H_{ab} = \arctan\left(\frac{b}{a}\right), \tag{3.44}$$

$$C_{ab} = \sqrt{a^2 + b^2}. \tag{3.45}$$

Zur besseren Unterscheidbarkeit von Farbton- und Sättigungs/Chroma-Werten zwischen dem *HSI-Farbmodell* und dem *CIE-Lab-Farbmodell* werden bei letzterem noch die Indizes H_{ab} bzw. C_{ab} angefügt, die anzeigen, dass die Werte aus den *ab*-Koordinaten gewonnen werden. Die L-Koordinaten (Luminanz / Intensität) werden unverändert übernommen.

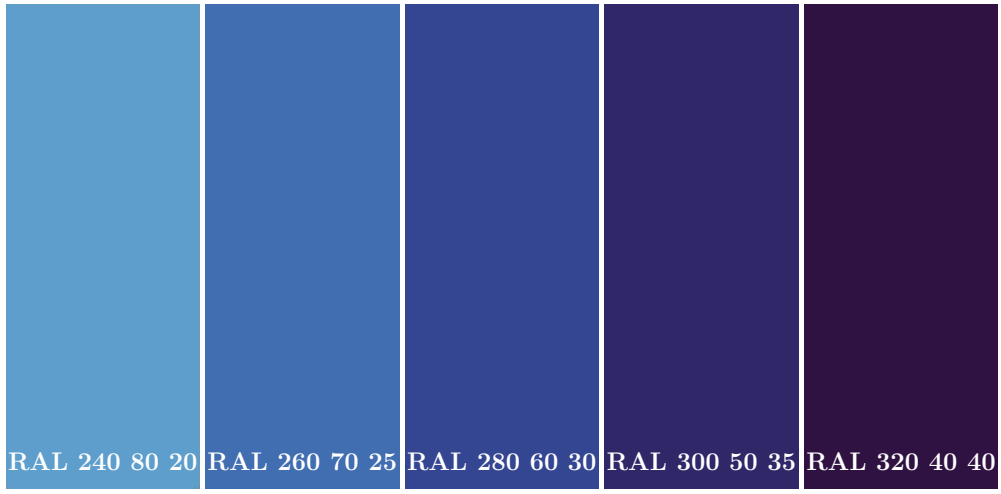
RAL DESIGN System

Eine Anwendung des *CIE-Lab-Farbmodells* auf Basis der HLC-Koordinaten, die in nahezu jedem Baumarkt an der Farbmischtheke angetroffen werden kann, ist das RAL DESIGN System. Denn RAL DESIGN-Farben werden systematisch als CIE-HLC-Farbwerte festgelegt, danach werden die Rezepturen entwickelt, bis Soll- und Ist-Farbton um nicht mehr als $\Delta E_{ab} = 0.7$ voneinander abweichen.

Die RAL DESIGN-Farbnamen sind 7-stellig: „RAL HHH LL CC“. Dabei ist:

- „HHH“ der Farbton (Hue) gemäß Formel 3.44, d.h. ein Wert zwischen 010 und 360 Grad, der im 10-Grad-Abstand abgegriffen wird (im Orange-Gelb-Bereich im 5-Grad-Abstand), der Wert 000 ist für Grautöne reserviert,
- „LL“ die Luminanz (Helligkeit) gemäß Formel 3.37, d.h. ein Wert zwischen 00 (Schwarz) und 100 (Weiß), der in Schritten von $\Delta E_{ab} = 10$ abgetastet wird,
- „CC“ die Chromazität (Sättigung) gemäß Formel 3.45, d.h. ein Wert zwischen 00 (Grauton der entsprechenden Helligkeit) und 60 bis 90 (je nach Farbton und Helligkeit), der in Schritten von $\Delta E_{ab} = 5$ bzw. 10 abgetastet wird.

Damit lässt sich zu jeder RAL DESIGN-Farbe eine Nachbarfarbe finden, die einen maximalen Farbabstand von $\Delta E_{ab} = 10$ aufweist. Dies kann man z.B. nutzen, um einen fein abgestuften Farbübergang zwischen zwei beliebigen Farben herzustellen, wie in Bild 3.22 dargestellt.



(a) (b) (c) (d) (e)

Bild 3.22: Farbübergang zwischen den Farben (a) RAL 240 80 20 und (e) RAL 320 40 40 in drei Stufen, bei denen die Unterschiede in H, L, C gleichmäßig aufgeteilt werden: (b) RAL 260 70 25. (c) RAL 280 60 30. (d) RAL 300 50 35.

3.5.9 Mathematisches Modell für Farbbilder

Im Rahmen dieser Darstellung wird das RGB-Farbmodell verwendet. Bei der Digitalisierung eines Farbbildes werden mit Farbfiltern Rot-, Grün- und Blauauszüge angefertigt. Jeder der drei Abtastvorgänge ist, abgesehen von der Verwendung eines Farbfilters, identisch mit der Digitalisierung bei Grautonbildern (Kapitel 3.4). Ein Bildpunkt eines digitalisierten Farbbildes besteht aus drei Maßzahlen für rot, grün und blau (Bild 3.23).

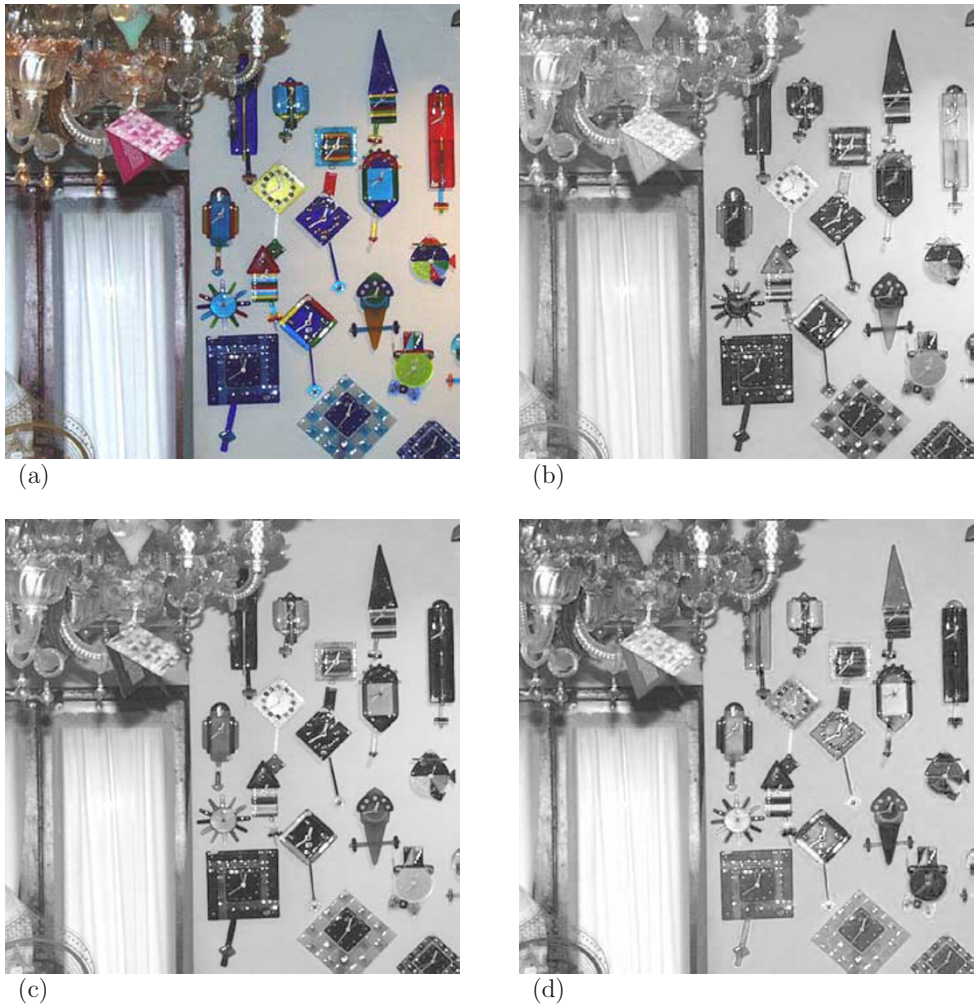
Das mathematische Modell von Kapitel 3.4 muss für Farbbilder erweitert werden, da die Bildmatrix jetzt dreidimensional ist:

$$
\begin{array}{ll}
G = \{0, 1, 2, ..., 255\} & \text{Grauwertmenge mit 256 Grauwerten} \\
\mathbf{S} = (s(x, y, n)) & \text{dreidimensionale Bildmatrix des digitalisierten Farbbildes} \\
x = 0, 1, ..., L-1 & L \text{ Bildzeilen} \\
y = 0, 1, ..., R-1 & R \text{ Bildspalten} \\
n = 0, 1, 2 & N = 3 \text{ Kanäle} \\
\mathbf{s}(x, y) = (g_0, g_1, g_2)^T & \text{Bildpunkt in der Position } (x, y). \\
g_n \in G, n = 0, 1, 2 & \text{Maßzahl für rot, grün und blau.}
\end{array}
\tag{3.46}
$$

Die einzelnen Farbkomponenten werden in der Regel mit einem Byte codiert. Ein Bildpunkt eines digitalisierten Farbbildes umfasst also drei Byte oder 24 Bit. Bei praktischen Anwendungen treten aber wesentlich weniger als die $2^{24} = 16777216$ theoretisch möglichen Farbtöne auf. Moderne Grafikkarten verwenden allerdings heute meistens 36 Bit.

Die Werte der Farbkomponenten liegen hier nicht, wie in Kapitel 3.5.4 dargestellt, im Intervall [0,1], sondern in der Grauwertmenge $G = \{0, 1, 2, ..., 255\}$. Bei Umrechnung in andere Farbmodelle ist das unter Umständen zu beachten. Wegen der diskretisierten Werte können sich bei der Umrechnung Fehler ergeben.

Die Farbe ist ein wichtiges Merkmal für die Segmentierung von Bildern. Die Farbe als Merkmal für eine Bildsegmentierung wird in Kapitel 12 ausführlich besprochen. Dort werden auch einige Verfahren zur Weiterverarbeitung von Farbbildern, etwa die Erstellung von Indexbildern, erläutert. Die Verwendung des HSI-Modells bringt bei manchen Verarbeitungsschritten Zeitvorteile, da geringere Datenmengen verarbeitet werden müssen. Soll z.B. eine Bildsegmentierung mit der Farbe als Merkmal durchgeführt werden, so müssen im RGB-Modell alle drei Farbauszüge berücksichtigt werden. In der HSI-Darstellung kann die Segmentierung möglicherweise allein anhand des Farbwertes durchgeführt werden. In den Kapiteln 20 bis 21 werden häufig Beispiele mit dem Merkmal Farbe verwendet.



(a) (b)

(c) (d)

Bild 3.23: Beispiel eines Farbbildes (a), Rotauszug (b), Grünauszug (c) und Blauauszug (d).

3.6 Multispektral- und mehrkanalige Bilder

Bei vielen Anwendungen werden multisensorielle Systeme eingesetzt, die u.a. auch Wellenlängen erfassen, die für das menschliche Auge nicht mehr sichtbar sind (z.B. ultraviolett, nahes oder thermisches infrarot). Digitalisierte Bilder dieser Art bezeichnet man als *Multispektralbilder*. Bild 3.24 stellt einen Ausschnitt aus einem Multispektralbild des Fernerkundungssatellitensystems SPOT dar, das unter anderem mit einem Multispektralscanner die Spektralbereiche $500nm - 590nm$ (grün), $610nm - 680nm$ (rot) und $790nm - 890nm$ (nahes infrarot) mit einer räumlichen Auflösung von $20m$ aufzeichnet.

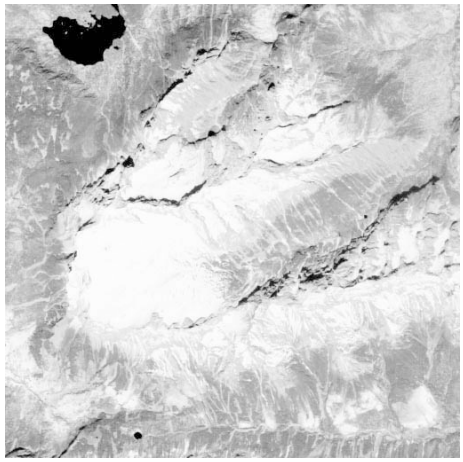
Ein Multispektralbild kann wie das digitalisierte Farbbild als dreidimensionale Bildmatrix $\mathbf{S} = (s(x, y, n))$ dargestellt werden, nur dass für den Index n der Definitionsbereich erweitert wird: $n = 0, 1, ..., N - 1$. Ein digitalisiertes Farbbild ist ein Spezialfall eines Multispektralbildes:

$G = \{0, 1, 2, ..., 255\}$	Grauwertmenge mit 256 Grauwerten	
$\mathbf{S} = (s(x, y, n))$	dreidimensionale Bildmatrix	
$x = 0, 1, ..., L - 1$	L Bildzeilen	
$y = 0, 1, ..., R - 1$	R Bildspalten	(3.47)
$n = 0, 1, ..., N - 1$	N Kanäle	
$\mathbf{s}(x, y) = (g_0, g_1, ..., g_{N-1})^T$	Bildpunkt in der Position (x, y).	
$g_n \in G, n = 0, 1, ..., N - 1$	Maßzahl im Kanal n.	

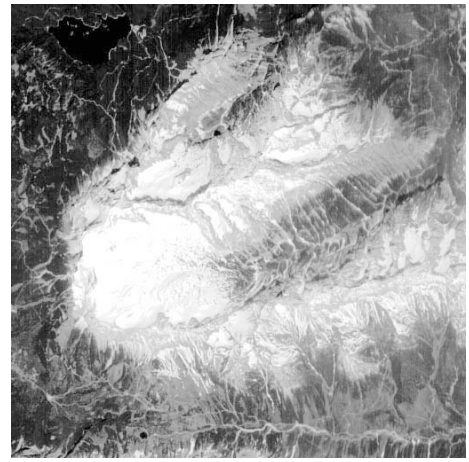
Oft haben die einzelnen Kanäle eines Bildes nicht mehr die Bedeutung von Spektralinformation. Das ist z.B. dann der Fall, wenn die Spektralkanäle mit Operatoren zur Kantenextraktion behandelt wurden oder wenn ein Bild aus mehreren Spektralkanälen und zusätzlich aus einem Kanal mit dem digitalen Höhenmodell des Landschaftsausschnittes zusammengesetzt ist. Bilder dieser Art werden als *mehrkanalige Bilder* bezeichnet. Das mathematische Modell muss dazu nicht erweitert werden. Eine grafische Darstellung eines 4-kanaligen Bildes zeigt Bild 3.25. Ein Multispektralbild ist ein Spezialfall eines mehrkanaligen Bildes.

An dieser Stelle seien noch die Begriffe Pseudofarbdarstellung, Falschfarbendarstellung und Echtfarbendarstellung erläutert.

Bei der *Pseudofarbdarstellung* ist das Ausgangsbild ein einkanaliges Grauwertbild $\mathbf{S} = (s(x, y))$ mit $s(x, y) \in G$. Zu diesem Bild liegt eine *look-up-Tabelle* (LuT), oft auch *Palette*) vor. Die *look-up*-Tabelle ist eine Tabelle mit 256 Einträgen mit den relativen Adressen 0 bis 255. Der Grauwert $s(x, y)$ wird als Index interpretiert (daher auch der häufig verwendete Name *Indexbild*), der auf einen der Einträge der *look-up*-Tabelle zeigt. Zur Darstellung auf einem Displaysystem oder bei der Ausgabe auf einem Drucker wird der in der *look-up*-Tabelle eingetragene Wert $LuT[s(x, y)]$ verwendet. Dies kann ein 8-Bit-Grauwert g oder ein 24-Bit-RGB-Farbtripel (r, g, b) sein. Wenn z.B. die *look-up*-Tabelle mit den Werten $LuT[i] = i, \quad i = 0, ..., 255$ geladen ist, wird das Bild so dargestellt, wie es in der Bildmatrix $\mathbf{S}$ gespeichert ist. Das invertierte Bild (Negativbild) erhält man mit $LuT[i] = 255 - i, \quad i = 0, ..., 255$. Durch passende Wahl der *look-up*-Tabelle lassen sich,



(a)



(b)

(c)

Bild 3.24: Beispiel eines Multispektralbildes des Fernerkundungssatelliten SPOT. Der Bildausschnitt zeigt das Wettersteingebirge mit der Zugspitze. Es sind die drei Spektralbereiche (a) grün ($500nm - 590nm$), (b) rot ($610nm - 680nm$) und (c) nahes infrarot ($590nm - 890nm$) abgebildet. Die Bodenauflösung beträgt $20m$.

in Verbindung mit einem geeigneten Displaysystem, meistens in Videoechtzeit, Helligkeits- und Kontrastveränderungen durchführen. Ist der LuT-Eintrag ein 24-Bit-RGB-Farbtripel, so wird auf dem Displaysystem die Farbe ausgegeben, die diesem Farbtripel entspricht, vorausgesetzt, dass das Displaysystem 24-Bit-Farbtripel darstellen kann. Wie aus RGB-Farbbildern mit 24 Bit pro Bildpunkt Indexbilder mit nur 8 Bit pro Bildpunkt erstellt werden können, wird in Kapitel 12 ausführlich beschrieben.

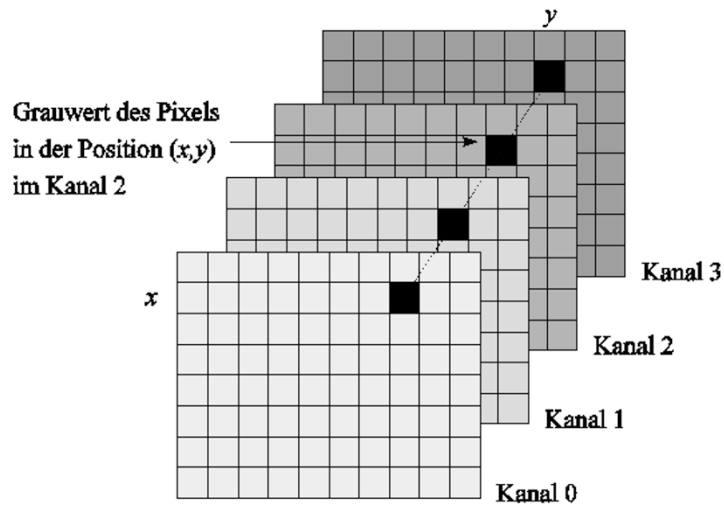


Bild 3.25: Beispiel eines 4-kanaligen Bildes mit einer dreidimensionalen Bildmatrix. Ein Bildpunkt ist hier ein 4-dimensionaler Vektor.

Der Begriff *Falschfarbendarstellung* wird fast nur in der Fernerkundung verwendet. Damit ist gemeint, dass den RGB-Kanälen eines Displaysystems nicht die RGB-Kanäle eines Farb- oder Multispektralbildes, sondern andere Kanalkombinationen zugeordnet werden. Man könnte z.B. dem Rotkanal des Displaysystems den Grünkanal eines Multispektralbildes **S**, dem Grünkanal des Displaysystems den Blaukanal von **S** und dem Blaukanal des Displaysystems einen Infrarotkanal von **S** zuordnen.

Von einer *Echtfarbendarstellung* spricht man, wenn das Darstellungssystem die durch die menschliche Physiologie gestellten Anforderungen erfüllt. Dazu gehören mindestens drei Farbkanäle, ausreichender Werteumfang der Farbsignale und hinreichend sorgfältig gewählte Grundfarben.

3.7 Bildfolgen

Die letzte Verallgemeinerung des mathematischen Modells für digitalisierte Bilddaten sind Bildfolgen. Ein in Kapitel 3.6 definiertes mehrkanaliges Bild kann zu verschiedenen Zeitpunkten $t = 0, 1, 2, ...$ aufgezeichnet werden. Das Modell wird dazu wie folgt erweitert:

$G = \{0, 1, 2, ..., 255\}$	Grauwertmenge mit 256 Grauwerten.
$\mathbf{S} = (s(x, y, n, t))$	vierdimensionale Bildmatrix
$x = 0, 1, ..., L - 1$	L Bildzeilen
$y = 0, 1, ..., R - 1$	R Bildspalten
$n = 0, 1, ..., N - 1$	N Kanäle
$t = 0, 1, ..., T - 1$	T Bilder (Frames)
$(\mathbf{s}(x, y)) = (\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{T-1})$	Bildpunkt in der Position (x, y) als Matrix geschrieben.
$\mathbf{g}_t = (g_{t0}, g_{t1}, ..., g_{tN-1})^T$	N-dimensionaler Vektor der Grauwerte des N-kanaligen Bildes $\mathbf{S}$ zum Zeitpunkt t.
$g_{tn} \in G$	Grauwert des Bildpunktes in der Position (x, y) zum Zeitpunkt t im Kanal n.

(3.48)

Als Beispiel kann die Digitalisierung eines Farbvideofilms dienen: Hier fallen bei der derzeitigen Technik 25 Vollbilder pro Sekunde an, die von einem Farbvideoframegrabber pro Bildpunkt in ein 24-Bit-RGB-Tripel umgesetzt werden. Ein Problem war hier lange Zeit nicht die Digitalisierung, sondern die Abspeicherung der Datenmengen, die bei längeren Bildsequenzen anfallen. Das Fassungsvermögen der heutigen Festplatten hat hier Abhilfe geschaffen. Bild 3.26 zeigt eine Grauwertbildfolge mit acht Teilbildern. Bei Bildfolgen werden die einzelnen Teilbilder, die zu verschiedenen Zeitpunkten aufgenommen wurden, oft als *Frames* bezeichnet.

In der *Bilddatencodierung* werden Techniken angewendet, die die enormen Datenmengen verdichten, ohne dass dabei ein merklicher Qualitätsverlust auftritt. Eine Einführung in diese Thematik wird z.B. in [Heyn03] gegeben.

Für die hier behandelten Verfahren wird oft auf Indizes, die im Rahmen eines Algorithmus nicht benötigt werden, aus Gründen der Übersichtlichkeit verzichtet. Dazu einige Beispiele:

$\mathbf{S} = (s(x, y))$	Statisches Grauwertbild ohne Zeitachse, z.B. ein digitalisiertes Grauwertbild (Halbtonbild).
$\mathbf{S} = (s(x, y, t))$	Dynamisches Grauwertbild mit Zeitachse. Zu den Zeitpunkten $t = 0, 1, ..., T - 1$ fallen die Grauwertbilder $(s(x, y, t))$ an, z.B. digitalisierte Grauwertbildfolgen (Videograutonbilder) mit 50 Halbbildern pro Sekunde.
$\mathbf{S} = (s(x, y, n))$	Statisches, mehrkanaliges Bild ohne Zeitachse, z.B. ein digitalisiertes Farbbild.
$\mathbf{S} = (s(x, y, n, t))$	dynamisches, mehrkanaliges Bild (Bildfolge), z.B. digitalisiertes Videofarbbild mit Rot-, Grün- und Blauauszug, 50 Halbbilder pro Sekunde.

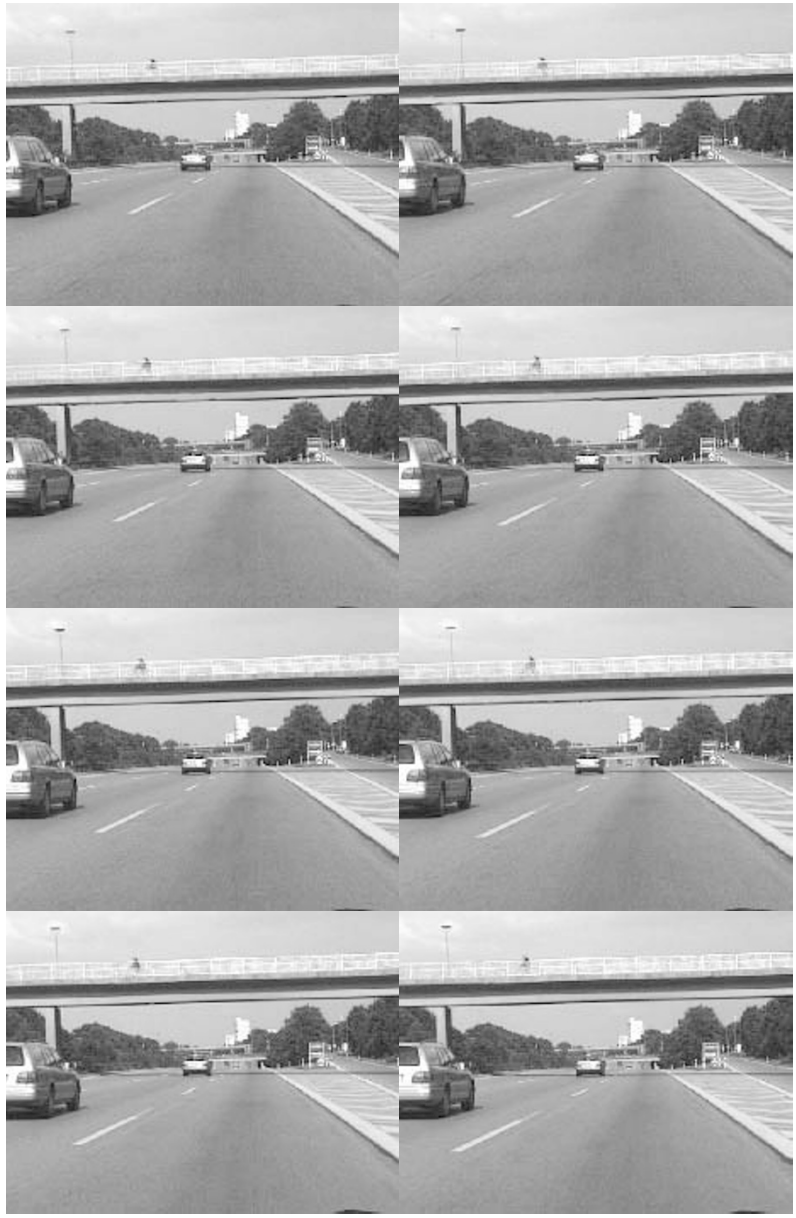


Bild 3.26: Beispiel einer Bildfolge mit acht Teilbildern (Frames).

3.8 Weitere mathematische Modelle für Bilder

In diesem Abschnitt sind weitere mathematische Modelle für Bilder zusammengestellt. Im Rahmen dieses Buches werden weitgehend die Modelle von oben verwendet, in denen digitalisierte Bilder als mehrdimensionale Felder aufgefasst wurden. In der Literatur findet man häufig auch andere Betrachtungsweisen, so z.B. Bilder als Funktionen von zwei reellen oder diskreten Variablen. Auch auf die statistische Beschreibung von Bildern mit Zufallsvariablen wird in diesem Abschnitt eingegangen.

3.8.1 Bilder als relle Funktionen zweier reeller Variablen

Ein Grauwertbild ist ein zweidimensionales Gebilde, bei dem sich die Helligkeit von Punkt zu Punkt ändert. Dieser Sachverhalt kann mathematisch mit reellen Funktionen der beiden (Orts-) Variablen x und y beschrieben werden: $s(x, y)$ ist der Grauwert an der Stelle (x, y). Gewöhnlich wird angenommen, dass sich diese Funktionen ,,analytisch wohlverhalten", so z.B., dass sie integrierbar sind oder dass sie eine Fouriertransformierte besitzen. Weiter wird angenommen, dass die Ortskoordinaten x und y nicht negativ und beschränkt sind. Für die Grauwerte $s(x, y)$ wird angenommen, dass sie beschränkt sind. Damit ergibt sich folgende Darstellung:

$$\begin{aligned} s(x,y) \quad & \text{reelle Funktion der beiden reellen (Orts-)Variablen } x \text{ und } y, \\ & \text{wobei gilt:} \\ & 0 \le x \le L-1, \\ & 0 \le y \le R-1 \text{ und} \\ & g_{min} \le s(x,y) \le g_{max}. \end{aligned} \tag{3.49}$$

Soll in dieser Schreibweise ein Farb- oder Multispektralbild oder ein allgemeines N-kanaliges Bild dargestellt werden, so wird es als Vektor geschrieben:

$$\mathbf{s} = \Big(s_0(x,y), s_1(x,y), ..., s_{N-1}(x,y)\Big)^T. \tag{3.50}$$

Die Funktion $s_n(x, y)$ steht hier für den Kanal n, $n = 0, 1, ..., N-1$.

3.8.2 Bilder als (diskrete) Funktionen zweier diskreter Variablen

Die Digitalisierung (Rasterung des Ortsbereiches und Quantisierung der Grauwerte) bewirkt, dass die Grauwerte des Bildes nur mehr an bestimmten, diskreten Stellen (x, y) vorliegen. Damit ist

$$\begin{aligned} s(x,y) \quad & \text{eine Funktion der beiden diskreten (Orts-)Variablen} \\ & x \text{ und } y \text{ mit} \\ & x = 0, 1, ..., L-1, \\ & y = 0, 1, ..., R-1 \text{ und} \\ & g_{min} \leq s(x,y) \leq g_{max}. \end{aligned} \tag{3.51}$$

In (3.51) wurden die Grauwerte wie im kontinuierlichen Fall (3.49) als beschränkt angenommen. Liegen die Grauwerte ebenfalls quantisiert vor, so tritt an die Stelle des Grauwertintervalls $[g_{min}, g_{max}]$ die Grauwertmenge $G' = \{z_1, ..., z_k\}$.

Ein allgemeines N-kanaliges Bild kann wie in (3.50) als N-dimensionaler Vektor der N diskreten Funktionen $s(x, y)$ geschrieben werden.

Wird $s(x, y)$ nicht als diskrete Funktion zweier Variablen, sondern als Matrix aufgefasst, so ergibt sich der Querbezug zu den mathematischen Modellen in den Abschnitten 3.3 bis 3.7. Dort wurde einschränkend statt der allgemeinen Grauwertmenge G' die spezielle Grauwertmenge $G = \{0, 1, ..., 255\}$ verwendet.

3.8.3 Bilder als Zufallsprozesse

In manchen Anwendungsgebieten, z.B. bei der Übertragung von Bildern, ist es sinnvoll, Bilder nicht *deterministisch* sondern *statistisch* zu beschreiben. Dabei sind verschiedene Betrachtungsweisen möglich. Sie werden im Folgenden kurz erläutert.

Eine Möglichkeit ist die Interpretation *aller* Grauwerte $g = s(x, y)$ eines Bildes als Realisationen *einer* Zufallsvariablen Z, wobei hier die Anordnung der Grauwerte im (x, y)-Koordinatensystem keine Rolle spielt. Diese Zufallsvariable besitzt eine bestimmte Verteilung, die z.B. durch die Verteilungsfunktion beschrieben werden kann. Die Verteilungsfunktion ist definiert als die Wahrscheinlichkeit, dass die Zufallsvariable Z Werte annimmt, die kleiner oder gleich einem Wert z sind:

$$F(z) = p(Z \leq z). \tag{3.52}$$

Dabei kann es sich um eine *stetige* oder *diskrete Verteilung* handeln. Sind als Grauwerte eines Bildes **S** alle reellen Zahlen eines Intervalls zulässig und besitzt $F(z)$ eine stetige Ableitung $f(z)$, so wird **S** mit einer stetigen Zufallsgröße Z beschrieben. Die Funktion $f(z)$ heißt die *Dichte* der Verteilung.

Ein Bild $\mathbf{S} = (s(x,y))$ mit der Grauwertmenge $G = \{0, 1, ..., 255\}$ wird dagegen mit einer diskreten Zufallsvariablen Z beschrieben, die die Grauwerte g_i mit den Wahrscheinlichkeiten p_i, $i = 0, 1, ...255$, annimmt. Dies wird im Folgenden vorausgesetzt. Die *Wahrscheinlichkeitsfunktion* $f(z)$ ist dann definiert als

$$f(z) = \begin{cases} p_i & \text{für } z = g_i, \\ 0 & \text{für alle übrigen } z. \end{cases} \tag{3.53}$$

Das in Abschnitt 3.10 erläuterte Histogramm der relativen Häufigkeiten ist eine Schätzung der Wahrscheinlichkeitsfunktion $f(z)$. Die relative Summenhäufigkeit ist eine Schätzung der Verteilungsfunktion $F(z)$.

Der *Erwartungswert* der Zufallsgröße Z ist:

$$E(Z) = \sum_i g_i p_i \tag{3.54}$$

und die *Streuung* (*Varianz*) von Z ist

$$\sigma^2 = E\Big((Z - E(Z))^2\Big) = \sum_i \Big(g_i - E(Z)\Big)^2 p_i. \tag{3.55}$$

Der Mittelwert in Abschnitt 3.10 ist ein Schätzwert für den Erwartungswert und die mittlere quadratische Abweichung ein (allerdings nicht erwartungstreuer) Schätzwert für die Streuung. Die Größe σ wird als *Standardabweichung* bezeichnet.

Es wird jetzt ein N-kanaliges Bild $\mathbf{S} = (s(x,y,n))$, $n = 0, 1, ..., N-1$ vorausgesetzt. Die Grauwerte des Kanals n werden als Realisationen der Zufallsvariablen Z interpretiert. Das bedeutet, dass $\mathbf{S}$ durch eine N-dimensionale Zufallsgröße

$$Z = \Big(Z_0, Z_1, ..., Z_{N-1}\Big) \tag{3.56}$$

beschrieben wird.

Die *Kovarianz* des Kanals n_1 mit dem Kanal n_2 ist definiert als:

$$V_{n_1,n_2} = E\Big((Z_{n_1} - E(Z_{n_1}))(Z_{n_2} - E(Z_{n_2}))\Big). \tag{3.57}$$

Falls $n_1 = n_2$, ergibt sich die oben erläuterte Streuung. Einen Schätzwert für die Kovarianz bekommt man mit

$$v_{n_1,n_2} = \frac{1}{M} \sum_{x=1}^{L-1} \sum_{y=1}^{R-1} \Big(s(x,y,n_1) - m_{\mathbf{S},n_1}\Big)\Big(s(x,y,n_2) - m_{\mathbf{S},n_2}\Big). \tag{3.58}$$

Dieser Ausdruck kann ähnlich wie bei der einfacheren Berechnung der mittleren quadratischen Abweichung umgeformt werden (Abschnitt 3.10):

$$v_{n_1,n_2} = \frac{1}{M} \sum_{x=1}^{L-1} \sum_{y=1}^{R-1} s(x,y,n_1)s(x,y,n_2) - m_{\mathbf{S},n_1} m_{\mathbf{S},n_2}. \tag{3.59}$$

Die Kovarianz kann für jedes Paar (i, j), $i, j \in \{0, 1, ..., N-1\}$ von Kanälen des Bildes $\mathbf{S}$ berechnet werden. Man erhält damit die *Kovarianzmatrix* $\mathbf{C}$:

$$\mathbf{C} = (v_{i,j}) = \begin{pmatrix} v_{0,0} & v_{0,1} & ... & v_{0,N-1} \\ v_{1,0} & v_{1,1} & ... & v_{1,N-1} \\ ... & ... & ... & ... \\ v_{N-1,0} & v_{N-1,1} & ... & v_{N-1,N-1} \end{pmatrix}. \tag{3.60}$$

Im Weiteren wird die Schätzung $\mathbf{C}$ vereinfachend als Kovarianzmatrix bezeichnet. $\mathbf{C}$ ist wegen $v_{i,j} = v_{j,i}$ symmetrisch zur Hauptdiagonale.

Werden die Elemente der Hauptdiagonale der Kovarianzmatrix zu 1 normiert, so wird daraus die *Matrix der Korrelationskoeffizienten*:

$$\mathbf{K} = (r_{i,j}) = \left(\frac{v_{i,j}}{\sqrt{v_{i,i} v_{j,j}}} \right). \tag{3.61}$$

Die Korrelationskoeffizienten liegen im Intervall $-1 \leq r_{i,j} \leq +1$. Die Zufallsvariablen Z_i und Z_j sind *unkorreliert*, wenn $r_{i,j} = 0$ ist.

Nun zu einer etwas anderen Betrachtungsweise. Hier wird auch die Position (x, y) eines Bildpunktes berücksichtigt. Dazu ein beispielhafter Versuchsaufbau: Ein Original wird mit einer Videokamera aufgezeichnet und über einen Videodigitalisierer (*framegrabber*) in einen Rasterbildspeicher geschrieben. Die Videokamera tastet das Original mit einer Bildwiederholungsrate von 50 Halbbildern pro Sekunde ab, so dass der *framegrabber* 50 Halbbilder pro Sekunde in den Rasterbildspeicher schreibt.

Der Grauwert $s(x, y)$ des Bildpunktes in der Position (x, y) im Rasterbildspeicher ist die codierte mittlere Helligkeit eines kleinen Flächenausschnittes des Originals. Dieser an sich konstante Grauwert wird durch verschiedene Quellen verrauscht, z.B. durch die atmosphärischen Bedingungen zwischen Originalbild und Videokamera, durch elektronisches Rauschen in der Videokamera oder durch Quantisierungsrauschen im *Framegrabber*. Das hat zur Folge, dass der Grauwert $s(x, y)$ nicht konstant ist, sondern um einen bestimmten Mittelwert schwankt.

Das Verhalten des Grauwertes in der Position (x, y) wird durch die Zufallsvariable $Z_{T,(x,y)}$ beschrieben. Der zusätzliche Index T steht für den Zufallsversuch T mit den Ereignissen $T = t_0, t_1, ...$. Im vorliegenden Beispiel kann dazu folgende Interpretation gegeben werden: Zum Zeitpunkt t_i tritt das Ereignis „Bild zum Zeitpunkt t_i“ ein, das durch die diskrete Funktion $Z_{t_i,(x,y)}$ beschrieben wird. Damit sind unterschiedliche Interpretationen möglich:

- Für einen festen Wert t ist $Z_{T,(x,y)}$ eine Bildfunktion der beiden diskreten (Orts-)Variablen x und y. Für $i = 0, 1, 2, ...$ ist somit eine Menge von Bildfunktionen gegeben.
- Für einen festen Punkt (x, y) und für variables t ist $Z_{T,(x,y)}$ eine Zufallsvariable für den Bildpunkt in der Position (x, y).

Unter diesen Voraussetzungen wird $Z_{T,(x,y)}$ als *zweidimensionaler Zufallsprozess* oder *stochastischer Prozess* (*random field*) bezeichnet. Damit können statistische Kenngrößen für den Zufallsprozess $Z_{T,(x,y)}$ formuliert werden:

- Der *Erwartungswert* oder *Mittelwert* des Zufallsprozesses für den Punkt (x, y):

$$E\Big(Z_{T,(x,y)}\Big). \tag{3.62}$$

- Die *Streuung* (*Varianz*) des Zufallsprozesses für den Punkt (x, y):

$$E\Big((Z_{T,(x,y)} - E(Z_{T,(x,y)}))^2\Big). \tag{3.63}$$

- Die *Autokorrelation* des Zufallsprozesses als Erwartungswert des Produktes der beiden Zufallsvariablen $Z_{T,(x_1,y_1)}$ und $Z_{T,(x_2,y_2)}$:

$$K_{(x_1,y_1),(x_2,y_2)} = E\Big(Z_{T,(x_1,y_1)} Z_{T,(x_2,y_2)}\Big). \tag{3.64}$$

- Die *Autokovarianz* des Zufallsprozesses

$$\begin{aligned} V_{(x_1,y_1),(x_2,y_2)} &= \\ &= E\Big((Z_{T,(x_1,y_1)} - E(Z_{T,(x_1,y_1)}))(Z_{T,(x_2,y_2)} - E(Z_{T,(x_2,y_2)}))\Big) = \\ &= K_{(x_1,y_1),(x_2,y_2)} - E\Big(Z_{T,(x_1,y_1)}\Big) E\Big(Z_{T,(x_2,y_2)}\Big). \end{aligned} \tag{3.65}$$

Oft wird für den Erwartungswert $E(Z_{T,(x,y)}) = 0$ angenommen. Dann kann ein Zufallsprozess allein durch seine Autokorrelation $K_{(x_1,y_1),(x_2,y_2)}$ beschrieben werden. Wird die oben beschriebene Versuchsanordnung so gewählt, dass anstelle des Originalbildes, das während des gesamten Versuchs konstant blieb, alle möglichen zweidimensionalen Grauwertverteilungen gleich wahrscheinlich auftreten können, so wird der Mittelwert $E(Z_{T,(x,y)})$ unabhängig von der Position (x, y) und die Autokorrelation verschiebungsinvariant:

$$K_{(x_1,y_1),(x_2,y_2)} = K_{(x_1+\Delta x,y_1+\Delta y),(x_2+\Delta x,y_2+\Delta y)}. \tag{3.66}$$

Setzt man für $(\Delta x, \Delta y) = (-x_2, -y_2)$ so ergibt sich

$$K_{(x_1,y_1),(x_2,y_2)} = K_{(x_1-x_2,y_1-y_2),(0,0)} = K_{(\alpha,\beta)}, \tag{3.67}$$

wobei α und β die Koordinatendifferenzen $x_1 - x_2$ und $y_1 - y_2$ sind.

Ein Zufallsprozess dieser Art heißt *homogen*. Er kann durch seine Autokorrelation beschrieben werden, die statt von einem Punktepaar (x_1, y_1) und (x_2, y_2) nur von der Koordinatendifferenz von Punktepaaren abhängt. Bei vielen Anwendungen werden in der Literatur Bilder als homogene Zufallsprozesse vorausgesetzt. Schätzungen dieser Kenngrößen berechnen sich sinngemäß wie oben. Ein Schätzwert für den Mittelwert:

$$m_{(x,y)} = \frac{1}{T} \sum_{t=0}^{T-1} s(x, y, t), \tag{3.68}$$

ein Schätzwert für die Streuung:

$$q_{(x,y)} = \frac{1}{T} \sum_{t=0}^{T-1} s(x, y, t)^2 - m_{(x,y)}^2; \tag{3.69}$$

und ein Schätzwert für die Kovarianz:

$$v_{(x_1,y_1),(x_2,y_2)} = \frac{1}{T} \sum_{t=0}^{T-1} s(x_1, y_1, t) s(x_2, y_2, t) - m_{(x_1,y_1)} m_{(x_2,y_2)}. \tag{3.70}$$

Die Kovarianz wird für alle Paare von Bildpunkten (x_k, y_k), (x_l, y_l) berechnet. Die Kovarianzmatrix ist dann die Matrix der Schätzwerte und hat die Dimension $M \cdot M$, wobei $M = L \cdot R$ die Anzahl der Bildpunkte von $\mathbf{S}$ ist.

3.9 Datenreduktion und Datenkompression

Da digitalisierte Bilddaten sehr viel Speicherplatz benötigen, wird oft eine Reduzierung des Speicherbedarfs notwendig. Zunächst kann man auf digitalisierte Bilddaten, die aus der Sicht des Betriebssystems nichts anderes als in Dateien gespeicherte Daten sind, alle Verfahren anwenden, die in der PC-Welt zur Datenkompression verwendet werden (z.B. ZIP oder RAR Verfahren zur verlustfreien Kompression von Daten).

Andrerseits gibt es viele Verfahren, die speziell die Tatsache berücksichtigen, dass es sich um Bilddaten handelt. Zunächst aber für diesen Bereich zwei Begriffserläuterungen: Bilddatenreduktion und Bilddatenkompression.

Bei der *Bilddatenreduktion* werden von den Originalbilddaten Bestandteile weggelassen, die im speziellen Anwendungsfall nicht oder nur gering relevant sind. Aus der reduzierten Darstellung kann das Original nicht mehr rekonstruiert werden. Ein Beispiel ist die Reduzierung der Grauwertmenge von 256 auf z.B. 16 Graustufen.

Aus Bildern, die mit Verfahren der *Bilddatenkompression* verarbeitet wurden, kann das Original wieder eindeutig und fehlerfrei rekonstruiert werden. Als Beispiel hierzu kann die *run-length*-Codierung (Kapitel 23) von Binärbildern dienen.

Eine weitere Beurteilungsmöglichkeit der Eignung von Bilddatenreduktions- und -kompressionsverfahren bei einer bestimmten Anwendung ist die Frage, ob es die reduzierte oder

komprimierte Form erlaubt, Bildverarbeitungs- und Mustererkennungsverfahren darauf anzuwenden. Wird ein Bild mit einer Archivierungssoftware komprimiert, so ist es sicher nicht möglich, auf die komprimierten Daten einen Bildverarbeitungsoperator, wie beispielsweise den Laplace-Operator, anzuwenden. Man muss die Bilddaten zuerst dekomprimieren, dann den Bildverarbeitungsoperator anwenden und dann das Ergebnis bei Bedarf wieder komprimieren.

Anders ist es als Beispiel bei der *run-length*-Codierung: Hier ist es möglich, mit den komprimierten Bilddaten viele sinnvolle Verarbeitungsschritte durchzuführen. Die Motivation für den Einsatz der *run-length*-Codierung kann somit nicht nur der Gesichtspunkt der Datenkompression, sondern auch die effiziente Verwendung mancher Bildverarbeitungsoperatoren sein.

Wichtige Verfahren zur Bilddatenreduktion wurden von der *Joint Photografic Experts Group*, JPEG, vorgeschlagen und standardisiert. Diese Verfahren bauen auf einer blockweisen Cosinustransformation auf. Für Bildfolgen wurde das MPEG-Format („Moving Picture Experts Group“) definiert. An dieser Stelle sei nochmals auf das schon zitierte Buch [Heyn03] verwiesen.

Die „Portable Video Research Group“, PRVG, an der Stanford University hat zu diesen Formaten das Softwarepaket *PRVG-JPEG/MPEG codec* entwickelt, das im Internet zusammen mit Handbüchern verfügbar ist.

3.10 Charakterisierung digitalisierter Bilder

In diesem Abschnitt sind wichtige Eigenschaften von digitalisierten Bilddaten und Kenngrößen, die diese Eigenschaften beschreiben, zusammengestellt.

3.10.1 Mittelwert und mittlere quadratische Abweichung

Es sei $\mathbf{S} = (s(x, y))$ ein einkanaliges Grauwertbild mit L Zeilen und R Spalten. Der *mittlere Grauwert* des Bildes $\mathbf{S}$, auch *Mittelwert* von $\mathbf{S}$ genannt, berechnet sich gemäß:

$$m_{\mathbf{S}} = \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s(x, y), \tag{3.71}$$

wobei $M = L \cdot R$ die Anzahl der Bildpunkte von $\mathbf{S}$ ist. Als Beispiel ist der Mittelwert von Bild 3.27-a $m_{\mathbf{S}} = 129.26$.

Der Mittelwert eines Bildes sagt aus, ob das Bild insgesamt dunkler oder heller ist. Eine Aussage über den Kontrast lässt sich aus dem Mittelwert nicht ableiten. So haben z.B. ein Bild, das nur den Grauwert 127 enthält (Bildgröße beliebig) und ein Bild, das ein Schachbrettmuster mit den Grauwerten 0 und 254 enthält, denselben Mittelwert 127.

Eine Kenngröße, die eine Aussage über den Kontrast im Bild zulässt, ist die *mittlere quadratische Abweichung*:

$$q_{\mathbf{S}} = \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} \Big(s(x, y) - m_{\mathbf{S}}\Big)^2. \tag{3.72}$$

Bild 3.27-a hat eine mittlere quadratische Abweichung $q_{\mathbf{S}} = 4792.79$. Das oben erwähnte homogene Bild mit dem Grauwert 127 hat die mittlere quadratische Abweichung $q_{\mathbf{S}} = 0$, während sich für das Schachbrettmusterbild $q_{\mathbf{S}} = 127^2 = 16129$ ergibt.

Zur einfacheren Berechnung der mittleren quadratischen Abweichung lässt sich (3.72) durch die Anwendung der binomischen Formel umformen:

$$
\begin{aligned}
q_{\mathbf{S}} &= \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} \left(s^2(x,y) - 2m_{\mathbf{S}} s(x,y) + m_{\mathbf{S}}^2 \right) = \\
&= \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s^2(x,y) - \frac{2m_{\mathbf{S}}}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s(x,y) + \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} m_{\mathbf{S}}^2 = \\
&= \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s^2(x,y) - m_{\mathbf{S}}^2.
\end{aligned}
\tag{3.73}
$$

Mit (3.71) und (3.73) lassen sich der Mittelwert und die mittlere quadratische Abweichung eines Bildes in einem Durchgang berechnen.

Bei mehrkanaligen Bildern oder Bildfolgen können Mittelwert und mittlere quadratische Abweichung für jeden Kanal oder für jedes Teilbild getrennt berechnet werden. Für ein mehrkanaliges Bild $\mathbf{S} = (s(x,y,n)), \quad n = 0,1,...,N-1$ wird dies im Folgenden dargestellt. Die Vektoren

$$
\begin{aligned}
&\mathbf{m_S} = \left(m_{\mathbf{S},0}, m_{\mathbf{S},1}, ..., m_{\mathbf{S},N-1} \right)^T \quad \text{und} \\
&\mathbf{q_S} = \left(q_{\mathbf{S},0}, q_{\mathbf{S},1}, ..., q_{\mathbf{S},N-1} \right)^T
\end{aligned}
$$

werden dann als *Mittelwertvektor* und *Vektor der mittleren quadratischen Abweichungen* des mehrkanaligen Bildes $\mathbf{S}$ bezeichnet.

3.10.2 Histogramme

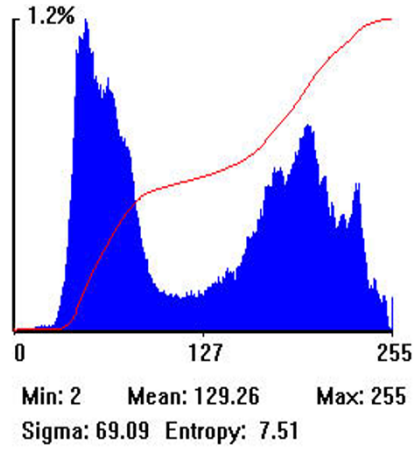
Mittelwert und mittlere quadratische Abweichung sind einfache Maßzahlen zur Charakterisierung der Verteilung der Grauwerte eines Bildes $\mathbf{S} = (s(x,y))$. Mehr Aufschluss über die Verteilung der Grauwerte gibt das *Histogramm der relativen Häufigkeiten* von $\mathbf{S}$:

$$
p_{\mathbf{S}}(g) = \frac{a_g}{M}, \quad g = 0,1,...,255. \tag{3.74}
$$

Dabei sind g die Grauwerte der Grauwertmenge G und a_g die Häufigkeit des Auftretens des Grauwertes g in $\mathbf{S}$. Da das Histogramm über die Anzahl M der Bildpunkte von $\mathbf{S}$ normiert ist, gilt:

$$
\sum_{g=0}^{255} p_{\mathbf{S}}(g) = 1, \text{ also auch } 0 \le p_{\mathbf{S}}(g) \le 1. \tag{3.75}
$$

(a)



(b)

Bild 3.27: (a) Testbild zur Veranschaulichung der in diesem Abschnitt behandelten Kenngrößen für digitalisierte Bilder. Das Bild hat einen Mittelwert $m_{\mathbf{S}} = 129.26$ und eine mittlere quadratische Abweichung $q_{\mathbf{S}} = 4792.79$. (b) Histogramm: Auf der Abszisse werden die Grauwerte g von 0 bis 255 aufgetragen. Zu jedem Grauwert g wird auf der Ordinate ein Balken gezeichnet, der in der Höhe dem Wert $p_{\mathbf{S}}(g)$ in % entspricht. Da die Werte alle sehr klein sind, wurde die Ordinate entsprechend skaliert. Die monoton steigende Kurve repräsentiert die relativen Summenhäufigkeiten. Sie liegen zwischen 0 und 1. Für diese Kurve gilt eine andere Skalierung der Ordinate.

0 - 15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01
16 - 31	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.02	0.01	0.01	0.02	0.03	0.05	0.07	0.09	0.13
32 - 47	0.16	0.18	0.24	0.31	0.42	0.48	0.59	0.72	0.88	0.99	1.05	1.04	1.10	1.09	1.09	1.12
48 - 63	1.17	1.16	1.13	1.06	1.04	1.04	0.94	0.95	0.93	0.90	0.92	0.86	0.90	0.89	0.92	0.95
64 - 79	0.90	0.90	0.89	0.87	0.83	0.79	0.75	0.73	0.73	0.71	0.72	0.69	0.68	0.68	0.65	0.59
80 - 95	0.56	0.49	0.45	0.42	0.39	0.37	0.34	0.31	0.29	0.26	0.26	0.21	0.20	0.19	0.17	0.17
96 - 111	0.16	0.15	0.14	0.14	0.14	0.15	0.13	0.13	0.13	0.13	0.13	0.12	0.13	0.14	0.13	0.12
112 - 127	0.11	0.12	0.12	0.12	0.14	0.13	0.13	0.15	0.13	0.13	0.12	0.12	0.14	0.13	0.14	0.00
128 - 143	0.13	0.15	0.15	0.16	0.16	0.16	0.17	0.17	0.18	0.17	0.20	0.18	0.17	0.20	0.19	0.22
144 - 159	0.23	0.21	0.22	0.21	0.22	0.19	0.23	0.23	0.23	0.25	0.28	0.28	0.31	0.35	0.35	0.34
160 - 175	0.35	0.37	0.36	0.37	0.41	0.41	0.45	0.43	0.48	0.51	0.54	0.53	0.54	0.54	0.54	0.59
176 - 191	0.61	0.59	0.60	0.60	0.61	0.59	0.57	0.54	0.56	0.58	0.58	0.60	0.62	0.63	0.66	0.66
192 - 207	0.68	0.72	0.72	0.72	0.76	0.76	0.77	0.76	0.76	0.74	0.74	0.67	0.64	0.62	0.55	0.56
208 - 223	0.57	0.57	0.57	0.52	0.46	0.43	0.42	0.48	0.47	0.42	0.38	0.39	0.40	0.43	0.43	0.43
224 - 239	0.41	0.38	0.40	0.42	0.43	0.49	0.52	0.55	0.52	0.55	0.47	0.38	0.34	0.29	0.25	0.19
240 - 255	0.16	0.17	0.15	0.19	0.19	0.15	0.13	0.11	0.10	0.13	0.13	0.11	0.05	0.01	0.00	0.12

Tabelle 3.2: Numerische Ausgabe der relativen Häufigkeiten in % zu Testbild 3.27-a

Bild 3.27-b zeigt das Histogramm von Bild 3.27-a. Auf der Abszisse werden die Grauwerte g und auf der Ordinate die dazugehörigen relativen Häufigkeiten $p_{\mathbf{S}}(g)$ in % aufgetragen. Tabelle 3.2 enthält das Histogramm in tabellarischer Form.

Der Mittelwert und die mittlere quadratische Abweichung können leicht aus dem Histogramm berechnet werden:

$$m_{\mathbf{S}} = \frac{1}{M} \sum_{g=0}^{255} g M p_{\mathbf{S}}(g) = \sum_{g=0}^{255} g p_{\mathbf{S}}(g) \tag{3.76}$$

und

$$q_{\mathbf{S}} = \frac{1}{M} \sum_{g=0}^{255} (g - m_{\mathbf{S}})^2 M p_{\mathbf{S}}(g) = \sum_{g=0}^{255} (g - m_{\mathbf{S}})^2 p_{\mathbf{S}}(g). \tag{3.77}$$

Zur Erläuterung der Bedeutung des Histogramms eines Bildes werden im Folgenden einige Beispiele gegeben. Für ein homogenes Bild $\mathbf{S} = (s(x,y)) = (g_k)$ ergibt sich trivialerweise ein Histogramm

$$p_{\mathbf{S}}(g) = \begin{cases} 1 & \text{falls } g = g_k, \\ 0 & \text{sonst.} \end{cases} \tag{3.78}$$

Ein Zweipegelbild $\mathbf{S} = (s(x,y))$, in dem nur die beiden Grauwerte $s(x,y) = g_k$ oder $s(x,y) = g_l$ auftreten, hat folgendes Histogramm:

$$p_{\mathbf{S}}(g) = \begin{cases} p_{\mathbf{S}}(g_k) & \text{falls } g = g_k, \\ p_{\mathbf{S}}(g_l) = 1 - p_{\mathbf{S}}(g_k) & \text{falls } g = g_l, \\ 0 & \text{sonst.} \end{cases} \tag{3.79}$$

Diese Form des Histogramms haben auch Binärbilder, die ein Sonderfall eines Zweipegelbildes sind.

Bei einem dunklen Bild mit wenig Kontrast sind vor allem die relativen Häufigkeiten $p_{\mathbf{S}}(g)$ für kleine Werte von g besetzt, während bei einem hellen Bild mit wenig Kontrast vor allem die großen Werte von g hohe relative Häufigkeiten aufweisen. Das Histogramm eines kontrastreichen Bildes zeichnet sich dadurch aus, dass im Idealfall alle relativen Häufigkeiten $p_{\mathbf{S}}(g) = 1/256$ sind. Das bedeutet, dass alle Grauwerte gleich oft auftreten. Ein Bild, das vorwiegend einen dunklen und einen hellen Bereich enthält, erzeugt ein Histogramm mit zwei lokalen Maxima. Histogramme dieser Art heißen *bimodal*. Abschließend sei noch darauf hingewiesen, dass aus dem Histogramm nicht auf die örtliche Anordnung der Grauwerte in der Bildmatrix $\mathbf{S}$ geschlossen werden kann. Ein Bild mit zwei deutlich abgegrenzten dunklen und hellen Bildbereichen kann dasselbe bimodale Histogramm erzeugen wie ein Bild, in dem die gleichen Grauwerte zufällig verteilt sind. Werden die relativen Häufigkeiten $p_{\mathbf{S}}(g)$ aufsummiert, so erhält man die *relativen Summenhäufigkeiten*:

$$h_{\mathbf{S}}(g) = \sum_{k=0}^{g} p_{\mathbf{S}}(k), \quad g = 0, 1, ..., 255. \tag{3.80}$$

Wegen

$$\sum_{g=0}^{255} p_{\mathbf{S}}(g) = 1 \text{ gilt: } 0 \leq h_{\mathbf{S}}(g) \leq 1. \tag{3.81}$$

Die relative Summenhäufigkeit wird z.B. bei der Modifikation des Histogramms zur Verbesserung des Kontrastes verwendet (Kapitel 4).

Die Berechnung des Histogramms ist nicht auf einkanalige Grauwertbilder beschränkt. In der multivariaten Klassifizierung (Kapitel 20) spielen N-dimensionale Histogramme eine wichtige Rolle. Das N-dimensionale Histogramm des N-kanaligen Bildes $\mathbf{S} = (s(x,y,n)), n = 0, 1, ..., N-1$ ist definiert als

$$p_{\mathbf{S}}(g_0, g_1, ..., g_{N-1}) = \frac{a_{g_0 g_1 ... g_{N-1}}}{M}, \tag{3.82}$$

wobei $a_{g_0,g_1,...,g_{N-1}}$ die Häufigkeit der Grauwertkombination mit dem Grauwert g_n im Kanal n ist und M die Anzahl der Bildpunkte.

Im zweidimensionalen Fall kann das Histogramm bildlich dargestellt werden: Die relative Häufigkeit $p_{\mathbf{S}}(g_0, g_1)$ wird als Grauwert codiert in ein Bild $\mathbf{S}_a$ der Größe $256 \cdot 256$ Bildpunkte in der Position $(x,y) = (g_0, g_1)$ aufgetragen, also:

$$s_a(x,y) = s_a(g_0, g_1) = c \cdot p_{\mathbf{S}}(g_0, g_1), \tag{3.83}$$

wobei c ein Skalierungsfaktor ist. Bild 3.28 ist ein Beispiel einer Darstellung dieser Art. Aus der speziellen Form läßt sich ablesen, dass die Farbkanäle stark korreliert sind.

Aus dem N-dimensionalen Histogramm können die eindimensionalen Histogramme der N Kanäle abgeleitet werden. Im zweidimensionalen Fall $p_{\mathbf{S}}(g_0, g_1)$ gilt z.B.:

$$p_{\mathbf{S},\mathbf{0}}(g) = \sum_{k=0}^{255} p_{\mathbf{S}}(g,k) \tag{3.84}$$

und

$$p_{\mathbf{S},\mathbf{1}}(g) = \sum_{l=0}^{255} p_{\mathbf{S}}(l,g), \quad g = 0,1,...,255. \tag{3.85}$$

Die Umkehrung gilt natürlich nicht: Aus den N eindimensionalen Histogrammen kann das N-dimensionale Histogramm nicht abgeleitet werden.

3.10.3 Entropie

Bei Fragestellungen, wie man die Datenmenge eines digitalisierten Bildes reduzieren kann, benötigt man ein Maß für den mittleren Informationsgehalt eines Bildes. Eine Maßzahl dafür ist die *Entropie*. Sie berechnet sich für ein einkanaliges Grauwertbild $\mathbf{S} = (s(x,y)), G = \{0,1,...,255\}$ mit dem Histogramm $p_{\mathbf{S}}(g)$ zu

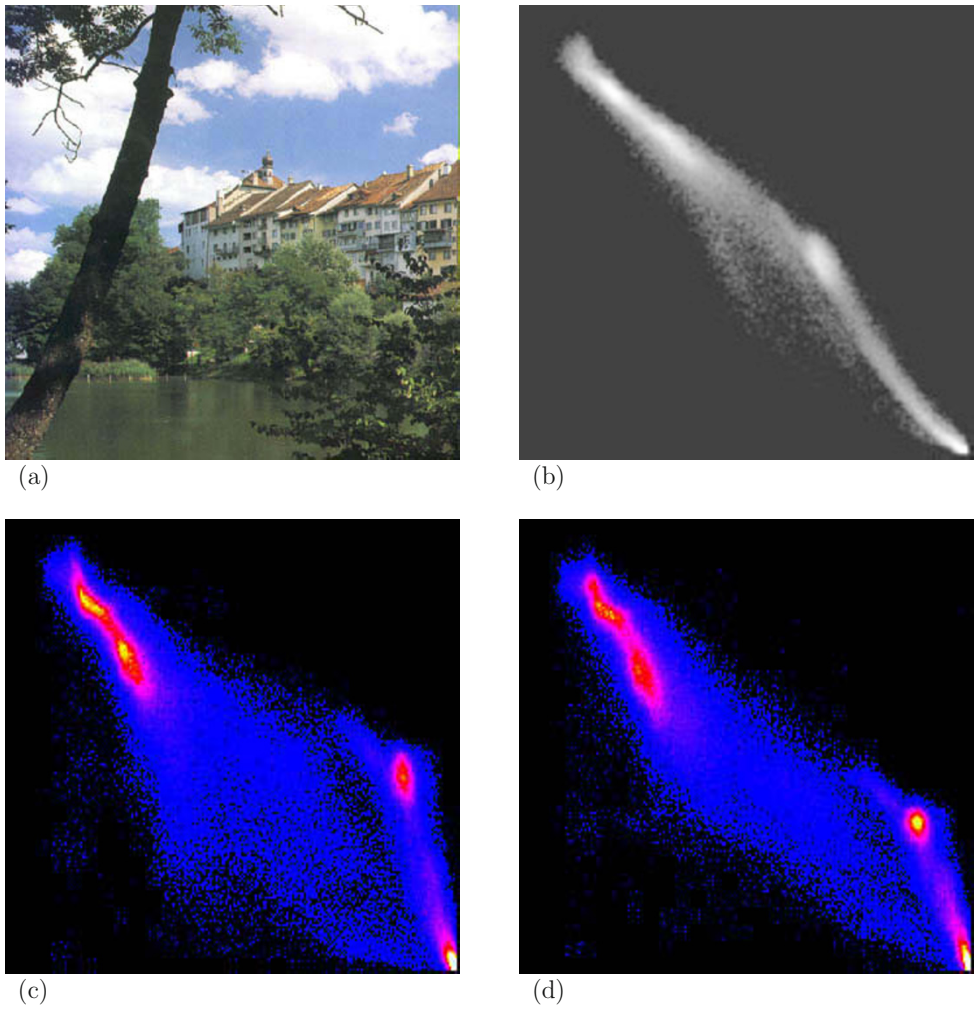
$$H = -\sum_{g=0}^{255} \Big(p_{\mathbf{S}}(g) \cdot log_2 p_{\mathbf{S}}(g) \Big). \tag{3.86}$$

Die Entropie kann auch als ein Maß für die mittlere apriori-Unsicherheit pro Bildpunkt oder die gemittelte Anzahl der notwendigen Bit pro Bildpunkt interpretiert werden. Ein Bild mit der Entropie H, bei dem die Grauwerte der Bildpunkte gleichverteilt sind, kann ohne Informationsverlust nicht mit weniger als H Bit pro Bildpunkt codiert werden. In der Praxis kann diese Annahme allerdings selten vorausgesetzt werden. Durch den Bildinhalt werden Korrelationen zwischen benachbarten Bildpunkten hergestellt, die nicht vernachlässigbar sind. So ist die Entropie hier nur eine Abschätzung für den extremen Fall.

Dazu noch einige Beispiele. Ein homogenes Bild $\mathbf{S} = (s(x,y)) = g$ hat die Entropie $H = 0$, während sich für ein Zweipegelbild mit $p_{\mathbf{S}}(g_1) = 0.5$ und $p_{\mathbf{S}}(g_2) = 0.5$ die Entropie $H = 1$ berechnet. Ein Bild, in dem alle Grauwerte mit derselben relativen Häufigkeit $p_{\mathbf{S}}(g) = 1/256$ auftreten, hat die Entropie $H = 8$. Das bedeutet, dass die Grauwerte der Bildpunkte mit 8 Bit codiert werden müssen. Die Entropie von Bild 3.27-a ist $H = 7.52$.

Ein aus der Entropie abgeleitetes Maß für die Symmetrie des Histogramms ist der *Anisotropiekoeffizient*:

$$\alpha = \frac{-\sum_{g=0}^{k} \Big(p_{\mathbf{S}}(g) \cdot log_2 p_{\mathbf{S}}(g) \Big)}{H}. \tag{3.87}$$



(a) (b)

(c) (d)

Bild 3.28: Mehrdimensionale Histogramme. (a) Farbtestbild (RGB-Bild). (b) Bildliche Darstellung des zweidimensionalen Histogramms des Rot- und des Grünauszugs. Nach unten sind die Grauwerte des ersten Kanals (Rotauszug) und nach rechts die Grauwerte des zweiten Kanals (Grünauszug) aufgetragen. Die relativen Häufigkeiten werden als Grauwert codiert. Die spezielle Form zeigt, dass die beiden Kanäle korreliert sind. (c) Zweidimensionales Histogramm: Rot-/Blauauszug. (d) Zweidimensionales Histogramm: Grün-/Blauauszug. Hinweis: bei den Histogrammen (c) und (d) wurden die relativen Häufigkeiten pseudofarbcodiert, um für den menschlichen Betrachter die Unterscheidbarkeit zu erhöhen (die relativen Häufigkeiten steigen von schwarz über blau zu rot, gelb und weiß an).

Hier ist k der kleinste mögliche Grauwert aus G, für den gilt:

$$\sum_{g=0}^{k} p_{\mathbf{S}}(g) \geq 0.5. \tag{3.88}$$

Symmetrische Histogramme haben ein α von 0.5. Die Abweichung von diesem Wert ist ein Hinweis auf zunehmende Asymmetrie. Der Anisotropiekoeffizient kann z.B. bei der Untersuchung, ob ein Histogramm bimodal ist oder nicht, verwendet werden.

3.10.4 Grauwertematrix (*co-occurrence*-Matrix)

Ein weiteres wichtiges Hilfsmittel zur Beschreibung von Bildeigenschaften ist die *Grauwertematrix* (*Grauwertübergangsmatrix*, *co-occurrence*-Matrix). Sie darf nicht verwechselt werden mit der Bildmatrix $\mathbf{S} = (s(x, y))$ eines digitalisierten Bildes.

Zur Definition der Grauwertematrix wird zunächst eine Relation zwischen Paaren von Bildpunktpositionen (x_1, y_1) und (x_2, y_2) festgelegt.

Beispiele:

(a) $(x_1, y_1)\rho_1(x_2, y_2)$ falls (x_2, y_2) der rechte Nachbar von (x_1, y_1) ist.

(b) $(x_1, y_1)\rho_2(x_2, y_2)$ falls gilt: $(x_2, y_2) = (x_1 + \Delta x, y_1 + \Delta y)$. (x_2, y_2) ist hier der Nachbar von (x_1, y_1) bezüglich des Verschiebungsvektors $(\Delta x, \Delta y)$.

Die Grauwertematrix von $\mathbf{S} = (s(x, y)), G = \{0, 1, 2, ..., 255\}$ bezüglich der Relation ist dann die Matrix

$$\mathbf{W}_{\mathbf{S},\rho}(g_1, g_2) = \Big(a_{g_1,g_2}\Big), \tag{3.89}$$

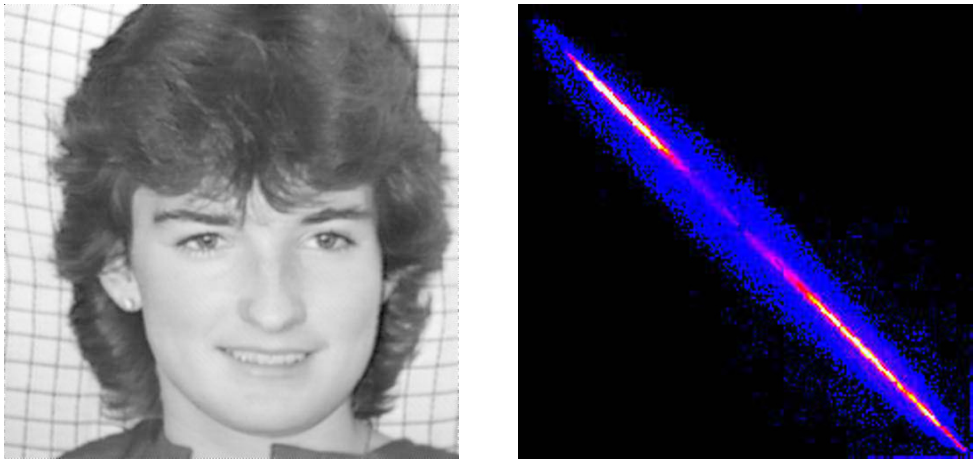
wobei a_{g_1,g_2} die Häufigkeit der Grauwertkombination $(g_1, g_2) = \Big(s(x_1, y_1), s(x_2, y_2)\Big)$ bezüglich der Relation ρ ist.

Dazu ein Beispiel. Das Bild $\mathbf{S}$ mit $G = \{0, 1, 2, 3\}$ sei gegeben durch

0	0	1	1	2	3
0	0	0	1	2	3
0	0	1	2	3	3
0	1	1	2	3	3
1	2	2	3	3	3
2	2	3	3	3	3

Als Relation wird $\rho = \rho_1$ („rechter Nachbar“) verwendet. Es ergibt sich dann die Grauwertematrix

$$\mathbf{W}_{\mathbf{S},\rho}(g_1, g_2) = \begin{pmatrix} 4 & 4 & 0 & 0 \\ 0 & 2 & 5 & 0 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 0 & 7 \end{pmatrix}.$$



(a) (b)

Bild 3.29: (a) Testbild mit 256 Graustufen. (b) Bildliche Darstellung der Grauwertematrix (*co-occurrence*-Matrix) des Testbildes bezüglich der Relation „rechter Nachbar“.

Die Grauwertematrix wird vor allem bei der Bildsegmentierung mit der Oberflächenstruktur (Textur) als Merkmal verwendet. Die Elemente der Hauptdiagonalen der Grauwertematrix repräsentieren die ungefähre Größe der homogenen Bildbereiche. Das Element in der Position $(g_1, g_2), g_1 \neq g_2$, gibt die ungefähre Länge der Grenze zwischen den Bereichen mit dem Grauwert g_1 und dem Grauwert g_2 wieder. Bei einem Bild mit wenig Kontrast wird der Bereich um die Hauptdiagonale der Grauwertematrix stark besetzt sein, während bei einem Bild mit viel Kontrast die linke untere Ecke und die rechte obere Ecke stark besetzt sein werden. Bild 3.29-b zeigt eine bildliche Darstellung der Grauwertematrix zu Bild 3.29-a (Relation ρ: „rechter Nachbar“).

Kapitel 4

Modifikation der Grauwerte

In diesem und allen folgenden Kapiteln werden Verfahren erläutert, die letztlich auf eine, dem jeweiligen Anwendungsfall angepasste Interpretation des Bildinhaltes hinauslaufen. Die Bildinterpretation kann visuell durch einen Bearbeiter erfolgen. Dann werden Verarbeitungstechniken eingesetzt, die Bildinformationen, die bei einer speziellen Anwendung redundant oder sogar störend sind, unterdrücken und dafür wichtige Bildinhalte deutlicher hervorheben. Andererseits können Bilder auch automatisch oder zumindest teilautomatisch interpretiert werden. Auf Verfahren dieser Art wird ab Kapitel 11 näher eingegangen. Zunächst werden einfache Methoden der Bildverbesserung durch Analyse und Modifikation der Grauwertverteilung eines Bildes untersucht. Diese Verfahren dienen der besseren visuellen Interpretierbarkeit. Sie können aber auch Vorverarbeitungsschritte für nachfolgende Bildsegmentierung und Bildinterpretation sein (ab Kapitel 11).

4.1 Anwendungen

Häufig werden Bilder mit Hilfe eines Videodigitalisierers eingezogen und auf dem Monitor des Bildverarbeitungssystems dargestellt. Durch eine interaktive Veränderung der Grauwerte kann hier oft eine bessere bildliche Reproduktion erzielt werden. Die Grauwertmodifikation kann auch über im System gespeicherte *look-up*-Tabellen, die für bestimmte Anwendungsfälle vorgesehen sind, erfolgen. Auch eine dynamische Veränderung der dargestellten Grauwerte kann mit entsprechend schnellen Bildverarbeitungssystemen in Videoechtzeit durchgeführt werden.

Ein anderer Problemkreis ist die Ausgabe von Grauwertbildern über einfache Schwarz-/Weißdrucker. Es zeigt sich oft, dass die Darstellung auf dem Monitor nicht schlecht aussieht, jedoch nach dem Ausdrucken ein ziemlich kontrastarmes Bild vorliegt. Das liegt an der geringeren Anzahl der Graustufen, die ausgedruckt werden können. Ein automatisches Verfahren, das dieses Problem in vielen Fällen löst, wird in diesem Kapitel beschrieben.

In dieses Anwendungsgebiet fällt auch noch die Pseudofarbdarstellung, bei der bestimmte Grauwerte eines einkanaligen Grauwertbildes auf dem Monitor farbig dargestellt werden und so dem Betrachter einprägsamer erscheinen.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_4

Eine andere Fragestellung liegt vor, wenn eine Trennung von Objekten vom Hintergrund durchgeführt werden soll. Dieser Problemkreis fällt in den Bereich der Bildsegmentierung. Die Binarisierung und die Äquidensitenbildung sind einfache Verfahren dazu.

Schließlich werden noch unterschiedliche Ansätze zur Reduktion der Grauwertemenge beschrieben.

4.2 Grundlagen der Grauwerttransformation

Im Folgenden wird ein einkanaliges Grauwertbild $\mathbf{S}_e = (s_e(x, y))$ mit der Grauwertmenge $G = \{0, 1, ..., 255\}$ vorausgesetzt. Eine Grauwerttransformation ist eine Abbildung f der Grauwertmenge G. Für f ist im Prinzip jede Funktion geeignet. Man wird in der Regel fordern, dass gilt:

$$f : G \to G. \tag{4.1}$$

Dann muss f beschränkt sein:

$$\min\{f\} > -\infty; \quad \max\{f\} < +\infty. \tag{4.2}$$

Ist dies der Fall, so kann f zu f_n normiert werden, sodass (4.1) erfüllt ist:

$$f_n(g) = \frac{f(g) - \min\{f\}}{\max\{f\} - \min\{f\}} \cdot c. \tag{4.3}$$

Dabei ist c ein Skalierungsfaktor, der geeignet zu wählen ist (z.B. $c = 255$). Die grafische Darstellung von $f_n(g)$ bezeichnet man als die *Gradationskurve*.

Wird f_n auf das Grauwertbild $\mathbf{S}_e$ angewendet, so berechnet sich das Ergebnisbild $\mathbf{S}_a$ wie folgt:

$$\begin{aligned} &\mathbf{S}_e \to \mathbf{S}_a : \\ &s_a(x, y) = f_n(s_e(x, y)),\ 0 \le x \le L - 1,\ 0 \le y \le R - 1. \end{aligned} \tag{4.4}$$

An dieser Stelle ein Hinweis zur Implementierung: In der Praxis wird man die Transformation $\mathbf{S}_e \to \mathbf{S}_a$ nie gemäß (4.4) realisieren, sondern immer über eine *look-up*-Tabelle (LuT). Eine *look-up*-Tabelle ist eine Tabelle mit 256 Speicherplätzen, die für die Grauwertmenge $G = \{0, 1, ..., 255\}$ die Werte von $f_n(g)$, $g \epsilon G$, enthält.

Die Transformationsfunktionen $f(g)$ oder $f_n(g)$ können auch interaktiv definiert werden. Dazu wird bei einem Grafik- oder Rasterbildspeichersystem mit dem Cursor der Verlauf einer Gradationskurve gezeichnet. Nach Maßgabe der festgelegten Kurve wird die *look-up*-Tabelle besetzt. Bei dieser Vorgehensweise kann die Skalierung der Grauwerte ganz auf das vorliegende Bildmaterial abgestimmt werden.

Der Algorithmus zur Skalierung der Grauwerte eines Bildes wird folgendermaßen skizziert:

A4.1: Skalierung der Grauwerte.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild mit der Grauwertmenge $G = \{0, 1, ..., 255\}$ (Eingabebild).

◇ $\mathbf{S}_a = (s_a(x,y))$ ein einkanaliges Grauwertbild mit der Grauwertmenge $G = \{0, 1, ..., 255\}$ (Ausgabebild).

Algorithmus:

(a) Festlegen oder Berechnen der Parameter der Funktion f_n.

(b) Berechnung der *look-up*-Tabelle. Für alle Grauwerte $g \epsilon G$:
$LuT(g) = f_n(g)$;

(c) Für alle Bildpunkte des Bildes $\mathbf{S}_e = (s_e(x,y))$:
$s_a(x,y) = LuT(s_e(x,y))$;

Ende des Algorithmus

Wenn das Eingabebild $\mathbf{S}_e$ ein mehrkanaliges Bild ist, z.B. ein Farbbild, kann die dargestellte Grauwerttransformation für jeden Kanal durchgeführt werden. Meistens ist es dann sinnvoll, für jeden Kanal eine eigene *look-up*-Tabelle zu verwenden.

4.3 Lineare Skalierung

Durch die lineare Skalierung eines Bildes wird die Grauwertverteilung eines Originalbildes $\mathbf{S}_e$ durch eine lineare Abbildung transformiert:

$$\mathbf{S}_e \to \mathbf{S}_a : s_a(x,y) = (s_e(x,y) + c_1)c_2 = c_2 s_e(x,y) + c_1 c_2. \tag{4.5}$$

Mit der Notation von Abschnitt 4.2 erhält man folgende Transformationsfunktion:

$$f(g) = (g + c_1)c_2 = c_2 g + c_1 c_2. \tag{4.6}$$

Man sieht sofort, dass sich für $c_1 = 0$ und $c_2 = 1$ die identische Abbildung ergibt. Das Histogramm von $\mathbf{S}_a$ berechnet sich aus dem Histogramm von $\mathbf{S}_e$ gemäß:

$$p_{\mathbf{S}_a}(g_a) = \frac{a_{g_a}}{M} = \frac{a_{(g_e - c_1)/c_2}}{M} \tag{4.7}$$

Ist $c_1 > 0$, so wird zu den Grauwerten von $\mathbf{S}_e$ eine Konstante addiert und das Bild wird insgesamt heller, während für $c_1 < 0$ das Bild dunkler wird. Das zu $\mathbf{S}_a$ gehörige Histogramm

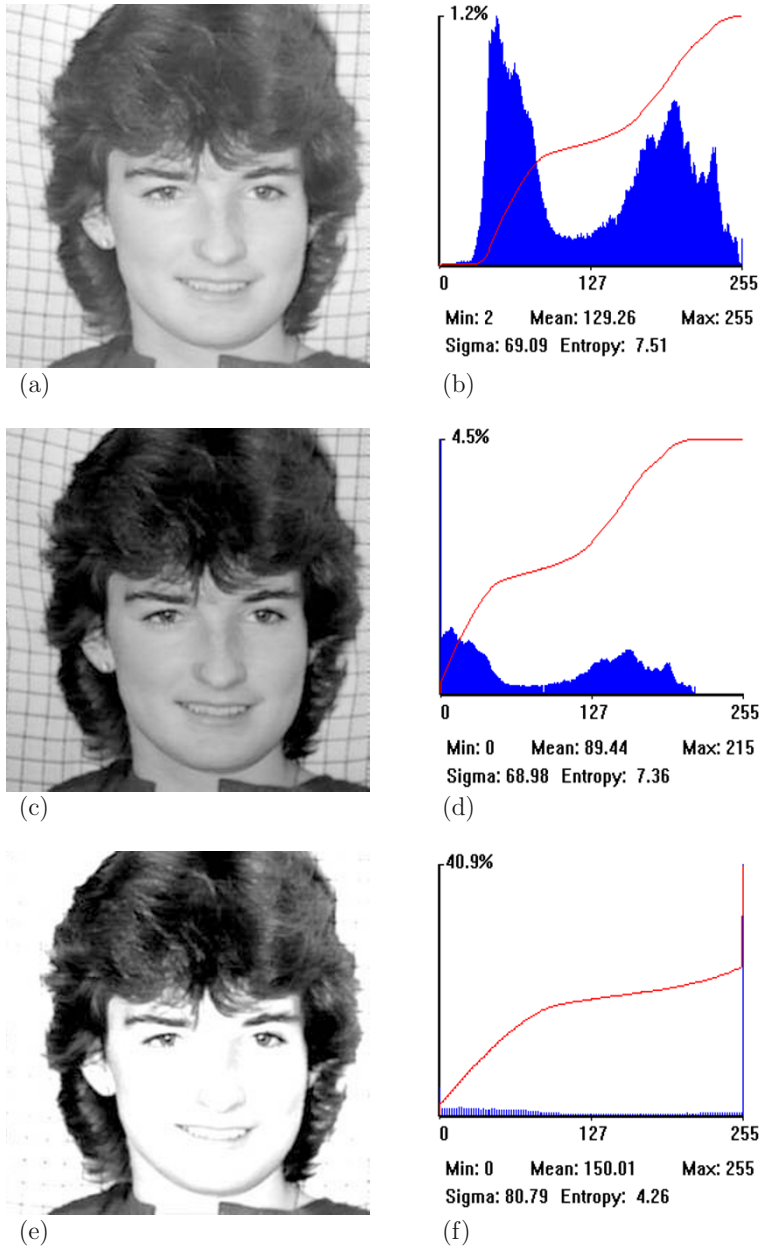


Bild 4.1: Beispiele zur linearen Skalierung eines Grauwertbildes. (a) Original. (b) Histogramm des Originals. (c) Verschiebung der Grauwerte mit $c_1 = -40$. (d) Histogramm nach der Verschiebung. Das veränderte Aussehen des Histogramms erklärt sich zum einen aus der Verschiebung der Grauwerte und zum anderen aus der unterschiedlichen Skalierung der Ordinate. Man sieht, dass Grauwerte auf der linken Seite des Histogramms „verloren gegangen“ sind. (e) Kontrastanreicherung durch Multiplikation mit $c_2 = 2$. (f) Histogramm des skalierten Bildes. Auch hier wurden Grauwerte abgeschnitten.

ist gegenüber dem von $\mathbf{S}_e$ nach links ($c_1 < 0$) oder nach rechts ($c_1 > 0$) verschoben (Bild 4.1).

Die Konstante c_2 bewirkt eine Änderung des Kontrastes. Für $|c_2| > 1$ wird das Histogramm breiter und das Bild $\mathbf{S}_a$ kontrastreicher, wogegen für $|c_2| < 1$ das Histogramm schmäler und $\mathbf{S}_a$ kontrastärmer wird. Diese Sachverhalte können auch anhand des Mittelwertes und der mittleren quadratischen Abweichung verifiziert werden:

$$m_{\mathbf{S}_a} = \frac{1}{M}\sum_{x=0}^{L-1}\sum_{y=0}^{R-1} s_a(x,y) = \frac{1}{M}\sum_{x=0}^{L-1}\sum_{y=0}^{R-1}\Big(c_2 s_e(x,y) + c_1 c_2\Big) = c_2 m_{\mathbf{S}_e} + c_1 c_2. \tag{4.8}$$

$$\begin{aligned} q_{\mathbf{S}_a} &= \frac{1}{M}\sum_{x=0}^{L-1}\sum_{y=0}^{R-1}\Big(s_a(x,y) - m_{\mathbf{S}_a}\Big)^2 = \\ &= \frac{1}{M}\sum_{x=0}^{L-1}\sum_{y=0}^{R-1}\Big(c_2 s_e(x,y) + c_1 c_2 - (c_2 m_{\mathbf{S}_e} + c_1 c_2)\Big)^2 = \\ &= c_2^2 \frac{1}{M}\sum_{x=0}^{L-1}\sum_{y=0}^{R-1}\Big(s_e(x,y) - m_{\mathbf{S}_e}\Big)^2 = c_2^2 q_{\mathbf{S}_e}. \end{aligned} \tag{4.9}$$

Als Grauwertmenge bei Grauwertbildern wird $G = \{0, 1, ..., 255\}$ verwendet. Durch die Transformation (4.5) oder (4.6) kann diese Festlegung verletzt werden, da $f(g)$ nicht normiert ist. Es muss also die Transformation (4.5) so modifiziert werden, dass die Grauwerte von $\mathbf{S}_a$ immer in G liegen. Aus der linearen Skalierung wird dann eine *stückweise lineare Skalierung*:

$$\mathbf{S}_e \to \mathbf{S}_a : s_a(x,y) = \begin{cases} 0, & \text{falls } (s_e(x,y) + c_1)c_2 \le 0 \\ 255, & \text{falls } (s_e(x,y) + c_1)c_2 > 255 \\ (s_e(x,y) + c_1)c_2, & \text{sonst.} \end{cases} \tag{4.10}$$

Die normierte Skalierungsfunktion f_n ist hier stückweise linear. Sie kann somit wie folgt geschrieben werden:

$$f_n(g) = \begin{cases} 0, & \text{falls } c_2 g + c_1 c_2 \le 0, \\ 255, & \text{falls } c_2 g + c_1 c_2 > 255, \\ (g + c_1)c_2 = c_2 g + c_1 c_2, & \text{sonst.} \end{cases} \tag{4.11}$$

Die stückweise lineare Skalierung kann in $\mathbf{S}_a$ Informationsverlust bewirken, da alle Bildpunkte, deren Grauwerte durch die Transformation kleiner als 0 (oder größer als 255) würden, in $\mathbf{S}_a$ den Grauwert 0 (bzw. 255) erhalten und somit nicht mehr unterschieden werden können (siehe Bildfolge 4.1). Dieser Effekt muss aber nicht unbedingt nachteilig sein. Es gibt viele Beispiele, bei denen auf diese Weise nicht benötigte Bildinformation „ausgeblendet“ wird.

Auf jeden Fall muss die Bestimmung der Parameter c_1 und c_2 sorgfältig durchgeführt werden. Eine Möglichkeit ist es, sie aus dem Histogramm zu ermitteln. Dazu werden der

minimale (min) und der maximale (max) Grauwert des Bildes $\mathbf{S}_e$ bestimmt. Die beiden Parameter berechnen sich dann gemäß:

$$c_1 = -min \text{ und } c_2 = \frac{255}{max - min}. \tag{4.12}$$

Bei dieser Transformation geht keine Bildinformation verloren, da alle Grauwerte von $\mathbf{S}_e$ im Intervall $[min,max]$ liegen und min in $\mathbf{S}_a$ auf den Wert 0 und max auf den Wert 255 abgebildet wird.

Es kann aber vorkommen, dass sich das Bild $\mathbf{S}_a$ gegenüber $\mathbf{S}_e$ kaum verändert hat, nämlich dann, wenn min und max zufällige Resultate eines leicht verrauschten Bildes sind, die keine Signifikanz für den eigentlichen Bildinhalt haben. Es ist z.B. möglich, dass durch Fehler bei der Digitalisierung die Grauwerte einiger Bildpunkte auf 0 oder 255 gesetzt werden. Dann wäre die Transformation (4.10) mit der Parameterberechnung (4.12) eine identische Abbildung. Aus diesem Grund ist es oft sinnvoll, die Skalierungsparameter in (4.12) mit zwei Werten $min' > min$ und $max' < max$ zu berechnen. Hier ist dann aber Vorsicht geboten, da durch ungeschickte Wahl Bildinformation verloren geht.

Bevor weitere Techniken zur Bestimmung der Parameter c_1 und c_2 untersucht werden noch ein Hinweis auf die Implementierung solcher Grauwertskalierungen. Um Rechenzeit zu sparen, wird die Implementierung nicht gemäß (4.10) durchgeführt, sondern wie in Abschnitt 4.2 dargestellt, über eine *look-up*-Tabelle, die einmal vor der eigentlichen Grauwerttransformation besetzt wird. Die Vorgehensweise dazu wurde in Algorithmus **A4.1** angegeben.

Bei einem Betriebssystem mit einer grafischen Oberfläche oder mit Hilfe eines Rasterbildspeichersystems, lässt sich einfach eine Skalierung der Daten im Bildfenster oder im Bildwiederholungsspeicher einrichten. Eine *look-up*-Tabelle wird anfangs mit der identischen Abbildung, also mit den Werten von 0 bis 255, vorbesetzt. Die Skalierungsparameter erhalten anfänglich die Werte $c_1 = 0$ und $c_2 = 1$. Der Cursor wird im Bildfenster in die Bildmitte gesetzt. Bei jeder Bewegung werden die Koordinaten des Cursors (x, y) ausgelesen. Die Skalierungsparameter werden dann, wie in Algorithmus **A4.2** angegeben, aus der Position des Cursors berechnet.

Die *look-up*-Tabelle wird nach Maßgabe der neuen Skalierungsparameter neu besetzt. Horizontale Bewegungen des Lichtpunktes bewirken Veränderungen in der Helligkeit und vertikale Bewegungen Veränderungen im Kontrast.

Die Berechnung von c_1 und c_2 geht sehr schnell. Wenn das Lesen und Laden der *look-up*-Tabelle des Grafik- oder Rasterbildspeichersystems sehr schnell geht, kann die Skalierung interaktiv in Echtzeit durchgeführt werden, wobei die Originaldaten im Bildfenster erhalten bleiben. Es besteht mit dieser Methode somit die Möglichkeit, zu einem Bild interaktiv eine Helligkeits- und Kontrastmanipulation durchzuführen. Wenn die gewünschte Darstellung gefunden ist, wird die *look-up*-Tabelle übernommen und das Originalbild (z.B. zur Bildausgabe über einen Drucker) mit der *look-up*-Tabelle modifiziert. Diese Technik lässt sich auch bei Farbbildern einsetzen.

A4.2: Interaktive Berechnung von c_1 und c_2.

Voraussetzungen und Bemerkungen:

◇ Es wird angenommen, dass die *x_cursor_position* und die *y_cursor_position* zwischen 1 und 512 liegen.

Algorithmus:

(a) Den *cursor* des Bildverarbeitungssystems in die Bildmitte setzen.

(b) Solange kein Abbruch-Code (z.B. mit der rechten oder linken Maus-Taste) gegeben wird:

(ba) Aktuelle *cursor*-Position lesen.

(bb) Falls sich die *cursor*-Position geändert hat:

(bba) Berechnung der Parameter c_1 und c_2 etwa gemäß:
$c_2 = (512 - y\_cursor\_position)/y\_cursor\_position;$
$c_1 = 127 - c_2 \cdot x\_cursor\_position/2;$

(bbb) Berechnung der neuen *look-up*-Tabelle gemäß Algorithmus **A4.1** und Formel (4.11).

(bbc) Abspeichern der neuen *look-up*-Tabelle in der Hardware des Bildverarbeitungssystems.

Ende des Algorithmus

Die Skalierungsparameter können auch so bestimmt werden, dass das skalierte Bild $\mathbf{S}_a$ einen vorgegebenen Mittelwert und eine vorgegebene mittlere quadratische Abweichung hat. Es sei

$m_{\mathbf{S}_e}$ der Mittelwert des Originalbildes $\mathbf{S}_e$,

$q_{\mathbf{S}_e}$ die mittlere quadratische Abweichung von $\mathbf{S}_e$,

$m_{\mathbf{S}_a}$ der vorgegebene Mittelwert von $\mathbf{S}_a$,

$q_{\mathbf{S}_a}$ die vorgegebene mittlere quadratische Abweichung von $\mathbf{S}_a$.

Dann liefert die Transformation (4.10) mit den Skalierungsparametern

$$c_1 = \frac{m_{\mathbf{S}_a}\sqrt{q_{\mathbf{S}_e}}}{\sqrt{q_{\mathbf{S}_a}}} - m_{\mathbf{S}_e} \text{ und } c_2 = \frac{\sqrt{q_{\mathbf{S}_a}}}{\sqrt{q_{\mathbf{S}_e}}} \tag{4.13}$$

ein Bild $\mathbf{S}_a$ mit den vorgegebenen Werten für $m_{\mathbf{S}_a}$ und $q_{\mathbf{S}_a}$. Durch eine Transformation dieser Art können z.B. zwei Bilder desselben Objektes, die unter verschiedenen Aufnahmebedingungen erzeugt wurden, in der Helligkeit und im Kontrast angeglichen werden.

4.4 Äquidensiten (*gray level slicing*)

Eine Spezialisierung der allgemeinen Skalierung von Abschnitt 4.2 ist die Erzeugung eines *Äquidensitenbildes* $\mathbf{S}_a$ aus dem Original $\mathbf{S}_e$ (*gray level slicing*). Die normierte Skalierungsfunktion $f_n(g)$ wird dazu stückweise konstant gewählt:

$$f_n(g) = g_k \text{ für } l_k \leq g < l_{k+1} \text{ und } k = 0, 1, ... \tag{4.14}$$

wobei l_k, g_k und g aus dem Definitionsbereich $G = \{0, 1, ..., 255\}$ sind. Bild 4.2-a zeigt ein Beispiel einer Gradationskurve dieser Art, Bild 4.2-b die dazugehörige Äquidensitendarstellung des Testbildes (mit schwarzen Übergangsrändern) und Bild 4.2-c das Histogramm des Äquidensitenbildes.

Die Formel (4.15) ist ein weiteres Beispiel, wie die Skalierungsfunktion in einem Anwendungsfall aussehen könnte:

$$f_n(g) = \begin{cases} 0 & \text{falls } 0 \leq g = s_e(x,y) \leq 11, \\ 100 & \text{falls } 11 < g = s_e(x,y) \leq 111, \\ 200 & \text{falls } 111 < g = s_e(x,y) \leq 177, \\ 0 & \text{falls } 177 < g = s_e(x,y) \leq 255. \end{cases} \tag{4.15}$$

Hier werden alle Grauwerte des Originals $\mathbf{S}_e$, die z.B. zwischen 12 und 111 (eingeschlossen) liegen, im Ausgabebild $\mathbf{S}_a$ durch den Grauwert 100 dargestellt, Grauwerte, die zwischen 112 und 177 (eingeschlossen) liegen, durch den Grauwert 200. Die übrigen Grauwerte werden im Ausgabebild mit dem Grauwert 0 codiert. Das dazu gehörige Bildbeispiel ist die Bildfolge 4.3. Bild 4.3-a zeigt zwei unterschiedliche Stecker. Mit obiger Skalierungsfunktion f_n wurden die beiden „Objekte“ ausgeblendet (Bild 4.3-c). Jeweils rechts daneben sind die Histogramme mit den relativen Summenhäufigkeiten. Der Bildhintergrund des Originals erfasst etwa 84% der gesamten Bildfläche. Bei Bildern, die mit stückweise konstanten Skalierungsfunktionen verarbeitet wurden, sind die relativen Summenhäufigkeiten stufenförmig (Bilder 4.2-c und 4.3-d).

Es ist auch möglich, bestimmte Grauwertbereiche unverändert aus dem Original zu übernehmen. So könnte man im obigen Beispiel den linken Stecker mit dem Grauwert 200 codieren, den Hintergrund mit 0 und den rechten Stecker mit seinen Originalgrauwerten aus dem Eingabebild $\mathbf{S}_e$:

$$f_n(g) = \begin{cases} 0 & \text{falls } 0 \leq g = s_e(x,y) \leq 11, \\ s_e(x,y) & \text{falls } 11 < g = s_e(x,y) \leq 111, \\ 200 & \text{falls } 111 < g = s_e(x,y) \leq 177, \\ 0 & \text{falls } 177 < g = s_e(x,y) \leq 255. \end{cases} \tag{4.16}$$

Die mit (4.15) erzeugten Äquidensiten heißen *Äquidensiten 1. Ordnung*. Bei *Äquidensiten 2. Ordnung* werden nur die Ränder der Flächen ausgegeben. Dazu werden zunächst die Äquidensiten 1. Ordnung berechnet. In einem zweiten Schritt werden die Differenzen zu den Nachbarn gebildet und mit einem Skalierungsfaktor ins Ausgabebild übertragen (Algorithmus **A4.3**).

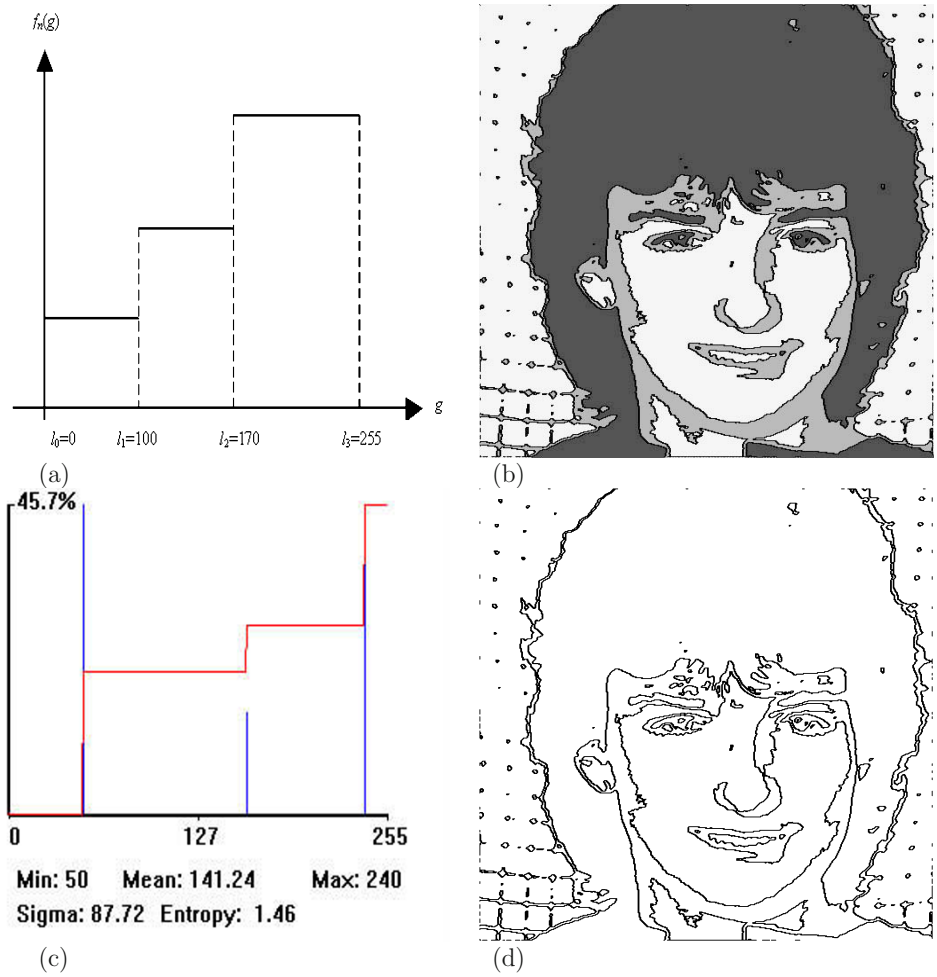
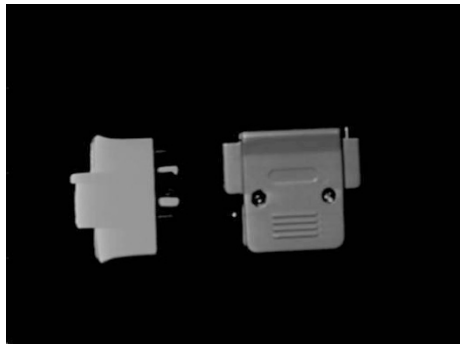
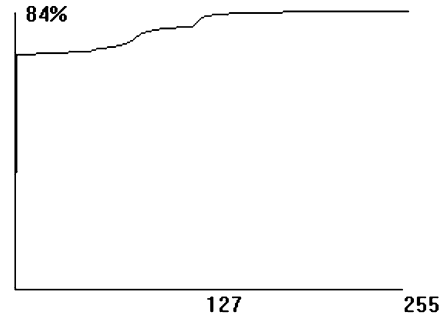


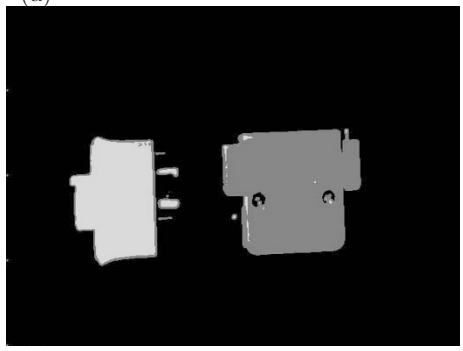
Bild 4.2: Beispiel zur Äquidensitendarstellung. (a) Stückweise konstante Skalierungsfunktion. (b) Äquidensitendarstellung des Testbildes (mit schwarzen Übergangsrändern). (c) Histogramm der Äquidensitendarstellung. (d) Äquidensiten 2. Ordnung.



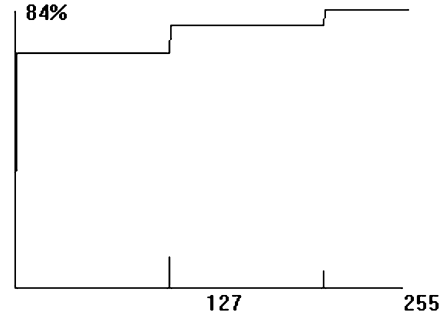
(a)



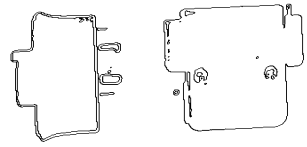
(b)



(c)



(d)



(e)

Bild 4.3: Äquidensitendarstellung. (a) Das Original zeigt zwei unterschiedliche Stecker. (b) Das Histogramm und die Summenhäufigkeiten des Originals. Es sind ca. 84% der Grauwerte sehr nahe bei null. (c) Die beiden „Objekte“ wurden ausgeblendet. Da aber sowohl im rechten als auch im linken Stecker jeweils Grauwerte des anderen Steckers enthalten sind, ist eine eindeutige Trennung nicht möglich. (d) Histogramm mit Summenhäufigkeiten des Äquidensitenbildes. Bei Bildern, die mit stückweise linearen Skalierungsfunktionen verarbeitet wurden, sind die Summenhäufigkeiten stufenförmig. (e) Äquidensiten 2. Ordnung. Die doppelten Ränder im linken Stecker entstehen durch Grauwerte, mit denen der rechte Stecker codiert wurde.

A4.3: Äquidensiten 2. Ordnung.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild.

◇ $\mathbf{S}_a = (s_a(x,y))$ sei das Ausgabebild.

Algorithmus:

(a) Berechnung des Äquidensitenbildes $\mathbf{S}_{aeq}$.

(b) Für alle Pixel $s_{aeq}(x,y)$ von $\mathbf{S}_{aeq}$:

(ba) Bilde: $diff = 4s_{aeq}(x,y) - s_{aeq}(x,y-1) - s_{aeq}(x,y+1) - s_{aeq}(x-1,y) - s_{aeq}(x+1,y)$;

(bb) Falls $diff = 0 : s_a(x,y) = 255$;

(bc) Falls $diff \neq 0 : s_a(x,y) = 0$;

Ende des Algorithmus

Diese Technik kann verwendet werden, wenn auf einfache Art der Rand der „Objekte“ ermittelt werden soll (Bild 4.3-e). Allerdings sind die Ergebnisse nur brauchbar, wenn die Grauwertintervalle deutlich voneinander getrennt liegen und im Hintergrund diese Grauwerte nicht auftreten. Von *Äquidensiten gemischter Ordnung* spricht man, wenn im Ausgabebild sowohl die grau codierten Flächen als auch deren Ränder dargestellt werden.

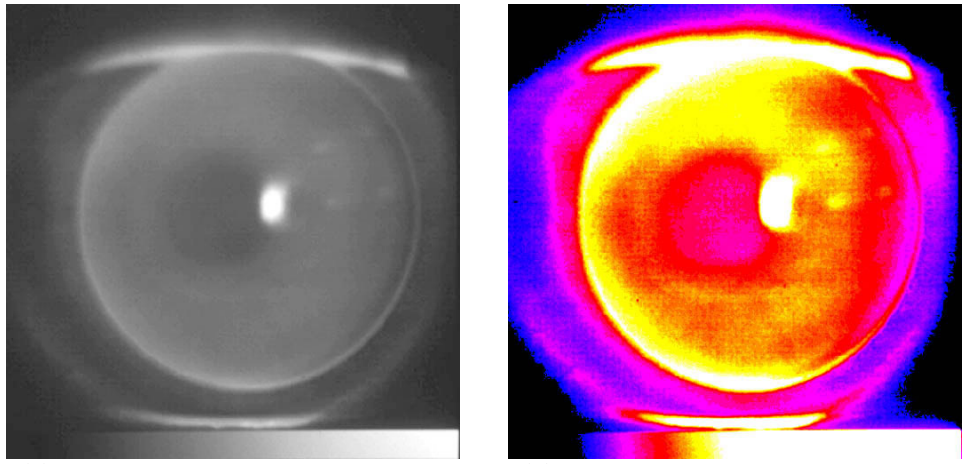
Wenn das Originalbild verrauscht ist, wird das Äquidensitenbild an den Rändern der Flächen mehr oder weniger stark ausgefranst. Dann ist es notwendig, vor der Äquidensitenbildung einen Glättungsoperator (Grundlagen dazu in Kapitel 5) anzuwenden. Sollen bei der Glättung die 8-Nachbarn eines Pixels $s(x,y)$ berücksichtigt werden, so kann der „bewegte Mittelwert“ mit folgendem Filterkern verwendet werden:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{4.17}$$

Auch größere Nachbarschaften sind sinnvoll, allerdings ist dabei zu beachten, dass damit feine Bildstrukturen mehr und mehr verloren gehen.

Die Bilder 4.4-a bis 4.4-f zeigen ein Beispiel zu dieser Anwendung. Bild 4.4-a ist das Original, ein etwas verrauschtes, digitalisiertes Fernsehbild. Daneben ist das mit einem $5 \cdot 5$ bewegten Mittelwert verarbeitete Bild. Die Bilder 4.4-c und 4.4-d zeigen jeweils die Äquidensitendarstellung mit vier Grauwertintervallen. Man sieht, dass sich die homogenen Flächen im rechten Bild deutlicher ausprägen. Wenn nach der Äquidensitenbildung das Bild weiterverarbeitet werden soll, z.B. in eine Strichzeichnung für ein CAD-System, so ist das linke Bild 4.4-e unbrauchbar, während das rechte Bild 4.4-f als Urskizze in einem CAD-System weiter verwendet werden könnte.

Bild 4.4: (a) Verrauschtes, digitalisiertes Fernsehbild. (b) Geglättetes Bild mit einer Umgebung von $5 \cdot 5$ Nachbarn. (c) Äquidensiten aus dem Original mit vier Graustufen. (d) Äquidensiten nach dem bewegten Mittelwert $(5 \cdot 5)$. Man sieht hier, dass sich die homogenen Flächen wesentlich deutlicher ausprägen als im linken Bild. (e) Äquidensiten 2. Ordnung aus dem Original. (f) Äquidensiten 2. Ordnung aus dem geglätteten Bild. Dieses Bild wäre z.B. als Urskizze für ein CAD-System brauchbar.



(a) (b)

Bild 4.5: (a) Bei der Anpassung von Kontaktlinsen wird eine fluoreszierende Kontrastflüssigkeit verwendet. In der vorliegenden Anwendung wurden dem Linsenanpasser die Bilder zusätzlich in digitalisierter Form bewegungskompensiert auf einem Monitor angeboten. (b) Durch die dabei verwendete Pseudofarbdarstellung wurde die Verteilung der Kontrastflüssigkeit deutlich sichtbar gemacht.

Die *Pseudofarbdarstellung*, die schon in den vorhergehenden Abschnitten angesprochen wurde, ist eng mit der Äquidensitenbildung verwandt. Hier werden den Grauwerten oder Grauwertintervallen des Originalbildes Farben zugeordnet. Soll ein bestimmter Bildbereich dem Betrachter deutlich gemacht werden, so ist eine farbliche Absetzung in der Regel vorzuziehen. Das menschliche Auge kann nämlich nur etwa 30 unterschiedliche Grauabstufungen unterscheiden. Untersuchungen haben aber gezeigt, dass es bei der Unterscheidung von Farben wesentlich leistungfähiger ist (möglicherweise um mehrere Zehnerpotenzen).

Je nach Anwendungsfall können dem Betrachter durch die entsprechende Farbwahl zusätzlich bestimmte Eigenschaften wie z.B. „warm“ und „kalt“, „häufig“ und „selten“ oder „früher“ und „später“ suggeriert werden. Die Bilder 4.5-a und 4.5-b zeigen ein Beispiel dazu.

In manchen Anwendungsfällen entstehen einprägsame Äquidensitenbilder, wenn die Flächen nicht durch Grauwerte, sondern durch synthetische Muster wiedergegeben werden. So können z.B. bei einer Landnutzungskartierung aus Luft- oder Satellitenbilddaten die verschiedenen Landnutzungsflächen durch Symbole markiert werden, die sich an der Darstellung in topografischen Karten orientieren.

Ein Spezialfall der Äquidensitenbildung ist die Erzeugung eines Zweipegel- oder Binärbildes. Diese Techniken werden im nächsten Abschnitt behandelt.

4.5 Erzeugen von Binärbildern

Die Erzeugung eines Binär- oder Zweipegelbildes ist ein Sonderfall der stückweise linearen Skalierung (Äquidensitenbildung). Die Skalierungsfunktion sieht hier folgendermaßen aus:

$$f_n(g) = \begin{cases} g_0, & 0 = l_0 \leq g = s_e(x,y) < l_1, \\ g_1, & l_1 \leq g < l_2, \\ g_2 = g_0, & l_2 \leq g \leq l_3 = 255. \end{cases} \tag{4.18}$$

Bei einer reinen *Binarisierung* wird ein Schwellwert c vorgegeben. Alle Grauwerte des Bildes $\mathbf{S}_e$, die kleiner als der Schwellwert sind, werden im Ausgabebild $\mathbf{S}_a$ auf den Grauwert $g_0 = 0$ gelegt, alle anderen auf $g_1 = 1$:

$$f_n(g) = \begin{cases} g_0 = 0 & \text{falls } g = s_e(x,y) \leq c, \\ g_1 = 1 & \text{sonst.} \end{cases} \tag{4.19}$$

Bei einer anderen Variante bleiben die Grauwerte bestimmter Bildbereiche in $\mathbf{S}_a$ erhalten, während die Grauwerte anderer Bildbereiche durch einen neuen Grauwert dargestellt werden. Eine Anwendung davon ist z.B. die Trennung eines abgebildeten Objektes vom Hintergrund. Dies ist aber nur möglich, wenn sich die Grauwerte von Hintergrund und Objekt nicht überlappen. Die Skalierungsfunktion könnte dabei festgelegt sein gemäß:

$$f_n(g) = \begin{cases} g_0 = 0, & 0 \leq g = s_e(x,y) < l_1, \\ g_1 = g, & l_1 \leq g \leq 255. \end{cases} \tag{4.20}$$

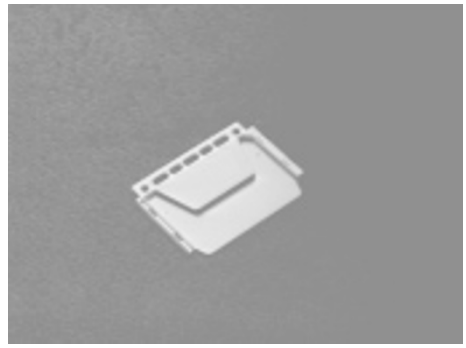
Die Grauwerte von 0 bis $l_1 - 1$ werden durch diese Skalierungsfunktion auf den Grauwert $g_0 = 0$ abgebildet, die Grauwerte von l_1 bis 255 bleiben erhalten. Verarbeitungen dieser Art fallen in den Bereich einfacher Bildsegmentierungen.

Werden anstatt der Grauwerte $g_0 = 0$ und $g_1 = 1$ beliebige Grauwerte g_0 und g_1 verwendet, so spricht man von einem *Zweipegelbild* (Kapitel 3). Die Darstellung als Zweipegelbild (etwa mit $g_0 = 0$ und $g_1 = 255$) ist dann vorzuziehen, wenn das binarisierte Bild auf einem Displaysystem dargestellt werden soll, da die Grauwerte 0 und 1 nicht zu unterscheiden sind.

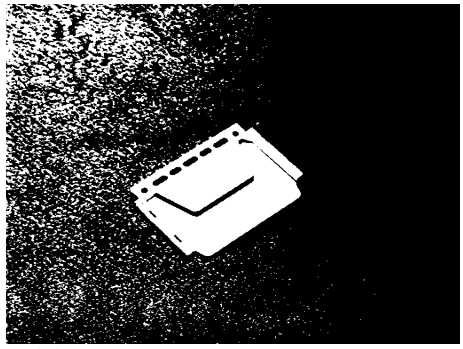
Die Binarisierung eines Grauwertbildes wird in der Praxis meistens dazu verwendet, um abgebildete Objekte vom Hintergrund zu trennen. Deshalb kann sie auch als ein einfaches Verfahren zur *Segmentierung* angesehen werden. Im Folgenden werden einige einfache Verfahren der Bildsegmentierung durch Binärbilderzeugung vorgestellt.

Wie bei der Äquidensitentechnik bilden sich die Flächen homogener aus, wenn ein leicht verrauschtes Bild zunächst mit einem Glättungsoperator behandelt wird.

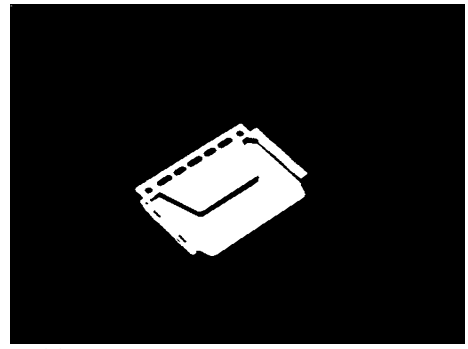
In Formel (4.19) wird für das gesamte Bild ein Schwellwert c verwendet. Man bezeichnet diese Art deshalb auch als *Binarisierung mit fixem Schwellwert*. Der Schwellwert c kann mit unterschiedlichen Verfahren festgelegt werden. Zunächst kann er natürlich aus dem Histogramm von $\mathbf{S}_e$ abgelesen oder automatisch ermittelt werden.



(a)



(b)



(c)

Bild 4.6: (a) Original. (b) Als Schwellwert wurde der Mittelwert m des Bildes verwendet, der in diesem Fall ungeeignet ist, da er das Rauschen im Bild segmentiert. (c) Hier wurde die Schwelle etwas höher gelegt.

Bei deutlich bimodalen Histogrammen ist das Minimum zwischen den beiden Maxima sicher ein geeigneter Schwellwert. Auch der mittlere Grauwert m des gesamten Bildes kann sich als Schwellwert eignen, wenn die zu trennenden hellen und dunklen Bildbereiche etwa gleich groß sind. Ist das nicht der Fall, so kann der Mittelwert m als Schwellwert c auch ungeeignet sein, da er z.B. bei einem leicht verrauschten Bild das Rauschen segmentiert (Bild 4.6).

Viele Bildverarbeitungssysteme bieten die Möglichkeit, den Schwellwert c auch interaktiv, z.B. mit einer Rollkugel oder einer Maus festzulegen. Verfahrenstechnisch wird hier ähnlich vorgegangen wie in Algorithmus **A4.2**. Hier kann das Ergebnis sofort auf dem Monitor des Displaysystems kontrolliert werden.

Soll ein Objekt segmentiert werden, dessen Größe ungefähr bekannt ist, so kann die

p%-Methode verwendet werden. Es wird vorausgesetzt, dass das zu segmentierende Objekt etwa $p\%$ des gesamten Bildbereichs ausmacht. Man wird bei der Binarisierung den Schwellwert c so wählen, dass er $p\%$ „Objekt-Bildpunkte“ und $(100-p)\%$ „Hintergrundbildpunkte“ erzeugt.

A4.4: Binärbilderzeugung mit der p%-Methode.

Voraussetzungen und Bemerkungen:

$\diamond$ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild.

Algorithmus:

(a) Ermittle den Schwellwert c so, dass gilt:
$\sum_{k=0}^{c} p(k) \leq 1 - \frac{p\%}{100}$;

(b) Binarisiere $\mathbf{S}_e$ mit dem Schwellwert c.

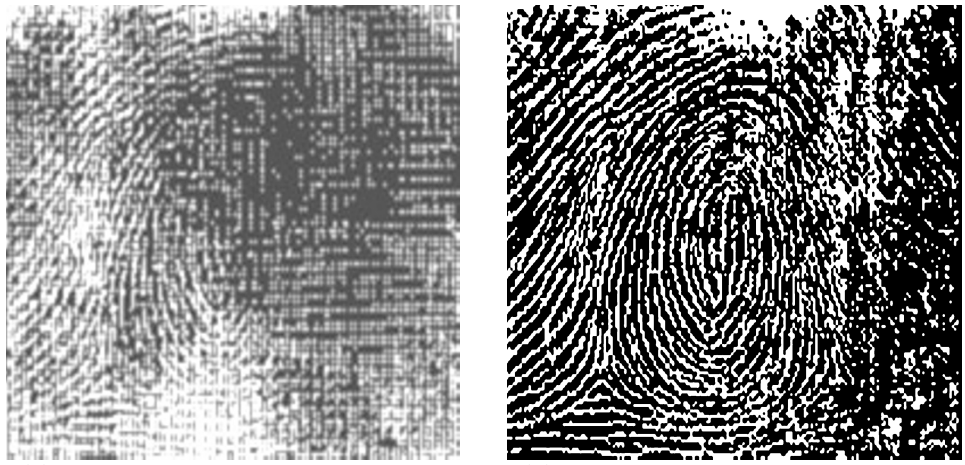
Ende des Algorithmus

Diese Verfahren sind aber nur dann geeignet, wenn der Hintergrund im gesamten Bild im gleichen Grauwertbereich liegt. Dies ist in der Praxis häufig nicht der Fall, z.B. bei ungleichmäßiger Ausleuchtung bei der Bildaufzeichnung. Die Grenze der Anwendbarkeit der einfachen Schwellwertverfahren ist auch erreicht, wenn die Grauwerte des Bildhintergrundes auch im Objekt auftreten. Dazu zeigt Bild 4.7-a ein Beispiel: Durch die unterschiedliche Schwärzung gehen die Papillaren im rechten Bildteil weitgehend verloren.

Eine Möglichkeit sind dann Verfahren mit *dynamischem Schwellwert*. Der Schwellwert c wird hier (im extremen Fall) für jeden Bildpunkt $s(x,y)$ aus einer Umgebung $U(s(x,y))$ (z.B. 3·3, 5·5, usw.) berechnet. Wie groß die Umgebung zu bemessen ist, muss im jeweiligen Anwendungsfall aus dem Bildmaterial festgelegt werden. Für die gewählte Umgebung wird das Histogramm berechnet und daraus der Schwellwert c abgeleitet. Zur Berechnung von c kann ohne weitere Prüfung der Mittelwert $m_{U(s(x,y))}$ der Umgebung verwendet werden, wenn sichergestellt ist, dass in jeder Umgebung $U(s(x,y))$ beide „Modi“ (Hintergrund und zu segmentierendes Objekt) auftreten. Für eine $m \cdot m$-Umgebung sieht die Berechnung des Schwellwertes c folgendermaßen aus:

$$c = \frac{1}{m^2} \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} s(x+k-i, y+k-j), k = \frac{m-1}{2} \tag{4.21}$$

Die Schwellwertberechnung muss nicht unbedingt für jeden Bildpunkt durchgeführt werden. Wenn sich der Hintergrund nur langsam ändert, so kann das Fenster, das zur Berechnung verwendet wird, auch um mehr als einen Bildpunkt verschoben werden. Für die dazwischen liegenden Bildpunkte wird der jeweils letzte berechnete Wert verwendet.



(a) (b)

Bild 4.7: Beispiel zur Binärbilderzeugung mit dynamischem Schwellwert. (a) zeigt das Original eines Fingerabdruckes. Durch die unterschiedliche Schwärzung gehen die Papillaren im rechten Bildteil weitgehend verloren. Durch den Einsatz eines dynamischen Schwellwertverfahrens konnte die Darstellung in (b) wesentlich verbessert werden.

Die Annahme, dass in den Umgebungen $U(s(x,y))$ der Bildpunkte immer beide Modi auftreten, ist nicht immer zutreffend. In solchen Fällen muss das Histogramm der jeweiligen Umgebung berechnet und eine *Bimodalitätsprüfung* durchgeführt werden. Dabei wird geprüft, ob das Histogramm bimodal ist, d.h., ob sich zwei lokale Maxima deutlich ausprägen. Ein Algorithmus dazu könnte etwa wie folgt aussehen:

A4.5: Bimodalitätsprüfung.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild.

◇ $U(s_e(x,y))$ sei eine Umgebung des Bildpunktes in der Position (x,y).

Algorithmus:

(a) Von Null aufsteigend wird im Histogramm des Ausschnittes das erste lokale Maximum ermittelt. Der Grauwert und der Wert des Maximums werden mit $(gmax_1, max_1)$ bezeichnet.

(b) Anschließend wird überprüft, ob in einem bestimmten Abstand von $gmax_1$ (Fangbereich) ein weiteres lokales Maximum ist, das größer als max_1 ist. Falls dies der Fall ist, wird $(gmax_1, max_1) = (gmax'_1, max'_1)$ gesetzt.

(c) Von 255 absteigend wird jetzt im Histogramm das erste lokale Maximum $(gmax_2, max_2)$ gesucht.

(d) Wie oben wird im Fangbereich geprüft, ob ein lokales Maximum $max'_2 > max_2$ existiert. Falls dies erfüllt ist, wird $(gmax_2, max_2) = (gmax'_2, max'_2)$ gesetzt.

(e) Jetzt wird untersucht, ob zwischen $gmax_1$ und $gmax_2$ ein weiteres lokales Maximum liegt. In diesem Fall wäre das Histogramm sicherlich nicht bimodal.

(f) Es sei $(gmin, min)$ das Minimum zwischen $(gmax_1, max_1)$ und $(gmax_2, max_2)$ und $minmax$ das kleinere der beiden Maxima.

(g) Falls nun der Wert von $min/minmax$ kleiner als eine vorgegebene Schranke ist, wird das Histogramm des Ausschnittes als bimodal akzeptiert. Als Schwellwert kann in diesem Fall $c = gmin$ oder der mittlere Grauwert der Umgebung verwendet werden, da ja sichergestellt ist, dass das Histogramm bimodal ist.

(h) Falls die obige Bedingung nicht erfüllt ist, wird für die Binarisierung der alte Schwellwert weiter verwendet.

Ende des Algorithmus

Bild 4.8 ist ein Beispiel zu einem dynamischen Schwellwertverfahren mit Bimodalitätsprüfung. Bild 4.8-a zeigt das synthetische Originalbild, das aus einem dezimal ausgedruckten Graukeilbalken besteht, der einem Graukeil als Hintergrund überlagert wurde. Das dynamische Schwellwertverfahren mit Bimodalitätsprüfung ermöglicht die einwandfreie Trennung von ,,Hintergrund" und ,,Objekt" (Bild 4.8-b).

Abschließend sei noch auf die mehrdimensionalen Schwellwertverfahren hingewiesen. Hier werden in einem mehrkanaligen Bild zur Segmentierung eines Objektes für jeden Kanal Schwellwerte festgelegt, so dass die Trennung in einem mehrdimensionalen Merkmalsraum durchgeführt wird. Auf diese Techniken wird später noch näher eingegangen (Abschnitt 20.7).

Eine andere Möglichkeit ist es, den Hintergrund *vor* der Binarisierung zu korrigieren. Dazu werden zwei unterschiedliche Lösungsverfahren vorgestellt.

Bei manchen Anwendungen ist es nicht möglich, die Beleuchtung so einzurichten, dass die Objekte auf einem sauber ausgeleuchteten Bildhintergrund liegen. Das ist z.B. bei mikroskopischen Aufnahmen oder bei Durchlichtaufnahmen mit einem Leuchttisch der Fall. Wenn sich aber an der Beleuchtung und am Bildausschnitt nichts ändert, so kann man ein Kalibrierungsbild $\mathbf{K}$ erzeugen, indem man den Bildausschnitt *ohne* Objekt digitalisiert (Bild 4.9-b). Dieses Kalibrierungsbild kann man jetzt zur Korrektur des Hintergrundes

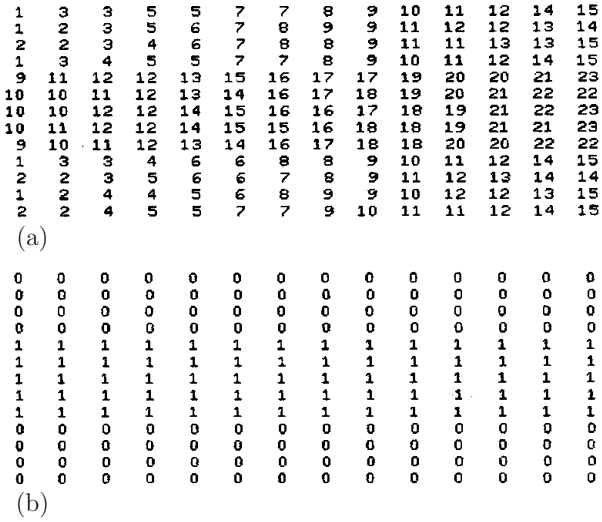


Bild 4.8: Synthetisches Bildbeispiel zur dynamischen Schwellwertberechnung mit Bimodalitätsprüfung. (a) zeigt einen dezimal ausgedruckten Ausschnitt aus einem Graukeil, dem ein Graukeilbalken überlagert wurde. Beide ,,Modi" wurden zufällig gestört. (b) Das dynamische Schwellwertverfahren ermöglicht die einwandfreie Trennung zwischen ,,Objekt" und ,,Hintergrund".

eines Bildes $\mathbf{S}_e$ mit Objekt (Bild 4.9-a) verwenden, indem man z.B. als Ausgabebild $\mathbf{S}_a$ (Bild 4.9-d) die absoluten Differenzen zwischen $\mathbf{S}_e$ und $\mathbf{K}$ berechnet:

$$\mathbf{S}_e \rightarrow \mathbf{S}_a : s_a(x,y) = |s_e(x,y) - k(x,y)|. \tag{4.22}$$

Der Vergleich der Binarisierung des Originals (Bild 4.9-c) und des Differenzbildes (Bild 4.9-e) zeigt die Verbesserung. Werden in der weiteren Verarbeitung die Originalgrauwerte des Objekts benötigt, so kann man das binarisierte Differenzbild als Maske verwenden und damit das Objekt aus dem Original ausblenden.

Bei einer anderen Variante werden morphologische Operationen im Grauwertbild herangezogen, wie z.B. Dilatation und Erosion (Grundlagen dazu in Kapitel 6): wenn es gelingt, das Objekt ganz auszublenden, also ein Bild zu erzeugen, in dem nur mehr der Hintergrund vorhanden ist, so kann durch eine Differenzbildung zwischen Original und Hintergrundbild der Einfluss der unterschiedlichen Beleuchtung eliminiert werden. Der Algorithmus dazu lautet wie folgt:

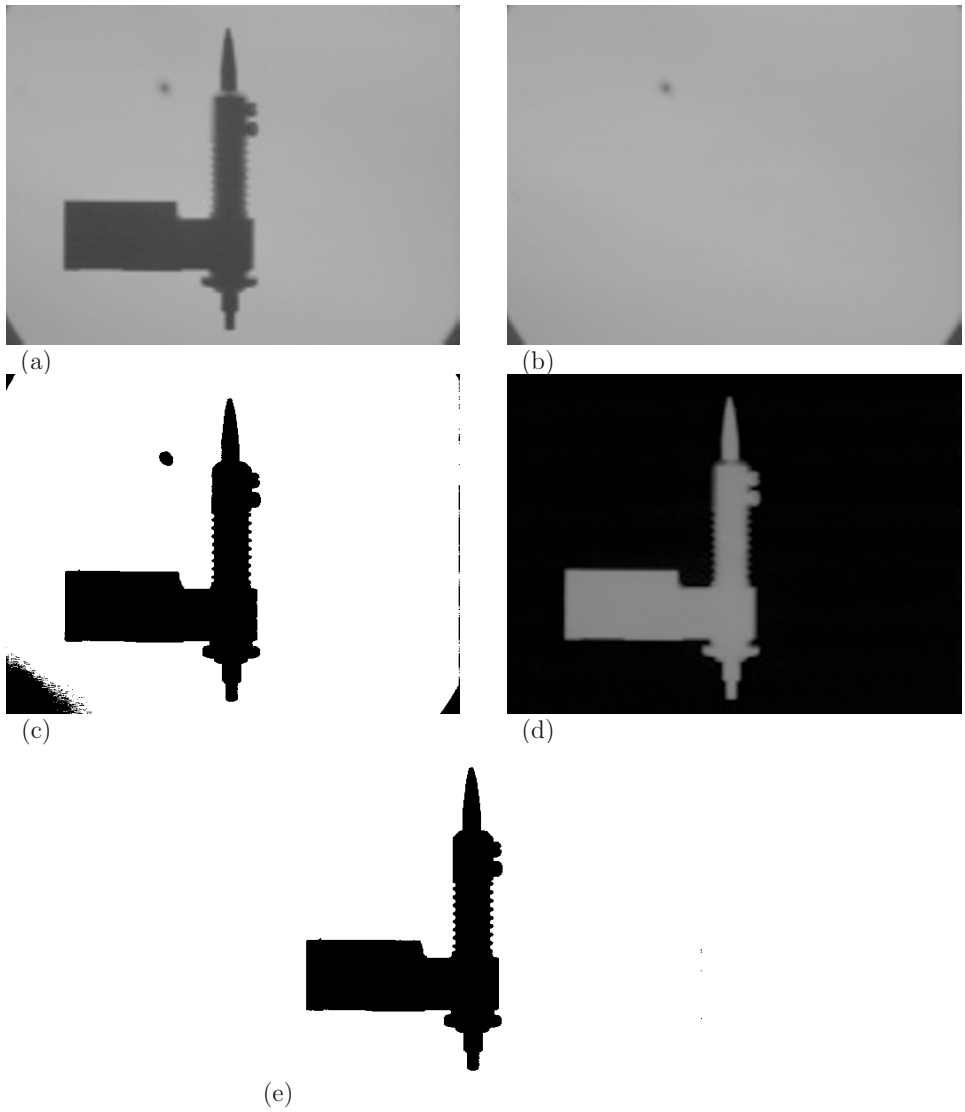


Bild 4.9: (a) Original. (b) Bildhintergrund (Kalibrierungsbild). (c) Binarisierung des Originals. Die Trennung des Objekts vom Hintergrund ist nicht sauber möglich. (d) Differenzbild: Es enthält die Absolutbeträge der Differenzen des Originals und des Kalibrierungsbildes. (e) Binarisierung des Differenzbildes. Das Ergebnis kann z.B. als Maske zum Ausblenden des Objektes aus dem Original verwendet werden.

A4.6: Elimination von Einflüssen durch die Beleuchtung.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y))$ ein einkanaliges Grauwertbild.

◇ Es wird angenommen, dass sich die „Objekte“ dunkel vom hellen Hintergrund abheben.

Algorithmus:

(a) Berechnung eines Hilfsbildes $\mathbf{T}$ durch n Dilatationen, gefolgt von n Erosionen (*closing*-Operation).

(b) Berechnung des Differenzbildes $\mathbf{U}$: $\mathbf{U} = 127 + \mathbf{S}_e - \mathbf{T}$.

(c) Binarisiere $\mathbf{U}$ mit einem fixen Schwellwert c.

Ende des Algorithmus

Die Anzahl n der Dilatationen und Erosionen im Algorithmus **A4.6** hängt von der „Dicke” der Objekte ab. Man sieht sofort, dass das Verfahren dann geeignet ist, wenn das Objekt aus dünnen Linien besteht, die durch die Dilatationen verschwinden. Ist aufgrund des Bildinhaltes nur eine geringe Anzahl von Dilatationen durchzuführen (z.B $n = 2$ oder $n = 3$), so kann möglicherweise auf die nachfolgenden Erosionen verzichtet werden. Die Bilder 4.10-a bis 4.10-e sind Beispiele zu dieser Vorgehensweise.

Abschließend sei noch bemerkt, dass mit schnellen Bildverarbeitungssystemen, wie z.B. Grafikkarten (Band I Abschnitt 13.6.3), die morphologischen Operationen durchaus in Videoechtzeit berechnet werden können. Damit kann auch die eben beschriebene Korrektur sehr schnell durchgeführt werden.

4.6 Logarithmische und exponentielle Skalierung

Bei der linearen Skalierung werden alle Bereiche der Grauwertmenge G gleichmäßig verändert, mit Ausnahme der Grauwerte, die durch die Transformation kleiner als 0 oder größer als 255 würden. Bei einem Bild könnte es aber wünschenswert sein, dass z.B. in dunklen Bildbereichen der Kontrast stärker angehoben wird als in hellen Bildbereichen. Dieser Effekt kann mit einer logarithmischen Funktion f_n erzielt werden. Es werden drei Parameter c_1, c_2 und c_3 benötigt, für die $0 < c_1 < c_2$ und $-255 < c_3 < +255$ gelten soll.

$$\begin{aligned} &min = \log(c_1); \quad max = \log(c_2); \\ &eps = (c_2 - c_1)/255; \\ &f_n(g) = 255 \cdot \frac{\log(c_1 + g \cdot eps) - min}{max - min}; \quad g \epsilon G. \\ &LuT(g + c_3) = f_n(g) \end{aligned} \tag{4.23}$$

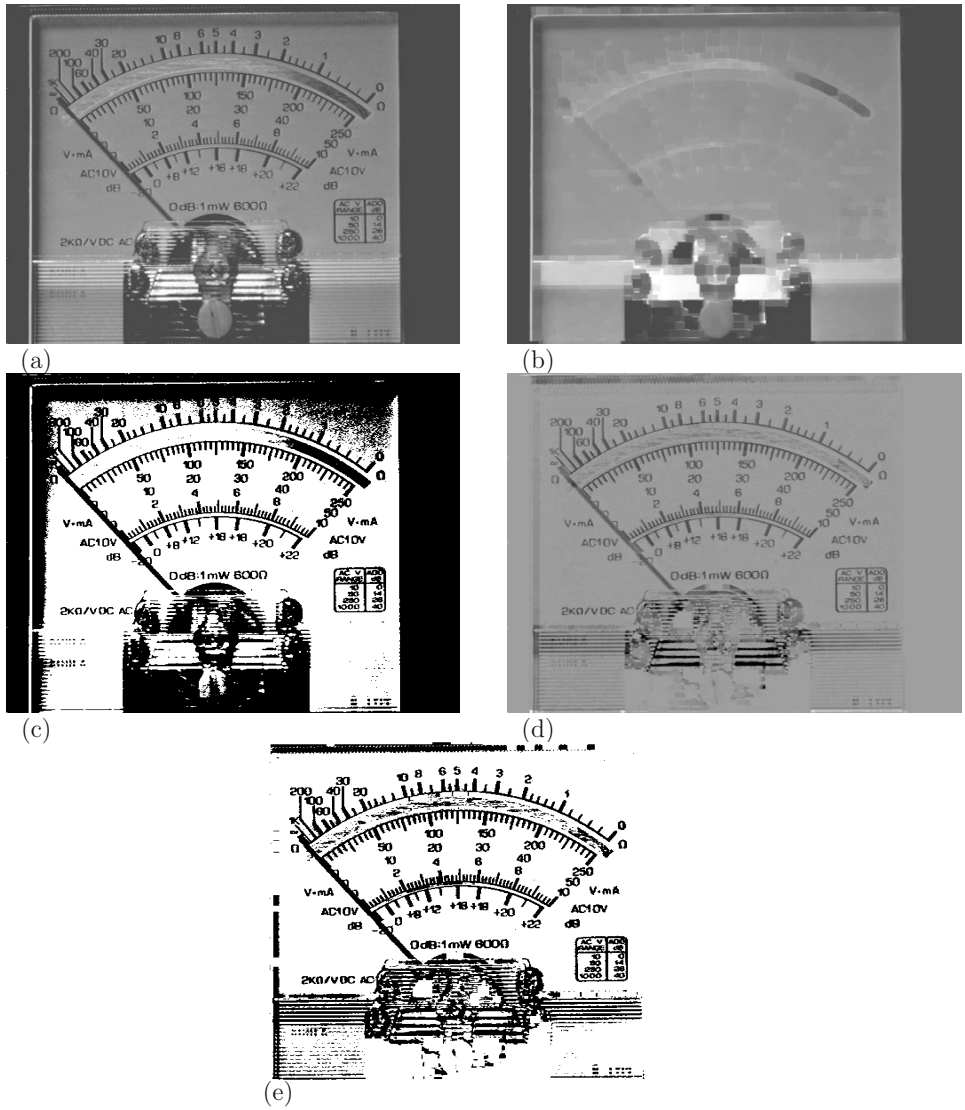


Bild 4.10: Binärbilderzeugung in Verbindung mit morphologischen Operationen. (a) Original. (b) *closing*-Operation: Vier Dilatationen gefolgt von vier Erosionen (Hintergrundbild). (c) Fixer Schwellwert ohne Korrektur des Hintergrundes. (d) Differenzbild: 127 + Original - Hintergrundbild. (e) Fixer Schwellwert nach einer Korrektur des Hintergrundes.

Durch die Wahl von c_1 und c_2 kann der Steigungsverlauf von f_n beeinflusst werden. Mit c_3 wird f_n im Grauwertintervall G nach links oder rechts verschoben.

Eine exponentielle Skalierung erhält man, wenn in (4.23) statt des Logarithmus die Exponentialfunktion verwendet wird. Es muss dann $c_1 < c_2$ gelten, die Beschränkung für c_3 bleibt wie oben bestehen.

Beispiele zur nichtlinearen Skalierung zeigen die Bilder 4.11-a bis 4.11-e. Bild 4.11-a ist das Original. Als Zielsetzung soll eine Darstellung gefunden werden, bei der sowohl das Zifferblatt als auch die LCD-Anzeige deutlich zu sehen sind. Bei Bild 4.11-b wurde eine lineare Skalierungsfunktion f_n verwendet. Das gesteckte Ziel wurde damit nicht erreicht. Bei Bild 4.11-c wurde eine logarithmische Funktion f_n mit $c_1 = 0.0001$, $c_2 = 20.0$ und $c_3 = 38$ verwendet. Der Kontrast in den dunklen Bereichen wurde merklich angehoben, was zur Folge hat, dass die LCD-Anzeige im unteren Bildbereich gut zu lesen ist. Der Kontrast in den hellen Bildbereichen wurde dagegen nicht so stark angehoben. Bild 4.11-d entstand mit Hilfe einer exponentiellen Funktion f_n ($c_1 = -6$, $c_2 = +6$, $c_3 = -155$). Hier werden die hellen Bildbereiche, also im wesentlichen der Bildhintergrund, besser kontrastiert. Die Bilder 4.11-e und 4.11-f zeigen die grafische Darstellung der Skalierungsfunktion .

An dieser Stelle sei noch vermerkt, dass die nichtlinearen Skalierungsverfahren durchaus eine Berechtigung im Rahmen der Bildvorverarbeitung für eine nachfolgende Informationsextraktion haben. Eine logarithmische Skalierung z.B. in Verbindung mit einer nachfolgenden Hochpassfilterung und einer anschließenden exponentiellen Skalierung wirkt sich so aus, dass dunkle Bildbereiche von der Hochpassfilterung stärker betroffen sind als helle.

4.7 Ebnen der Grauwerte

Das *Ebnen* eines Grauwertbildes oder die *Histogrammlinearisierung* ist ebenfalls eine nichtlineare Transformation der Grauwerte eines Bildes. Hier wird die Funktion f_n aber nicht vorgegeben, sondern aus dem zu verarbeitenden Bild berechnet und so auf die gegebene Grauwertverteilung des Bildes abgestimmt. Der Algorithmus sieht folgendermaßen aus:

A4.7: Ebnen der Grauwerte.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild mit der Grauwertmenge $G = \{0, 1, ..., 255\}$.

Algorithmus:

(a) Berechnung des Histogramms $p(g)$ des Bildes.

(b) Berechnung der relativen Summenhäufigkeiten:

$$h(g) = \sum_{k=0}^{g} p(k), \quad g = 0, 1, ..., 255;$$

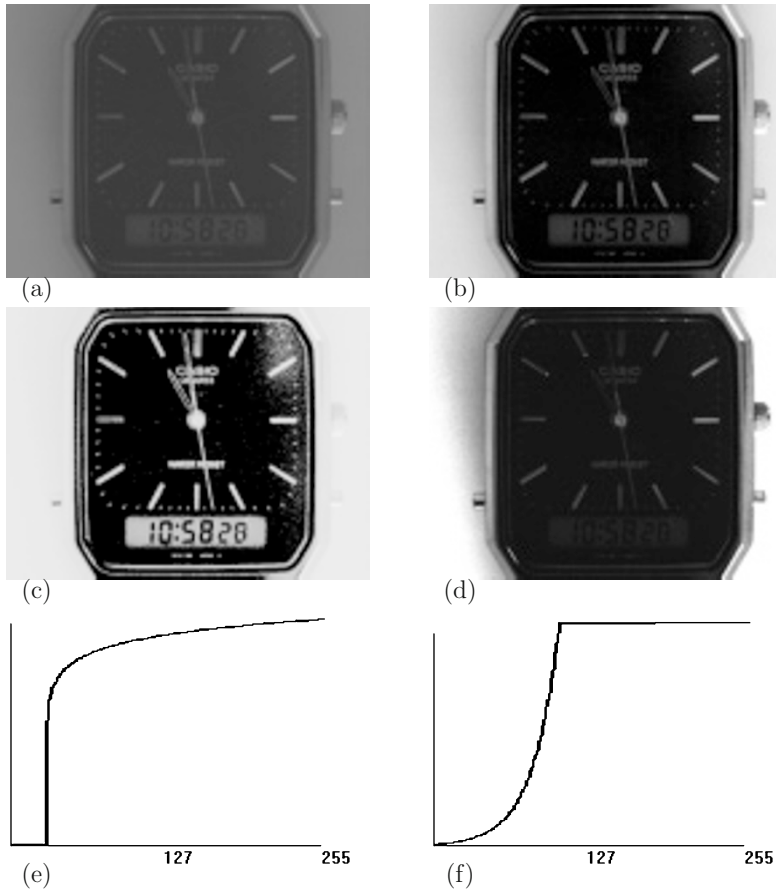


Bild 4.11: Beispiele zur nichtlinearen Skalierung. (a) Original. Als Zielsetzung soll eine Darstellung gefunden werden, bei der sowohl das Zifferblatt als auch die LCD-Anzeige deutlich zu sehen sind. (b) Lineare Skalierungsfunktion f_n: Das gesteckte Ziel wurde damit nicht erreicht. (c) Logarithmische Funktion f_n: Der Kontrast in den dunklen Bereichen wurde merklich angehoben, wodurch die LCD-Anzeige gut zu lesen ist. In den hellen Bereichen wurde der Kontrast nicht so stark verändert. (d) Exponentielle Skalierungsfunktion f_n: Es wurde vor allem der helle Bildhintergrund stark kontrastiert. (e) Skalierungsfunktion zu (c). (f) Skalierungsfunktion zu (d).

(c) Berechnung der Skalierungsfunktion f_n:
$f_n(g) = 255 \cdot h(g)$;

(d) Berechnung der *look-up*-Tabelle und Skalierung des Bildes gemäß Algorithmus **A4.1**.

Ende des Algorithmus

Wie sich diese Transformation auswirkt, wird anhand der Bilder 4.12 und 4.13 erläutert. Die Bilder 4.12-a und 4.12-b zeigen das Original und das transformierte Bild. Die Histogramme und die Skalierungsfunktionen sind in den Bildern 4.12-c und 4.12-d abgebildet. Man sieht deutlich, dass das transformierte Bild kontrastreicher ist und feine Bildstrukturen, z.B. in der linken oberen Ecke, besser zu sehen sind. Ein nochmaliges Ebnen eines bereits geebneten Bildes würde am Ergebnis nichts mehr ändern. Das sieht man auch am speziellen Verlauf der relativen Summenhäufigkeiten eines geebneten Bildes.

Wenn im Eingabebild alle 256 möglichen Grauwerte auftreten, sollte für das Ausgabebild gelten: $p(g) = \frac{1}{256} = 0.39\%$. Dass das Histogramm diesen Sachverhalt nicht richtig wiedergibt, hat drei Gründe: Erstens ist der Maßstab der Histogramme in Richtung der Ordinate sehr stark vergrößert, zweitens treten bei dieser Transformation der Grauwerte Rundungsfehler auf, und drittens fehlen im transformierten Bild einige Grauwerte, was an den Lücken im Histogramm zu erkennen ist. Wenn die daneben liegenden Grauwerte gleichmäßig auf die Lücken verteilt werden, wird der theoretische Sachverhalt besser angenähert.

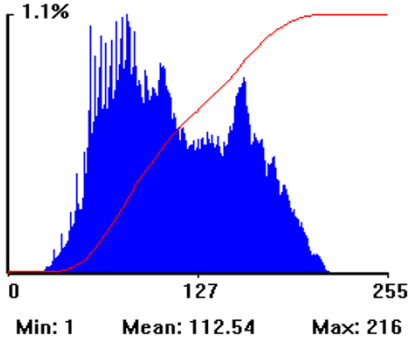
Bei manchen Bildern kann die Histogrammlinearisierung aber auch nicht geeignet sein. Meistens ist das bei Bildern der Fall, die große homogene Flächen mit nahe beieinander liegenden Grauwerten haben. Als Beispiel dient Bild 4.13-a. Es hat ein bimodales Histogramm (Bild 4.13-c). Die relativen Summenhäufigkeiten und somit auch der Verlauf der Skalierungsfunktion f_n sind in das Histogramm eingeblendet. Die Funktion verdeutlicht, dass sowohl in dunklen als auch in hellen Bildbereichen der Kontrast angehoben wird, während er bei den mittleren Grauwerten nahezu unverändert bleibt. Das hat zur Folge, dass z.B. im helleren Bildbereich des transformierten Bildes 4.13-b das Rauschen stark hervortritt. Bild 4.13-d zeigt das Histogramm des skalierten Bildes. Hier gilt dasselbe wie bei Bild 4.12-d.

Man hat hier also eine parameterfreie Methode. Wo kann sie sinnvoll angewendet werden? Sollen digitalisierte Grauwertbilder über einfache Schwarz-/Weißdrucker ausgegeben werden, so hat man das Problem, dass diese Ausgabegeräte in der Regel nicht über die 256 verschiedenen Graustufen verfügen, wie ein Monitor des Bildverarbeitungsystems. Ist das Originalbild verhältnismäßig kontrastarm, so wird es nach der Ausgabe noch kontrastärmer sein. Gibt man jedoch das geebnete Bild aus, so hat man in der Regel eine ausreichend gute Darstellung.

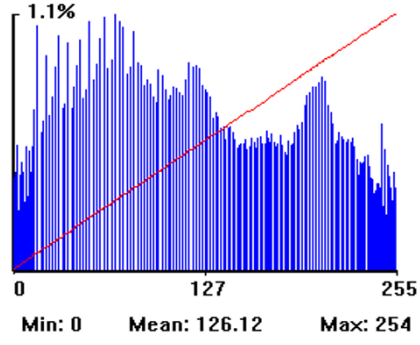
Ein weiteres Beispiel zum Ebnen der Grauwerte ist das Angleichen von Farbauszügen. Ein RGB-Bild kann mit Hilfe einer Farbvideokamera und eines Farbvideo-Digitalisierers aufgezeichnet werden. Für einen Bildpunkt werden von Systemen dieser Art meistens 24

(a)

(b)



(c)



(d)

Bild 4.12: Beispiele zum Ebnen der Grauwertverteilung. (a) Original (Amalienburg im Schloßpark Nymphenburg in München; aus dem Stadtarchiv). (b) Transformiertes Bild. Man sieht deutlich, dass das Bild kontrastreicher als das Original ist. Feine Bildstrukturen, wie z.B. in der linken oberen Bildecke, sind besser zu sehen. (c) Histogramm und Skalierungsfunktion, gebildet aus den relativen Summenhäufigkeiten des Originalbildes. (d) Histogramm und relative Summenhäufigkeiten des transformierten Bildes. Ein nochmaliges Ebnen eines bereits geebneten Bildes würde nichts mehr verändern. Das sieht man auch an der speziellen Form der relativen Summenhäufigkeiten des geebneten Bildes.

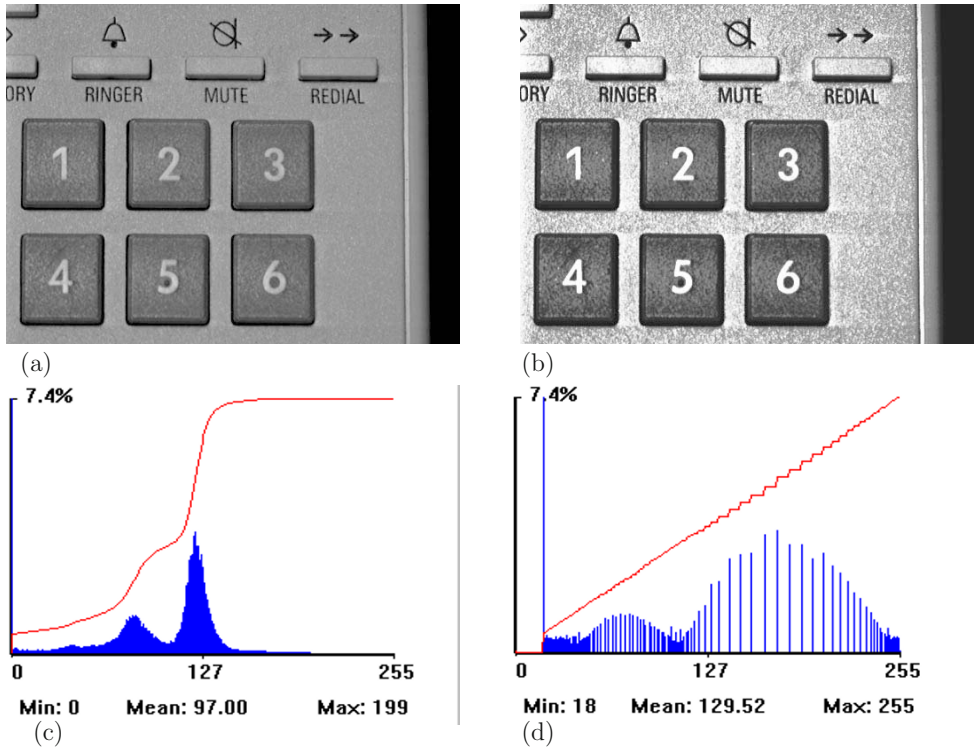
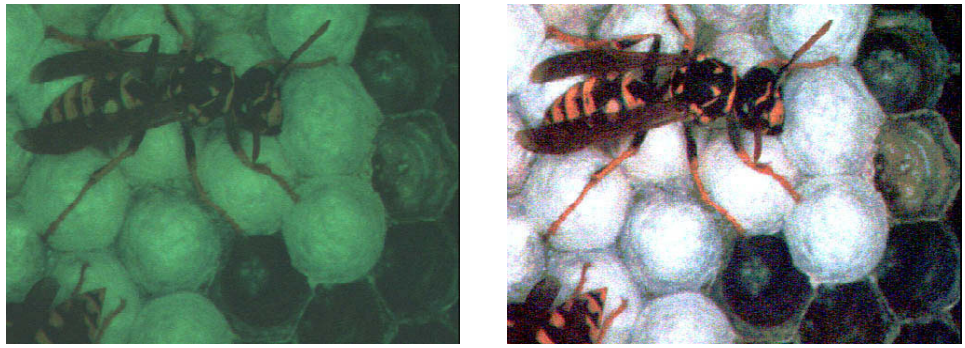


Bild 4.13: Beispiele zum Ebnen der Grauwertverteilung. (a) Testbild. (b) Transformiertes Bild. In hellen Bildbereichen tritt das Rauschen deutlich hervor. Das liegt daran, dass das Original größere homogene Bildbereiche mit ähnlichen Grauwerten besitzt. Die Grauwerte dieser Intervalle werden im Ausgabebild stark aufgespreizt. (c) Histogramm und Skalierungsfunktion des Originals. (d) Histogramm und relative Summenhäufigkeiten des transformierten Bildes. Das linke Häufigkeitsmaximum entspricht dem Bereich mit dem Grauwert 0 am rechten Bildrand des Originals. Im Histogramm des Originals ist dieser Balken durch die Ordinate verdeckt.



(a) (b)

Bild 4.14: RGB-Farbbilder, die mit Rot-, Grün- und Blaufiltern und einem normalen 8-Bit-Framegrabber erzeugt wurden. Als Vorlage zur Digitalisierung diente hier ein Farbdruck. Die Voraussetzung, dass sich der zu digitalisierende Bereich nicht bewegt, war somit gegeben. (a) Originalbild nach der Digitalisierung. (b) Werden die Farbauszüge getrennt geebnet, so ergibt sich oft eine bessere Farbqualität.

Bit verwendet. Bei Anwendungen, bei denen sich das Beobachtungsgebiet nicht bewegt, kann auch mit einer Schwarz-/Weißkamera, einem normalen 8-Bit Video-Digitalisierer und Farbfiltern gearbeitet werden. Nacheinander werden hier drei Digitalisierungen mit einem Rot-, einem Grün- und einem Blaufilter durchgeführt.

Um eine gute farbliche Darstellung zu erzielen, können die drei Farbauszüge getrennt voneinander mit den hier erläuterten Skalierungsverfahren behandelt werden. Bei der Verwendung von Farbfiltern, deren spektrale Bandbreiten nicht genau bekannt sind und die auch nicht exakt zusammenpassen, ergeben sich Farbdarstellungen, die oft weit vom Original entfernt sind. Praktische Erfahrungen haben gezeigt, dass das Ebnen der drei Farbauszüge oft die Farbqualität wesentlich verbessert. Bildbeispiele dazu zeigen die Bilder 4.14-a und 4.14-b.

4.8 Kombination einer Grauwertskalierung mit einer Hochpassfilterung

Steht bei der Anwendung der Grauwertskalierung das Erzeugen eines hinsichtlich der Helligkeit und des Kontrastes guten Bildes im Vordergrund, so kann dies durch die Kombination mit einem hochpassgefilterten Bild erzielt werden. Der theoretische Hintergrund ist in Bild 4.15 vereinfacht am Beispiel einer eindimensionalen Funktion $s(x)$ dargestellt. Der Verlauf von $s(x)$ (Bild 4.15-a) kann als Querschnittsprofil durch eine Grauwertkante in einem Bild

interpretiert werden. Die erste und zweite Ableitung zeigen die Bilder 4.15-b und 4.15-c. Wird die zweite Ableitung der Originalfunktion $s(x)$ überlagert, z.B. durch $s(x) - s''(x)$, so ergeben sich in den Bereichen der Funktion mit starken Steigungsänderungen „Überschwinger“ (Bild 4.15-d).

Auf ein Grauwertbild übertragen heißt das, dass sich eine Grauwertkante besser abhebt, weil sie von einem schmalen, zuerst hellen, dann dunklen Streifen eingesäumt wird.

Als diskrete Nachbildung der zweiten Ableitung eines Bildes kann z.B. der Laplace-Operator (Grundlagen dazu in Kapitel 5) verwendet werden. Als Filterkerne können die folgenden Masken dienen:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \text{oder} \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{4.24}$$

Im Folgenden ist der Algorithmus dazu kurz zusammengestellt:

A4.8: Grauwertskalierung mit einer Hochpassfilterung.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y))$ ein einkanaliges Grauwertbild.

Algorithmus:

(a) Berechnung des skalierten Bildes $\mathbf{T} = (t(x, y))$ durch eine Skalierungsfunktion f_n.

(b) Anwendung des Laplace-Operators auf das Originalbild $\mathbf{S}_e$. Das Ergebnis sei $\mathbf{S}''$.

(c) Überlagerung von $\mathbf{S}''$ und des skalierten Bildes $\mathbf{T}$ etwa gemäß:

$t(x, y) - c \cdot s''(x, y)$;

Der Parameter c steuert das Gewicht der Überlagerung. Er sollte bei praktischen Anwendungsfällen anhand einiger Testbilder ermittelt werden.

Ende des Algorithmus

Die Bildfolge 4.16-a bis 4.16-c zeigt Beispiele dazu. Hier wurde das Originalbild mit einer linearen Skalierungsfunktion transformiert (Bild 4.16-b). Das skalierte Bild wurde dann mit dem Laplace-gefilterten Original überlagert. Da beide Teilverfahren mit geeigneter Bildverarbeitungshardware in Videoechtzeit durchgeführt werden können, lässt sich auch das hier erläuterte „Schärfen“ eines Grauwertbildes (*image sharpening*) in Videoechtzeit durchführen.

Gute Ergebnisse erhält man in der Regel auch, wenn man bei etwas verrauschten Bildern zuerst eine Medianfilterung (Kapitel 6) durchführt und dann das Ergebnis, wie oben beschrieben, hochpassfiltert.

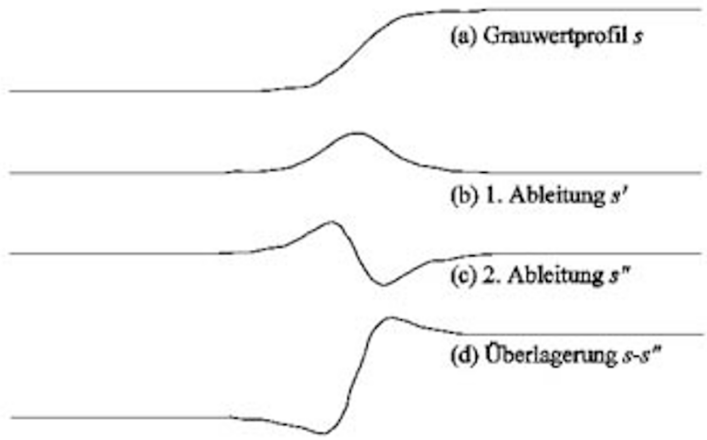


Bild 4.15: (a) Grauwertprofil $s(x)$ einer Grauwertkante in einem Bild. (b) und (c) Erste und zweite Ableitung von $s(x)$. (d) Überlagerung der Originalfunktion mit ihrer zweiten Ableitung, gemäß $s(x) - s''(x)$.

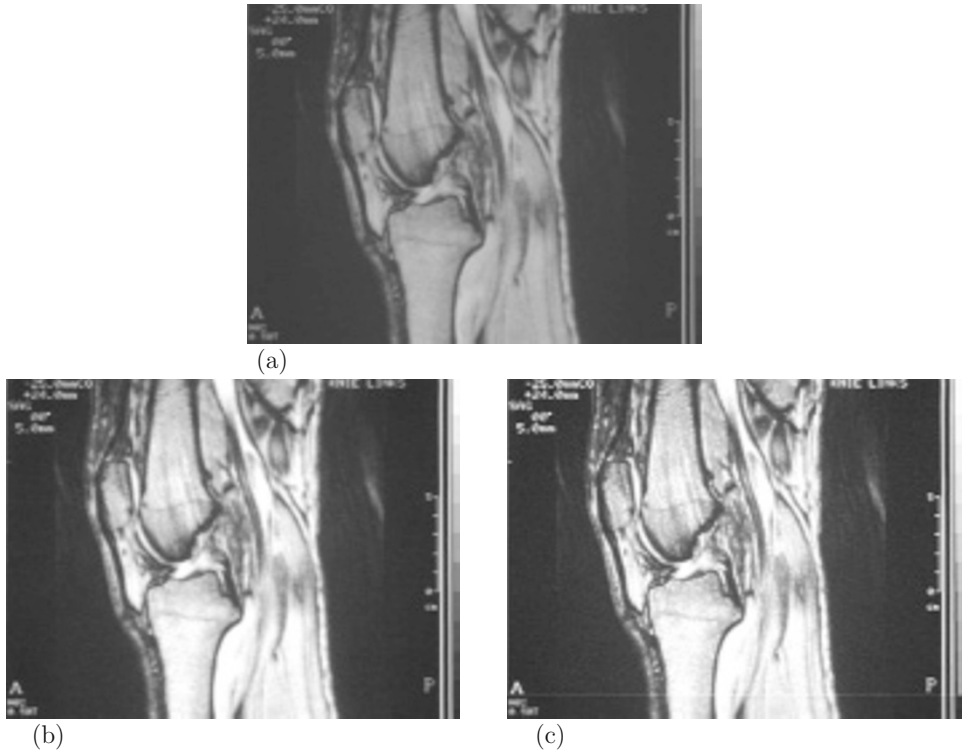
4.9 Kalibrierung der Grauwerte

Digitalisierungssysteme (allgemeiner: bildaufzeichnende Systeme) sollten monoton arbeiten. Damit ist gemeint, dass gleiche Grautöne, die im Original an verschiedenen Stellen auftreten, im digitalisierten Bild denselben Grauwert haben sollten. Das ist in der Praxis selten der Fall. Bei einer fotografischen Aufnahme können z.B. bereits bei der Belichtung des Fotomaterials *Randabschattungen* (*Vignettierungen*) auftreten, die sich je nach der Qualität des Objektivs mehr oder weniger stark auswirken. Selbst eine fehlerfreie Digitalisierung kann diesen Effekt nicht mehr eliminieren. Als weitere verfälschende Einflüsse können aufgezählt werden: die Filmentwicklung, die unterschiedliche Ablenkung des Kathodenstrahls bei der Aufzeichnung mit einer Videokamera oder die unterschiedlichen Arbeitspunkte der einzelnen Sensoren eines Abtasters mit einem Sensorfeld.

Zunächst wird untersucht, wie eine multiplikative Störung der Grauwerte kalibriert werden kann. Es sei $\mathbf{S}_e = (s_e(x, y))$ das digitalisierte Grauwertbild mit der überlagerten Störung. Dann kann für die Grauwerte von $\mathbf{S}_e$ geschrieben werden:

$$s_e(x, y) = e(x, y) \cdot s_u(x, y), \tag{4.25}$$

wobei $e(x, y)$ der multiplikative Störfaktor und $s_u(x, y)$ der ungestörte Grauwert in der Position (x, y) ist. Die Korrektur des bildaufzeichnenden Systems, einschließlich der Digitalisierung, wird mit Hilfe eines Kalibrierungsbildes, z.B. einer homogenen, grauen Vorlage, durchgeführt:



(a)

(b)

(c)

Bild 4.16: Skalierung der Grauwerte in Verbindung mit einer Hochpassfilterung. (a) Kernspintomografie des linken Knies. (b) Skaliertes Bild mit einer linearen Skalierungsfunktion f_n. (c) Zusätzliche Überlagerung des skalierten Bildes mit dem Laplace-gefilterten Original. Man sieht, dass feine Linienstrukturen besser sichtbar werden.

$\mathbf{S}_k = (s_k(x,y))$ digitalisiertes Kalibrierungsbild,
$\mathbf{S}_{k'} = (s_{k'}(x,y))$ digitalisiertes Kalibrierungsbild, das sich bei einer fehlerfreien Aufzeichnung und Digitalisierung ergeben müßte.

Für $\mathbf{S}_{k'}$ kann $s_{k'}(x,y) = c$ angenommen werden, da eine fehlerfrei aufbereitete graue Vorlage ein homogen graues Bild liefern müßte. Mit diesem Kalibrierungsbild können jetzt die Störfaktoren ermittelt werden:

$$s_k(x,y) = e(x,y) \cdot c \text{ oder } e(x,y) = \frac{s_k(x,y)}{c}. \tag{4.26}$$

Die Grauwerte des korrigierten Bildes berechnen sich daraus gemäß:

$$s_u(x,y) = \frac{s_e(x,y)}{e(x,y)} = c \cdot \frac{s_e(x,y)}{s_k(x,y)} \tag{4.27}$$

Ein anderes Kalibrierungsproblem tritt bei Abtastgeräten auf, die eine Bildzeile parallel mit einem Feld von Sensoren erfassen (Bild 4.17-a), wie z.B. ein Scanner. Ein Problem dabei ist, dass die einzelnen Sensoren nicht alle so exakt abgestimmt werden können, dass sie für denselben Grauton nach der Digitalisierung auch denselben Grauwert liefern. Ein so digitalisiertes Bild ist dann in Zeilenrichtung durch Streifen gestört.

Der Sensor y (abkürzend für: der Sensor, der die Bildspalte y abtastet) und der Sensor $y+1$ erzeugen somit für denselben Grauton die Grauwerte

$$c + f_y(g) \text{ und } c + f_{y+1}(g). \tag{4.28}$$

Die Korrekturfunktion $f_y(g)$ für die Bildspalte y ist abhängig vom Grauwert g, da der Sensor im dunklen Grautonbereich anders arbeitet als im hellen. Wenn in einem bestimmten Grautonbereich ein linearer Verlauf der Sensorempfindlichkeit angenommen werden kann, ist es möglich, mit einer hellen und einer dunklen Kalibrierungsvorlage die Korrekturfunktion $f_y(g)$ als Gerade zu ermitteln. Es sei:

$\mathbf{S}_H = \big(s_H(x,y)\big) = \big(c_H + f_y(g)\big)$ das helle, digitalisierte Kalibrierungsbild,
$\mathbf{S}_D = \big(s_D(x,y)\big) = \big(c_D + f_y(g)\big)$ das dunkle, digitalisierte Kalibrierungsbild,
$m_{y,H}$ der Mittelwert der Bildspalte y im hellen Kalibrierungsbild und
$m_{y,D}$ der Mittelwert der Bildspalte y im dunklen Kalibrierungsbild.

Mit diesen Voraussetzungen berechnet sich die Korrekturgerade für die Bildspalte y zu:

$$f_y(g) = \big(m_{y,H} - m_{y,D}\big) \cdot \frac{g - c_D}{c_H - c_D} + m_{y,D}. \tag{4.29}$$

Für c_D und c_H kann der Mittelwert des dunklen oder hellen Kalibrierungsbildes verwendet werden. Bild 4.17-b zeigt die grafische Darstellung der Kalibrierungsgeraden. Die Bilder 4.17-c und 4.17-d zeigen ein praktisches Beispiel.

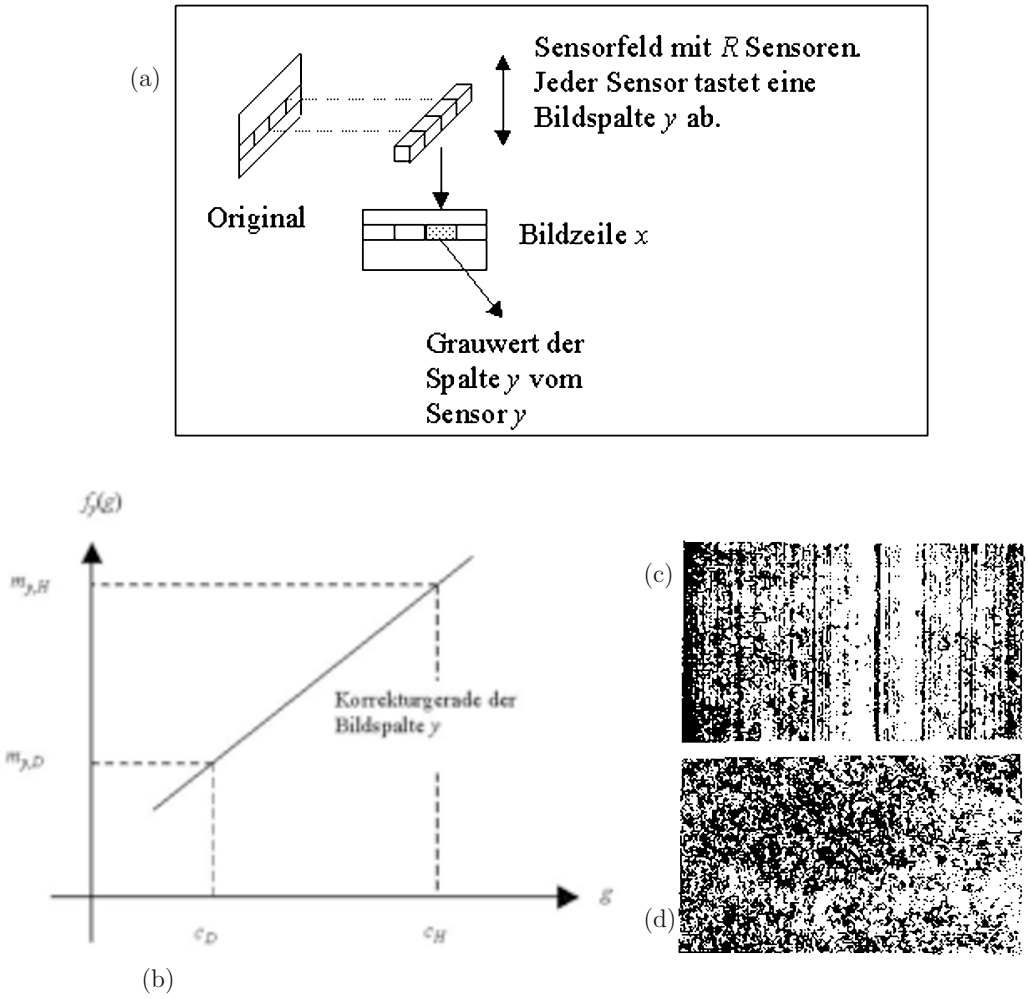


Bild 4.17: Kalibrierung bei CCD-Scannern. (a) Abtastung einer Vorlage mit einem Sensorfeld, das eine Bildzeile parallel erfasst. (b) Grafische Darstellung der Korrekturgeraden für die Bildspalte y. (c) Streifenmuster vor der Kalibrierung mit starker Kontrastverstärkung. (d) Kalibriertes Bild.

4.10 Berechnung einer neuen Grauwertmenge

Wird zu einem digitalisierten Grauwertbild $\mathbf{S}_e = (s_e(x,y))$ das Histogramm $p_{\mathbf{S}_e}(g)$ berechnet, so zeigt es oft eine Grauwertverteilung, die nicht alle Grauwerte der Grauwertmenge $G = \{0, 1, ..., 255\}$ umfasst. In solchen Fällen können die Grauwerte, wie in diesem Kapitel beschrieben, skaliert werden, sodass sie die gesamte Grauwertmenge ausnützen. Die Bilddaten können aber auch komprimiert gespeichert werden, wenn sich zeigt, dass statt der ursprünglichen 8 Bit weniger als 8 Bit pro Bildpunkt ausreichend sind. Das ist z.B. dann der Fall, wenn

$$\Delta g = g_{max} - g_{min} \leq 128, \tag{4.30}$$

wobei g_{min} und g_{max} der minimale und der maximale im Bild $\mathbf{S}_e$ auftretende Grauwert sind. Wie viele Bit zur komprimierten Speicherung notwendig sind, ergibt sich aus der Größe der Differenz Δg in (4.30). Falls z.B. $127 \geq \Delta g \geq 64$ werden 7 Bit entsprechend 128 Grauwerten, also $G' = \{0, 1, ..., 127\}$ oder falls $63 \geq \Delta g \geq 32$ werden 6 Bit, entsprechend 64 Grauwerten, also $G' = \{0, 1, ..., 63\}$ benötigt.

Einen weiteren Hinweis auf die Anzahl der benötigten Bit pro Bildpunkt gibt die Entropie (Abschnitt 3.10):

$$H = -\sum_{g=0}^{255} \Big(p_{\mathbf{S}_e}(g) \cdot \log_2 p_{\mathbf{S}_e}(g) \Big). \tag{4.31}$$

Für das Bild 4.1-a in Kapitel 4.3 ist der minimale Grauwert $g_{min} = 2$ und der maximale Grauwert $g_{max} = 255$. Die Differenz $\Delta g = 253$ zeigt, dass das Bild ohne Informationsverlust nicht nach der hier beschriebenen Methode mit weniger als 8 Bit pro Bildpunkt gespeichert werden kann. Dies bestätigt auch der Wert der Entropie $H = 7.51$.

Lässt man geringe Informationsverluste zu, die den visuellen Eindruck der Bilder oft nicht beeinflussen, so kann mit den im Folgenden beschriebenen Beispielen eine weitere Erhöhung der Speicherungsdichte erreicht werden. Zu Bild 4.1-a sind in Bild 4.18 die Binärbilder der einzelnen Bitebenen dargestellt. Das Binärbild zur Bitebene $i, 0 < i < 7$, wird dadurch erzeugt, dass von jedem Grauwert $s(x,y)$ in seiner Darstellung als Dualzahl nur das Bit in der Position i bildlich dargestellt wird:

$$s_i(x,y) = \begin{cases} 1 & \text{falls } \Big(s_e(x,y).\text{AND}.2^i \Big) \neq 0 \\ 0 & \text{sonst.} \end{cases} \tag{4.32}$$

Beim Binärbild zur Bitebene $i = 0$ wird so z.B. von jedem Grauwert nur das niederwertigste Bit dargestellt.

Bild 4.18 zeigt, dass die Bitebenen 0 und l fast keine Bildinformation enthalten und daher weggelassen werden können. Das bedeutet in diesem Beispiel eine Reduktion von 8 auf 6 Bit pro Bildpunkt, also von 256 auf 64 verschiedene Grauwerte.

Bei allen Ermittlungen einer neuen Grauwertmenge sollte noch der resultierende Kompressions- oder Reduktionsfaktor gegenüber der Tatsache abgewogen werden, dass dann

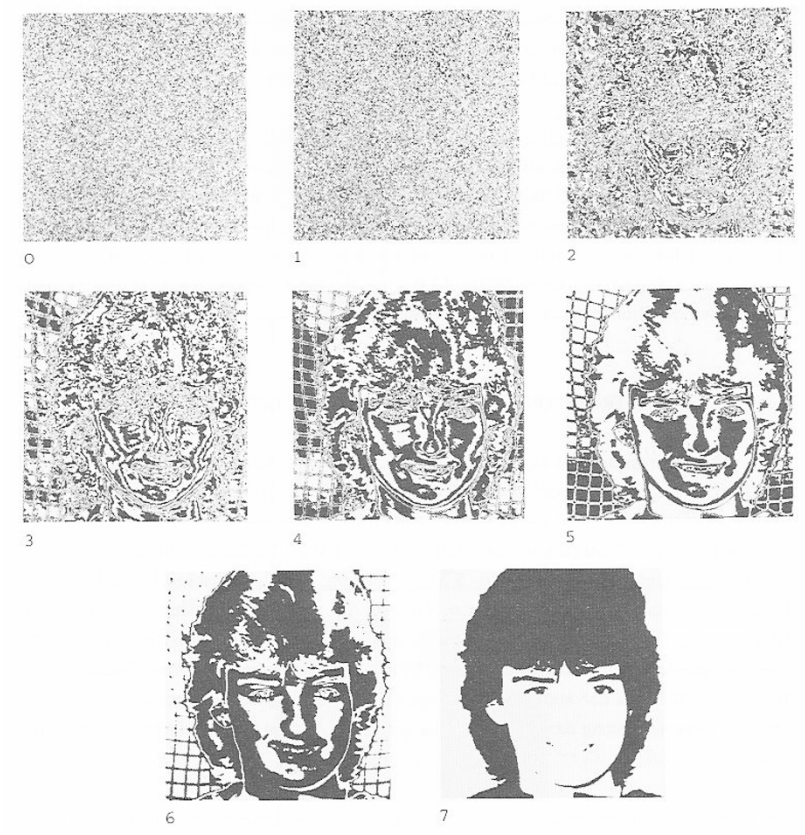


Bild 4.18: Binärbilder der einzelnen Bitebenen zu Bild 4.1-a. Man sieht deutlich, dass die Bitebenen 0 und l fast keine Bildinformation enthalten und daher weggelassen werden können.

der Grauwert eines Bildpunktes nicht mehr ein ganzes Byte belegt, wodurch sich u.U. erhöhte Rechenzeiten bei den Zugriffen auf die einzelnen Bildpunkte ergeben. So wäre z.B. eine Datenreduktion von 8 auf 7 Bit in einer byteorientierten Rechenanlage nicht sinnvoll, da bei jedem Zugriff auf einen Bildpunkt der zugehörige Grauwert durch Maskieren und Verschieben der Bitmuster von zwei Byte ermittelt werden muss.

4.11 Rekonstruktion des höchstwertigen Bit

Auch das höchstwertige Bit eines Grauwertbildes $\mathbf{S}_e = (s_e(x, y)$ kann zur datenkomprimierten Speicherung weggelassen werden. Das Original lässt sich aus der komprimierten Darstellung wieder rekonstruieren, wenn die Annahme zutrifft, dass zwischen benachbarten Bildpunkten kein allzu abrupter Übergang in den Grauwerten ist. Dieses Verfahren lässt sich am besten am Beispiel eines Graukeils (Bild 4.19-a) erläutern. Die grafische Darstellung des Grauwertverlaufs entlang der markierten Strecke $\overline{AB}$ ergibt ein Geradenstück. Wird das höchstwertige Bit 7, das nur bei den Grauwerten $128 \leq s(x, y) \leq 255$ gesetzt ist, maskiert, erhält man den gestrichelten Verlauf (Bild 4.19-b). Den dazugehörigen modifizierten Graukeil zeigt Bild 4.19-c. Soll aus der komprimierten Darstellung das Original rekonstruiert werden, so wird der Grauwertsprung daran erkannt, dass die Differenz zum rechten Nachbarn größer als l ist.

Bei natürlichen Bildern ist die Rekonstruktion nicht ganz so einfach, da man zunächst nicht weiß, ob der Grauwert am Zeilenanfang ursprünglich im Intervall $\left[0, 127\right]$ oder $\left[128, 255\right]$ lag. Wird jedoch diese Information in einem Bit zusätzlich mit abgespeichert, so kann bei der Rekonstruktion an jedem Zeilenanfang eine Hilfsgröße richtig auf 0 oder 128 gesetzt werden. Außerdem können im Verlauf der Grauwerte einer Bildzeile mehrere Sprungstellen auftreten. Bei jeder Sprungstelle muss dann die Hilfsgröße auf 128 oder 0 umgeschaltet werden. Schließlich muss noch für die Schwelle c ein geeigneter Wert festgelegt werden, der davon abhängt, wie groß die Grauwertübergänge im Original sind. Bild 4.20 zeigt das mit diesem Verfahren komprimierte Bild 4.1-a.

Nachdem in Bild 4.1-a die Bitebenen 0 und l weggelassen werden können und die Bitebene 7 durch stetige Fortsetzung rekonstruiert werden kann, lassen sich somit die Grauwerte ohne wesentlichen Informationsverlust mit 5 Bit pro Bildpunkt darstellen.

4.12 Reduktion der Grauwertmenge durch Differenzbildung

Eine weitere Möglichkeit der Reduktion (Kompression) der zur Darstellung eines Grauwertbildes notwendigen Grauwertmenge ergibt sich, wenn nicht wie bisher jeder Bildpunkt getrennt für sich, sondern im Rahmen seiner 4- oder 8-Nachbarschaft betrachtet wird. Ein einfaches Beispiel hierzu ist die Berechnung und Speicherung der Grauwertdifferenzen zum jeweiligen rechten Nachbarn innerhalb einer Bildzeile:

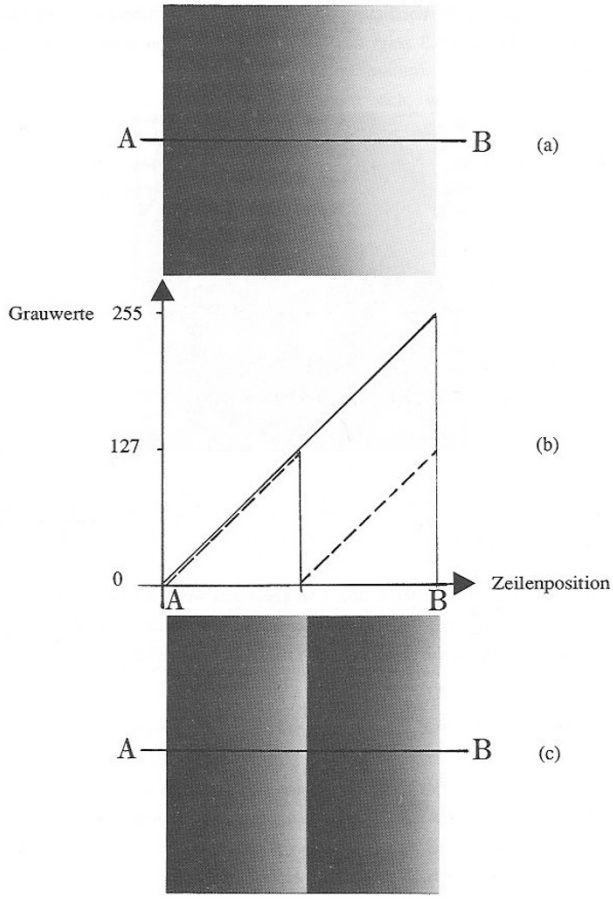


Bild 4.19: Rekonstruktion des höchstwertigen Bit. (a) Graukeil (b) Grafische Darstellung des Grauwertverlaufs entlang der Strecke $\overline{AB}$ vor und nach dem Ausblenden des höchstwertigen Bit 7. (c) Graukeil, bei dem Bit 7 ausgeblendet wurde.

Bild 4.20: Testbild mit ausgeblendetem Bit 7. Das Original kann durch stetige Fortsetzung der Grauwerte innerhalb der Bildzeilen rekonstruiert werden.

$$\mathbf{S}_e \rightarrow \mathbf{S}_a :$$
$$s_a(x,y) = s_e(x,y) - s_e(x,y+1) + c,\ 0 \le x \le L-1,\ 0 \le y \le R-2. \tag{4.33}$$

Die Konstante c muss dabei so gewählt werden, dass negative Grauwerte vermieden werden. Das Verfahren lässt sich am eindrucksvollsten wieder anhand des Testbildes 4.1-a darstellen. Bild 4.21-a zeigt die bildliche Darstellung des transformierten Originals, das aus den Grauwertdifferenzen zum jeweiligen linken Zeilennachbarn besteht. Bei der Differenzberechnung in (4.33) wurde $c = 127$ verwendet. Zur Verdeutlichung wurde das Differenzenbild etwas im Kontrast angereichert. Das dazugehörige Histogramm zeigt Bild 4.21-b. Daraus kann abgeleitet werden, dass die Differenzen etwa im Intervall $-23 \le s_a(x,y) \le 25$ liegen. Zur Darstellung dieser 48 verschiedenen Grauwerte werden dann 6 Bit pro Bildpunkt, also $G' = \{0, 1, ..., 63\}$ verwendet. Da die Standardabweichung der Differenzen sich zu $\sigma = 7.7$ berechnet, kann das Differenzenbild ohne allzu großen Informationsverlust auch mit weniger Grauwerten datenreduziert gespeichert werden, z.B. mit 4 Bit pro Bildpunkt, was einem maximalen Grauwertdifferenzbetrag von 16 Grauwerten entspricht.

Um die Rekonstruktion zu ermöglichen, wird z.B. in jeder Bildzeile der erste Bildpunkt mit seinem Originalgrauwert $s(x,0)$ gespeichert. Zu den in der Zeile folgenden Bildpunkten wird nur mehr die Differenz zum rechten Nachbarn gemäß (4.33) abgelegt. Bei der Rekonstruktion kann jetzt, ausgehend vom Originalgrauwert am Zeilenanfang, durch Addieren oder Subtrahieren der jeweiligen Differenz der ursprüngliche Grauwert ermittelt werden. Es genügt, auch nur den Originalgrauwert in der Position $(0,0)$ zu speichern. Beim Übergang in eine neue Bildzeile ist dann der Grauwert $s_a(x,0)$ die Differenz zum Originalgrauwert.

Das hier erläuterte Differenzbildungsverfahren ist nur ein einfaches Beispiel der Speicherung von Transformationskoeffizienten anstatt der Originalgrauwerte. Weitere Möglichkei-

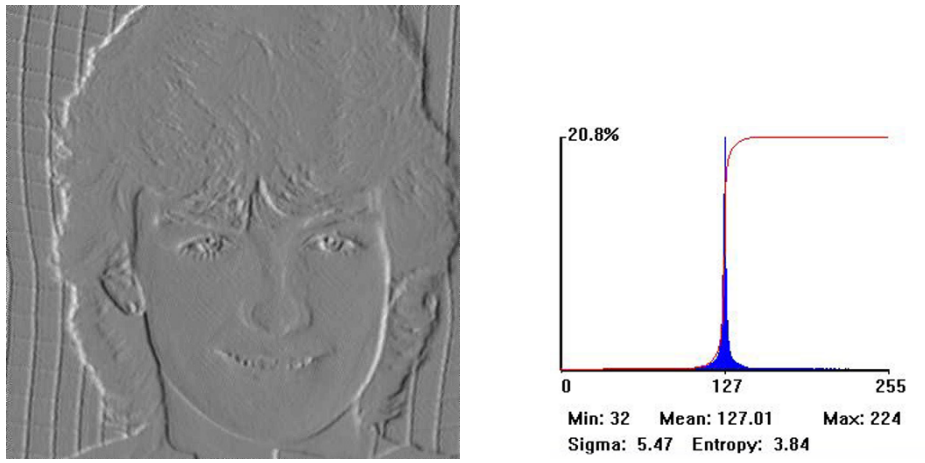


Bild 4.21: Differenzbildung (a) Bildliche Darstellung der Grauwertdifferenzen zum jeweils rechten Zeilennachbarn. Zur Verdeutlichung wurde das Bild etwas im Kontrast angehoben, (b) Histogramm zu (a). Zu den Grauwertdifferenzen wurde, um negative Grauwerte zu vermeiden, jeweils die Konstante 127 addiert.

ten sind z.B. die Speicherung der Fourierkoeffizienten (Abschnitt 8.4) oder die Speicherung der Koeffizienten einer Hauptkomponententransformation (Abschnitt 13.4).

Kapitel 5

Operationen im Ortsbereich

Die Grauwerttransformationen von Kapitel 4 waren rein bildpunktbezogen, d.h. es wurden keine Grauwertinformationen von benachbarten Bildpunkten verwendet. Es gibt nun aber auch viele Operationen, die ausgehend von einem bestimmten Bildpunkt, eine Umgebung dieses Bildpunktes berücksichtigen. Da hier Umgebungen eines Bildpunktes mit den Ortskoordinaten (x, y) verwendet werden, bezeichnet man derartige Verfahren als *Operationen im Ortsbereich*.

Ein einfaches Beispiel ist die Mittelung der Grauwerte in einer Umgebung des Bildpunktes in der Position (x, y). Ein so erzeugtes Ausgabebild wird, verglichen mit dem Original, unschärfer wirken, da sich die Grauwertübergänge (Grauwertkanten) nicht mehr so deutlich ausprägen. Man kann auch sagen, dass das „Grauwertgebirge" geglättet wird. Zur Verdeutlichung dieser Operation ist in der Bildfolge 5.1-a bis 5.1-d ein Beispiel gegeben. Bild 5.1-a zeigt das Original eines Grauwertbildes, 5.1-b das geglättete Bild. Darunter ist in 5.1-c das Original als Projektion des dreidimensionalen „Grauwertgebirges" mit Hilfe einer Liniengrafik abgebildet. Schließlich zeigt 5.1-d das geglättete Bild als Liniengrafik.

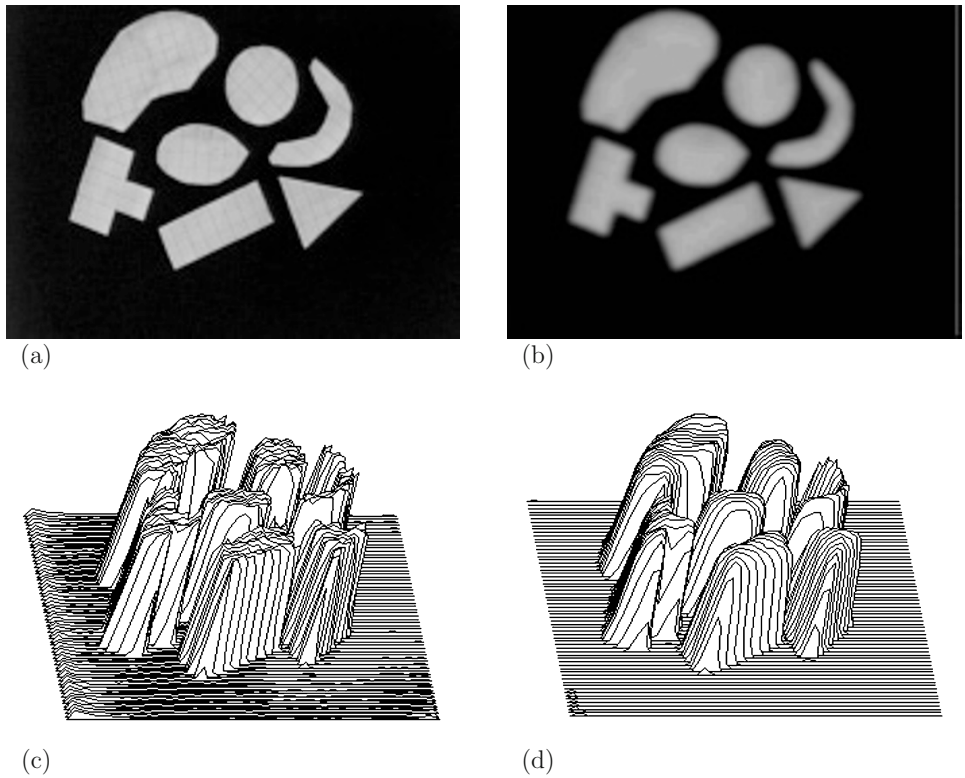
Andere Operationen dieser Art werden verwendet, um die Grauwertkanten deutlicher hervorzuheben. Sie können damit ein Verarbeitungsschritt in Richtung Kantenextraktion sein. Wieder andere derartige Operatoren werden zur Elimination von gestört aufgezeichneten Bildpunkten oder Bildzeilen verwendet. Eine wichtige Klasse von Operatoren wird unter dem Oberbegriff der Rangordnungsoperatoren zusammengefasst. Beispiele dazu sind die Erosion und die Dilatation. Diese Operationen werden in Kapitel 6 behandelt.

Einige Operationen dieser Art können als digitale Filter im Ortsbereich interpretiert werden. Der Zusammenhang zwischen den digitalen Filtern im Ortsbereich und den digitalen Filtern im Frequenzbereich wird in Kapitel 8 dargestellt. Die Verfahren in diesem Kapitel können auch als Vorverarbeitungsschritte verstanden werden, wenn sich z.B. eine Bildsegmentierung (Kapitel 11) anschließt.

Nach einer Übersicht der Anwendungen, die zu den hier dargestellten Verfahren passen, folgt ein kurzes Grundlagenkapitel zu Filteroperationen im Ortsbereich. Auf der Grundlage dieser Verfahren werden im Folgenden die unterschiedlichsten Anwendungen erläutert.

Beispielimplementierungen der Filteroperationen im Ortsbereich für die Bildverarbeitung auf Grafikkarten sind in Band I Kapitel 13.6 beschrieben.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_5



(a) (b)

(c) (d)

Bild 5.1: Beispiel zur Glättung des „Grauwertgebirges“. (a) Original. (b) Geglättetes Original. (c) Darstellung des Originals als zweidimensionale Projektion mit Hilfe einer Liniengrafik. (d) Geglättetes Bild als Liniengrafik.

5.1 Anwendungen

Bilddaten, die mit Videosensoren aufgezeichnet werden, sind oft mehr oder weniger stark verrauscht. Dieses Rauschen ist bei einer rein visuellen Verarbeitung der Bilder in der Regel nicht störend und teilweise auch gar nicht sichtbar. Werden diese Bilddaten aber weiteren Verarbeitungsschritten, wie z.B. der Bildsegmentierung, zugeführt, so können sich die Einflüsse des Rauschens verfälschend auf die Ergebnisse auswirken.

In diesem Kapitel werden Verfahren erläutert, bei denen Glättungsoperatoren eingesetzt werden, um z.B. verrauschte Einzelbilder zu verbessern. Eine Bildverbesserung kann auch durch eine Bildakkumulation entlang der Zeitachse bei einer Bildsequenz erzielt werden.

Ein anderes Anwendungsgebiet ist das Herausarbeiten von Kanten und Linien, die in einem Bild enthalten sind. Die in diesem Kapitel erläuterten Differenzenoperatoren sind ein erster Schritt in diese Richtung. Die ausführliche Untersuchung von Kanten und Linien erfolgt in Kapitel 7.

Außerdem werden Algorithmen besprochen, die es erlauben, einzelne gestörte Pixel oder gestörte Bildzeilen zu eliminieren.

5.2 Grundlagen: Filteroperationen im Ortsbereich

Als Eingabebild wird ein einkanaliges Grauwertbild $\mathbf{S}_e = (s_e(x, y))$ verwendet. Die Bildpunkte $s_a(x, y)$ des Ausgabebildes $\mathbf{S}_a$ ergeben sich durch eine gewichtete, additive Verknüpfung des Bildpunktes $s_e(x, y)$ mit benachbarten Bildpunkten. Die Nachbarschaft und die Gewichte der Nachbarbildpunkte wird dabei durch eine Maske (*Faltungsmaske*, *Filterkern*) $\mathbf{H} = (h(u, v))$ festgelegt. Diese Operation, die einer *Faltung* des Bildes $\mathbf{S}_e$ mit der Maske $\mathbf{H}$ entspricht und auch als *digitale Filterung im Ortsbereich (Ortsraum)* bezeichnet wird, kann wie folgt geschrieben werden:

$$\mathbf{S}_e \rightarrow \mathbf{S}_a : \tag{5.1}$$

$$s_a(x, y) = \frac{1}{m^2} \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_e(x + k - u, y + k - v) \cdot h(u, v),$$

wobei m die Größe der Maske $\mathbf{H}$ angibt. In der Praxis werden meistens quadratische Masken mit $m = 3, 5, 7, ...$ verwendet. Für den Parameter k gilt: $k = \frac{m-1}{2}$.

Für den Randbereich des Eingabebildes $\mathbf{S}_e$ ergeben sich mit (5.1) Probleme, da bei der Verknüpfung auch Bildpunkte verwendet werden, die außerhalb des Bildbereichs von $\mathbf{S}_e$ liegen (z.B. für $x = 0$ oder $y = 0$). Zur Lösung dieses Problems bieten sich unterschiedliche Möglichkeiten an.

Als erste Möglichkeit kann man bei der Berechnung des Ausgabebildes $\mathbf{S}_a$ und einer Maske mit z.B. $m = 3$ die Indizes x und y jeweils nur von eins bis zur vorletzten Zeile (Spalte) laufen lassen. Das hat aber zur Folge, dass das Ausgabebild hier um einen ein Pixel breiten Rand kleiner wird. Dieser Effekt, dass das Ausgabebild kleiner ist als das Eingabebild, ist bei manchen Anwendungen störend oder sogar unerwünscht.

Bild 5.2: Periodische Fortsetzung des Bildes $\mathbf{S} = (s(x, y))$.

Als zweite Möglichkeit kann der Bildrand von $\mathbf{S}_e$ unverändert in das Ausgabebild $\mathbf{S}_a$ übernommen werden. Dann kann es passieren, dass bei weiteren Verarbeitungsschritten, z.B. bei einer Skalierung der Grauwerte (Kapitel 4), die Grauwerte in diesem Randbereich Störungen verursachen.

Drittens ist auch die Codierung der Bildpunkte des Randes mit einem konstanten Grauwert (etwa 0, 127 oder 255) denkbar.

Schließlich kann das Problem auch gelöst werden, indem man sich das Eingabebild $\mathbf{S}_e$ periodisch fortgesetzt vorstellt und so die fehlenden Pixel vom oberen bzw. unteren und

rechten bzw. linken Bildrand verwendet wie Bild 5.2 zeigt.

Diese Vorgehensweise ist vor dem Hintergrund der diskreten, zweidimensionalen Fouriertransformation sinnvoll, da dort eine solche periodische Fortsetzung vorausgesetzt wird (Kapitel 8). Allerdings erscheint eine solche periodische Fortsetzung nicht immer sinnvoll. Als Beispiel sei ein Bild mit einem dunklen Vordergrund im unteren Teil und einem hellen Hintergrund im oberen Teil betrachtet: Bei der periodischen Fortsetzung würden hier bei der Berechnung des Mittelwertes am oberen Bildrand die Grauwerte des dunklen unteren Bildbereichs erfasst werden. Es ist hier somit eine Spiegelung des Bildes sinnvoller.

Je nachdem, wie die Koeffizienten des Filterkerns $\mathbf{H}$ aussehen, ergeben sich unterschiedliche *digitale Filter im Ortsbereich*. Treten nur positive Koeffizienten auf, so spricht man auch von *Summenoperatoren*, bei positiven und negativen Koeffizienten von *Differenzenoperatoren*. Lassen sich die Koeffizienten als Stützstellen einer linearen Funktion auffassen, so werden die Operationen als *lineare digitale Filter im Ortsbereich*, anderenfalls als *nicht lineare digitale Filter im Ortsbereich* bezeichnet.

Im Folgenden werden einige Beispiele zu Filterkernen gegeben. Eine Operation mit dem Filterkern

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{5.2}$$

wird als *bewegter Mittelwert* bezeichnet. Er wurde in (5.2) mit $m = 3$ dargestellt. Er kann aber auch mit größeren Filterkernen angewendet werden. Der bewegte Mittelwert bewirkt eine Glättung der Grauwerte. Eine Beispielimplementierung für die Bildverarbeitung auf Grafikkarten befindet sich in Band I Kapitel 13.6.2.1.

Es sind auch Varianten denkbar: Soll z.B. der Bildpunkt in der Position (x, y) doppelt gewichtet werden, so kann der zugehörige Koeffizient doppelt so groß wie die anderen gewählt werden. Wenn dabei der Mittelwert des gesamten Bildes nicht verändert werden soll, so ist darauf zu achten, dass die Summe der Filterkoeffizienten den Wert m^2 ergibt. Ein Beispiel dazu ist der folgende Filterkern:

$$\mathbf{H} = \begin{pmatrix} 0.9 & 0.9 & 0.9 \\ 0.9 & 1.8 & 0.9 \\ 0.9 & 0.9 & 0.9 \end{pmatrix} \tag{5.3}$$

Der nichtlineare *Gauß-Tiefpass* hat den Filterkern

$$\mathbf{H} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}. \tag{5.4}$$

Der Name rührt daher, dass seine Elemente eine zweidimensionale Gauß'sche Glocke grob annähern. Wenn auch hier die Operation den Mittelwert des Bildes erhalten soll, so müssen die Elemente dieses Filterkerns mit $\frac{9}{16}$ multipliziert werden.

Differenzenoperatoren werden in der Regel verwendet, um Grauwertübergänge deutlicher herauszuarbeiten oder Kanten zu extrahieren. Bei der Berechnung kann hier auf den Normierungsfaktor $\frac{1}{m^2}$ in (5.1) verzichtet werden. Wenn das Ergebnis wieder in die Grauwertmenge $G = \{0, ..., 255\}$ abgebildet werden soll, so ist zu beachten, dass die Werte geeignet skaliert werden.

Wird eine partielle Differentiation in Zeilen- und Spaltenrichtung nachgebildet, so erhält man die Filterkerne

$$\mathbf{H}_x = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{und} \quad \mathbf{H}_y = \begin{pmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{5.5}$$

Nach dieser Darstellung der Grundprinzipien werden in den folgenden Abschnitten die unterschiedlichen Verfahren ausführlich beschrieben.

5.3 Glätten der Grauwerte eines Bildes

Die einfachste Form der Glättung der Grauwerte eines Bildes ist die Berechnung des Mittelwertes einer vorgegebenen Umgebung eines Bildpunktes. Aus Gründen der Symmetrie wird meistens eine quadratische Umgebung mit $3 \cdot 3, 5 \cdot 5, 7 \cdot 7, ...$ Bildpunkten verwendet. Es sei $\mathbf{S}_e = (s_e(x, y))$ ein Grauwertbild mit $G = \{0, 1, ..., 255\}$. Die Größe des quadratischen Umgebungsfensters wird mit m bezeichnet. Der Operator

$$\mathbf{S}_e \to \mathbf{S}_a : s_a(x, y) = \frac{1}{m^2} \sum_{u=-k}^{+k} \sum_{v=-k}^{+k} s_e(x - u, y - v), \tag{5.6}$$

mit $m = 3, 5, 7, ...$ und $k = (m - 1)/2$, weist dem Bildpunkt $s_a(x, y)$ den Mittelwert der $m \cdot m$-Umgebung des Bildpunktes $s_e(x, y)$ in $\mathbf{S}_e$ als neuen Grauwert zu. Für $m = 3$ werden z.B. folgende Bildpunkte in die Berechnung mit einbezogen:

$$\begin{array}{lll} s_e(x-1, y-1) & s_e(x-1, y) & s_e(x-1, y+1) \\ s_e(x, y-1) & s_e(x, y) & s_e(x, y+1) \\ s_e(x+1, y-1) & s_e(x+1, y) & s_e(x+1, y+1). \end{array}$$

Die Probleme, die sich für den Randbereich ergeben, können mit den unterschiedlichen Vorgehensweisen gelöst werden, die in Abschnitt 5.2 dargestellt wurden.

Bilder, die mit diesem Operator, der auch *bewegter Mittelwert* oder *gleitender Mittelwert* (*moving average*) genannt wird, verarbeitet werden, wirken im Vergleich zum Original weicher oder etwas unschärfer. In Bildbereichen mit homogenen Grauwerten hat der bewegte Mittelwert keine Auswirkung, wogegen Grauwertübergänge geglättet werden. Ein Beispiel zeigt die Bildfolge 5.3.

Der Mittelwert von $\mathbf{S}_a$ ist derselbe wie der von $\mathbf{S}_e$, während die Streuung von $\mathbf{S}_a$ kleiner wird, was z.B. bedeuten kann, dass im Bild $\mathbf{S}_e$ überlagertes Rauschen verringert wird. Dies

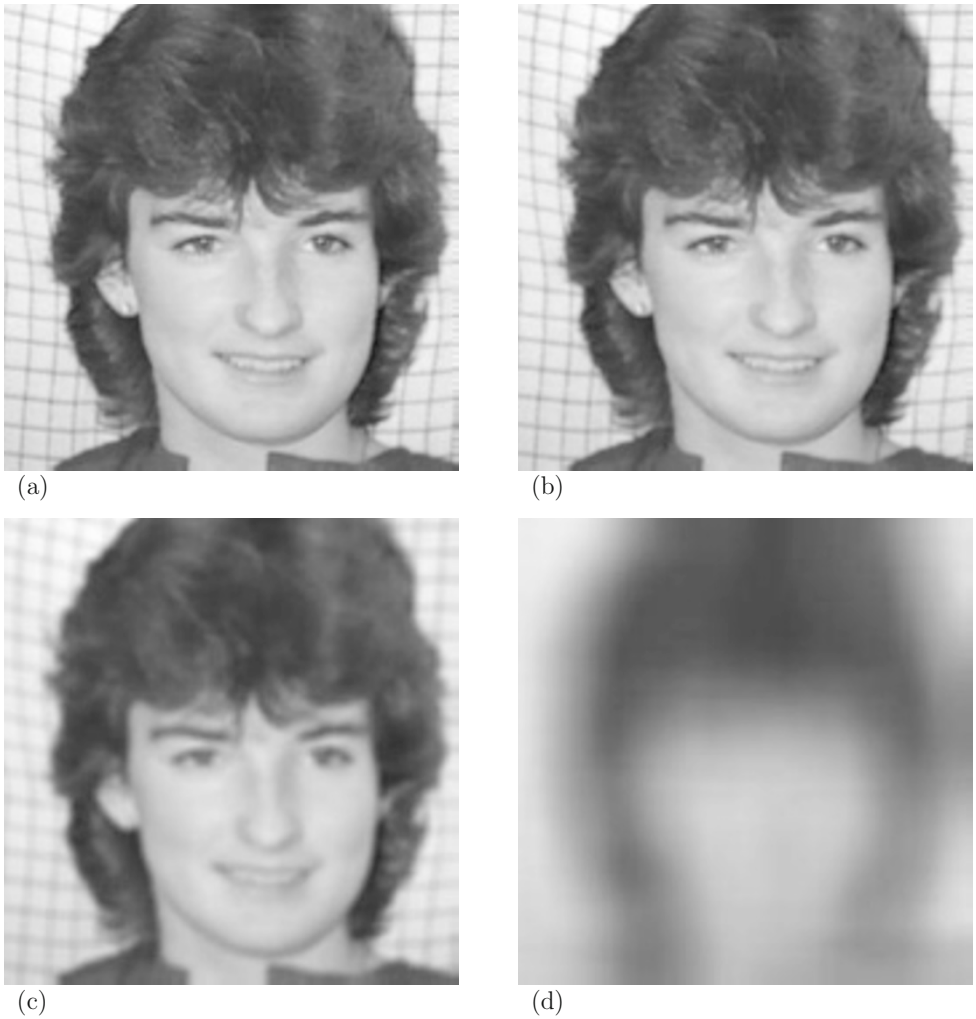


Bild 5.3: Beispiele zum bewegten Mittelwert. (a) Original. (b) Umgebung mit $3 \cdot 3$ Bildpunkten. (c) Umgebung mit $11 \cdot 11$ Bildpunkten. (d) Umgebung mit $101 \cdot 101$ Bildpunkten. Bei allen Beispielen wurde der Randbereich durch eine Spiegelung des Originals berechnet.

läßt sich auch rechnerisch nachweisen. Für die Grauwerte von $\mathbf{S}_e$ wird angenommen, dass ihnen ein Rauschanteil $z(x,y)$ additiv überlagert ist:

$$s_e(x,y) = s_0(x,y) + z(x,y). \tag{5.7}$$

Die Rauschanteile $z(x,y)$ werden durch Zufallsvariable $Z_{(x,y)}$ beschrieben, die den Erwartungswert 0 und die Streuung σ^2 haben. Die Streuung des Rauschens wird auch als *Rauschleistung* bezeichnet. Die Zufallsvariablen seien unabhängig und unkorreliert. Damit läßt sich für $s_a(x,y)$ schreiben:

$$s_a(x,y) = \frac{1}{m^2}\Big(\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} s_0(x-u,y-v) + \sum_{u=-k}^{+k}\sum_{v=-k}^{+k} z(x-u,y-v)\Big). \tag{5.8}$$

Für den Erwartungswert und die Streuung der Zufallsvariablen

$$Z = \frac{1}{m^2}\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} Z_{x-u,y-v} \tag{5.9}$$

ergibt sich:

$$E(Z) = E\Big(\frac{1}{m^2}\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} Z_{x-u,y-v}\Big) = \frac{1}{m^2}\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} E\Big(Z_{x-u,y-v}\Big) = 0. \tag{5.10}$$

$$E(Z^2) = E\Big(\Big(\frac{1}{m^2}\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} Z_{x-u,y-v}\Big)^2\Big) = \tag{5.11}$$

$$= \frac{1}{m^4}E\Big(\Big(\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} Z_{x-u,y-v}\Big)^2\Big) =$$
$$= \frac{1}{m^4}\sum_{u=-k}^{+k}\sum_{v=-k}^{+k} E\Big(\Big(Z_{x-u,y-v}\Big)^2\Big) = \frac{1}{m^2}\cdot\sigma^2.$$

Für die Umformung in (5.11) wird die Unabhängigkeit und die Unkorreliertheit von zwei Zufallsvariablen $Z_{x-u_1,y-v_1}$ und $Z_{(x-u_2,y-v_2)}$ verwendet:

$$E\Big(Z_{(x-u_1,y-v_1)}Z_{(x-u_2,y-v_2)}\Big) = E\Big(Z_{(x-u_1,y-v_1)}\Big)\cdot E\Big(Z_{(x-u_2,y-v_2)}\Big) = 0. \tag{5.12}$$

Die Beziehungen (5.10) und (5.11) bestätigen die obigen Vermutungen: Der Erwartungswert bleibt erhalten und die Streuung des Rauschanteils wird durch den Faktor $1/m^2$ reduziert.

Der bewegte Mittelwert kann auch als Faltungsoperation, wie in Abschnitt 5.2 erläutert, aufgefasst werden. Die Faltung zweier Funktionen $s(x, y)$ und $h(x, y)$ der diskreten Variablen x und y ist definiert als:

$$\sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} s(x-u, y-v) \cdot h(u, v). \tag{5.13}$$

Da beim bewegten Mittelwert nur über eine endliche $m \cdot m$-Umgebung des Bildes $\mathbf{S}_e$ gemittelt wird, kann (5.6) als Faltung des Bildausschnittes mit einer Maske $\mathbf{H} = (h(u, v))$ geschrieben werden:

$$s_a(x, y) = \frac{1}{m^2} \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_e(x+k-u, y+k-v) \cdot h(u, v) \tag{5.14}$$

mit $k = (m-1)/2$ und $m = 3, 5, 7, ...$ Die Maske $\mathbf{H}$ hat beim bewegten Mittelwert mit z.B. $m = 3$ das Aussehen:

$$\mathbf{H} = \Big(h(u, v)\Big) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{5.15}$$

Bei der Berechnung des bewegten Mittelwertes kann berücksichtigt werden, dass beim Übergang von der Position (x, y) zu $(x, y+1)$ nicht die ganze Summe neu berechnet werden muss, sondern der neue Mittelwert durch Subtraktion des Mittelwertes des Spaltenvektors

$$\Big(s_e(x-k, y-k), s_e(x-k+1, y-k), ..., s_e(x+k, y-k)\Big)^T$$

und Addition des Mittelwertes des Spaltenvektors

$$\Big(s_e(x-k, y+1+k), s_e(x-k+1, y+1+k), ..., s_e(x+k, y+1+k)\Big)^T$$

berechnet werden kann.

Bei der Implementierung des bewegten Mittelwertes können diese speziellen Eigenschaften des Filterkerns ausgenützt werden. Ein Algorithmus dazu ist im Folgenden angegeben.

A5.1: Bewegter Mittelwert.

Voraussetzungen und Bemerkungen:

- ◇ $\mathbf{S}_e = (s_e(x, y))$ ein einkanaliges Grauwertbild mit $G = \{0, 1, ..., 255\}$ als Grauwertmenge (Eingabebild).
- ◇ $\mathbf{S}_a = (s_a(x, y))$ ein einkanaliges Grauwertbild $G = \{0, 1, ..., 255\}$ als Grauwertmenge (Ausgabebild).

◇ Die spezielle Behandlung des Bildrandes, wie in Abschnitt 5.2 dargestellt, wird hier nicht berücksichtigt.

Algorithmus:

(a) Für alle Bildzeilen $x = 0, 1, ..., L-1$ des Bildes $\mathbf{S}_e = (s_e(x, y))$:

(aa) Berechne die folgenden Spaltensummen für $j = -k, ..., +k$ und $k = \frac{m-1}{2}$:

$$s_j = \sum_{i=-k}^{+k} s_e(x+i, j);$$

(ab) Für alle Bildspalten $y = 0, 1, ..., R-1$ des Bildes $\mathbf{S}_e = (s_e(x, y))$:

(aba) Berechne den Grauwert des Ausgabebildpunktes:

$$s_a(x, y) = \frac{1}{m^2} \sum_{j=-k}^{+k} s_j;$$

(abb) Berechne eine neue Spaltensumme:

$$s_{k+1} = \sum_{i=-k}^{+k} s_e(x+i, y+k+1);$$

(abc) Verschiebe die Spaltensummen. Für $j = -k, ..., +k$:

$$s_j = s_{j+1};$$

Ende des Algorithmus

Die einzelnen Elemente von $\mathbf{H}$ geben das Gewicht an, mit dem die Grauwerte der $m \cdot m$-Umgebung in die Summation eingehen. Verwendet man für $\mathbf{H}$ die Maske

$$\mathbf{H} = \Big(h(u, v)\Big) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{5.16}$$

so wird der Grauwert in der Mitte der Umgebung doppelt bewertet. Allerdings wäre dann der Mittelwert von $\mathbf{S}_a$ anders als der von $\mathbf{S}_e$. Der Mittelwert bleibt erhalten, wenn die Elemente von $\mathbf{H}$ so normiert sind, dass ihre Summe m^2 ergibt. Statt (5.16) ist dann die Maske

$$\mathbf{H} = \Big(h(u, v)\Big) = \begin{pmatrix} 0.9 & 0.9 & 0.9 \\ 0.9 & 1.8 & 0.9 \\ 0.9 & 0.9 & 0.9 \end{pmatrix} \tag{5.17}$$

zu verwenden.

Der bewegte Mittelwert wird in der Praxis oft zur Glättung der Grauwerte vor anderen Operationen eingesetzt. So vermittelt z.B. die Äquidensiten- oder Pseudofarbdarstellung eines geglätteten Bildes einen besseren visuellen Eindruck als die entsprechende Darstellung

des Originals. Allerdings geht bei der Mittelwertbildung Bildinformation verloren, was aber bei einer anschließenden Äquidensiten- oder Pseudofarbdarstellung in der Regel keine Rolle spielt, da hier ja sowieso Grauwertintervalle zu einem Grauwert zusammengefasst werden.

Je nach der Größe von m werden durch den bewegten Mittelwert mehr oder weniger große Umgebungen des Bildpunktes in der Position (x, y) in die Mittelung einbezogen. Ein Nachteil ist dabei allerdings, dass Grauwertübergänge, also Kanten, geglättet und somit unschärfer werden. Die Bilder 5.4-a, 5.4-b, 5.4-c und 5.4-d zeigen ein Beispiel hierzu. Bild 5.4-d ist aus der Differenz zwischen dem Original und dem zehnmal hintereinander gefilterten Bild 5.4-c entstanden. Man kann so die Bildinformation darstellen, die durch die Filteroperation verloren ging: Das sind aber gerade die deutlichen Grauwertübergänge (Kanten). Dieser Sachverhalt wird bei den Laplace-Pyramiden (Kapitel 17) verwendet.

In Kapitel 4 wurde im Rahmen der Binärbilderzeugung, der Äquidensitendarstellung, sowie der Pseudofarbdarstellung bereits auf die Verwendung des bewegten Mittelwertes als Vorverarbeitungsschritt hingewiesen.

Bei der Verwendung anderer Filterkerne, z.B. dem Gauß-Tiefpass von (5.4), erhält man ähnliche Ergebnisse.

5.4 Differenzenoperatoren

Werden für die Elemente der Maske $\mathbf{H}$ in (5.14) auch negative Werte zugelassen, so ergeben sich *Differenzenoperatoren*. Ist die Summe der Elemente von $\mathbf{H}$ gleich 0, so berechnet sich mit (5.14) für homogene Bildbereiche der Wert 0. Bei Grauwertübergängen liefert der Operator eine Maßzahl für die „Stärke“ des Übergangs. Zur praktischen Berechnung kann in (5.14) auf den Normierungsfaktor $1/m^2$ verzichtet werden.

Eine einfache Anwendung ist die Berechnung der Differenzen benachbarter Bildpunkte in Zeilen- oder in Spaltenrichtung. Ist $s(x)$ eine stetige Funktion, so ist die erste Ableitung von $s(x)$ definiert als

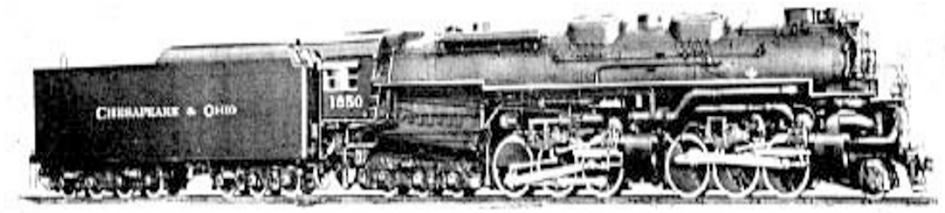
$$\frac{\mathrm{d}s}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{s(x + \Delta x) - s(x)}{\Delta x}. \tag{5.18}$$

Die sinngemäße Differenzenbildung bei einer Funktion $s(x)$ mit diskretem x lautet:

$$\frac{s(x+1) - s(x)}{1} = s(x+1) - s(x). \tag{5.19}$$

Da nun ein Grauwertbild $\mathbf{S} = (s(x, y))$ eine Funktion mit zwei diskreten Variablen x und y ist, ergibt sich als Gradient

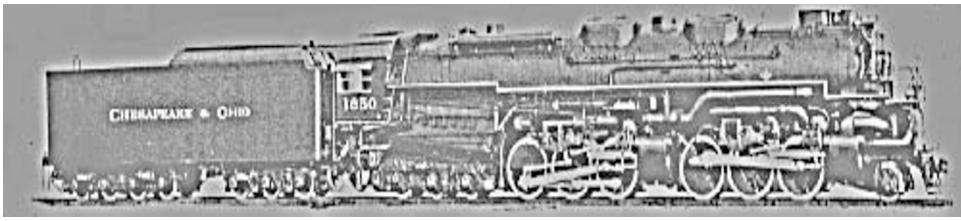
$$\text{grad } s(x, y) = \nabla s(x, y) = (\frac{\partial s}{\partial x}, \frac{\partial s}{\partial y})^T = \begin{pmatrix} s(x+1, y) - s(x, y) \\ s(x, y+1) - s(x, y) \end{pmatrix} \tag{5.20}$$



(a)

(b)

(c)



(d)

Bild 5.4: (a) Originalbild „Chesapeake & Ohio“. (b) Bewegter Mittelwert($m = 3$): Man sieht deutlich, dass Kanten unschärfer sind als im Original. (c) Hier wurde der Operator zehnmal hintereinander angewendet. (d) Differenzbild zwischen Original und dem zehnmal gefilterten Bild. Man sieht hier die Bildinhalte, die durch die Filterung verloren gingen.

(a)

(b)

Bild 5.5: Differenzenoperator (Sobeloperator), angewendet auf das Testbild. (a) Ergebnis der $\mathbf{H}_x$-Maske. (b) Ergebnis der $\mathbf{H}_y$-Maske.

Die Maske $\mathbf{H}$ in (5.14) hat damit für die partiellen Ableitungen aus 5.20 das folgende Aussehen:

$$\mathbf{H}_x = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{H}_y = \begin{pmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{5.21}$$

Differenzenoperatoren, die die beiden Masken $\mathbf{H}_x$ oder $\mathbf{H}_y$ verwenden, sind also die diskreten Analoga zur Berechnung der ersten Ableitung in x- oder y-Richtung einer stetigen Funktion zweier Variablen.

Wie schon erwähnt, werden Differenzenoperatoren vor allem da eingesetzt, wo es darum geht, Grauwertkanten und Linien aus einem Bild zu extrahieren. Die einfache Differenzbildung mit den Faltungskernen (5.21) ist dazu in der Praxis zu anfällig gegenüber Störungen, etwa durch Rauschen. Aus diesem Grund werden zur Differenzbildung auch die Grauwerte weiterer Nachbarn verwendet. Statt (5.21) könnte z.B. auch verwendet werden:

$$\mathbf{H}_x = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{H}_y = \begin{pmatrix} 1 & -1 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \end{pmatrix}. \tag{5.22}$$

Bessere Ergebnisse liefert der *Sobeloperator* :

$$\mathbf{H}_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \qquad \mathbf{H}_y = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \tag{5.23}$$

da hier die Differenzbildung jeweils zur übernächsten Zeile (Spalte) berechnet wird und kleine Störungen benachbarter Zeilen (Spalten) nicht in das Ergebnis eingehen. Die Rauschminderung wird hier durch Mittelung quer zur Richtung des Gradienten bewirkt. Die Anwendung dieser beiden Masken auf das Testbild zeigen Bild 5.5-a und Bild 5.5-b. Für die bildliche Darstellung wurde zu den Differenzen der Grauwert 127 addiert. Außerdem wurde zur besseren bildlichen Darstellung der Bildkontrast etwas angehoben.

Die bis jetzt besprochenen Differenzenoperatoren sprechen am stärksten auf horizontal oder vertikal verlaufende Grauwertkanten an, sie sind also richtungsabhängig. Beispiele für Implementierung der Differenzenoperatoren auf Grafikkarten sind in Band I Kapitel 13.6.2 beschrieben.

Ein weiterer richtungsabhängiger Differenzenoperator ist der *Kompassgradient*, der aus insgesamt acht Masken für acht Richtungen besteht:

$$\begin{aligned}
\mathbf{H}_{\text{Ost}} &= \begin{pmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{pmatrix} & \mathbf{H}_{\text{Nordost}} &= \begin{pmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{pmatrix} \\
\mathbf{H}_{\text{Nord}} &= \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{pmatrix} & \mathbf{H}_{\text{Nordwest}} &= \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{pmatrix} \\
\mathbf{H}_{\text{West}} &= \begin{pmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{pmatrix} & \mathbf{H}_{\text{Südwest}} &= \begin{pmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{pmatrix} \\
\mathbf{H}_{\text{Süd}} &= \begin{pmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix} & \mathbf{H}_{\text{Südost}} &= \begin{pmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix}
\end{aligned} \tag{5.24}$$

Man nimmt oftmals als Gesamtergebnis des Kompassgradienten pro Pixel das maximale Teilergebnis der Faltung mit den 8 Faltungskernen. Somit ist das Ergebnis sensitiv zu Strukturen im Bild, die in Richtung der Faltungskerne verlaufen.

Der *Gradient* einer stetigen Funktion zweier Variablen ist (5.20):

$$\left(\frac{\partial s(x,y)}{\partial x}, \frac{\partial s(x,y)}{\partial y} \right)^T. \tag{5.25}$$

Der *Betrag* des Gradienten ist:

$$\sqrt{\left(\frac{\partial s(x,y)}{\partial x} \right)^2 + \left(\frac{\partial s(x,y)}{\partial y} \right)^2} \tag{5.26}$$

und die *Richtung* des Gradienten berechnet sich aus:

$$\varphi = \arctan \left(\frac{\partial s(x,y)}{\partial y} \right) \Big/ \left(\frac{\partial s(x,y)}{\partial x} \right). \tag{5.27}$$

Im diskreten Fall werden z.B. mit den Masken des Sobeloperators (5.23) die Funktionen $s_x(x,y)$ und $s_y(x,y)$ berechnet und anschließend *Betrag* und *Richtung* des Gradienten:

$$\sqrt{s_x(x,y)^2 + s_y(x,y)^2} \text{ und } \arctan(\frac{s_y(x,y)}{s_x(x,y)}). \tag{5.28}$$

Soll eine bildliche Darstellung des Betrags des Gradienten erfolgen, so ist in (5.28) zu beachten, dass die berechneten Werte in die Grauwertmenge G abgebildet werden. Die Bilder 5.6-a/b zeigen den so aus Bild 5.5-a und Bild 5.5-b berechneten Gradienten (Betrag und Richtung).

Richtungsunabhängige Differenzenoperatoren liegen vor, wenn die Maske $\mathbf{H}$ punktsymmetrisch ist. Am häufigsten wird hier der *Laplace-Operator* verwendet. Im kontinuierlichen Fall ist er definiert als:

$$\frac{\partial s^2(x,y)}{\partial x^2} + \frac{\partial s^2(x,y)}{\partial y^2}. \tag{5.29}$$

Die Maske für sein diskretes Analogon ist:

$$\mathbf{H} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}. \tag{5.30}$$

Oft werden statt dessen auch

$$\mathbf{H} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \text{ oder } \mathbf{H} = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix} \tag{5.31}$$

verwendet.

Der Laplace-Operator, der etwas anfälliger gegenüber Bildstörungen ist, liefert in Verbindung mit (5.14) in einem Durchgang Maßzahlen für den Betrag des Grauwertgradienten.

Allerdings ist zu beachten, dass der Laplace-Operator als diskrete Nachbildung der 2. Ableitung, angewendet auf eine beliebig geneigte Ebene, den Wert 0 ergibt, obwohl die Gradienten dieser Fläche nicht notwendigerweise den Wert 0 haben.

Die Differenzenoperatoren werden, wie schon oben erwähnt, vor allem zur Extraktion von Grauwertkanten eingesetzt. Sie sind hier nur ein erster Schritt in einer Folge von weiteren Verarbeitungen, wie z.B. Relaxation, Schwellwertbildung oder Linienverfolgung. Diese Problemkreise werden in Kapitel 7 noch näher untersucht. Implementierungen der Differenzenoperatoren auf Grafikkarten stehen in Band I Kapitel 13.6.2.

(a)

(b)

Bild 5.6: Gradienten zum Testbild nach der Anwendung des Sobeloperators. (a) Betrag. (b) Richtung. Die Winkel wurden so codiert, dass gleiche Richtungen durch den gleichen Grauwert dargestellt sind.

5.5 Elimination isolierter Bildpunkte

Manchmal treten in Bildern einzelne Pixelstörungen auf. Ein Beispiel dazu sind Luftbilddaten aus der Fernerkundung, bei denen durch störende Einflüsse auf den Sensor gelegentlich einzelne Bildpunkte übersteuert werden. Eine derartige Störung könnte etwa wie folgt aussehen:

...	...	...	...	...	...	...
...	18	22	27	24	16	...
...	19	21	28	25	20	...
...	18	21	255	24	31	...
...	17	24	26	25	32	...
...	14	19	22	19	25	...
...	...	...	...	...	...	...

In einer Umgebung von Bildpunkten mit einem Mittelwert um etwa 20 hat ein Bildpunkt den Grauwert 255.

Mit einem bewegten Mittelwert oder ähnlichen Glättungsoperatoren (Abschnitt 5.3) wird das gesamte Bild unschärfer und der fehlerhafte Wert wird etwas verkleinert. Bei der Verwendung eines Medianfilters (Kapitel 6) kann die Bildpunktstörung eliminiert werden. Die Grauwerte der Umgebung der Störung werden dabei aber auch verändert, was bei manchen Anwendungen nicht wünschenswert ist.

Um den gestörten Bildpunkt zu erkennen und zu eliminieren, ohne dabei die Grauwerte der Umgebung zu verändern, vergleicht man den Grauwert jedes Bildpunktes mit dem Mittelwert m seiner 8-Nachbarn. Übersteigt der Unterschied einen bestimmten Schwellwert c, so wird der Bildpunkt als gestört erkannt und durch den Mittelwert m ersetzt:

$$\mathbf{S}_e \rightarrow \mathbf{S}_a : \tag{5.32}$$

$$s_a(x,y) = \begin{cases} m, & \text{falls } \mid m - s_e(x,y) \mid > c, \\ s_e(x,y), & \text{sonst.} \end{cases}$$

Dabei ist

$$m = \frac{1}{8} \sum_{u=0}^{2} \sum_{v=0}^{2} s_e(x+1-u, y+1-v) \cdot h(u,v). \tag{5.33}$$

mit der Filtermaske

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{5.34}$$

Dieses Verfahren kann immer eingesetzt werden, wenn sich die Bildstörungen im Grauwert deutlich von den korrekten Bildpunkten abheben. Wird der Schwellwert c zu klein

gewählt, so ist die Wirkung dieses Operators mit der des bewegten Mittelwertes (Abschnitt 5.3) zu vergleichen.

Durch sinngemäße Modifikation der verwendeten Maske können auch Bildstörungen, die mehr als einen Bildpunkt verfälscht haben, mit diesem Operator detektiert und eliminiert werden. Ist z.B. bekannt, dass immer zwei aufeinander folgende Bildpunkte in einer Bildzeile gestört sind, so kann die folgende Maske eingesetzt werden:

$$\mathbf{H} = \Big(h(u,v)\Big) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{5.35}$$

Es ist aber zu bemerken, dass bei ausgedehnten Bildstörungen die Effektivität dieses Operators mehr und mehr abnimmt.

5.6 Elimination gestörter Bildzeilen

Manchmal treten in Bildern gestörte Bildzeilen auf, die entweder ganz weiß oder schwarz sind oder ein zufälliges Grauwertmuster aufweisen. Die Detektion solcher Bildzeilen kann mit Hilfe der Berechnung der Korrelation von aufeinander folgenden Bildzeilen durchgeführt werden: Bei natürlichen Bildern ist der Übergang der Grauwerte von einer Bildzeile zur nächsten in der Regel nicht allzu abrupt, was bedeutet, dass der Korrelationskoeffizient zwischen benachbarten, ungestörten Bildzeilen relativ groß ist. Dagegen ist er bei einer korrekten und einer gestörten Bildzeile klein. Damit ist also die folgende Vorgehensweise angebracht:

- Die Verarbeitung wird mit einer korrekten Bildzeile begonnen.
- Zu je zwei aufeinander folgenden Bildzeilen x und $x+1$ wird der Korrelationskoeffizient $r_{x,x+1}$ berechnet.
- Liegt der Betrag $\mid r_{x,x+1} \mid$ über einem gewissen Schwellwert c, so wird angenommen, dass die Bildzeile $x+1$ korrekt ist.
- Ist der Betrag $\mid r_{x,x+1} \mid < c$, so wird die Bildzeile $x+1$ als fehlerhaft betrachtet.

Der Korrelationskoeffizient für die Bildzeilen des Eingabebildes $\mathbf{S}_e$ wird dazu wie folgt berechnet. Es sei $v_{x,x+1}$, die Kovarianz der Bildzeilen x und $x+1$:

$$v_{x,x+1} = \frac{1}{R} \sum_{y=0}^{R-1} \Big(s_e(x,y) - m_x\Big)\Big(s_e(x+1,y) - m_{x+1}\Big). \tag{5.36}$$

m_x und m_{x+1} sind die mittleren Grauwerte der Bildzeilen x und $x+1$. Durch Normierung der Kovarianz durch die Wurzel des Produktes der Streuungen $v_{x,x}$ und $v_{x+1,x+1}$ erhält man den Korrelationskoeffizienten:

$$r_{x,x+1} = \frac{v_{x,x+1}}{\sqrt{v_{x,x} v_{x+1,x+1}}}. \tag{5.37}$$

Der Betrag $\mid r_{x,x+1} \mid$ liegt zwischen 0 und 1, wobei sich starke Korrelation durch Werte nahe der 1 ausdrückt. Zur Bestimmung eines passenden Schwellwertes c ist zu sagen, dass er am besten durch Testen am jeweiligen Bildmaterial zu ermitteln ist.

A5.2: Elimination gestörter Bildzeilen.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild mit $G = \{0,1,...,255\}$ als Grauwertmenge (Eingabebild).

◇ $\mathbf{S}_a = (s_a(x,y))$ ein einkanaliges Grauwertbild mit $G = \{0,1,...,255\}$ als Grauwertmenge (Ausgabebild).

◇ Die Verarbeitung wird mit einer korrekten Bildzeile begonnen.

Algorithmus:

(a) Für je zwei aufeinander folgende Bildzeilen x und $x+1$ wird der Korrelationskoeffizient $r_{x,x+1}$ berechnet.

(aa) Liegt der Betrag $\mid r_{x,x+1]} \mid$ über einem Schwellwert c, so wird angenommen, dass die Bildzeile $x+1$ korrekt ist.

(ab) Ist der Betrag $\mid r_{x,x+1]} \mid < c$, so wird die Bildzeile $x+1$ als fehlerhaft betrachtet und z.B. durch die Zeile x ersetzt.

Ende des Algorithmus

Bei der praktischen Anwendung dieses Verfahrens sind noch einige Punkte zu berücksichtigen. Wie schon oben erwähnt, muss die Startbildzeile korrekt sein. Eine passende Startzeile kann entweder interaktiv festgelegt oder durch den Vergleich der Grauwerte von benachbarten Bildpunkten ermittelt werden. Falls gestörte Bildzeilen auch direkt aufeinander folgend auftreten können, muss berücksichtigt werden, dass zwei gestörte Bildzeilen, die z.B. beide ganz weiß oder schwarz sind, maximal korreliert sind. Andererseits werden gestörte Bildzeilen, die zufällige Grauwertmuster aufweisen, in der Regel geringer korreliert sein.

Abschließend ist noch zu untersuchen, wie eine detektierte, gestörte Bildzeile eliminiert werden kann. Die einfachste Methode ist die Ersetzung dieser Bildzeile durch die vorhergehende korrekte, die zur Berechnung des Korrelationskoeffizienten verwendet wurde. Bei dieser Methode kann allerdings bei schräg verlaufenden Grauwertkanten eine ungewünschte Stufung auftreten. Andere Methoden interpolieren die Grauwerte zwischen zwei oder mehreren benachbarten Bildzeilen.

5.7 Bildakkumulation bei Bildfolgen

Wenn Bildfolgen, etwa der Form $\mathbf{S} = (s(x, y, t))$, $t = 0, 1, 2, ...$, vorliegen, so können Rauschanteile auch durch eine Summation entlang der Zeitachse vermindert werden:

$$\mathbf{S}_e \rightarrow \mathbf{S}_a : \tag{5.38}$$

$$s_a(x, y) = \frac{1}{T} \sum_{t=0}^{T-1} s_e(x, y, t);$$

Hier wird über eine Bildsequenz von T Einzelbildern akkumuliert. Es muss allerdings vorausgesetzt werden, dass sich im Bildausschnitt nichts bewegt. Der Vorteil gegenüber einem bewegten Mittelwert über ein Einzelbild liegt darin, dass durch die Akkumulation entlang der Zeitachse zwar Rauschen verringert werden kann, jedoch Kanten im Bild erhalten bleiben.

Bei sehr stark verrauschten Bildsignalen, bei denen sich die eigentliche Bildinformation vom Rauschen kaum mehr unterscheiden läßt, kann durch die Akkumulation über einen genügend langen Zeitraum eine deutlichere Hervorhebung der Bildinformation erzielt werden. In der Radioastronomie werden diese Techniken der Bildakkumulation erfolgreich eingesetzt. Über einen längeren Zeitraum ist hier allerdings die Voraussetzung, dass sich das Beobachtungsgebiet nicht bewegt, nicht gegeben. Da aber die Bewegung bekannt ist, kann sie durch ein Nachführen des Sensors (des Radioteleskops) kompensiert werden.

In der Medizin wird die Bildakkumulation ebenfalls verwendet, z.B. in der Angiocardiografie. Hier werden durch die Kombination von Bildakkumulation, Verwendung von Kontrastmitteln und der Differenzbildung von akkumulierten Zwischenergebnissen gute Bildverbesserungserfolge erzielt. Eine ausführliche Darstellung der Auswertung von Bildfolgen ist in Kapitel 13 zusammengestellt.

Kapitel 6

Rangordnungsoperatoren und mathematische Morphologie

6.1 Anwendungen

In den folgenden Abschnitten werden Verfahren beschrieben, die in den Bereich der Rangordnungsoperatoren und der mathematischen Morphologie gehören. Nach einem Grundlagenabschnitt über mathematische Morphologie wird besprochen, wie sich z.B. vereinzelte Bildpunkte eliminieren lassen oder wie sich der Rand eines Segments ermitteln lässt. Auch bestimmte Formen können durch die Kombination verschiedener morphologischer Operationen gezielt extrahiert werden. Eine Beispielimplementierung der hier beschriebenen Rangordnungsoperatoren für die Bildverarbeitung auf Grafikkarten ist in Band I Kapitel 13.6.3 zu finden.

6.2 Grundlagen: Mathematische Morphologie

Die mathematische Morphologie ist eine Theorie, die sich mit der Verknüpfung von Mengen befasst. Angewendet auf Bildvorlagen können durch die Kombination verschiedener morphologischer Operatoren viele Bildverarbeitungsprobleme, wie z.B. eine Kantenextraktion, eine Skelettierung, eine Segmentierung oder das Zählen von Segmenten behandelt werden. In diesem Abschnitt werden die Grundlagen dazu zusammengefasst.

Sehr allgemein wird die mathematische Morphologie in [Serr82] und [Serr88] behandelt. Wichtige Mengenoperationen sind hier, neben den bekannten Operationen wie Durchschnitt, Vereinigung, Differenz und Komplement, die *Dilatation* und die *Erosion* von Mengen. Dilatation und Erosion werden in dieser mengentheoretischen Darstellung folgendermaßen definiert: Es seien X und K beliebige Mengen. Die Menge K wird als *strukturierendes Element* (manchmal auch *Kern*) bezeichnet. Die Menge K wird in alle möglichen Positionen y verschoben. Die so verschobene Menge wird mit K_y bezeichnet. Die Dilatation der Menge X durch das strukturierende Element K stellt die Menge aller Punkte y dar,

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_6

○ ○ ● ○ ○	Elementarraute
○ ○ ● ○	Elementarrechteck
○ ○ ○ ○ ● ○ ○ ○ ○	8-Nachbarn
○ ○ ○ ○ ● ○ ○ ○ ○	schräges Element

Bild 6.1: Beispiele für strukturierende Elemente. Mit • ist der Bezugspunkt und mit ○ sind die Nachbarn markiert.

bei denen der Durchschnitt von X und K_y nicht leer ist:

$$Y = X \oplus K = \{y \mid K_y \cap X \neq \emptyset\}. \tag{6.1}$$

Bei der Erosion wird gefordert, dass K_y eine Teilmenge von X ist:

$$Y = X \ominus K = \{y \mid K_y \subseteq X\} = \{y \mid K_y \cap X = K_y\}. \tag{6.2}$$

Aufbauend auf diesen grundlegenden Operationen werden in den oben zitierten Arbeiten weitere morphologische Operationen definiert und auf ihre algebraischen Eigenschaften untersucht. Auf diese zum Teil recht anspruchsvollen Ausführungen soll im Rahmen dieser Darstellung verzichtet werden. Vielmehr werden im folgenden die morphologischen Operationen praxisbezogen für Binär- und Grauwertbilder definiert. Eine ausführliche Zusammenstellung dazu ist in [Abma94] zu finden.

Es sei $\mathbf{S}_e = (s_e(x, y))$ ein Grauwertbild. Als strukturierendes Element wird eine Maske K (ein *Kern* K) verwendet, die festlegt, welche Bildpunkte zum Bildpunkt in der Position (x, y) als benachbart betrachtet werden. Der Bildpunkt des strukturierenden Elements, der auf die Position (x, y) gelegt wird, heißt *Bezugspunkt*. In Bild 6.1 sind einige Beispiele dazu gegeben.

Der Bezugspunkt des strukturierenden Elements wird bei der Verarbeitung auf die Bildpunkte in den Positionen (x, y), $x = 0, 1, ..., L-1$ und $y = 0, 1, ..., R-1$ gelegt. Probleme, die dabei mit den Bildrändern auftreten, sind in ähnlicher Weise wie in Abschnitt 5.2 beschrieben zu behandeln. Die Grauwerte der durch das strukturierende Element definierten Nachbarn werden der Größe nach geordnet und bilden eine *Rangfolge* f. Operatoren, die auf dieser Vorgehensweise aufbauen, werden *Rangordnungsoperatoren* genannt.

Dazu ein Beispiel: Gegeben sei der folgende Ausschnitt aus einem Grauwertbild:

	0	1	2	3	4	...	Spalte y
0	53	68	47	36	27	...	
1	57	67	52	41	31	...	
2	61	64	19	43	35	...	
3	63	38	58	47	41	...	
...	...	...	...	...	...	...	
Zeile x							

Als Bezugspunkt wird der Punkt $(x, y) = (2, 2)$ gewählt. Mit der Elementarraute als strukturierendes Element ergibt sich dann:

$$\begin{array}{ccc} & 52 & \\ 64 & 19 & 43 \\ & 58 & \end{array} \quad \begin{array}{c} \text{geordnete} \\ \rightarrow \\ \text{Rangfolge} \end{array} \quad f = \{19, 43, 52, 58, 64\}$$

Mit dem schrägen Element als strukturierendes Element ergibt dagegen:

$$\begin{array}{ccc} & 52 & 41 \\ 64 & 19 & 43 \\ 63 & 38 & 58 \end{array} \quad 31 \quad \begin{array}{c} \text{geordnete} \\ \rightarrow \\ \text{Rangfolge} \end{array} \quad f = \{19, 31, 38, 41, 43, 52, 58, 63, 64\}$$

Je nachdem, durch welches Element der geordneten Rangfolge f der Bildpunkt in der Position des Bezugspunktes ersetzt wird, ergeben sich unterschiedliche Operationen:

Eine *Dilatation* erhält man, wenn der Grauwert in der Position des Bezugspunkts durch den *maximalen* Wert der Rangfolge f ersetzt wird. Hier dehnen sich die helleren Bildbereiche auf Kosten der dunkleren Bildbereiche aus. Die Ausdehnungsrichtung kann dabei durch die Form des strukturierenden Elements beeinflusst werden.

Bei einer *Erosion* wird der Grauwert des Bezugspunktes durch den *minimalen* Wert der Rangfolge f ersetzt, was eine Ausdehnung der dunklen Bildbereiche auf Kosten der hellen bewirkt. Ein Beispiel zu Dilatationen und Erosionen wurde in Abschnitt 4.5 gegeben.

Eine weitere Operation ist das *Medianfilter*: Hier wird der Grauwert des Bezugspunkts durch den *mittleren* Wert der Rangfolge f ersetzt. Ein Medianfilter ist z.B. zur Verbesserung verrauschter Bilder geeignet. Eine Anwendung dazu ist in Abschnitt 6.3 enthalten.

Ist das Eingabebild $\mathbf{S}_e$ ein Binärbild, so können die Grauwerte der Bildpunkte auch als Werte 0 und 1 von Boole'schen Variablen aufgefasst werden. In diesem Fall können die morphologischen Operationen auch durch logische Operatoren realisiert werden, so die Dilatation etwa durch die ODER-Verknüpfung: Die 0/1-Werte der durch das strukturierende Element festgelegten Nachbarschaft werden mit ODER verknüpft und das Ergebnis wird dem Bildpunkt in der Position des Bezugspunktes zugewiesen. Die Erosion lässt sich sinngemäß durch die UND-Verknüpfung der benachbarten Bildpunkte realisieren.

Hier sei noch auf eine schnelle Implementierungsmöglichkeit der morphologischen Operationen hingewiesen ([Abma94]). Anstatt das strukturierende Element auf die Positionen (x, y) zu legen und die benachbarten Bildpunkte zu betrachten, kann das Eingabebild als Ganzes in die einzelnen Positionen des strukturierenden Elements verschoben und mit dem

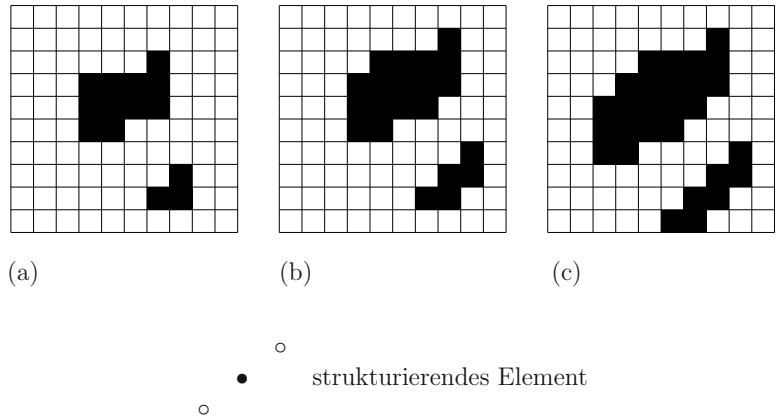


Bild 6.2: Beispiel für eine schnelle Implementierung von morphologischen Operationen durch SHIFT- und logische Operationen: Dilatation mit dem abgebildeten strukturierenden Element. (a) Originalbild (schwarze Bildpunkte: 1, weiße Bildpunkte: 0). (b) Verschiebung des Originals in die rechte obere Position des strukturierenden Elements und ODER-Verknüpfung mit dem Original. (c) Verschiebung des Originals in die linke untere Position des strukturierenden Elements und ODER-Verknüpfung mit dem Ergebnis von Teilbild (b).

Original entsprechend verknüpft werden (ODER bei der Dilatation oder UND bei der Erosion). Bild 6.2 zeigt ein Beispiel dazu. Voraussetzung für diese Verarbeitungstechnik sind schnelle Verschiebungsoperationen von Binärbildern und schnelle logische Verknüpfungen von zwei Binärbildern.

Auch bei Grauwertbildern kann die oben beschriebene Technik verwendet werden. An die Stelle des logischen ODER tritt dann die Maximumbildung und an die Stelle des logischen UND die Minimumbildung.

Bei Grauwertbildern werden manchmal die morphologischen Operationen etwas allgemeiner verwendet. Im Folgenden wird mit $\mathbf{K} = (k(i,j))$ das strukturierende Element bezeichnet. Dilatation und Erosion lassen sich dann wie folgt schreiben:

$$\mathrm{dil}(x,y) = \max_{i,j}\{s_e(x+i, y+j) + k(i,j)\}, \tag{6.3}$$

$$\mathrm{ero}(x,y) = \min_{i,j}\{s_e(x+i, y+j) - k(i,j)\}. \tag{6.4}$$

Dabei laufen die Indizes i und j über den Geltungsbereich des strukturierenden Elements. Stellt man sich ein Grauwertbild als „Grauwertgebirge“ (Funktionswerte $s(x,y)$ an der Stelle (x,y)) vor, so ist das strukturierende Element $\mathbf{K}$ hier ein dreidimensionales Gebil-

de, das über das Grauwertgebirge gleitet. Bei einer Dilatation werden dann die Grauwerte entsprechend inkrementiert, bei der Erosion dekrementiert.

Nun noch eine Bemerkung, wie man beliebig geformte strukturierende Elemente verwenden kann, wenn man aus Programmierungsgründen **K** als rechteckigen Bereich vereinbart hat: Man kann z.B. festlegen, dass nur Werte mit $k(i,j) \geq 0$ als Nachbarn gewertet werden. Negative Werte markieren dann Positionen, die nicht in die Nachbarschaft einbezogen werden sollen.

Häufig werden auch mehrere Operationen miteinander kombiniert. Eine Folge von n Erosionen und anschließenden n Dilatationen wird *opening*-Operation genannt. n Dilatationen, gefolgt von n Erosionen heißen *closing*-Operation. Mit diesen Operatoren können gezielt dunkle Strukturen auf hellem Hintergrund bzw. helle Strukturen auf dunklem Hintergrund bearbeitet werden.

Eine weitere Verknüpfungsmöglichkeit ist die Verbindung einer *opening*- oder *closing*-Operation mit einer Differenzbildung mit dem Originalbild $\mathbf{S}_e$. Diese Operationen sind geeignet, wenn man gezielt bestimmte Strukturen aus einem Bild ausblenden will.

Werden bei einem Binärbild, das mit dem Grauwert 1 codierte Segmente auf einem mit 0 codierten Hintergrund enthält, abwechselnd Dilatationen mit einer anschließenden UND-Verknüpfung mit dem Original durchgeführt, so kann man gezielt Segmente ausblenden. Es ist damit möglich, alle Segmente zu extrahieren, die den Bildrand berühren oder die Löcher enthalten. Wird ein Segment „markiert“, so kann durch diese Operation, die auch *grassfire*-Operation genannt wird, dieses Segment aus dem Bild ausgeblendet werden.

6.3 Median Filter

Wie im Grundlagenabschnitt 6.2 erläutert wird bei der Medianfilterung der Grauwert des Bezugspunktes durch den mittleren Wert der geordneten Folge f ersetzt. Im Beispiel von Abschnitt 6.2 mit der Elementarraute ergibt sich

$$\begin{array}{ccccc} & & 52 & & \\ & 64 & 19 & 43 & \\ & & 58 & & \end{array} \quad \begin{array}{c} \text{Ersetzung} \\ \rightarrow \\ \text{med}(f) = 52 \end{array} \quad \begin{array}{ccccc} & & 52 & & \\ & 64 & 52 & 43 & \\ & & 58 & & \end{array}$$

und mit dem schrägen Element ergibt sich

$$\begin{array}{ccccc} & & 52 & 41 & 31 \\ & 64 & 19 & 43 & \\ 63 & 38 & 58 & & \end{array} \quad \begin{array}{c} \text{Ersetzung} \\ \rightarrow \\ \text{med}(f)=43 \end{array} \quad \begin{array}{ccccc} & & 52 & 41 & 31 \\ & 64 & 43 & 43 & \\ 63 & 38 & 58 & & \end{array} .$$

Für die Medianfunktion med$f$ gelten die folgenden Rechenregeln, die die Addition einer Konstanten zu den Werten der Folge und die Multiplikation der Folge mit einem Faktor beschreiben:

$$\text{med}\{c + f(k)\} = c + \text{med}\{f(k)\} \text{ und } \text{med}\{c \cdot f(k)\} = c \cdot \text{med}\{f(k)\}. \tag{6.5}$$

(a)

(b)

Bild 6.3: (a) Original eines Fernsehbildes, das aufgrund eines schlecht eingestellten Senders stark verrauscht ist. (b) Mediangefiltertes Bild. Man sieht deutlich die verbesserte Bildqualität.

Es gilt jedoch nicht allgemein, dass der Medianwert der Summe von zwei Folgen gleich der Summe der Medianwerte ist.

Aus der Beschreibung eines Medianfilters ist zu ersehen, dass ein Medianfilter gut zur Elimination isolierter, fehlerhafter Bildpunkte geeignet ist. Auch Rauschen in einem Bild kann mit einem Medianfilter abgeschwächt werden. Gegenüber dem bewegten Mittelwert (Abschnitt 5.3) ergibt sich hier der Vorteil, dass die Grauwertübergänge (Kanten) besser erhalten bleiben. Allerdings ist im Vergleich mit dem bewegten Mittelwert ein erhöhter Rechenaufwand zu bewältigen. Mit schneller Bildverarbeitungshardware ist das aber kein Problem.

Die Bilder 6.3-a und 6.3-b zeigen ein Beispiel. Es wurde ein Fernseh-Videobild digitalisert, das aufgrund eines schlecht eingestellten Senders stark verrauscht ist. Die Anwendung eines Medianfilters hat hier eine wesentliche Verbesserung der Bildqualität gebracht.

Die Medianfilterung lässt sich auch mit anderen Vorverarbeitungsschritten kombinieren. In Abschnitt 4.8 wurde die Grauwertskalierung in Verbindung mit einer Hochpassfilterung erläutert. Bei manchen Anwendungen lässt sich die Bildqualität noch verbessern, wenn das skalierte und hochpassgefilterte Bild anschließend mit einem Medianfilter bearbeitet wird.

Neben der eben diskutierten Anwendung können mit einem Medianfilter auch einzelne, gestörte Bildpunkte eliminiert werden (Abschnitt 5.5).

6.4 Dilatation und Erosion im Binärbild

In diesem Abschnitt wird ein Binärbild $\mathbf{S}_e = (s_e(x, y))$ mit $G = \{0, 1\}$ vorausgesetzt. Als strukturierendes Element wird das Element „8-Nachbarn“ verwendet. Bei der Dilatation wird der binäre Grauwert des Bezugspunktes durch das Maximum der geordneten Folge f ersetzt. Wenn also in der durch das strukturierende Element definierten Nachbarschaft ein Bildpunkt den Grauwert 1 besitzt, so wird der Bezugspunkt durch 1 ersetzt. Dies lässt sich auch durch die logische Verknüpfung der benachbarten Bildpunkte darstellen:

$$\begin{aligned} &\mathbf{S}_e \to \mathbf{S}_a : \\ &\quad s_a(x,y) \;=\; s_e(x-1,y-1) \vee s_e(x-1,y) \vee s_e(x-1,y+1) \vee \\ &\qquad\qquad s_e(x,y-1) \vee s_e(x,y) \vee s_e(x,y+1) \vee \\ &\qquad\qquad s_e(x+1,y-1) \vee s_e(x+1,y) \vee s_e(x+1,y+1). \end{aligned} \tag{6.6}$$

Durch diese Verknüpfung behält ein Bildpunkt mit $s_e(x, y) = 1$ in $\mathbf{S}_a$ seinen Grauwert, während ein Bildpunkt $s_e(x, y) = 0$ in $\mathbf{S}_a$ den Grauwert 1 erhält, wenn in seiner 8-Nachbarschaft eine 1 auftritt. Dies bewirkt die Vergrößerung (Expansion) aller Flächen mit dem Grauwert 1 um einen Rand der Breite eines Bildpunktes. Außerdem werden kleine Flächen mit dem Grauwert 0 eliminiert. Bild 6.4-b und Bild 6.4-c zeigen die ein- bzw. zweimalige Expansion der weißen Flächen, ausgehend von Bild 6.4-a.

Die Erosion ist die entgegengesetzte Operation: Der binäre Grauwert des Bezugspunktes wird durch das Minimum der geordneten Folge f ersetzt. Das bewirkt mit obigem strukturierenden Element das Verkleinern (Kontraktion) aller Flächen mit dem Grauwert 1 um einen Rand der Breite eines Bildpunktes. Ein Bildpunkt mit $s_e(x, y) = 0$ behält in $\mathbf{S}_a$ seinen Grauwert, während $s_e(x, y) = 1$ in $\mathbf{S}_a$ nur dann den Grauwert 1 behält, wenn alle Nachbarn den Grauwert 1 besitzen. Dies kann durch die Boole'sche Und-Verknüpfung erreicht werden:

$$\begin{aligned} &\mathbf{S}_e \to \mathbf{S}_a : \\ &\quad s_a(x,y) \;=\; s_e(x-1,y-1) \wedge s_e(x-1,y) \wedge s_e(x-1,y+1) \wedge \\ &\qquad\qquad s_e(x,y-1) \wedge s_e(x,y) \wedge s_e(x,y+1) \wedge \\ &\qquad\qquad s_e(x+1,y-1) \wedge s_e(x+1,y) \wedge s_e(x+1,y+1). \end{aligned} \tag{6.7}$$

Da in der Boole'schen Algebra der Zusammenhang $\neg(a \wedge b) = \neg a \vee \neg b$ gilt, kann (6.7) so umgeformt werden, dass die Kontraktion des Grauwertes 1 durch eine Expansion des Grauwertes 0 erreicht werden kann. Diese Umformung ist auch anschaulich einleuchtend. Bild 6.4-d und Bild 6.4-e zeigen die ein- bzw. zweimalige Kontraktion des Grauwertes 1.

Es ist klar, dass durch die inverse Operation das Original nicht mehr reproduziert werden kann, da ja z.B. isolierte Bildpunkte mit $s_e(x, y) = 0$ bei der Dilatation eliminiert werden und bei der Erosion nicht mehr auftauchen. Diese aufeinander folgende Anwendung

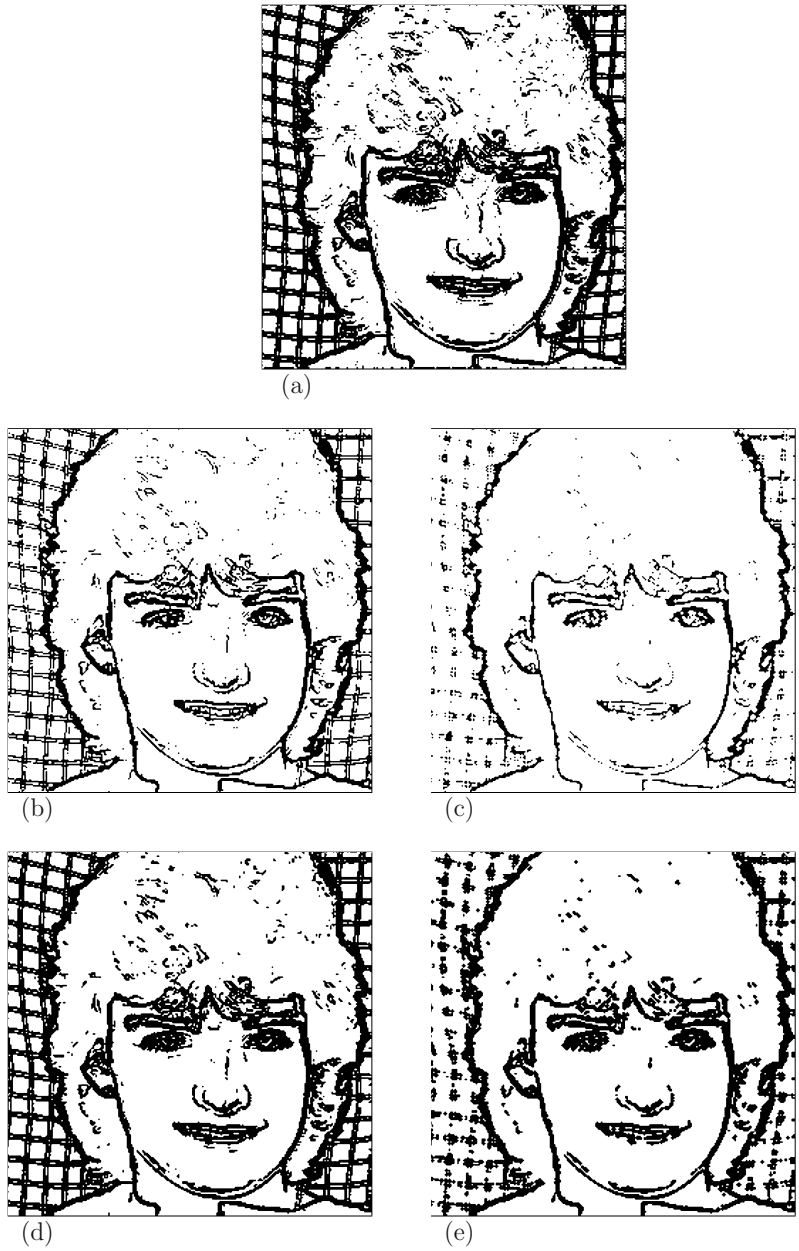
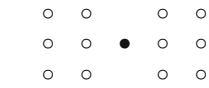


Bild 6.4: Expansions- und Kontraktionsoperatoren. (a) Originalbinärbild der Kantenextraktion. (b) und (c) Expansion (Dilatation) der Flächen mit dem Grauwert 1. (d) und (e) Kontraktion (Erosion) der Flächen mit dem Grauwert 1 durch Expansion der Flächen mit dem Grauwert 0.

von Dilatationen und Erosionen wird in der Praxis oft als *closing* (*Fermeture*) bezeichnet. Die Kombination von Erosionen mit anschließenden Dilatationen heißt dementsprechend *opening* (*Ouverture*).

Ein anderes einfaches Beispiel ist die Extraktion des Randes einer mit 1-en codierten Fläche in einem Binärbild. Dazu wird z.B. eine Dilatation durchgeführt und das Ergebnis mit dem Original mit der Boole'schen Operation XOR (eXclusives OR) verknüpft.

Die Richtung der Wirkungsweise einer Dilatation oder Erosion lässt sich durch die Gestalt des strukturierenden Elementes beeinflussen. In den obigen Beispielen zur Verarbeitung von Binärbildern wurde ein symmetrisches Element verwendet. Soll z.B. eine Dilatation verstärkt in Spaltenrichtung wirken, so könnte etwa das folgende strukturierende Element verwendet werden:



Durch entsprechendes apriori-Wissen kann somit durch geeignete Wahl des strukturierenden Elements die Wirkung dieser morphologischen Operationen auf den jeweiligen Anwendungsfall angepasst werden.

6.5 Morphologie im Grauwertbild

Die morphologischen Operationen lassen sich sinngemäß auch auf Grauwertbilder $\mathbf{S}_e = (s_e(x,y))$ mit $G = \{0,1,...,255\}$ anwenden. Es wird im Folgenden mit SE das strukturierende Element und mit $U(SE(x,y))$ die Menge der durch das strukturierende Element festgelegten Nachbarn eines Bildpunktes in der Position (x,y) bezeichnet. Die Dilatation und die Erosion können dann geschrieben werden:

$$\begin{aligned} \mathrm{dil}(x,y) &= \max\Big(U(SE(x,y))\Big) + c_g \\ \mathrm{ero}(x,y) &= \min\Big(U(SE(x,y))\Big) - c_g \end{aligned} \tag{6.8}$$

Dabei ist c_g ein Gewichtsfaktor, der abhängig von den Grauwerten von $g \in G$ als „Zuschlag" verwendet werden kann. Dann ist allerdings zu beachten, dass im Ergebnisbild z.B. nach einer Dilatation Grauwerte auftreten, die im Originalbild nicht vorhanden waren.

Anhand eines einfachen Beispiels werden im Folgenden die Verwendung von morphologischen Operationen im Grauwertbild erläutert: Die Problemstellung sei, die Messmarken in den vier Ecken des Bildes 6.5-a zu extrahieren. Durch eine Folge von Erosionen verschwinden die Messmarken allmählich (Bild 6.5-b). Wieviele Erosionen hintereinander ausgeführt werden, hängt von der Größe der Bildstrukturen ab. In diesem Beispiel wurden sechs Erosionen mit der Elementarraute durchgeführt. Besitzen die Bildstrukturen eine Vorzugsrichtung, so kann dies durch das strukturierende Element berücksichtigt werden. Im nächsten Schritt werden ebensoviele Dilatationen wie vorher Erosionen ausgeführt

(Bild 6.5-c). Die hellen Messmarken erscheinen dabei nicht mehr, aber größere helle Bildstrukturen erhalten wieder etwa ihre ursprüngliche Form. Das Differenzbild zwischen dem Original und dem erodierten/dilatierten Bild (*opening*-Operation) enthält nur diejenigen Bildinformationen, die durch die morphologischen Operationen verlorengegangen sind, hier im wesentlichen die Messmarken. Da die rechteckigen Objekte an den Ecken nicht exakt reproduziert wurden, treten in diesem Bereichen Störungen auf. Zu beachten ist auch, dass die Spiegelung auf der Kappe des Faserschreibers durch die *opening*-Operation eliminiert wurde. Eine Binarisierung von Bild 6.5-d liefert Bild 6.5-e. Die zusätzlichen Störungen sind deutlich zu erkennen. In einem weiteren Verarbeitungsschritt, in dem die Fläche und die Länglichkeit der schwarzen Bildstrukturen verwendet wurde, konnten die Messmarken extrahiert werden (Bild 6.5-f). Die weitere Verarbeitung besteht in diesem Beispiel aus der geometrischen Entzerrung des Originals, so dass die Messmarken mit ihren Schwerpunkten in den vier Bildecken liegen.

Bei segmentierten Bildern, bei denen die einzelnen Grauwerte der Bildpunkte Codes für die Zugehörigkeit zu einer Klasse sind (logische Bilder), können mit morphologischen Operationen punktweise Störungen eliminiert werden. Die beiden Bilder 6.6-a und 6.6-b zeigen ein Beispiel dazu. Hier wurden durch eine Reihe von Dilatationen und anschließenden Erosionen die punktweisen Störungen eliminiert. Allerdings gehen bei diesen Operationen feine Bildstrukturen verloren. Es kann auch passieren, dass der Zusammenhang der einzelnen Segmente unterbrochen wird. Das ist dann der Fall, wenn z.B. durch Dilatationen schmale Segmentstellen gelöscht und durch die anschließenden Erosionen nicht mehr rekonstruiert werden können (Bilder 6.6-c und 6.6-d).

Mit morphologischen Operationen wie Medianfilter, Dilatation und Erosion, können auch ganze Bildzeilen, die gestört sind, eliminiert werden. Wie bei der Korrektur einzelner Bildpunktstörungen ist hier auch der Nachteil in Kauf zu nehmen, dass die Grauwerte der Umgebungen ebenfalls verändert werden. Soll dies nicht der Fall sein, so können derartige Störungen durch die Berechnung der Korrelation aufeinander folgender Bildzeilen detektiert werden (Abschnitte 5.5 und 5.6).

Auch zur Verarbeitung von Kanten sind die morphologischen Operationen geeignet. Auf diese Thematik wird im Kapitel 7 näher eingegangen.

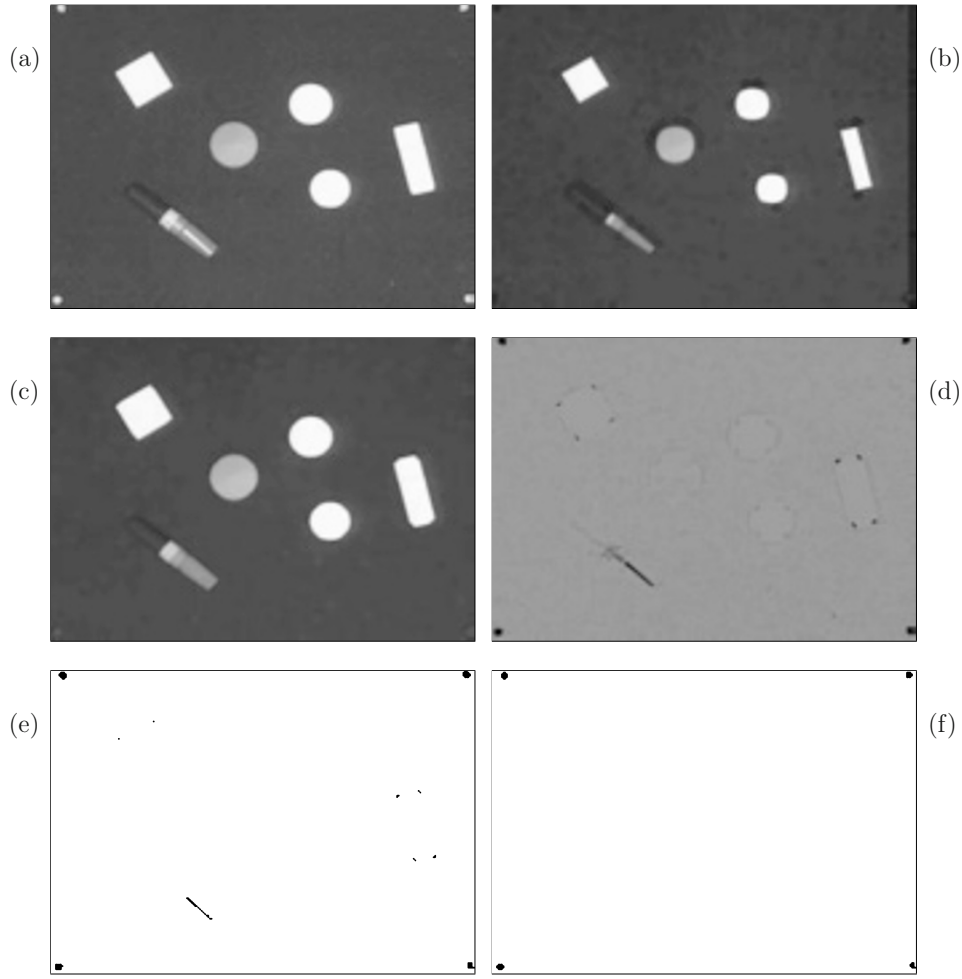
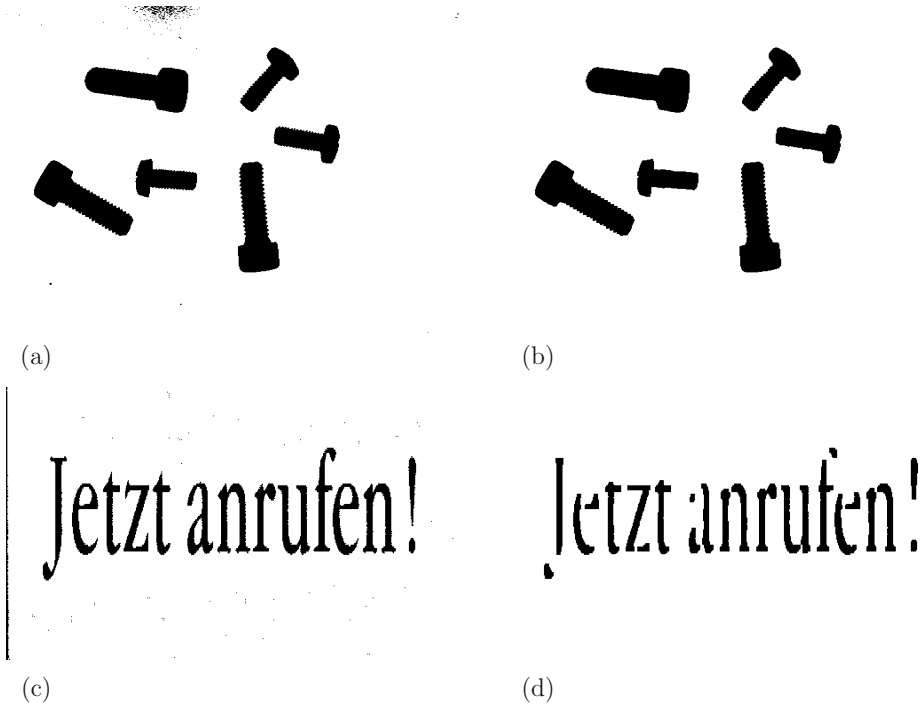


Bild 6.5: Beispiel zur Grauwertbildmorphologie. (a) Testbild mit mehreren Objekten und vier Messmarken in den Ecken. (b) Durch eine Folge von Erosionen verschwinden die hellen Messmarken. (c) Weitere Dilatationen reproduzieren etwa die Form größerer Bildstrukturen, die Messmarken erscheinen nicht mehr. (d) Das Differenzbild des Originals und des erodierten/dilatierten Bildes enthält die Bildinformationen, die verloren gingen. (e) Nach einer Binarisierung (Schwellwertbildung) sind deutlich die Messmarken und weitere Bildstörungen zu sehen. (f) Nach einer Formanalyse der schwarzen Bildteile (Flächeninhalt und Länglichkeit) konnten die Messmarken extrahiert werden. Die weitere Verarbeitung ist in diesem Beispiel eine geometrische Entzerrung des Originals, so dass die Messmarken in den Bildecken liegen.



(a) (b)

(c) (d)

Bild 6.6: (a) Binarisiertes Testbild. (b) Durch Dilatationen und anschließende Erosionen wurden die punktweisen Störungen eliminiert. Bei diesen Operationen gehen allerdings feine Bildstrukturen verloren. (c) Testbild mit Bereichen, bei denen die Segmente sehr schmal werden. (d) Die Dilatationen unterbrechen hier den Zusammenhang der Segmente.

Kapitel 7

Kanten und Linien

7.1 Anwendungen

In den folgenden Abschnitten werden Verfahren beschrieben, bei denen das Herausarbeiten von Grauwertübergängen wichtig ist. Dazu gehören einfache Verfahren der Kantenextraktion, aber auch komplizierte Algorithmen, die eine Kante über den Gradient beschreiben. Anschließend wird besprochen, wie sich der Rand eines Segments mit den Methoden der mathematischen Morphologie ermitteln lässt. Weiter folgen Beispiele zur Verarbeitung von Linien. Hier wird unter anderem auch die Skelettierung dargestellt.

7.2 Grundlegendes über Kanten und Linien

Physiologische Untersuchungen haben gezeigt, dass das menschliche Auge besser auf Diskontinuität als auf Kontinuität geeicht ist. Das ist auch plausibel, da diejenigen Bereiche, die Helligkeitsübergänge aufweisen, uns bei der Betrachtung eines Bildes in der Regel mehr Information geben, als weitgehend homogen graue oder einfarbige Bereiche. Für die Verarbeitung digitalisierter Bilder heißt das, dass der Versuch, die Verarbeitung, z.B. eine Segmentierung, kanten- und linienorientiert anzugehen, durchaus erfolgversprechend sein kann. In der Bildverarbeitung findet dieses Thema seit jeher Beachtung [Jaeh11, Gonz18, Toen05].

Dieser Abschnitt beschäftigt sich vorwiegend mit der Extraktion und der Weiterverarbeitung von Kanten und Linien. Das Ausgangsbildmaterial, auf das die Verfahren zur Kantenextraktion angewendet werden, kann dabei auf unterschiedliche Weise zustande kommen.

Ein Grauwertbild, das mit einem Videosensor oder einem Scanner aufgezeichnet wurde, ist ein mögliches Ausgangsbild für die Hervorhebung von Kanten. Zunächst muss untersucht werden, was unter einer Kante oder, präziser ausgedrückt, unter einer *Grauwertkante* zu verstehen ist. Dazu wird ein Grauwertbild $\mathbf{S}_e = (s_e(x, y))$ betrachtet, das einen hellen und einen dunklen Bildbereich enthält (Bild 7.1, oberes Teilbild)

Der Funktionsverlauf ist in der grafischen Darstellung der Grauwerte (*Grauwertprofil*)

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_7

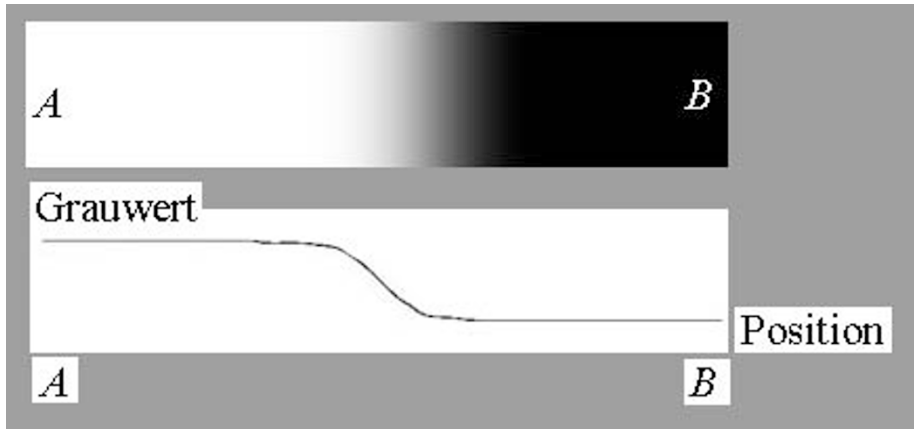


Bild 7.1: Oberes Teilbild: Bildausschnitt mit einer vergrößerten Grauwertkante. Unteres Teilbild: Grafische Darstellung des Grauwertverlaufs entlang der Strecke $\overline{AB}$ (Grauwertprofil).

entlang der Strecke $\overline{AB}$ durch eine große Steigung gekennzeichnet. Stellt man sich diesen Sachverhalt zweidimensional als „Grauwertgebirge" vor, so wird sich im Übergangsbereich eine „Rampe" ausprägen.

Eine Grauwertkante wird mit Hilfe des Gradienten (Abschnitt 5.4) charakterisiert: In homogenen oder nahezu homogenen Bildbereichen ist der Betrag des Gradienten gleich null oder sehr klein, während er im Bereich der Kante einen großen Betrag aufweist.

Die Charakterisierung einer Kante allein mit Hilfe des Gradientenbetrags ist noch nicht ausreichend, da ja z.B. ein Bildbereich mit starker Oberflächenstruktur auch viele Punkte mit großem Betrag des Gradienten aufweist. Allerdings werden hier die Richtungen mehr oder weniger zufällig verteilt sein. So wird man von einer Kante zusätzlich verlangen, dass sie sich in einem bestimmten Bildausschnitt durch eine Vorzugsrichtung des Gradienten auszeichnet, also kontinuierlich verläuft. Ab welcher Ausdehnung von einer Kante gesprochen wird, kann nicht pauschal gesagt werden, dies muss vielmehr im jeweiligen Anwendungsfall festgelegt werden.

In obiger Beschreibung wurde der Begriff der Kante als Grauwertkante erläutert. Kanten können aber auch in anderen Merkmalen auftreten, so z.B. eine Farbkante . Eine *Farb-* oder *Signaturkante* liegt vor, wenn in mindestens einem Farb- (Spektral-) Kanal eine Grauwertkante auftritt.

Werden durch ein Merkmal, das Bewegungen in einer Bildfolge erfasst (Kapitel 14), bewegte Bildbereiche von unbewegten Bildbereichen abgegrenzt, so könnte man von einer *Bewegungskante* sprechen.

Eine *Texturkante* entsteht im Übergangsbereich von unterschiedlichen Oberflächenstrukturen (Bild 7.2).

(a) (b)

Bild 7.2: Beispiel zu unterschiedlichen Oberflächenstrukturen (Texturen).
(a) Bildausschnitt mit verschiedenen Texturen. (b)Durch die Anwendung eines Texturmaßes werden Texturkanten berechnet.

Um den Übergangsbereich der beiden Texturen zu finden, werden hier texturbeschreibende Merkmale verwendet (Kapitel 15). Einige Beispiele dazu sind:

- Die Berechnung der Streuung für Umgebungen.
- Die Berechnung des Betrags und/oder der Richtung des Gradienten.
- Texturmaße, die aus der *co-occurrence*-Matrix abgeleitet werden (Abschnitt 3.10).
- Texturbeschreibende Parameter aus der Autokorrelation von Bildausschnitten, der Verarbeitung von Fourier-Koeffizienten oder der Verwendung anderer Transformationen.

Eine ausführliche Darstellung der Texturmerkmalsberechnung, auch in Verbindung mit der fraktalen Geometrie, wird in den Kapiteln 15, 17 und 18 gegeben.

Wenn die Texturmaßzahlen in die Grauwertmenge $G = \{0, ..., 255\}$ abgebildet werden, so ergibt sich ein Bild, dessen Grauwerte Maßzahlen für Texturen sind. Ein derartiges Bild kann natürlich als Eingabebild für eine Kantenextraktion dienen. Allerdings wird der berechnete Kantenverlauf, je nach der Größe der jeweiligen Texturen, den tatsächlichen Verlauf nur angenähert wiedergeben.

Nun zum Begriff der Linie. Eine Linie ist ein Gebilde, das beidseitig durch Kanten begrenzt ist, die einerseits nicht zu weit voneinander entfernt sind und zum anderen weitgehend parallel verlaufen (Bild 7.3). Bei welchem Abstand und welcher Ausdehnung der begrenzenden Kanten ein Gebilde als Linie angesehen wird oder nicht, ist wieder abhängig vom jeweiligen Problemkreis.

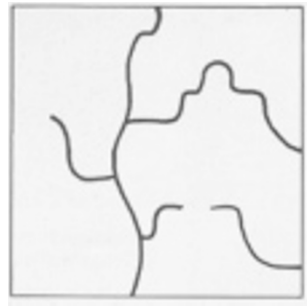


Bild 7.3: Verlauf einer Linie. Verzweigungen und Unterbrechungen können den Linienverlauf bestimmen.

Bei einer Linie in einem Luftbild kann es sich z.B. um eine Straße, eine Eisenbahnlinie oder um einen Flusslauf handeln. In einer Strichzeichnung sind die dunklen (oder hellen) Linien auf dem hellen (oder dunklen) Hintergrund die eigentlichen Informationsträger. Bei einem Fingerabdruck wiederum prägen sich die Papillaren als helle oder dunkle Linien aus.

Wird auf eine Grauwertkante ein lokaler Differenzenoperator (Abschnitt 5.4) angewendet, so wird im gefilterten Bild der Verlauf der Grauwertkante durch eine Linie wiedergegeben. Aus diesem Grund ist der Übergang zwischen kanten- und linienorientierter Bildsegmentierung fließend.

Eine kantenorientierte Bildsegmentierung läuft im wesentlichen so ab (Bild 7.4):

- Aus dem Originalbild (Bild 7.4-a) wird im ersten Schritt die Grauwertkante zwischen Objekt und Hintergrund extrahiert. Sie ist in Bild 7.4-b als Linie dargestellt.
- Im nächsten Schritt wird das kantenextrahierte Bild durch ein Schwellwertverfahren binarisiert. Dadurch wird logisch zwischen Kantenbereichen und keinen Kantenbereichen unterschieden.
- Mit Hilfe der zusätzlichen Information, ob das zu segmentierende Objekt links oder rechts der Kante liegt, kann jetzt eine Maske angefertigt werden, die logisch zwischen Hintergrund und dem Bereich des Objektes unterscheidet (Bild 7.4-c).
- Schließlich kann, falls benötigt, der Bereich des Objektes mit Hilfe der Maske aus dem Original ausgeblendet werden (Bild 7.4-d).

Im Folgenden ist zu untersuchen, wie die Bildpunkte auf der Grauwertkante ermittelt werden können. Dabei unterscheidet man zwischen zwei Vorgehensweisen. Bei der *parallelen Methode* werden alle Bildpunkte des zu verarbeitenden Bildes einem Extraktionsprozess unterzogen, während bei der *sequentiellen Methode*, ausgehend von einem bestimmten

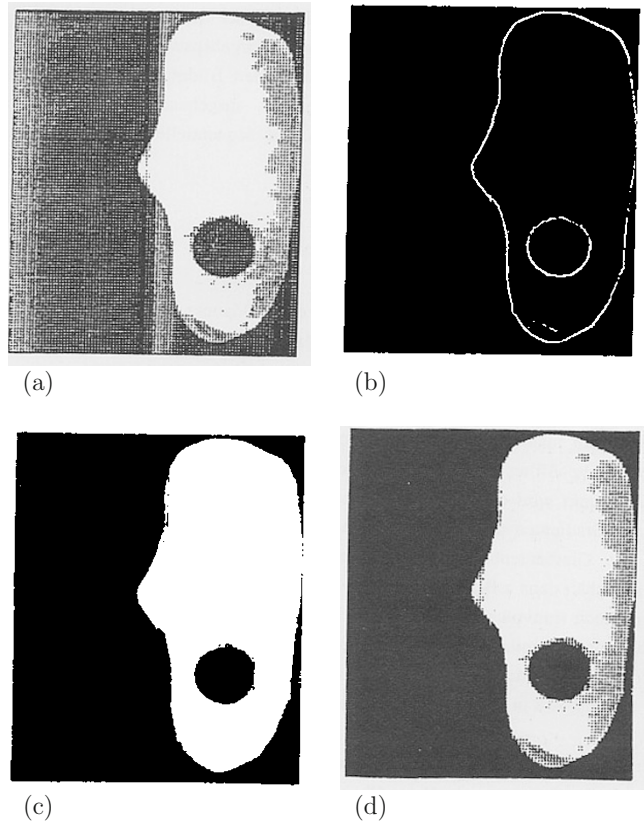


Bild 7.4: Kantenorientierte Bildsegmentierung. (a) Originalbild. (b) Extraktion der Kante zwischen Objekt und Hintergrund. (c) Maske: Hintergrund/Objekt. (d) Original mit ausgeblendetem Hintergrund.

Anfangspunkt, Nachfolgepunkte gesucht werden. Die sequentielle Methode wird auch als *Linienverfolgung* bezeichnet.

7.3 Einfache Verfahren zur Kantenextraktion

Mit Differenzenoperatoren (Abschnitt 5.4) werden Grauwertübergänge deutlicher herausgearbeitet. Einige Anwendungsbeispiele wurden bereits in vorhergehenden Kapiteln besprochen: Z.B. wurde in Abschnitt 4.8 dargestellt, dass eine Kontrastanreicherung durch die Veränderung der Grauskala und die Überlagerung mit dem Laplace-gefilterten Originalbild die Bildqualität für die visuelle Betrachtung verbessern kann.

Dieser Abschnitt beschäftigt sich vorwiegend mit der Extraktion und der Weiterverarbeitung von Kanten. Das Ausgangsbildmaterial, auf das die Verfahren zur Kantenextraktion angewendet werden, kann dabei, wie in Abschnitt 7.2 erläutert, auf unterschiedliche Weise zustande kommen.

Die Anwendung eines Differenzenoperators auf das (eventuell vorverarbeitete) Eingabebild ist der erste Schritt in Richtung Kantenextraktion. Geeignet sind dazu z.B. die Operatoren, die im Grundlagenabschnitt 5.4 aufgeführt sind.

Die Filterkerne (5.21) sind in der Praxis zu anfällig für Störungen. Häufig wird der Sobeloperator (5.23) verwendet. Er bringt bei den meisten Anwendungen gute Ergebnisse, auch wenn er dazu neigt, die Grauwertkanten verhältnismäßig dick darzustellen. Dadurch können feine Strukturen verloren gehen.

Bei Verwendung von (5.21) oder (5.23) ist zu beachten, dass mit diesen Filterkernen eine partielle Ableitung in Zeilen- und in Spaltenrichtung nachvollzogen wird. Um die Kanten herauszuarbeiten, muss anschließend noch der Betrag des Gradienten mit (5.26) berechnet werden.

Der Laplace-Operator (5.30) liefert direkt ein kantenextrahiertes Bild. Welche der vorgeschlagenen Varianten am besten geeignet ist, muss in jedem Anwendungsfall anhand des aktuellen Bildmaterials entschieden werden.

Für alle Filterkerne gilt die Bemerkung, dass die vorgeschlagene Größe nicht auf $m = 3$ beschränkt werden muss. Vielmehr können sie sinngemäß auch auf größere Umgebungen ausgedehnt werden. Allerdings ist bei größeren Filterkernen zu bedenken, dass sich dadurch höhere Rechenzeiten ergeben.

Nachdem die Kanten mit einem der vorgeschlagenen Operatoren vorbereitet wurden, folgt als nächster Schritt eine Segmentierung. Die Eigenschaft „ein Bildpunkt liegt auf einer Kante“ drückt sich durch einen hohen Betrag des Gradienten aus. Als Segmentierung kann daher eine einfache Schwellwertbildung (Binarisierung, Abschnitt 4.5) verwendet werden, bei der Bildpunkte mit einem hohen Gradientenbetrag als Kantenpunkte gewertet werden.

A7.1: Extraktion von Kanten.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild mit $G = \{0, 1, ..., 255\}$ als Grauwertmenge (Eingabebild). Dieses Bild kann auch das Ergebnis einer längeren Vorverarbeitung (z.B. einer Berechnung von Texturparametern) sein.

◇ $\mathbf{S}_a = (s_a(x,y))$ ein einkanaliges Ausgabebild. Der Grauwert 0 bedeutet „keine Kante“, der Grauwert 1 oder 255 bedeutet „Kante“.

Algorithmus:

(a) Anwendung eines Differenzenoperators (z.B. Sobeloperator) auf das Eingabebild $\mathbf{S}_e$.

(b) Berechnung des Betrags des Gradienten aus den beiden Richtungskomponenten $s_x(x,y)$ und $s_y(x,y)$ gemäß:
$g_b(x,y) = \sqrt{s_x(x,y)^2 + s_y(x,y)^2}$.

(c) Binarisierung der Gradientenbeträge mit einem geeigneten Schwellwert c:
$$s_a(x,y) = \begin{cases} 0, & \text{falls } g_b(x,y) \leq c, \\ 1, & \text{sonst.} \end{cases}$$

Ende des Algorithmus

Die Bilder 7.5-a, 7.5-b und 7.5-c zeigen ein Beispiel eines Originals und des daraus berechneten und binarisierten Kantenbildes.

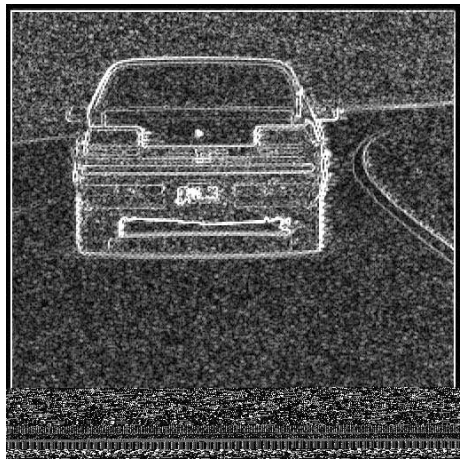
Bei der hier geschilderten, einfachen Methode zur Kantenextraktion tritt das Problem auf, dass Kanten, die im Eingabebild zwar vorhanden sind, die sich aber nicht so deutlich abheben wie andere Kanten, eventuell nicht erkannt werden. Das liegt z.B. daran, dass sich für derartige Kanten ein niederer Wert für den Gradientenbetrag ergibt, der bei der Binarisierung dann unter dem Schwellwert liegt. In der Bildfolge 7.5-a bis 7.5-c ist dieser Effekt deutlich zu sehen.

Wenn das Kantengebilde vollständig extrahiert werden soll, so müssen aufwändigere Methoden angewendet werden. Bevor auf die weitere Verarbeitung des binarisierten Kantenbildes, wie z.B. Skelettierung oder das Schließen unterbrochener Linien, eingegangen wird, werden in den folgenden Abschnitten Anregungen zu einer gezielteren Kantendetektion gegeben.

7.4 Parallele Kantenextraktion

Bei der parallelen Kantenextraktion wird für jeden Bildpunkt ein Maß berechnet, das als Wahrscheinlichkeit interpretiert werden kann, dass er zu einer Kante gehört. Die wesentlichen Schritte dabei sind im Folgenden wiederholt:

(a)



(b)

(c)

Bild 7.5: (a) Originalbild. (b) Kantenbild nach einem Sobeloperator und einer Berechnung der Gradientenbeträge. (c) Binarisiertes Bild. Hier werden z.B. im oberen Bildhintergrund Bildpunkte als Kanten ausgewiesen, wo im Original keine Kanten sind. Kanten, deren Gradientenbeträge unter dem Schwellwert liegen, werden nicht segmentiert.

Zuerst wird ein lokaler Differenzenoperator (z.B. Laplace Operator oder Sobel Operator) oder ein Hochpassfilter im Frequenzbereich angewendet (Abschnitt 5.4 und Kapitel 8). Jedem Bildpunkt wird dadurch eine Maßzahl zugewiesen, die sich entweder nur aus dem Betrag des Gradienten oder aus Betrag und Richtung zusammensetzt. Im nächsten Schritt werden die Gradientenbeträge einer Schwellwertbildung unterzogen. Ein so erzeugtes Binärbild enthält dann z.B. die Kanten des Originalbildes als Linien mit dem Grauwert 1. Es kann jetzt noch sinnvoll sein, das Binärbild von isolierten Einsen zu säubern, die Linien zu verdünnen und eventuelle Unterbrechungen zu überbrücken (z.B. mit morphologischen Operationen, Kapitel 6). Nachdem das Binärbild ,,gesäubert" wurde, kann dann die Maskenbildung, wie in Abschnitt 7.2 erläutert, durchgeführt werden.

Soll mit dieser parallelen Methode eine Linie extrahiert werden, so ist zu beachten, dass eine Linie nach der Filterung als Doppellinie erscheint. Dieser Sachverhalt kann u.U. als zusätzliche Information bei der weiteren Verarbeitung verwendet werden.

Bei der geschilderten Vorgehensweise wurde die Entscheidung, ob ein Bildpunkt auf einer Kante liegt oder nicht, nur vom Betrag des Gradienten und einer anschließenden Schwellwertbildung abhängig gemacht. Um hier mehr Sicherheit zu gewinnen, kann anstatt der Schwellwertbildung auch ein Klassifikator (Kapitel 20) verwendet werden. Dazu ein Beispiel (Bild 7.6-a): In einem Bild sei ein dunkles Objekt auf einem hellen Hintergrund abgebildet. Wird in einem zweidimensionalen Merkmalsraum auf der Abszisse der Grauwert eines Bildpunktes im Original aufgetragen und als Ordinate der Betrag des Gradienten, so werden sich die Verhältnisse ergeben, wie sie Bild 7.6-b zeigt. Der Klasse ,,Kante" ist in diesem Beispiel also der rechte obere Bereich des Merkmalsraumes zugeordnet. Dieser Sachverhalt kann z.B. mit einem fest-dimensionierten Klassifikator, der anhand einer Stichprobe trainiert wird, erfasst werden (Kapitel 20).

Da eine Grauwertkante meistens breiter als ein Bildpunkt ist, kann die Entscheidung, ob ein bestimmter Bereich auf der Kante liegt oder nicht, auch anhand von benachbarten Bereichen getroffen werden (Bild 7.7). Man könnte etwa immer $3 \cdot 3$ Bildpunkte große Bildausschnitte auf einmal zuordnen. Dazu berechnet man den Mittelwert des Gradientenbetrags dieses Ausschnittes und vergleicht ihn mit den entsprechenden Mittelwerten der 4-benachbarten oder 8-benachbarten $3 \cdot 3$-Umgebungen. Wenn der Mittelwert des Zentrums über einer Schwelle liegt, werden die $3 \cdot 3$ Bildpunkte der Klasse ,,Kante" zugeordnet.

Eine andere Vorgehensweise bietet sich an, wenn man eine Vermutung über den Verlauf einer Kante oder Linie in einem Bildausschnitt hat. In diesem Fall wird ein mathematisches Modell des Kanten- oder Linienverlaufs aufgestellt und mit dem tatsächlichen Verlauf im Bildausschnitt verglichen. Auch dazu ein Beispiel: Im Rahmen einer Kantenextraktion ergebe sich die Vermutung, dass in einem Bildausschnitt $s_e(x+k, y+k), k = 0, 1, ..., m-1$ eines Bildes $\mathbf{S}_e = (s_e(x, y))$ ungefähr diagonal von links oben nach rechts unten eine Grauwertkante verläuft. Diese Vermutung kann durch eine funktionale Approximation des Verlaufs der Kante modelliert werden, z.B. durch die Aufstellung einer Geradengleichung oder durch die Berechnung einer ,,Schablone" (*Template*) $\mathbf{H} = (h(u, v))$ der Größe $m \cdot m$,

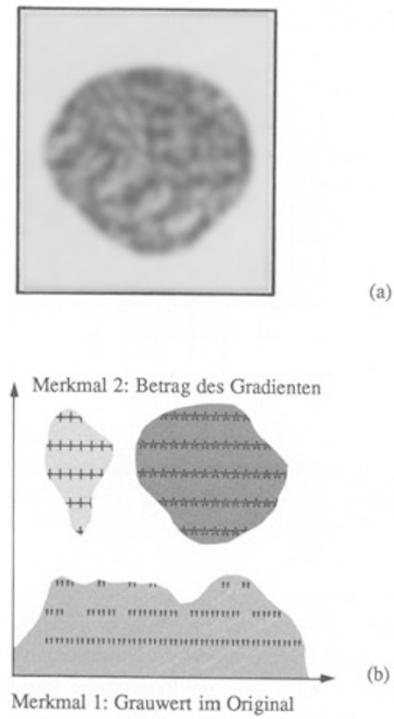


Bild 7.6: Klassifizierung der Bildpunkte im Merkmalsraum Grauwert/Betrag des Gradienten. (a) Originalbild: heller Hintergrund, dunkles Objekt. (b) Merkmalsraum: Grauwert/Betrag des Gradienten. + Bildpunkte mit kleinem Grauwert und großem Gradientenbetrag; * Bildpunkte mit großem Grauwert und großem Gradientenbetrag; " Bildpunkte mit kleinem Gradientenbetrag.

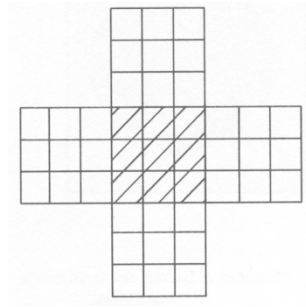


Bild 7.7: Zuordnung von $3 \cdot 3$-Ausschnitten anhand des mittleren Gradientenbetrags der Umgebung.

in der die Bildpunkte folgende Grauwerte besitzen:

$$h(u,v) = \begin{cases} 0, & u > v, \\ 1, & \text{sonst.} \end{cases} \tag{7.1}$$

Der zu überprüfende Bildausschnitt wird nun binarisiert und führt zur binären Bildmatrix $s_{bin}(x+k, y+k)$. Jetzt wird ein Maß d der Abweichung des Ausschnittes vom Modell berechnet:

$$d = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \Big(s_{bin}(x+u, y+v) - h(u,v) \Big)^2 = \tag{7.2}$$

$$= \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_{bin}(x+u, y+v)^2 - 2 \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_{bin}(x+u, y+v) h(u,v) + \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} h(u,v)^2.$$

Der erste Term der ausquadrierten binomischen Formel (7.2) ist für den betrachteten Bildausschnitt konstant, der letzte Term für den Modellausschnitt. Zur Berechnung des Maßes für die Abweichung ist somit der mittlere Term ausreichend. Die modellierte Lage der Kante im Bildausschnitt wird dann akzeptiert, wenn die Größe

$$d' = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} s_{bin}(x+u, y+v) h(u,v) > c \tag{7.3}$$

ist, mit einer geeigneten Schwelle c. Die Berechnung der Übereinstimmung zwischen Bildausschnitt und Modell gemäß (7.3) läuft somit auf die Berechnung der Korrelation hinaus.

7.5 Kantenextraktion mit Betrag und Richtung des Gradienten

Bei dem hier geschilderten *Kantendetektor* werden neben Gradientenbeträgen auch die Gradientenrichtungen ausgewertet. Der erste Schritt ist wie oben die Anwendung eines Differenzenoperator, der die Richtungskomponenten $s_x(x, y)$ und $s_y(x, y)$ des Gradienten liefert. Die Gradientenbeträge $g_b(x, y)$ werden mit (5.23) berechnet. Um zu einer betragsunabhängigen Kantenbewertung zu kommen, werden die Gradientenbeträge einer *Rangordnungsoperation* unterworfen. Die Gradientenbeträge in den $3 \cdot 3$–Umgebungen zweier Bildpunkte könnten etwa folgende Werte haben:

$$\begin{array}{ccc} 18 & 31 & 26 \\ 17 & 44 & 16 \\ 19 & 54 & 20 \end{array} \quad \text{und} \quad \begin{array}{ccc} 78 & 234 & 112 \\ 81 & 227 & 104 \\ 65 & 103 & 230 \end{array}. \tag{7.4}$$

Mit einem fixen Schwellwert ist es unmöglich, aus beiden Umgebungen die richtigen Kantenstücke zu erhalten.

Die geordneten Folgen (die *Rangordnungen*) dieser Umgebungen lauten:

$$16 \quad 17 \quad 18 \quad 19 \quad 20 \quad 26 \quad 31 \quad 44 \quad 54 \tag{7.5}$$

und

$$65 \quad 78 \quad 81 \quad 103 \quad 104 \quad 112 \quad 227 \quad 230 \quad 234. \tag{7.6}$$

Nun wird der Wert jedes Bildpunktes durch den Wert seiner Position in dieser Rangordnung ersetzt (*Positionsrang*). Man erhält dann:

$$\begin{array}{ccc} 3 & 7 & 6 \\ 2 & 8 & 1 \\ 4 & 9 & 5 \end{array} \quad \text{und} \quad \begin{array}{ccc} 2 & 9 & 6 \\ 3 & 7 & 5 \\ 1 & 4 & 8 \end{array}. \tag{7.7}$$

Es ist deutlich zu sehen, dass die beiden Kantenbereiche, ausgezeichnet durch hohe Gradientenwerte, im Positionsrang ebenfalls die hohen Werte erhalten. Außerdem werden beide Kantenausschnitte gleich bewertet, obwohl die Werte der Gradienten im linken Ausschnitt wesentlich kleiner sind. Legt man als Schwelle für die Klasse „Kante“ den Schwellwert $c \geq 7$ fest, so erhält man folgende binarisierte Ausschnitte, die den Kantenverlauf richtig wiedergeben:

$$\begin{array}{ccc} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{array} \quad \text{und} \quad \begin{array}{ccc} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}. \tag{7.8}$$

Nun einige Bemerkungen zu Problemen, die sich bei der Berechnung und Auswertung der Positionsrangbilder ergeben. Zunächst können in einer sortierten Folge gleiche Grauwerte auftreten. Es stellt sich die Frage, welchen Positionsrang man ihnen zuordnet. Der

höchste Rang schließt sich aus, da sonst in homogenen Gebieten alle Punkte diesen Rang erhalten würden und somit als Kante markiert würden. Der niederste Rang ist ebenfalls problematisch, da dadurch in Linienstücken häufige Unterbrechungen auftreten. Am besten ist wohl ein mittlerer Rang geeignet, der sich durch eine Mittelung und Rundung über die in Frage kommenden Rangpositionen berechnet.

Ein anderer Parameter, der das Ergebnis beeinflusst, ist die Umgebungsgröße, die zur Positionsrangberechnung verwendet wird. Hier ist, wie häufig in der Bildverarbeitung und Mustererkennung, zwischen repräsentativen Ergebnissen und benötigter Rechenzeit ein Kompromiss zu schließen. Um einigermaßen zusammenhängende Kantenstücke zu erhalten, sollte man schon eine $5 \cdot 5$– oder $7 \cdot 7$–Umgebung verwenden. Bei größeren Umgebungen wird der Rechenaufwand zu groß.

Schließlich beeinflusst die Schwelle, die zwischen „Kante“ und „nicht Kante“ unterscheidet, das Ergebnis wesentlich. Sie ist abhängig von der jeweiligen Umgebung und muss anhand des aktuellen Bildmaterials durch einige Versuche ermittelt werden. Bei $5 \cdot 5$– bzw. $7 \cdot 7$–Umgebungen kann das Probieren etwa mit einer Schwelle von 14 bzw. 25 beginnen.

Das binarisierte Positionsrangbild (im folgenden als *Rangbild* bezeichnet) wird auch in relativ homogenen Bildbereichen Kanten ausweisen. Diese können aber in Verbindung mit einer Richtungssystematik eliminiert werden.

Für die *Richtungssystematik* wird aus den Richtungskomponenten $s_x(x, y)$ und $s_y(x, y)$ ein Maß für die Gradientenrichtung an dieser Stelle ermittelt. Die Richtungswinkel φ könnten dazu gemäß

$$\varphi = \arctan \frac{s_y(x, y)}{s_x(x, y)} \tag{7.9}$$

berechnet werden. Aus Laufzeitgründen und aufgrund der Vorgehensweise, wie die Richtungssystematik ermittelt wird, ist es zu empfehlen, nicht die arctan–Berechnung von (7.9) zu verwenden, sondern die Werte $s_x(x, y)$ und $s_y(x, y)$ zu skalieren und auf ein umgebendes Quadrat mit einer Richtungseinteilung abzubilden. Die skalierten Werte zeigen dann direkt auf ein Richtungsfeld. Bild 7.8 zeigt ein Beispiel eines Richtungsquadrats der Größe $7 \cdot 7$.

Um in der Umgebung des Bildpunktes in der Position (x, y) systematisch nach Kantenstücken zu suchen, werden benachbarte Bildpunkte inspiziert, ob sie gleiche Gradientenrichtungen haben. Dabei sind unterschiedliche Strategien möglich, zwei davon werden kurz vorgestellt.

Beim ersten Verfahren (Bild 7.9) werden, ausgehend von der in der Position (x, y) gefundenen Richtung (im Beispiel die Richtung 5), die mit einem * markierten Bildpunkte untersucht. Richtungen, die innerhalb einer festzulegenden Toleranz liegen, erhöhen die Systematik. Die Richtungstoleranz ist dabei notwendig, da ja an der gerade untersuchten Position die Kante einen Knick machen könnte. Im Ergebnisbild (*Systematikbild*) sind die Grauwerte Maßzahlen für die Systematik, wobei ein großer Wert eine hohe Wahrscheinlichkeit für eine Kante ausdrückt.

Beim zweiten Verfahren wird ein schmälerer Richtungskanal verwendet als im ersten Verfahren (Bild 7.10). Er wird in drei Teile zerlegt und für jeden Teil werden Eigenschaften

9	8	7	6	5	4	3
10						2
11						1
12			x, y			0
13						23
14						22
15	16	17	18	19	20	21

Bild 7.8: Richtungsquadrat der Größe $7 \cdot 7$.

9	8	7	6*	5*	4*	3
10			*	*	*	2
11		*	*	*		1
12		*	5	*		0
13		*	*	*		23
14	*	*	*			22
15	16*	17*	18*	19	20	21

Bild 7.9: Richtungsquadrat der Größe $7 \cdot 7$. Die Richtungen der markierten Positionen werden bei einer vorgegebenen Richtung (z.B. 5) für den Bildpunkt in der Position (x, y) untersucht, ob sie einen positiven Beitrag zur Systematik liefern oder nicht.

berechnet, wie z.B. die Anzahl der vorkommenden Bildpunkte mit $s_x(x, y) = s_y(x, y) = 0$ (homogene Bildbereiche), die durchschnittliche Richtung auf einem Teilstück und die Summe der absoluten Abweichungen von der Hauptrichtung in der Position (x, y). Aus einer kombinierten Auswertung dieser Eigenschaften kann abgeleitet werden, ob Richtungssystematik vorliegt oder nicht. Das Ergebnisbild (Systematikbild) ist hier ein Binärbild.

Zur letztlich gewünschten Kantenextraktion wird in der Auswertung das Rangbild mit dem Systematikbild verknüpft. Versuche haben gezeigt, dass es besser ist, vom Rangbild auszugehen. Eine einfache UND-Verknüpfung der beiden Binärbilder hat sich nicht bewährt. Vielmehr werden im Rangbild zusammenhängende Kantenstücke von einer festzulegenden Mindestlänge ermittelt. Dann wird versucht, diese Kantenstücke im Systema-

9	8	7	6	5	4	3
10^1						2
11	[1]	[2]				1
12			10^2			0
13				[2]	[3]	23
14						22^3
15	16	17	18	19	20	21

Bild 7.10: Der Richtungskanal wird in drei Teilstücke aufgespaltet (hier markiert mit [1], [2] und [3]). Zu jedem Teilstück werden aus den Richtungen der betroffenen Bildpunkte Eigenschaften berechnet.

tikbild ebenfalls zu finden. Ist dies mit einer bestimmten Toleranz erfolgreich, so wird das Kantenstück in das endgültige Kantenbild übernommen. Der Toleranzbereich für die Übereinstimmung kann dabei wie folgt modelliert werden:

$$\alpha = \textit{Größe des Kantenstücks}; \tag{7.10}$$

$$\beta = \frac{\textit{Anzahl der übereinstimmenden Punkte}}{\alpha}.$$

Um zu einem akzeptablen Kantenbild zu kommen, wird eine minimale Kantengröße α und ein minimaler Übereinstimmungsgrad β festgelegt. In der Bildfolge 7.11 ist ein Beispiel zusammengestellt. Bild 7.11-a ist das Originalbild, Bild 7.11-b zeigt das Rangbild mit einer Umgebungsgröße von $7 \cdot 7$, Bild 7.11-c das dazu passende Systematikbild und 7.11-d das Kantenbild als Ergebnis.

Zusammenfassend lautet der gesamte Algorithmus zu diesem Kantendetektor folgendermaßen:

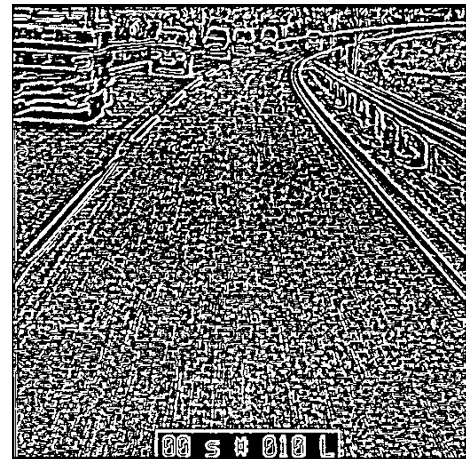
A7.2: Extraktion von Kanten mit Gradientenrichtung und Gradientenbetrag.

Voraussetzungen und Bemerkungen:

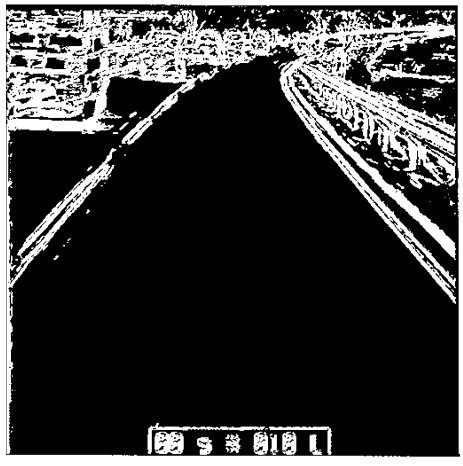
◇ $\mathbf{S}_e = (s_e(x,y))$ ein einkanaliges Grauwertbild mit $G = \{0, 1, ..., 255\}$ als Grauwertmenge (Eingabebild). Dieses Bild kann auch das Ergebnis mehrerer Vorverarbeitungsschritte sein (z.B. Texturparameter).

◇ $\mathbf{S}_a = (s_a(x,y))$ ein einkanaliges Ausgabebild. Der Grauwert 0 bedeutet „keine Kante“, der Grauwert 1 oder 255 bedeutet „Kante“.

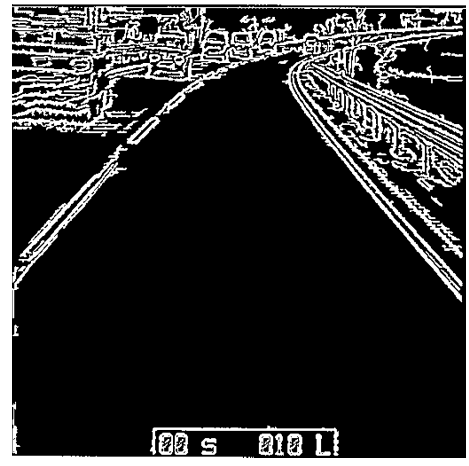
(a)



(b)



(c)



(d)

Bild 7.11: (a) Originalbild einer Straßenszene. (b) Rangbild. (c) Systematikbild. (d) Ergebnis der Kantenextraktion. Bemerkenswert ist, dass selbst die im Original kaum sichtbaren Stützen der Leitplanken am rechten Straßenrand deutlich zu sehen sind.

Algorithmus:

(a) Anwendung eines Differenzenoperators (z.B. Sobeloperator) auf das Eingabebild $\mathbf{S}_e$.

(b) Berechnung des Betrags des Gradienten aus den beiden Richtungskomponenten $s_x(x,y)$ und $s_y(x,y)$ gemäß:

$$g_b(x,y) = \sqrt{s_x(x,y)^2 + s_y(x,y)^2}.$$

(c) Berechnung des Rangbildes, wie es in den obigen Beispielen beschrieben wurde.

(d) Berechnung der Gradientenrichtung $g_r(x,y)$ durch Skalierung der Richtungskomponenten $s_x(x,y)$ und $s_y(x,y)$ auf die Größe des Richtungsquadrats gemäß Bild 7.10. Dies kann mit dem Bresenham-Algorithmus[1] durchgeführt werden. Die skalierten Werte d_x und d_y zeigen direkt auf einen Wert des Richtungsquadrats.

(e) Berechnung der Systematik (hier gemäß dem zweiten dargestellten Beispiel). Hier erfolgt eine Einteilung des Richtungskanals in drei Teile. Die Auswertung der Richtungen der drei Teile kann z.B. folgenden Weg gehen, bei dem Fälle für eine positiv bewertete Richtungssystematik aufgeführt werden:

(e1): Die Richtungsabweichung der gemittelten Richtung aller drei Teile von der Hauptrichtung $g_r(x,y)$ ist kleiner als ein Schwellwert. Im mittleren Teil des Richtungskanals treten keine Richtungen mit $d_x = d_y = 0$ auf. In den beiden anderen Teilen tritt mindestens eine mit der Hauptrichtung identische Richtung auf (gerade Kante).

(e2): Die Richtungsabweichung der gemittelten Richtungen von Teil 1 und Teil 2 ist akzeptabel. Teil 2 und Teil 3 erfüllen diese Forderung jedoch nicht. Die Streuung der Richtungen von Teil 3 liegen über der Schranke (gekrümmte Kante).

(e3): Wie die vorhergehende Bedingung, jedoch sinngemäß für Teil 2 und Teil 3 (gekrümmte Kante).

(e4): In Teil 1 liegen nur Punkte mit $d_x = d_y = 0$. Die Streuungen in Teil 2 und Teil 3 sind akzeptabel. Teil 3 enthält keine Punkte mit $d_x = d_y = 0$ (90^o- Ecke).

(e5): Wie die vorhergehende Bedingung, jedoch Teil 1 und Teil 3 vertauscht (90°-Ecke).

(e6): In Teil 1 liegen nur Punkte mit $d_x = d_y = 0$. Teil 2 beginnt mit einem solchen Punkt. Sonst wie die vierte Bedingung (90^o-Ecke).

[1] Bresenham-Algorithmus: Eine Fehlergröße e misst den Abstand zwischen dem tatsächlichen Geradenverlauf und dem gerade erzeugten Punkt im Koordinatenraster. Ist e positiv, so ist der Geradenverlauf über dem akuellen Punkt. Es wird die y-Koordinate erhöht und 1 von e abgezogen. Ist e negativ, so bleibt die y-Koordinate unverändert. Der Vorteil von diesem Algorithmus ist, dass er auch rein ganzzahlig implementiert werden kann.

(e7): In Teil 3 liegen nur Punkte mit $d_x = d_y = 0$. Teil 2 endet mit einem solchen Punkt. Sonst wie die fünfte Bedingung (90^o-Ecke).

Mit diesen Bedingungen wird für den Bildpunkt in der Position (x, y) ermittelt, ob eine Systematik vorliegt oder nicht.

(f) Berechnung des Kantenbildes durch eine Kombination des Rang- und des Systematikbildes wie oben beschrieben. Die Bewertung erfolgt dabei gemäß (7.10).

Ende des Algorithmus

Abschließend zu diesem Algorithmus noch einige Bemerkungen: Durch die Auswertung des Rang- und des Systematikbildes werden hier positive Eigenschaften der einzelnen Verarbeitungsschritte kombiniert. Sowohl Rang- als auch Systematikbild ermöglichen eine betragsunabhängige Kantendetektion. Die Rangauswertung liefert eine gute Geschlossenheit von Kantenzügen, während die Systematikauswertung bei der Entfernung von irrelevanten Kantenstücken in homogenen oder verrauschten Bildbereichen herangezogen wird. Beide Zwischenschritte (Rang und Systematik) und das Endergebnis sind Binärbilder (logische Bilder), in denen die Grauwerte der Bildpunkte die Bedeutung „liegt auf einer Kante“ und „liegt auf keiner Kante“ haben.

An vielen Stellen dieses Kantendetektors werden Schwellwerte verwendet. Das hat zur Folge, dass das Verfahren sehr sorgfältig auf die jeweilige Problemstellung zu adaptieren ist. Das Ergebnis wird dabei sicher verbessert, wenn in der Vorverarbeitung Störungen (Rauschen, Scannerfehler, usw.) korrigiert werden.

Da einzelne Verarbeitungsschritte durchaus rechenzeitintensiv sind, vor allem, wenn größere Umgebungen verwendet werden, wird man bei Echtzeitanwendungen möglicherweise Probleme bekommen. Hier kann die Verwendung von sehr schnellen allgemeinen Prozessoren, die Parallelisierung von Teilabläufen oder die Verwendung von Spezialhardware Abhilfe schaffen. Bei vielen praktischen Anwendungen ist es außerdem nicht notwendig, den gesamten Bildbereich zu verarbeiten, sondern die Verarbeitung kann auf einen kleinen Bildausschnitt (*area-of-interest*) beschränkt werden.

7.6 Der Canny-Kantendetektor

Einen Kantendetektor mit einigen interessanten Eigenschaften stellt der Canny-Kanten- detektor dar. Er wurde von Canny 1983 [Cann83, Cann86] vorgestellt. Damals waren schon eine Vielzahl von Operatoren bekannt, Canny's Beitrag war, mit einem mathematisch-signaltheoretischen Ansatz aus (mathematisch formulierten) Designkriterien einen bezüglich dieser Kriterien optimalen Kantendetektor abzuleiten.

Informell ausgedrückt sind dies folgende Kriterien:

- Geringe Fehler erster und zweiter Art, d.h. es sollen keine Kanten übersehen werden und keine Kanten an Stellen detektiert werden, an denen sich keine befinden,

- Genaue Lokalisierung von Kanten, d.h. ein Detektor sollte Kanten möglichst genau dort lokalisieren, wo sie auch im Originalbild zu finden sind,
- Nur eine Antwort des Detektors auf eine Kante, nicht mehrere Antworten an mehreren Stellen (im wesentlichen als Ergänzung zum zweiten Kriterium).

Canny betrachtet dabei das eindimensionale Signal senkrecht zur jeweils betrachteten Kante im Bild. Als bezüglich der obigen Kriterien optimalen Kantendetektor ermittelt er das Maximum der Ableitung des rauschgefilterten Signals.

Die Realisierung des Canny-Kantendetektors erfolgt - wie schon in Canny's Originalarbeit skizziert - meist in einem dreischrittigen Verfahren:

- Glättung des Bildes mit einem Gauss'schen Kern
- Bestimmung des Gradienten (Kantenfilterung)
- Kantentracking (Nicht-Maxima-Unterdrückung und Tracking mit Hysterese)

A7.3: Canny-Kantendetektor.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y))$ ein Grauwertbild;

◇ Eingabeparameter des Canny-Kantendetektors: σ: Standardabweichung für die Glättung mit einem Gauss'schen Kern; h, l: oberer und unterer Schwellwert für das Kantentracking mit Hysterese;

Algorithmus:

(a) Glätte $\mathbf{S}_e$ mit einem Gauss'schen Kern mit Standardabweichung σ;

(b) Bestimmung des Gradienten (Kantenfilterung):

(ba) Wende auf $\mathbf{S}_e$ Kantenfilteroperatoren in zwei orthogonalen Richtungen an;

(bb) Bilde aus dem Ergebnis Gradientenbetrag und -richtung;

(c) Kantentracking:

(ca) Unterdrücke im Gradientenbetragsbild diejenigen Pixel, die entlang der Gradientenrichtung kein Maximum darstellen;

(cb) Verfolge Kanten mit Hysterese: Ausgehend von Pixeln, die über der oberen Schwelle h liegen, markiere alle Pixel, die über der unteren Schwelle l liegen als zur Kante gehörend.

Ende des Algorithmus

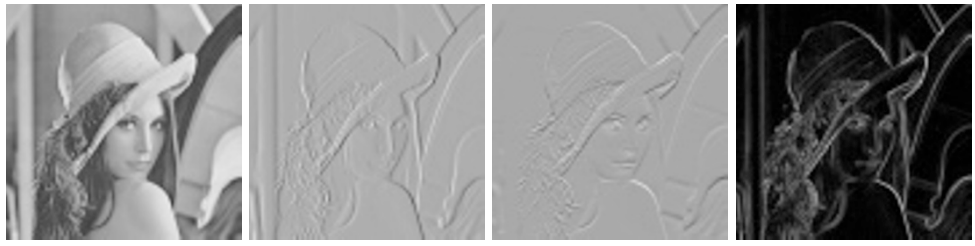
Bei der Glättung des Bildes wird ein Gauss'scher Kern mit - je nach Verrauschtheit des Bildmaterials - vorwählbarer Standardabweichung σ verwendet. Diese ist einer der Parameter des Canny-Kantendetektors. Ziel dieses Schrittes ist die Verminderung des Bildrauschens, da die hochfrequenten Rauschanteile sich nachteilig auf den nachfolgenden Schritt der Gradientenbildung auswirken.

Zur Gradientenbildung wird meist einer der bekannten Kantenfilteroperatoren in zwei orthogonalen Richtungen angewendet (Robert's Cross, Prewitt, Sobel, Bild 7.12-b,c), um daraus Betrag und Richtung des Gradienten zu bestimmen.

Im Gradienten-Betragsbild (Bild 7.12-d) werden dann diejenigen Pixel "unterdrückt", d.h. zu Null gesetzt (*non-maxima-suppression*), die entlang der Gradientenrichtung (also senkrecht zur Kante) kein Maximum aufweisen. Canny schlägt vor, für die Bestimmung der Nachbarpixel entlang des Gradienten zwischen den entsprechenden Pixeln der 8-er Nachbarschaft zu interpolieren. Allerdings wird hierzu auch häufig die Gradientenrichtung meist in die vier (betragsfreien) Richtungen der Achter-Nachbarschaft (*horizontal, vertikal, entlang der Hauptdiagonalen, entlang der Nebendiagonalen*) diskretisiert und die Maximumsunterdrückung entlang der drei benachbarten Pixel in dieser Richtung durchgeführt.

Auf die übrigbleibenden Kanten im Bild wird dann ein Linienverfolgungsverfahren mit Hysterese angewendet: Um schwache, oft auch aus Restrauschen entstehende, Kanten zu unterdrücken, werden nur diejenigen Kanten verwendet, bei denen mindestens ein Pixelwert über einer oberen Schranke h liegt. Um für diese Kanten Unterbrechungen durch schwächere Kantenbereiche zu vermeiden, wird eine Hysterese eingeführt, d.h. für eine einmal erkannte Kante reicht es aus, wenn alle Pixel über einer unteren Schranke l liegen.

Die obere und untere Schwellwerte (h, l) bilden neben der oben erwähnten Standardabweichung die anderen beiden der drei Parameter des Canny-Kantendetektors.



(a) (b) (c) (d)

Bild 7.12: (a) Die Original-Lena, (b) Gradient dx, (c) Gradient dy, (d) Betrag des Gradienten

Sollte die Topologie der detektierten Kanten in der jeweiligen Anwendung eine besondere Rolle spielen, so ist auf ein spezielles 'Feature' des Canny-Kantendetektors zu

(a) (b) (c) (d)

Bild 7.13: Auswirkungen unterschiedlicher Standardabweichungen zum Glätten des Bildes: (a) $\sigma = 1.0$, (b) $\sigma = 2.0$, (c) $\sigma = 3.0$, (d) $\sigma = 4.0$,

(a) (b) (c) (d)

Bild 7.14: Auswirkung unterschiedlicher Hystereseschwellen für das Kantentracking, leichte Glättung ($\sigma = 4.0$), starkes verbleibendes Rauschen: (a) $h = 0.8, l = 0.3$,(b) $h = 1.0, l = 0.6$,(c) $h = 1.0, l = 0.8$,(d) Detail mit offenen Y-Verbindungen

(a) (b) (c) (d)

Bild 7.15: Auswirkung unterschiedlicher Hystereseschwellen für das Kantentracking, starke Glättung ($\sigma = 4.0$), leichtes verbleibendes Rauschen: (a) $h = 1.0, l = 0.1$,(b) $h = 0.3, l = 0.3$,(c) $h = 1.0, l = 0.3$,(d) $h = 1.0, l = 0.6$

achten: Bedingt durch die Nicht-Maxima-Unterdrückung beim Kantentracking werden 'Y'-Verbindungen dreier (oder Kreuzungen mehrerer) Kanten nicht vollständig detektiert (siehe Detail aus der rechten, oberen Ecke des Originalbilds im Teilbild 7.14-d). Hier ist gegebenenfalls ein Nachbearbeitungsschritt zum Schließen der Y-Verbindungen oder eine alternative Trackingstrategie erforderlich.

7.7 Kanten und Linien mit morphologischen Operationen

Die mathematische Morphologie stellt der digitalen Bildverarbeitung und Mustererkennung mächtige Instrumente zur Bewältigung vieler Problemstellungen zur Verfügung. Die Grundlagen dazu sind in Kapitel 6 zusammengestellt. In diesem Abschnitt werden morphologische Verfahren erläutert, die im Zusammenhang mit der Verarbeitung von Kanten und Linien interessant sind.

Zuvor aber noch eine Zusammenfassung, wie Kanten- oder Linienbilder entstehen können: Grauwertkanten treten in Grauwertbildern an Helligkeitsübergängen auf, die eine gewisse Systematik aufweisen. Grauwertkanten können aber auch, wie schon oben dargestellt, das Ergebnis einer Textur- oder Farbmerkmalsverarbeitung sein. Wird auf ein derartiges Grauwertbild ein Verfahren zur Kantenextraktion angewendet (Abschnitte 5.4, 7.3 und 7.4), so sind im Ergebnisbild die markanten Grauwertübergänge in der Regel durch Linien dargestellt. Das so erzeugte Linienbild kann ein Grauwertbild oder ein Binärbild sein. Ist es ein Grauwertbild, so sagt ein hoher Wert aus, dass die Wahrscheinlichkeit, dass der Bildpunkt auf einer Grauwertkante liegt, hoch ist. Bei einem binären Linienbild bedeutet der Grauwert 0, dass der Bildpunkt zum Hintergrund, und der Grauwert 1, dass er zu einer Kante gehört.

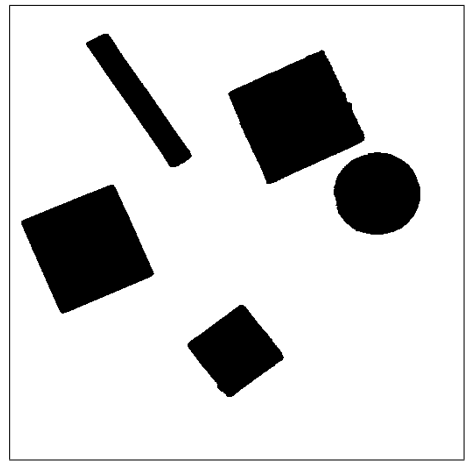
Linienbilder können natürlich auch das direkte Ergebnis einer Digitalisierung sein, so z.B. wenn eine Strichgrafik entweder per Software oder mit einem Digitalisierungsgerät (Videosensor, Scanner) in ein Rasterbild umgewandelt wird.

7.7.1 Extraktion des Randes von Segmenten

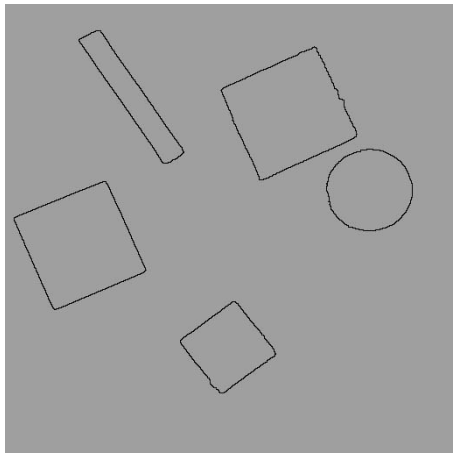
Ein einfaches Verfahren, wie man aus einem Binärbild, das unterschiedliche Segmente enthält, den Rand dieser Segmente gewinnen kann, ist eine Dilatation oder Erosion mit einer anschließenden Differenz zum Originalbild. Vorausgesetzt wird ein Binärbild (Zweipegelbild), in dem sich die Segmente dunkel (Grauwert 0) vor einem hellen Hintergrund (Grauwerte 1 oder 255) abheben.

Bei einer Dilatation und einer anschließenden Differenzbildung des dilatierten Bildes mit dem Original erhält man den inneren Rand der Segmente, während bei der Erosion und einer anschließenden Differenzbildung Ränder erzeugt werden, die die Segmente ganz enthalten (Bilder 7.16).

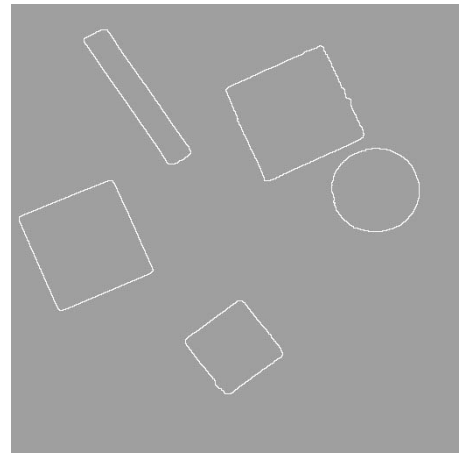
Dieses Verfahren ist auch bei Grauwertbildern möglich. Wird eine Dilatation durchgeführt, so vergrößern sich die hellen Bildbereiche auf Kosten der dunklen. Nach einer



(a)



(b)



(c)

Bild 7.16: (a) Originalbild: Dunkle Segmente auf hellem Hintergrund. (b) Dilatation und Differenz (Grauwert 127 addiert). (c) Erosion und Differenz (Grauwert 127 addiert).

Differenz mit dem Originalbild erhält man die Grauwertübergänge als Linien. Ähnlich wie bei den Differenzenoperatoren aus Abschnitt 5.4 ergibt sich auch hier das Problem, dass die Beträge dieses *morphologischen Gradienten* unterschiedliche Werte erhalten und so eine einfache Binarisierung oft nicht immer das gewünschte Ergebnis erzielt. Hier kann auch eine Betrachtung mit Positionsrang und Systematik (Abschnitt 7.5) weiter helfen. Die Bildfolge 7.17 zeigt ein Beispiel zum morphologischen Gradienten.

7.7.2 Verarbeitung von Linien

Bei der Binarisierung von Linienbildern lassen sich ebenfalls morphologische Operationen einsetzen. Handelt es sich z.B. um eine digitalisierte Strichgrafik, die mit einem Scanner oder einer Videokamera aufgezeichnet wurde, so kann es vorkommen, dass der Hintergrund durch ungleichmäßige Ausleuchtung oder durch Randabschattungen (Vignettierungen) gestört ist. Der Schwellwert bei der Binarisierung wird dann ortsabhängig. Dieses Problem kann manchmal mit einem dynamischen Schwellwert (Abschnitt 4.5) gelöst werden.

Wenn es möglich ist, durch eine *closing*-Operation in einem Eingabebild mit hellem Hintergrund die Linien ganz verschwinden zu lassen, so bleibt nur mehr der Hintergrund übrig. Durch eine Differenzbildung mit dem Original und eventuell der Addition einer Konstanten erhält man ein Ausgabebild, in dem der störende Hintergrund ausgeblendet ist. Dieses Bild kann dann gut binarisiert werden (Bildfolge 7.18).

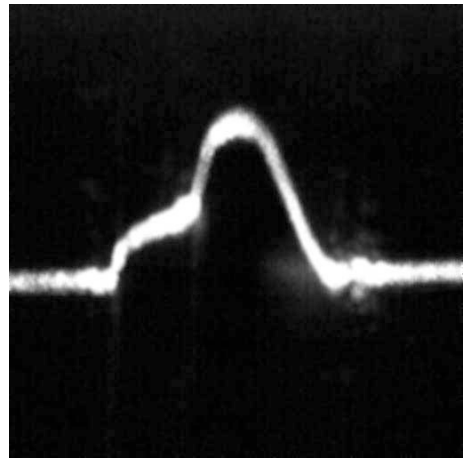
Auch nach einer Binarisierung können Linienbilder mit morphologischen Operationen korrigiert, verbessert und weiter verarbeitet werden. Eine einfache Korrektur besteht z.B. in der Elimination einzelner, verstreuter Bildpunkte (Abschnitt 5.5).

Häufig tritt auch der Fall ein, dass Linien unterbrochen sind, die eigentlich geschlossen sein sollten. Bei einer Vorzugsrichtung der Linien ist das Schließen von Lücken mit einem strukturierenden Element, das die Vorzugsrichtung berücksichtigt, einfach möglich. Bei einer Erosion des Vordergrundes schließen sich die Lücken, da die Erosion hier nur in der gewählten Richtung wirkt. Ein Beispiel dazu zeigt die Bildfolge 7.19.

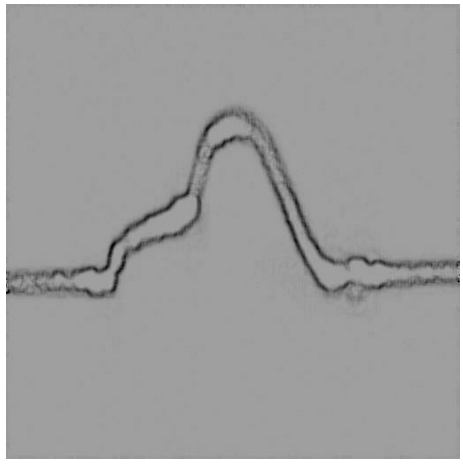
7.8 Skelettierung mit morphologischen Operationen

Nachdem ein Linienbild mit den in Abschnitt 7.7 beschriebenen Verfahren binarisiert wurde, ist eine oft benötigte Weiterverarbeitung das Verdünnen der Linien, so dass nur mehr eine ein Pixel breite Linie verbleibt, die in der Mitte der ursprünglichen, breiteren Linie verläuft. Diese Vorgehensweise nennt man *Skelettierung*. Ein Beispiel einer Anwendung hierzu wurde im vorhergehenden Abschnitt mit der Bildfolge 7.19 gegeben.

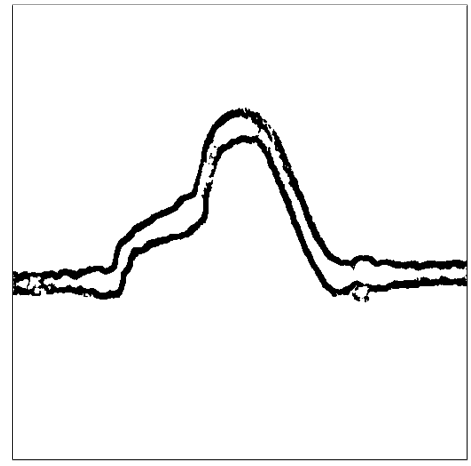
Die Skelettlinie oder das Skelett einer binär dargestellten Fläche (eines Segments) ist durch eine Reihe von Forderungen festgelegt, die jedoch nicht als exakte, mathematische Definitionen aufgefasst werden können. Das ist auch der Grund, warum sich in der einschlägigen Literatur eine große Anzahl von Arbeiten mit dieser Thematik befassen und



(a)



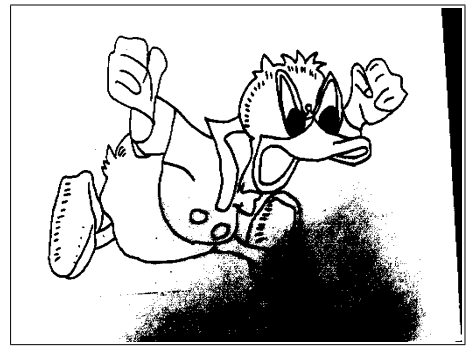
(b)



(c)

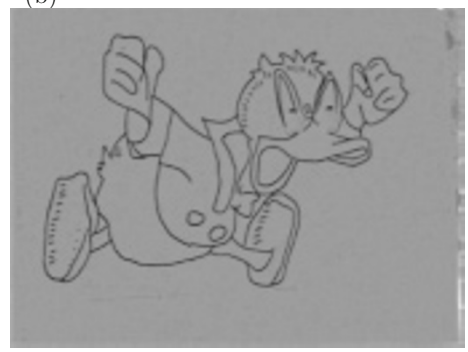
Bild 7.17: Beispiel zum morphologischen Gradienten im Grauwertbild. (a) Laserlichtband, das mit einer Spaltlinse auf eine Kleberaupe projiziert wurde. (b) Ergebnis nach zwei Dilatationen und einer Subtraktion vom Original (skaliert). (c) Binarisiertes Ergebnis.

(a)

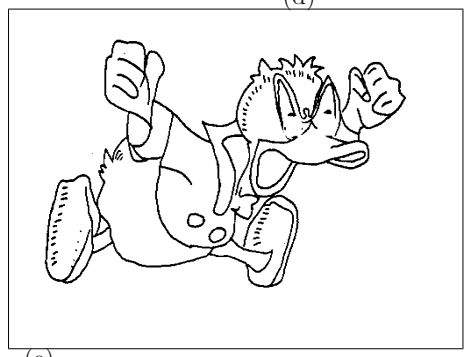


(b)

(c)



(d)



(e)

Bild 7.18: Beispiel zur Korrektur eines ungleichmäßig ausgeleuchteten Bildhintergrundes. (a) Originalbild. (b) Binärbild mit fixem Schwellwert: Der rechte untere Rand läuft zu. (c) Bildhintergrund nach einer *closing*-Operation (fünf Iterationen). (d) Differenz mit dem Original. (e) Binarisierung. Im Bereich der Augen ist der Bildvordergrund durch die *closing*-Operation nicht vollständig eliminiert worden. Damit sind die Störungen zu erklären. Das kann aber auch nutzbringend eingesetzt werden. In diesem Beispiel konnte dadurch der Keil am rechten Bildrand entfernt werden.

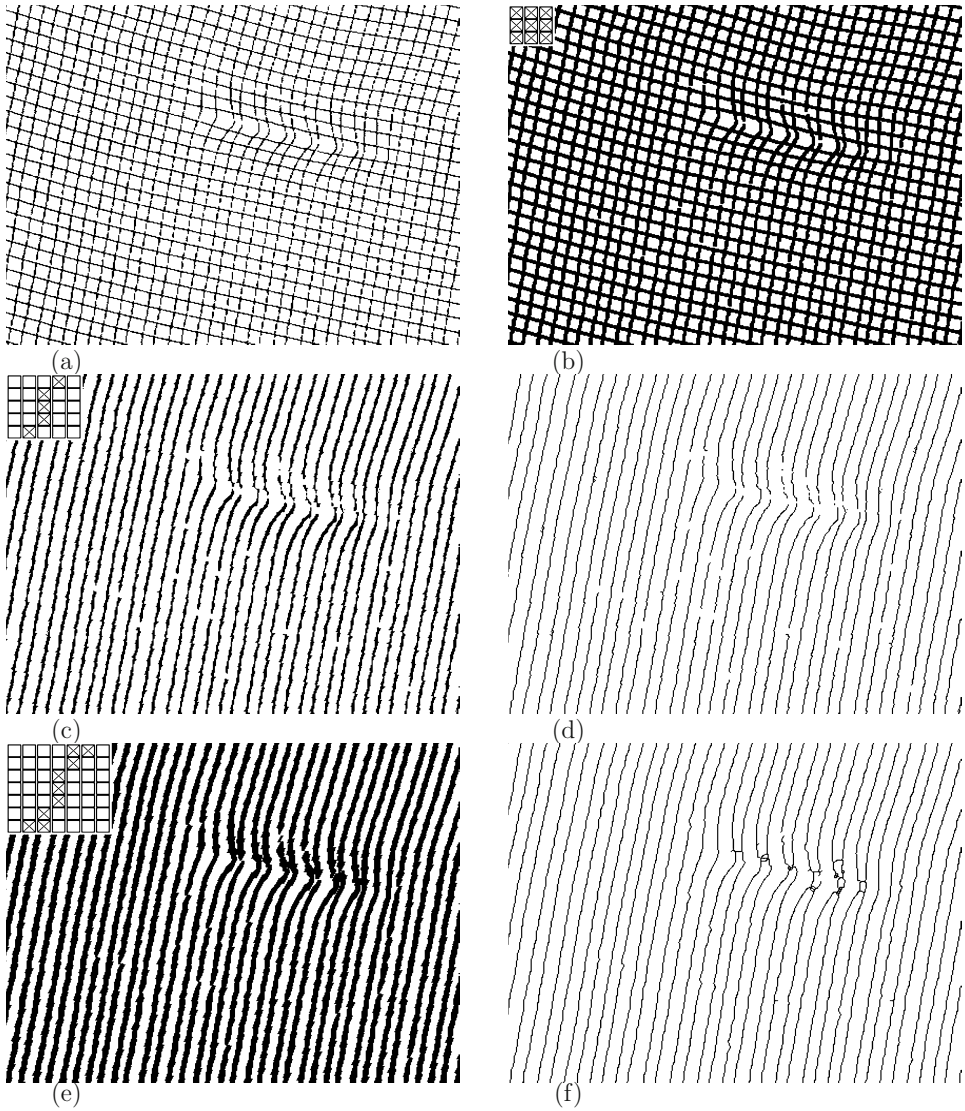


Bild 7.19: Beispiel zum Schließen von Lücken in einem binarisierten Linienbild. (a) Originalbild. Die Aufgabe besteht darin, die vertikalen Linien möglichst geschlossen zu extrahieren. (b) Um die Linien etwas zu verdicken, wurde eine Erosion mit dem links oben eingeblendeten strukturierenden Element durchgeführt. (c) Zur Elimination der horizontalen Linien wurde eine Dilatation mit dem links oben eingeblendeten strukturierenden Element durchgeführt. (d) Skelettierung (siehe nächster Abschnitt) zur Verdünnung der Linien. Einige Linien sind unterbrochen. (e) Erosion von Teilbild (c) mit dem links oben eingeblendeten strukturierenden Element. (f) Skelettierung zur Verdünnung der Linien. Die Linien sind, bis auf die Fehlerstelle, geschlossen.

dabei zu recht unterschiedlichen Ergebnissen kommen. Forderungen zur Skelettlinie lauten:

- Die Skelettlinie soll nur einen Bildpunkt breit sein.
- Die Skelettlinie soll ungefähr in der Mitte des Segments liegen.
- Das Skelett soll die ursprüngliche Form des Segments widerspiegeln. So darf das Skelett einer einfach zusammenhängenden Fläche nicht in mehrere Teile zerfallen.
- Das Skelettierungsverfahren soll unempfindlich gegen kleinere Störungen am Rand des Segments sein. Empfindliche Verfahren erzeugen bei ausgefransten Rändern viele Verästelungen.
- Der Skelettierungsalgorithmus muss nach einer bestimmten Anzahl von Iterationen stabil werden, d.h. es dürfen sich keine Änderungen mehr ergeben.

Es gibt unterschiedliche Verfahren zur Skelettierung. Eine Klasse von Verfahren geht vom Binärbild aus und entscheidet für alle Vordergrundpixel, ob sie entfernt werden können oder nicht. Die Entscheidung, ob ein gesetzter Vordergrundbildpunkt entfernt, also dem Hintergrund zugewiesen werden kann (das *Tilgungskriterium*), wird anhand von Umgebungen (meistens $3 \cdot 3$) entschieden.

Eine andere Klasse von Skelettierungsalgorithmen baut auf Konturverfolgungsverfahren auf. Hier liegt der Gedanke zugrunde, dass man nur diejenigen Bildpunkte überprüfen will, die auch tatsächlich zum Rand des Segments gehören. Man wird hier also die Randpunkte des Segments in einer Listenstruktur führen. Wenn ein Randpunkt aus dieser Liste entfernt wird, wird für diesen ein neuer, bisher nicht auf dem Rand liegender Segmentpunkt eingefügt.

Im Folgenden werden zwei Skelettierungsverfahren beschrieben, die beide zur ersten der oben dargestellten Klassen gehören.

In Kapitel 6 wurden neben den Grundlagen zur mathematischen Morphologie auch schnelle Implementierungsmöglichkeiten durch Verschiebeoperationen und Boole'sche Operationen erläutert. Diese Technik wird hier verwendet. Es wird im Folgenden angenommen, dass ein Binärbild mit den Grauwerten 0 (Hintergrundbildpunkt) und 1 (Vordergrundbildpunkt) vorliegt. Zunächst werden acht $3 \cdot 3$–Masken vorgegeben:

$$\begin{pmatrix} 0 & b & b \\ 0 & 1 & 1 \\ 0 & b & 1 \end{pmatrix} \begin{pmatrix} b & 1 & b \\ 0 & 1 & 1 \\ 0 & 0 & b \end{pmatrix} \begin{pmatrix} b & 1 & 1 \\ b & 1 & b \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} b & 1 & b \\ 1 & 1 & 0 \\ b & 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 1 & b & 0 \\ 1 & 1 & 0 \\ b & b & 0 \end{pmatrix} \begin{pmatrix} b & 0 & 0 \\ 1 & 1 & 0 \\ b & 1 & b \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ b & 1 & b \\ 1 & 1 & b \end{pmatrix} \begin{pmatrix} 0 & 0 & b \\ 0 & 1 & 1 \\ b & 1 & b \end{pmatrix} \tag{7.11}$$

Diese Masken beschreiben Konstellationen von $3 \cdot 3$ Bildpunkten, bei denen der mittlere Punkt von der Segmentfläche entfernt werden darf. Die Masken werden von links oben nach rechts unten mit $\mathbf{M_0}$ bis $\mathbf{M_7}$ bezeichnet. Dabei bedeuten die mit b (für „beliebig“) besetzten Positionen, dass an diesen Stellen entweder eine 0 oder eine 1 stehen kann.

Nach Maßgabe dieser Masken wird nun das Originalbild $\mathbf{S}_e$ in ein Hilfsbild $\mathbf{H}$ verschoben. Die Verschiebungsrichtungen werden dabei von 0 (östlich), bis 7 (südöstlich) numeriert. Abhängig davon, ob in der Maske in der Verschiebungsposition eine 0 oder 1 steht, wird das Hilfsbild negiert oder nicht. Positionen, die mit b besetzt sind, werden nicht berücksichtigt. Die Hilfsbilder für jede Verschiebungsrichtung werden alle mit UND verknüpft. Das so entstehende Hilfsbild $\mathbf{T}$ enthält alle Bildpunkte, die im Eingabebild $\mathbf{S}_e$ die Konstellation einer bestimmten Maske erfüllen. Dieses Bild $\mathbf{T}$ wird mit EXODER (exklusives ODER) mit dem Original verknüpft, wodurch die entsprechenden Pixel gelöscht werden. Für die Maske $\mathbf{M_0}$ ist diese Vorgehensweise im folgenden dargestellt:

$$
\begin{array}{l}
\text{Maske } M_0: \\
\quad \text{SHIFT}(\mathbf{S}_e, \mathbf{H}, 0) \\
\quad \text{NOT}(\mathbf{H}, \mathbf{H}) \\
\quad \text{AND}(\mathbf{S}_e, \mathbf{H}, \mathbf{T}) \\
\quad \text{SHIFT}(\mathbf{S}_e, \mathbf{H}, 1) \\
\quad \text{NOT}(\mathbf{H}, \mathbf{H}) \\
\quad \text{AND}(\mathbf{T}, \mathbf{H}, \mathbf{T}) \\
\quad \text{SHIFT}(\mathbf{S}_e, \mathbf{H}, 3) \\
\quad \text{AND}(\mathbf{T}, \mathbf{H}, \mathbf{T}) \\
\quad \text{SHIFT}(\mathbf{S}_e, \mathbf{H}, 4) \\
\quad \text{AND}(\mathbf{T}, \mathbf{H}, \mathbf{T}) \\
\quad \text{SHIFT}(\mathbf{S}_e, \mathbf{H}, 7) \\
\quad \text{NOT}(\mathbf{H}, \mathbf{H}) \\
\quad \text{AND}(\mathbf{T}, \mathbf{H}, \mathbf{T}) \\
\quad \text{EXODER}(\mathbf{S}_e, \mathbf{T}, \mathbf{S}_e)
\end{array}
\tag{7.12}
$$

Der dritte Parameter bei den SHIFT-Operationen gibt die Verschiebungsrichtung, wie oben festgelegt, an. Die Operationen von (7.12) werden nacheinander sinngemäß für alle Masken $\mathbf{M_0}$ bis $\mathbf{M_7}$ in einem Iterationsschritt durchgeführt. Es werden so viele Iterationen angeschlossen, bis in einem Schritt keine Bildpunkte mehr entfernt werden. Das Ergebnisbild enthält dann das Skelett. Der gesamte Algorithmus lautet somit wie folgt:

A7.4: Skelettierung mit morphologischen Operationen.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y))$ ein einkanaliges Binärbild; $G = \{0, 1\}$.

◇ **H** und **T** zwei Hilfsbilder.

Algorithmus:

(a) Setze $removed = 1$;

(b) Solange gilt $removed = 1$:

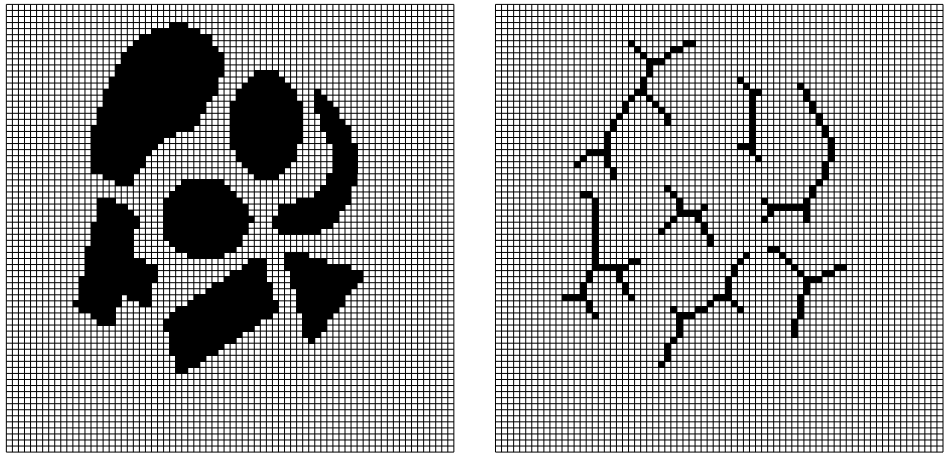
(ba) Setze $removed = 0$.

(bb) Führe die Operationen (7.12) sinngemäß für alle Masken $\mathbf{M_0}$ bis $\mathbf{M_7}$ durch.

(bc) Falls in einem Iterationsschritt Bildpunkte entfernt wurden: Setze $removed = 1$.

Ende des Algorithmus

Ein Beispiel zu diesen Verfahren zeigt Bild 7.20.



(a) (b)

Bild 7.20: Beispiel zur Skelettierung mit morphologischen Operationen. (a) Originalbild. (b) Skelett.

Bei dieser Skelettierungsmethode sind viele Verschiebeoperationen und logische Operationen notwendig. Das setzt voraus, dass diese grundlegenden Verknüpfungen von (Binär-)Bildern sehr schnell durchgeführt werden. Nur dann wird der dargestellte Algorithmus in akzeptabler Zeit ablaufen. Da es sich um einfache Einzeloperationen handelt, kann dieses Verfahren gut mit spezieller Hardware realisiert werden.

Ein anderes Skelettierungsverfahren, das bezüglich der Rechenzeit wesentlich schneller abläuft, ist im nächsten Abschnitt beschrieben. Allerdings liefert dieses Verfahren eine etwas „schlechtere“ Skelettlinie.

7.9 Skelettierung mit der Euler'schen Charakteristik

Dem zweiten Verfahren zur Skelettierung liegt die *Euler'sche Charakteristik* zugrunde. Die Euler'sche Charakteristik für Polygonnetze und für Polyeder ist die Zahl $E = e - k + f$, wobei e für die Zahl der Ecken, k für die Zahl der Kanten und f für die Zahl der Flächen (Polygonbereiche) steht. Der Euler'sche Polygonsatz sagt nun, dass für ein Polygonnetz $E = 1$ gilt. Im dreidimensionalen Raum gilt für (schlichtartige) Polyeder $E = 2$. Dazu ein einfaches Beispiel: Ein Würfel besitzt 8 Ecken, 12 Kanten und 6 Flächen, also ist $E = 8 - 12 + 6 = 2$.

Diese Betrachtungsweise lässt sich auch auf Binärbilder anwenden. Die Zahl E gibt hier die Anzahl der Segmente an. Die Bedeutung der Größen e, k und f ist etwas anders als oben. Hier ist e die Anzahl der Ecken der (z.B. schwarzen) Vordergrundbildpunkte, k die Anzahl der Kanten der Vordergrundbildpunkte und f die Anzahl der Vordergrundbildpunkte. Einen Vordergrundbildpunkt stellt man sich hier am besten als ein Quadrat mit vier Ecken, vier Kanten und einer Flächeneinheit vor. Gemeinsame Eckpunkte oder Kanten von Bildpunkten werden nur einmal gezählt. Bild 7.21 zeigt dazu einige Beispiele: Für das Binärbild 7.21-a ergibt sich $e = 11$, $k = 15$ und $f = 5$, also $E = 1$. Teilbild 7.21-b ergibt ebenfalls $E = 1$ ($e = 11$, $k = 14$ und $f = 4$). Das $5 \cdot 5$–Bild 7.21-c besitzt die Werte $e = 29$, $k = 38$ und $f = 12$, somit $E = 3$. Man sieht, dass E hier die Anzahl der einfach zusammenhängenden Segmente angibt. Diese Eigenschaft der Euler'schen Charakteristik, auf die in Abschnitt 24.10 nochmals eingegangen wird, kann zum Zählen der Segmente in einem Binärbild verwendet werden.

Wie kann die Euler'sche Charakteristik zur Skelettierung verwendet werden? Zur Berechnung der Entscheidung, ob ein Vordergrundbildpunkt entfernt werden soll, werden 3·3–Umgebungen betrachtet. Ein Bildpunkt darf entfernt werden, wenn sich dadurch die Anzahl der zusammenhängenden Flächen nicht ändert. So kann bei Bild 7.21-a der Bildpunkt in der Mitte dem Hintergrund zugewiesen werden. Wenn sich die Anzahl der zusammenhängenden Flächen nicht ändert, ist die Euler'sche Charakteristik vor und nach dem Entfernen gleich (in diesem Fall $E = 1$). Bei Bild 7.21-b darf der mittlere Punkt nicht entfernt werden, sonst würden zwei nicht zusammenhängende Flächen entstehen. Hier würde sich die Euler'sche Charakteristik von $E = 1$ auf $E = 2$ ändern.

Für den Skelettierungsalgorithmus heißt das, dass alle Punkte des Binärbildes im Rahmen ihrer $3 \cdot 3$-Umgebung untersucht werden. Ein Bildvordergrundpixel darf nur dann

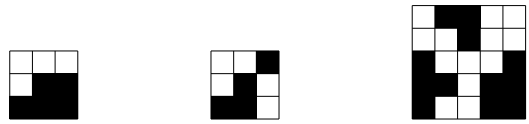


Bild 7.21: Beispiele zur Euler'schen Charakteristik in Binärbildern. (a) e=11, k=15 und f=5, also E=1. (b) e=11, k=14 und f=4, also E=1. (c) e=29, k=38 und f=12, somit E=3. Man sieht, dass E hier die Anzahl der einfach zusammenhängenden Segmente angibt.

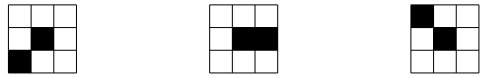


Bild 7.22: Vordergrundbildpunkte, die an Linienenden liegen, müssen gesondert behandelt werden.

entfernt werden, wenn die Euler'sche Charakteristik vor und nach dem Entfernen gleich bleibt.

Probleme ergeben sich noch bei Linienenden. Linienendpunkte sind Pixel, die nur einen Nachbarn besitzen, etwa wie Bild 7.22 zeigt.

Diese Bildpunkte werden gesondert behandelt. Es kann etwa gefordert werden, dass Vordergrundpunkte, die nur einen Nachbarn besitzen, nicht entfernt werden, obwohl sich die Euler'sche Charakteristik nicht ändert.

Als nächstes ist noch zu beachten, in welcher Reihenfolge das Bild durchlaufen wird. Zeilenweise von links oben nach rechts unten ist unpassend, da dann z.B. bei einem horizontal liegenden Balken die Skelettlinie am unteren Ende verlaufen würde, was einer der Forderungen widerspricht. Um dies zu umgehen, werden in einem Iterationsschritt mehrere Subiterationen durchgeführt. Bei jeder Subiteration wird nur eine Teilmenge der Bildpunkte des gesamten Bildes untersucht. Vorgeschlagen werden mindestens zwei Subiterationen. In der ersten werden nur diejenigen Pixel verarbeitet, bei denen beide (x, y)-Ortskoordinaten gerade oder beide ungerade sind. In der zweiten Subiteration werden die restlichen Bildpunkte verarbeitet.

Schließlich ist noch zu untersuchen, wann das Verfahren abbricht. In der Regel wird das Abbruchkriterium so formuliert, dass der Algorithmus beendet wird, wenn in einer Iteration keine Vordergrundpixel mehr entfernt wurden. Es müssen also in jeder Iteration alle Pixel untersucht werden, auch wenn vielleicht nur mehr ein oder zwei Pixel entfernt werden. Um hier Rechenzeit zu sparen kann das Bild in rechteckige Teilbilder unterteilt werden. Ein Teilbild wird in einem Iterationsschritt nur dann verarbeitet, wenn noch Bildpunkte entfernt werden können. Teilbilder, in denen das Skelett schon berechnet ist, werden nicht

mehr berücksichtigt.

Nun noch einige Bemerkungen zur Implementierung: Die Entscheidung, ob ein Vordergrundbildpunkt entfernt werden darf oder nicht, erfordert ein zweimaliges Berechnen der Euler'schen Charakteristik. Um diesen Rechenaufwand zu vermeiden, kann man eine Tabelle aufstellen, in der vermerkt ist, für welche Konstellationen von $3 \cdot 3$–Umgebungen der zentrale Punkt entfernt werden kann. Bei $3 \cdot 3$–Umgebungen gibt es 512 Konstellationsmöglichkeiten. Zu jeder Konstellation wird eine Adresse berechnet, die zwischen 0 und 511 liegt und somit einem Eintrag in der Tabelle zugeordnet ist.

Der gesamte Algorithmus läuft wie folgt ab:

A7.5: Skelettierung mit der Euler'schen Charakteristik.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y))$ ein einkanaliges Grauwertbild mit $G = \{0, 1, ..., 255\}$ als Grauwertmenge (Eingabebild).

◇ *threshold*: Schwellwert zur Binarisierung.

◇ *ETab* sei die Tabelle mit den Einträgen, ob ein Vordergrundpixel entfernt werden darf oder nicht.

◇ Es wird mit zwei Bildpuffern gearbeitet. Am Anfang wird der erste Puffer mit dem Eingabebild vorbesetzt. Das Ergebnis einer Iteration wird im zweiten Puffer erzeugt. Nach einer Iteration wird der Ausgabepuffer zum Eingabepuffer.

Algorithmus:

(a) Setze $removed = 1$;

(b) Solange gilt $removed = 1$:

(ba) Setze $removed = 0$.

(bb) Erste Subiteration: Betrachte alle Pixel in den Positionen (x, y), bei denen x und y beide gerade oder ungerade sind:

(bba) Betrachte für alle Vordergrundpixel die $3 \cdot 3$–Umgebung und berechne die Adresse *position* in der Tabelle *Etab*.

(bbb) Falls gilt $Etab[position] = 1$ und falls der Bildpunkt in der Position (x, y) mehr als einen Nachbarn besitzt, darf er im Ausgabebild dem Hintergrund zugewiesen werden. In diesem Fall wird $removed = 1$ gesetzt.

(bc) Zweite Subiteration: Setze das Ausgabebild zum Eingabebild und betrachte Positionen (x, y), für die x gerade und y ungerade oder x ungerade und y gerade ist:

(bca) (wie (bba)).

(bcb) (wie (bbb)).

(bcc) Vertausche die Puffer.

Ende des Algorithmus

Es sei nochmals darauf hingewiesen, dass aufgrund der etwas vagen Festlegung der Skelettlinie die verschiedenen Verfahren alle im Detail unterschiedliche Ergebnisse bringen. Auch das Verhalten an ausgefransten Segmenträndern ist nicht immer vorhersagbar.

Die weitere Verarbeitung von skelettierten Bildern hängt vom jeweiligen Anwendungsfall ab. Es können z.B. in einem nächsten Schritt die Skelettlinien vektorisiert werden. Das bedeutet, dass die Skelettlinien nicht mehr als Pixelfolgen, sondern durch Polygonzüge dargestellt werden. Dabei werden in der Regel die Skelettlinien „gesäubert“, d.h., dass ähnlich verlaufende Linienrichtungen zu einer Richtung zusammengefasst werden.

Ein weiterer Verarbeitungsschritt könnte die Transformation der Vektordarstellung in das Format von gängigen CAD-Systemen sein. Wenn das gelingt, kann eine ursprünglich als Rasterbild vorliegende Information letztlich in einem CAD-System weiter verarbeitet werden. Diese Anwendung kann hilfreich sein, wenn Bilder von (Bau-)Teilen, die mit einem Videosensor erfasst wurden, letztlich in einem CAD-System weiterverarbeitet werden sollen.

7.10 Relaxation

Die mathematisch statistische Methode der Relaxation ist ein interessantes Verfahren, das zur Kanten- und Linienextraktion verwendet wird. Zunächst wird der theoretische Hintergrund der Relaxation dargestellt, dann folgt eine praktische Anwendung aus dem Bereich der Kantenextraktion.

Bei der Relaxation wird folgender Sachverhalt vorausgesetzt:

(7.13)

$S = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_M\}$	sei eine Menge von M Objekten (z.B. eine Menge von Bildpunkten);
$L = \{l_1, l_2, ..., l_N\}$	sei eine Menge von N Eigenschaften, die die Objekte der Menge S besitzen können;
$p_i(l)$	sei die Wahrscheinlichkeit, dass l eine richtige Eigenschaft von $\mathbf{g}_i$ ist;
L_i	sei die Menge aller Eigenschaften, die für das Objekt $\mathbf{g}_i$ zulässig sind.

Es soll gelten:

$$\sum_{l \in L_i} p_i(l) = 1, \text{für } i = 1, 2, ..., M. \tag{7.14}$$

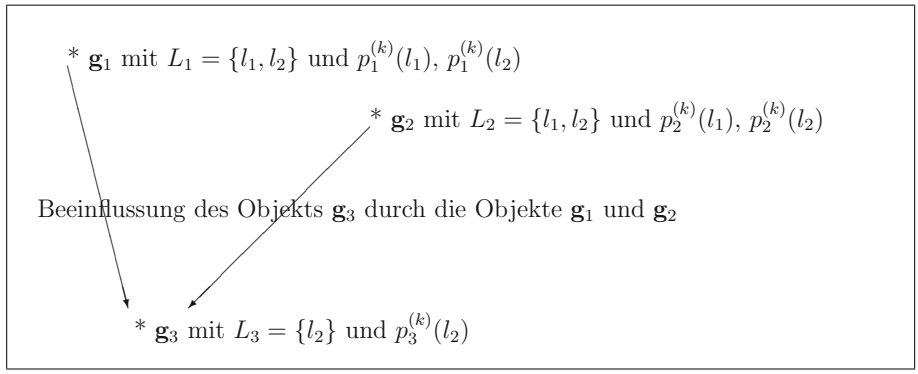


Bild 7.23: Relaxationsmodell mit $S = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ und $L = \{l_1, l_2\}$ im Iterationsschritt k

Das bedeutet, dass die Summe der Wahrscheinlichkeiten der Eigenschaften für das Objekt $\mathbf{g}_i$ eins ist. Die Wahrscheinlichkeiten werden jetzt als Startwerte einer Iteration betrachtet: $p_i^{(1)}(l) = p(l)$. Im k-ten Iterationsschritt ist

$$p_i^{(k)}(l), i = 1, 2, ..., M \tag{7.15}$$

die Wahrscheinlichkeit, dass die Eigenschaft l auf das Objekt $\mathbf{g}_i$ zutrifft. Die Objekte sollen sich nun gegenseitig beeinflussen, so dass beim Übergang von der k-ten zur $(k+1)$-ten Iteration die Wahrscheinlichkeiten

$$p_i^{(k+1)}(l), i = 1, 2, ..., M \tag{7.16}$$

neu berechnet werden. Bild 7.23 zeigt diesen Sachverhalt an einem Beispiel.

Die Wahrscheinlichkeiten $p_i^{(k+1)}(l)$ hängen dabei von den Wahrscheinlichkeiten $p_i^{(k)}(l)$ des Objektes $\mathbf{g}_i$ und von den Wahrscheinlichkeiten $p_n^{(k)}(l')$ aller benachbarten Objekte ab. Dabei muss in jedem Anwendungsfall festgelegt werden, was unter ,,benachbart" zu verstehen ist, ob z.B. als Nachbarn alle oder nur bestimmte Objekte zulässig sind. Die Einflussnahme ,,hängen ab von" wird wie folgt modelliert:

- Falls die Eigenschaften l und l' zusammenpassen (*vereinbar sind*), leistet $p^{(k)}(l')$ einen positiven Beitrag zu $p^{(k+1)}(l)$.
- Falls die Eigenschaften l und l' nicht zusammenpassen (*nicht vereinbar sind*), leistet $p^{(k)}(l')$ einen negativen Beitrag zu $p^{(k+1)}(l)$.
- Falls die Eigenschaften l und l' in keiner gegenseitigen Beziehung stehen (*gegenseitig neutral sind*), leistet $p^{(k)}(l')$ keinen Beitrag zu $p^{(k+1)}(l)$.

Dies kann durch eine Kompatibilitätsfunktion ausgedrückt werden:

$$r_{ij} : L_i \times L_j \to [-1, 1]\,. \tag{7.17}$$

Zur Neuberechnung der $p^{(k+1)}(l)$ wird zunächst ein Korrekturwert ermittelt:

$$q_i^{(k)}(l) = \sum_j d_{ij}\Big(\sum_{l'} r_{ij}(l, l')p_j^{(k)}(l')\Big). \tag{7.18}$$

Die erste Summe wird dabei über alle Objekte erstreckt, die das Objekt $\mathbf{g}_i$ beeinflussen können und die zweite Summe über alle Marken l' in der Menge L_j. Die Faktoren d_{ij} sind zusätzliche Gewichtsfaktoren, die den Abstand des Objektes $\mathbf{g}_j$ von $\mathbf{g}_i$ beschreiben und für die

$$\sum_j d_{ij} = 1 \tag{7.19}$$

vorausgesetzt wird. Damit kann $p^{(k)}(l)$ korrigiert werden:

$$p_i^{(k+1)}(l) = \frac{p_i^{(k)}(l)\left[1 + q_i^{(k)}(l)\right]}{\sum_{l \in L_i}\left(p^{(k)}(l)\left[1 + q_i^{(k)}(l)\right]\right)}. \tag{7.20}$$

Der Nenner in (7.20) gewährleistet, dass die Summe der neu berechneten Wahrscheinlichkeiten für das Objekt $\mathbf{g}_i$ wieder 1 ergibt.

Nach der Darstellung der Theorie jetzt ein praktisches Beispiel, bei dem die Relaxation zur Kantenverstärkung eingesetzt wird. Die Objekte der Relaxation sind die Bildpunkte $\mathbf{S} = (s(x, y))$ eines Grauwertbildes. Zu jedem Bildpunkt (x, y) wird (mit Hilfe eines lokalen Differenzenoperators, Abschnitt 5.4) der Betrag $mag(x, y)$ des Gradienten berechnet.

Die Menge der Eigenschaften sei $L = \{E, N\}$ mit der Bedeutung:

- E: der Bildpunkt liegt auf einer Kante und
- N: der Bildpunkt liegt auf keiner Kante.

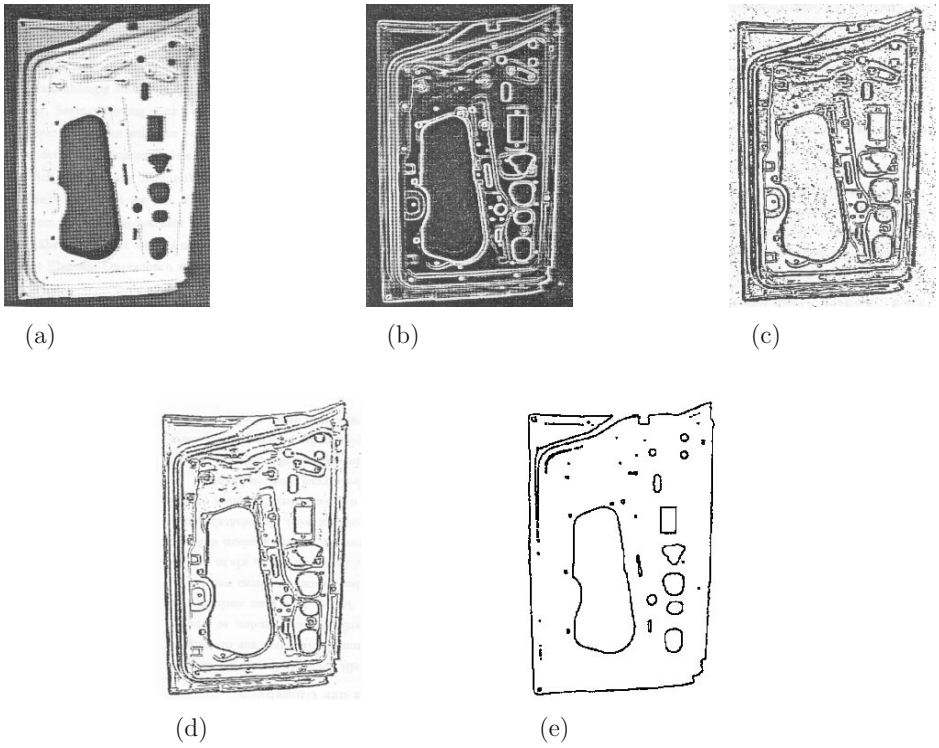
Damit ist klar, dass für jeden Bildpunkt beide Eigenschaften möglich sind. Es ist also $L_{(x,y)} = L$ zu setzen.

Als Anfangswerte der Iteration werden verwendet:

$$p_{(x,y)}^{(1)}(E) = \frac{mag(x, y)}{\max\{mag(x, y)\}} \text{und} \tag{7.21}$$

$$p_{(x,y)}^{(1)}(N) = 1 - p_{(x,y)}^{(1)}(E). \tag{7.22}$$

Die Bestimmung des maximalen Betrags des Gradienten wird dabei in einer bestimmten Umgebung von (x, y), etwa in der 4- oder 8-Nachbarschaft oder auch in einer größeren Umgebung durchgeführt. Damit ist auch die mögliche Einflussnahme der Objekte festgelegt,



(a) (b) (c)

(d) (e)

Bild 7.24: (a) Originalbild: Blechteil. (b) Betrag des Sobeloperators. (c) Invertierung von (b). (d) Binarisierung des invertierten Sobeloperatorbetrags. Dieses Bild ist das Ausgangsbild für die Relaxation. (e) Ergebnis der Relaxation.

deren Gewichtung z.B. für alle gleich sein kann. Schließlich muss noch die Kompatibilitätsfunktion angegeben werden. Dies geschieht hier als Kompatibilitätsmatrix:

$$\begin{pmatrix} r_{EE} & r_{EN} \\ r_{NE} & r_{NN} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{7.23}$$

Damit sind alle Parameter der Relaxation festgelegt und die Iterationen können gemäß (7.18) und (7.20) berechnet werden. Nach einer festgelegen Anzahl von Iterationen werden z.B. die Grauwerte des Originals mit den berechneten Wahrscheinlichkeiten multipliziert und so die Grauwertkanten verstärkt.

Die Bildfolge 7.24 zeigt ein Beispiel dazu. Ausgangspunkt der Relaxation war hier ein Sobelbetragsbild des Originals.

7.11 Grundlagen: Houghtransformation

Zielsetzung der *Houghtransformation* (gesprochen: „Haff"-Transformation) ist das Finden vorgegebener geometrischer Strukturen (*Referenzstrukturen*) in einem (segmentierten) Bild. Es wird geprüft, ob einzelne Segmente der Referenzstruktur ähnlich sind. Als geometrische Strukturen kommen z.B. Geraden, Kreise, Ellipsen, aber auch beliebige andere Formen in Frage. Eine wesentliche Eigenschaft der Houghtransformation ist ihre Robustheit. Damit ist gemeint, dass die im Bild auftretenden Strukturen der Referenzstruktur nicht gleich sein müssen, sondern durch Rauschen oder systematische Fehler gestört sein können. Die Houghtransformation ist auch erfolgreich einsetzbar, wenn die zu detektierenden Strukturen teilweise verdeckt, also nicht vollständig sind. Man kann daher von einer Rekonstruktion geometrischer Strukturen nach Maßgabe einer Referenzstruktur sprechen.

Die Houghtransformation wurde ursprünglich zur *Detektion kollinearer Bildpunkte* in Binärbildern verwendet [Houg62]. Die geometrische Struktur, die als Referenzstruktur dient, ist in diesem Fall die Gerade. Eine Gerade ist durch die Gleichung $ax + by + c = 0$ eindeutig bestimmt. Für $b \neq 0$ ist sie durch die beiden Parameter $m = -\frac{a}{b}$ (die Steigung) und $t = -\frac{c}{b}$ (den $y-$Achsenabschnitt) ebenfalls eindeutig bestimmt. Auch andere Gleichungen legen eine Gerade fest: Wird z.B. vom Koordinatenursprung das Lot auf die Gerade gefällt, so ist durch die Länge r des Lots und den Winkel φ des Lots mit der x- oder y-Achse die Gerade ebenfalls eindeutig festgelegt (Bild 7.25). Den Zusammenhang zwischen (x, y)-Koordinaten und (r, φ)-Koordinaten beschreibt die *Hesse'sche Normalform*:

$$r = x \cdot \cos\varphi + y \cdot \sin\varphi. \tag{7.24}$$

In einem Koordinatensystem, in dem auf der Abszisse die Werte von r und auf der Ordinate die Werte von φ aufgetragen werden, entspricht der Geraden ein Punkt. Dieses Koordinatensystem wird als (r, φ)-*Raum* bezeichnet.

Trägt man für alle Geraden des Geradenbüschels in einem Punkt A die Werte in den (r, φ)-Raum ein, so erhält man eine sinoidale Kurve (Bild 7.26). Das Büschel in einem zweiten Punkt B erzeugt ebenfalls eine derartige Kurve. Da die Büschel eine Gerade gemeinsam haben, schneiden sich beide Kurven im (r, φ)-Raum in einem Punkt. Wenn man somit die Büschel von vielen Punkten, die alle auf einer Geraden liegen, in den (r, φ)-Raum einträgt, so werden sich die dazugehörigen Kurven alle in einem Punkt schneiden.

Dieser Sachverhalt kann zur Detektion kollinearer Bildpunkte in einem Bild oder Bildausschnitt verwendet werden. Dazu wird der (r, φ)-Raum diskretisiert, z.B. die φ-Achse in 1°- oder 5°-Stufen und die Radiusachse je nach aktuellem Genauigkeitsbedarf. Der (r, φ)-Raum geht dann in ein endliches, zweidimensionales Feld über, das im Folgenden auch als *Akkumulator* bezeichnet wird. Jedes Element des Akkumulators entspricht genau einer Geraden im (x, y)-Koordinatensystem. Alle Elemente des Akkumulators werden anfänglich mit null vorbesetzt.

Im nächsten Schritt werden zu allen Vordergrundbildpunkten des binären Eingabebildes die Geradenbüschel berechnet und in den (r, φ)-Raum eingetragen, d.h. es werden die betroffenen Elemente des Akkumulators um eins erhöht. Falls nun im Eingabebild viele Vordergrundbildpunkte näherungsweise auf derselben Geraden liegen (Bild 7.27-b), so

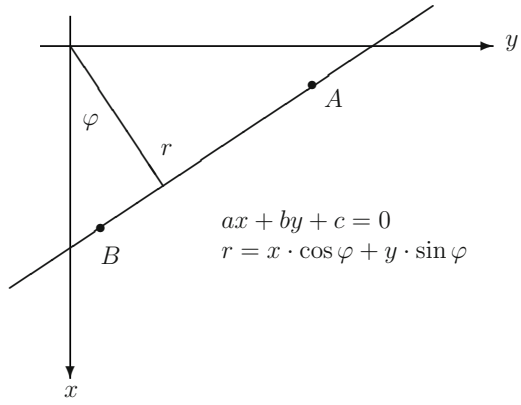


Bild 7.25: Gerade im (x, y)-Koordinatensystem. Wird vom Koordinatenursprung das Lot auf die Gerade gefällt, so legen die Länge r und der Winkel φ die Gerade eindeutig fest. Den Zusammenhang zwischen (x, y)-Koordinaten und (r, φ)-Koordinaten beschreibt die Hesse'sche Normalform.

wird im Akkumulator dasjenige Element einen hohen Wert aufweisen, in dem sich alle Büschelkurven schneiden (Bild 7.27-c). Die Aufgabe, im Originalbild kollineare Bildpunkte zu detektieren, reduziert sich nach dieser Verarbeitung auf die Suche des maximalen Elements des Akkumulators. Die zu diesem Element gehörigen r- und φ-Werte bestimmen die Gerade im (x, y)-Koordinatensystem, auf der die Vordergrundbildpunkte näherungsweise liegen.

Der Verlauf der Geraden kann dabei natürlich nur im Rahmen der gewählten Diskretisierung des (r, φ)-Koordinatensystems angegeben werden. Die Bilder 7.28-a bis 7.28-c zeigen weitere Beispiele und Problematiken zur Houghtransformation.

Die Auswertung des (r, φ)-Raumes kann eine reine Maximumsuche sein oder ein Clusteralgorithmus, mit dem dann auch versucht werden kann, bestimmte Bildstrukturen im Original zu erkennen. Als Beispiel dazu dient Bild 7.29. Es zeigt in Bild 7.29-a ein Quadrat und in Bild 7.29-b den (r, φ)-Raum, in dem sich vier Punkthäufungen ergeben, die den Geraden des Quadrates entsprechen.

Der Vorteil der Houghtransformation liegt darin, dass die Linien auch kleinere Unterbrechungen aufweisen dürfen und dass auch andere, verstreute Punkte auftreten können. Die Wahl des Bildausschnittes, auf den jeweils die Houghtransformation angewendet wird, ist von der Linienbreite abhängig und muss in konkreten Anwendungsfällen sorgfältig abgestimmt werden.

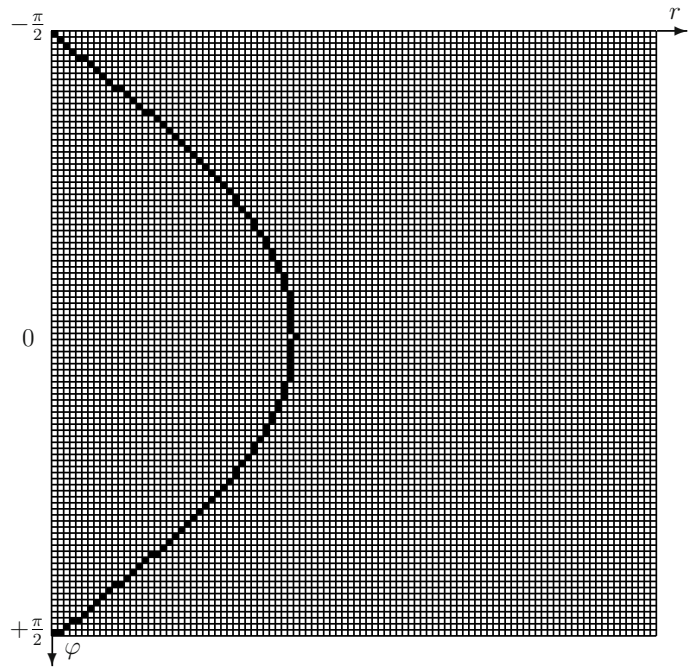


Bild 7.26: In einem (r, φ)-Koordinatensystem ((r, φ)- Raum) entspricht einer Geraden im (x, y)-Koordinatensystem ein Punkt. Die Geraden eines Büschels ergeben sinoidale Kurven.

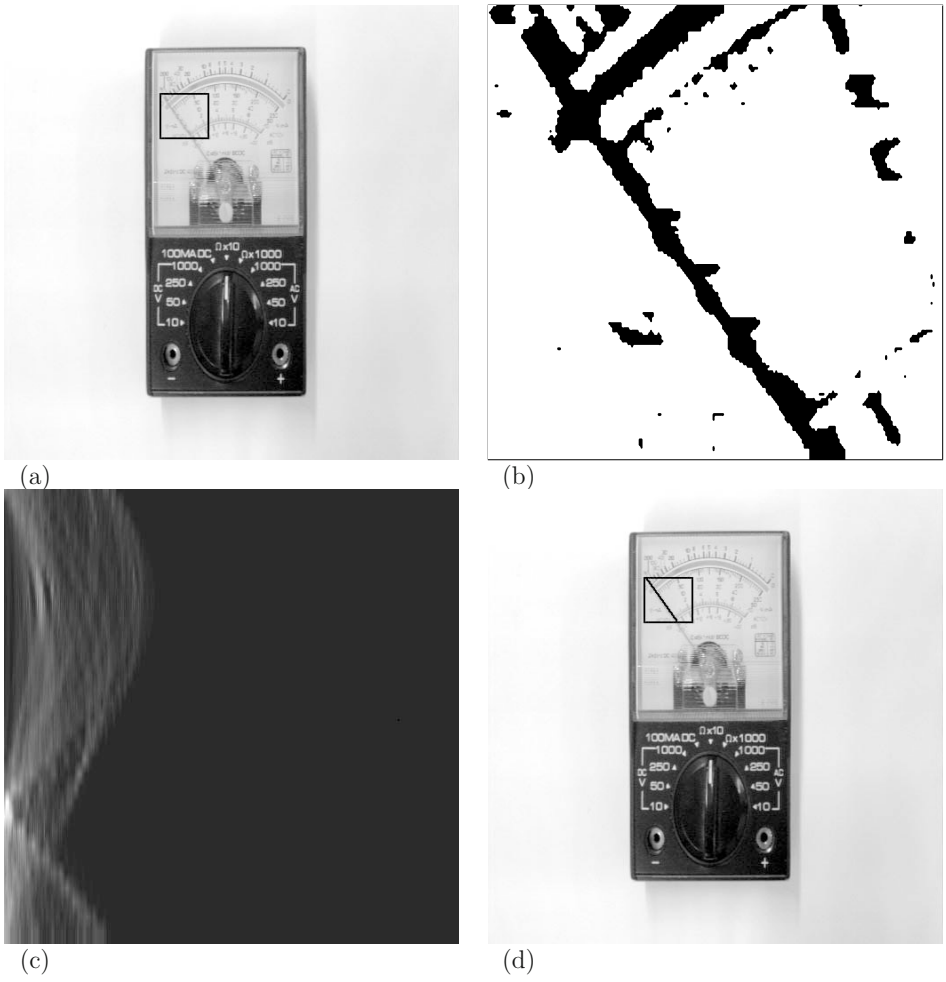


Bild 7.27: (a) Originalbild. Im markierten Ausschnitt soll die Stellung des Zeigers detektiert werden. (b) Markierter Ausschnitt, vergrößert und binarisiert. (c) Houghraum. Die Geradenbüschel aller schwarzen Vordergrundbildpunkte von Teilbild (b) wurden in den (r, φ)-Raum eingetragen. Da sich alle Büschelkurven von kollinearen Bildpunkten im (r, φ)-Raum in einem Punkt schneiden, haben nach dieser Auswertung einige Elemente des Akkumulators hohe Werte, die im Bild durch helle Grautöne dargestellt sind. Die Aufgabe der Detektion des Zeigers besteht jetzt nur mehr aus der Suche des maximalen Wertes des Akkumulators. (d) Das Maximum des Akkumulators liegt bei $r = 2Pixel$ und $\varphi = 124°$. Die detektierte Gerade ist in den Ausschnitt eingezeichnet. Das Lot von der linken oberen Ecke des Bildausschnitts auf diese Gerade bildet mit der nach unten gehenden x-Achse einen Winkel von 124°. Der Betrag ist etwa zwei Pixel.

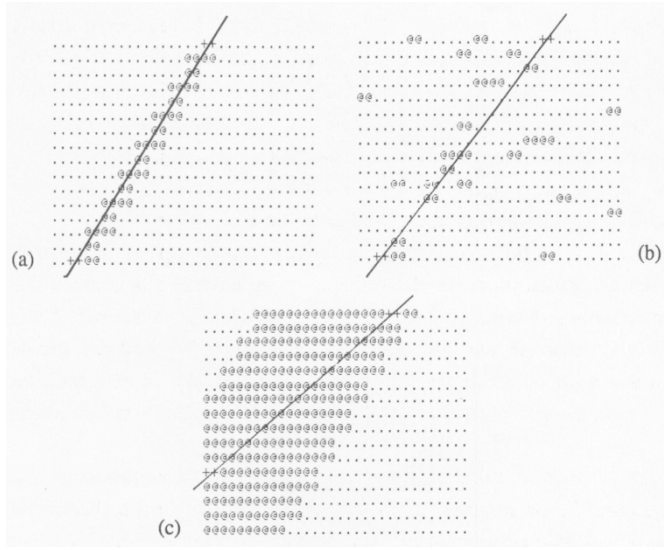


Bild 7.28: Beispiele zur Houghtransformation. (a) Detektion eines Geradenstücks (die Schnittpunkte der erkannten Gerade mit dem Bildrand sind durch '++' gekennzeichnet). (b) Detektion eines unterbrochenen Geradenstücks mit zusätzlichen Störungen. (c) Falls die zu detektierende Linie zu breit im Verhältnis zum Bildausschnitt ist, ergeben sich Ungenauigkeiten.

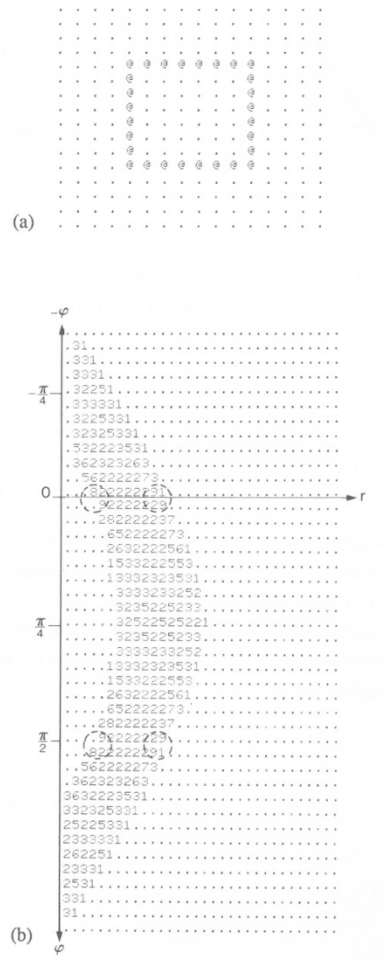


Bild 7.29: Beispiele zur Houghtransformation. (a) Beispiel eines Quadrates im Bildausschnitt. (b) Im (r, φ)-Raum ergeben sich vier Punkthäufungen.

7.12 Verallgemeinerte Houghtransformation

Die Ausführungen von Abschnitt 7.11 lassen allgemeinere Konzepte erkennen, die im Folgenden näher untersucht werden. Es besteht die Aufgabe, in einem Merkmalskanal eines (vorverarbeiteten) Bildes Bildstrukturen nach Maßgabe einer Referenzstruktur zu detektieren. Dabei können die Bildstrukturen bis zu einer bestimmten Grenze verrauscht, gestört oder verdeckt sein. Als geometrische Strukturen kommen z.B. Geraden, Kreise oder Ellipsen in Frage. In Abschnitt 7.13 wird eine Erweiterung auf beliebige Formen vorgestellt. Da bei Geraden, Kreisen oder Ellipsen nur eindimensionale Konturen untersucht werden, ist als Vorverarbeitung eine Kanten- oder Liniendetektion sinnvoll. Man könnte z.B. zu einem Grauwertbild die Gradientenbeträge (Abschnitt 5.4) berechnen und nur die Bildpunkte mit hohem Betrag in die weitere Verarbeitung einbeziehen.

Mit der Houghtransfomation werden im nächsten Schritt solche Bildpunkte gesucht, die ähnliche Merkmalseigenschaften besitzen. Es werden also lokale Merkmalsausprägungen zu globalen Ausprägungen kombiniert.

Die Darstellung der Standard-Houghtransformation von Abschnitt 7.11 zeigt drei Schritte, die der Reihe nach behandelt werden müssen:

- Die Parametrisierung der Referenzstruktur.
- Die akkumulierende Abbildung der Merkmalspunkte.
- Die Auswertung des Akkumulators.

7.12.1 Parametrisierung der Referenzstruktur

Für die Referenzstruktur muss eine geeignete Parametrisierung gefunden werden. Ein Beispiel für die Gerade wurde in Abschnitt 7.11 schon gegeben. Wie bereits erwähnt, können aber auch die Steigung und der y-Achsenabschnitt als Parameterpaar verwendet werden. Ein weiteres Beispiel sind zwei Werte s_1, s_2, die die Schnittpunkte der Geraden mit dem Bildrand angeben. Die Werte für s_1 und s_2 werden in einer festgelegten Orientierung entlang des Bildrandes gemessen (Bild 7.30).

Analog hierzu ergibt sich, dass für den Kreis drei Parameter (z.B. (x, y)-Koordinaten des Mittelpunkts und Radius) und für die Ellipse fünf Parameter (z.B. (x, y)-Koordinaten des Mittelpunkts, Längen der beiden Halbachsen und Drehwinkel) zur Definition notwendig sind. Eine geometrische Referenzstruktur wird somit durch ein *Parametertupel* beschrieben, das einen *Parameterraum* aufspannt. Die Anzahl der Parameter bestimmt die Dimension des Parameterraums, der auch *Transformationsraum* oder *Houghraum* genannt wird. Ein Parametertupel $(p_1, p_2, ..., p_N)$ entspricht genau einer Ausprägung der Referenzstruktur in einer definierten Lage.

Da der Parameterraum diskretisiert werden muss, sind nur solche Parametrisierungen von Bedeutung, bei denen sich für alle in Frage kommenden Lagen der Referenzstruktur endliche Werte ergeben. Werden Geraden als Referenzstruktur verwendet, so erfüllen die Parameterpaare (r, φ) oder (s_1, s_2) diese Forderung.

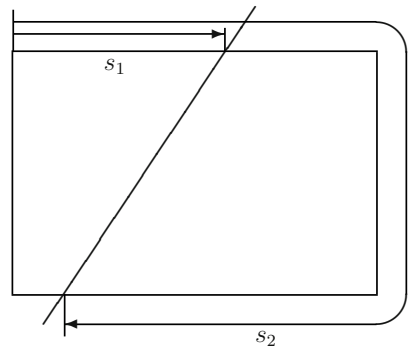


Bild 7.30: Parametrisierung einer Geraden durch die Angabe ihrer Schnittpunkte mit dem Bildrand.

Zur Diskretisierung werden die einzelnen Parameterskalen abgeschätzt und in (meistens äquidistante) Intervalle eingeteilt. Bei dieser Einteilung können Genauigkeitsanforderungen berücksichtigt werden. Durch die Diskretisierung des Houghraums wird das Akkumulatorfeld definiert. Jedem Element des Akkumulators entspricht ein Parametertupel und damit eine Referenzstruktur. Der Akkumulator wird anfangs auf null gesetzt. Bei einigen Varianten sind die Elemente des Akkumulators nicht Skalare, sondern Listen. In diesem Fall wird das Element in den Zustand „leere Liste“ gesetzt.

7.12.2 Akkumulierende Abbildung der Merkmalspunkte

In dieser Phase werden für jeden Merkmalspunkt (Vordergrundpunkt) die Lageparameter aller möglichen durch die Referenzstruktur festgelegten Strukturen berechnet, deren Mitglied der Punkt sein kann. Im Fall einer Geraden als Referenzstruktur ist das das Geradenbüschel im aktuellen Punkt. Für jedes sich ergebende Parametertupel wird die entsprechende Zelle des Akkumulators inkrementiert.

Die Inkrementierung kann im einfachsten Fall eine Erhöhung um eins sein. Aber auch kompliziertere Inkrementfunktionen sind möglich. Werden z.B. diejenigen Bildpunkte als Vordergrundpunkte betrachtet, deren Gradientenbetrag größer als ein vorgegebener Wert ist, so kann die Inkrementierung auch um diesen Betrag erfolgen.

7.12.3 Auswertung des Akkumulators

Der Auswertung des Akkumulatorfeldes liegt folgende Überlegung zugrunde: Strukturen, die der Referenzstruktur entsprechen und durch viele Merkmalspunkte gestützt sind, werden im Akkumulator einen hohen Wert aufweisen. Wenn man den maximalen Wert des

Akkumulators bestimmt, so kann man ziemlich sicher sein, dass das zugehörige Parametertupel die Lage einer Bildstruktur beschreibt, die der Referenzstruktur sehr ähnlich ist.

Bei der Auswertung des Akkumulators werden somit das globale Maximum oder lokale Maxima gesucht. Je mehr eine Struktur gestört, verrauscht oder verdeckt ist, je unähnlicher sie somit der Referenzstruktur ist, desto weniger markant werden sich die Maxima ausprägen. Es ist offensichtlich, dass das Problem der Auswertung des Akkumulators ein *cluster*-Analyseproblem ist und daher mit den einschlägigen Verfahren (z.B. mit dem *fuzzy*-Klassifikator von Kapitel 22) behandelt werden kann.

Nachdem eine Struktur auf diese Weise gefunden ist, können aus dem zugeordneten Parametertupel weitere Eigenschaften, wie z.B. die Größe, die Orientierung, die Länge der Kontur, usw. berechnet werden.

Im folgenden Abschnitt wird eine Erweiterung der Houghtransformation vorgestellt, die es erlaubt, beliebige Segmentstrukturen als Referenz zu verwenden. Dabei kann neben der konturbezogenen Vorgehensweise auch eine flächenhafte Analyse durchgeführt werden.

7.13 Erweiterte Houghtransformation

Die Houghtransformation wurde ursprünglich, wie oben dargestellt, zur Detektion von Geraden und Kurven in Bildausschnitten definiert. In diesem Abschnitt wird die Houghtransformation verallgemeinert und für Segmente verwendet, die durch ihre Konturkurve in einem binarisierten Kantenbild gegeben sind.

7.13.1 Erweiterte Houghtransformation der Randpunkte

Es wird ein Bild vorausgesetzt, in dem unterschiedliche Segmente enthalten sind. Es besteht die Zielsetzung, zu entscheiden, ob ein bestimmtes Segment mit einer für dieses Segment charakteristischen Form im Bild enthalten ist.

Als Vorverarbeitungsschritte sei eine Segmentierung (etwa eine Binarisierung, Abschnitt 4.5) und eine Randerkennung (Abschnitte 7.3 bis 7.11) der einzelnen Segmente durchgeführt worden. Die Randerkennung kann z.B. mit Hilfe einer *run-length*-Codierung, einer Segmentvereinzelung und einer Markierung der Randpunkte der einzelnen Segmente durchgeführt werden (Kapitel 23). Das gesamte Bild kann durch Rauschen gestört sein. Auch eine teilweise Verdeckung der Segmente ist zulässig.

Zu jedem Segment wird ein *Bezugspunkt* festgelegt. Als Bezugspunkt kann ein beliebiger Punkt verwendet werden. Es bietet sich z.B. der Flächenschwerpunkt an. Bild 7.31 zeigt ein Beispiel, wie ein derartiges Eingabebild aussehen könnte. Die Bezugspunkte der Segmente sind schraffiert markiert.

Die Aufgabe besteht darin zu entscheiden, ob ein vorgegebenes Segment, das in Form einer bestimmten Datenstruktur vorliegt, im Bild enthalten ist oder nicht. Dieses Segment wird im Folgenden als *Referenzsegment* und die dazugehörige Kontur als *Referenzkontur* bezeichnet. Vorausgesetzt wird, dass das zu detektierende Segment in gleicher Größe und Drehlage im Eingabebild enthalten sein muss.

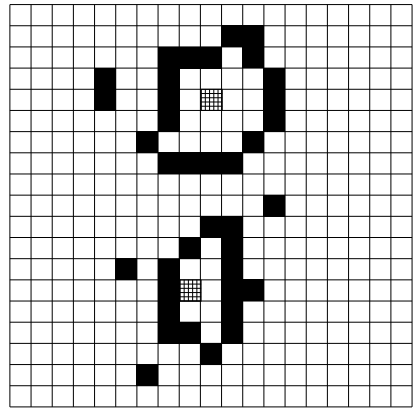


Bild 7.31: Beispiel zu einem vorverarbeiteten Bild, in dem zwei Segmente durch ihren Rand darstellt sind. Die Bezugspunkte der Segmente sind schraffiert markiert. Außerdem enthält das Bild einzelne Störungen.

Zu Beginn der Verarbeitung werden zunächst die Elemente eines zweidimensionalen *Akkumulatorfeldes* $\mathbf{A} = (a(x, y))$, das die Größe des Eingabebildes besitzt, auf null gesetzt. Es wird ein beliebiger Vordergrundpunkt P des binären Eingabebildes $\mathbf{S}_e$ betrachtet. Dieser Punkt könnte auch ein Punkt der Referenzkontur sein. Falls es ein solcher Punkt ist, müsste der Bezugspunkt des Referenzsegments in einer bestimmten, bekannten relativen Lage dazu liegen. Es werden der Reihe nach alle Möglichkeiten untersucht, wie die Referenzkontur relativ zu diesem Punkt liegen könnte. Jedesmal ergibt sich eine andere Lage für den Bezugspunkt. Diese Menge von möglichen Lagen des Bezugspunktes wird *Lösungsmenge für den Bezugspunkt* genannt. Bild 7.32 zeigt die Lösungsmenge für den markierten Punkt P. Das Referenzsegment ist darüber abgebildet. Man sieht, dass die Lösungsmenge punktsymmetrisch zur Referenzkontur liegt.

Im nächsten Schritt werden die Zähler im Akkumulatorfeld für die Punkte der Lösungsmenge um eins erhöht. Die geschilderten Schritte werden für alle in Frage kommenden Vordergrundpunkte des Eingabebildes durchgeführt. Abschließend wird im Akkumulatorfeld das Maximum ermittelt. Die Koordinaten des Maximums werden festgehalten. Falls der Wert des Maximums einen bestimmten Schwellwert übersteigt, ist das Segment gefunden und der Bezugspunkt liegt an der Stelle der Maximumkoordinaten. Je nach Wahl des Schwellwertes können im Eingabebild $\mathbf{S}_e$ mehr oder weniger große Abweichungen von der Form des zu detektierenden Referenzsegments auftreten.

Zusammengefasst lautet der Algorithmus für die erweiterte Houghtransformation wie folgt:

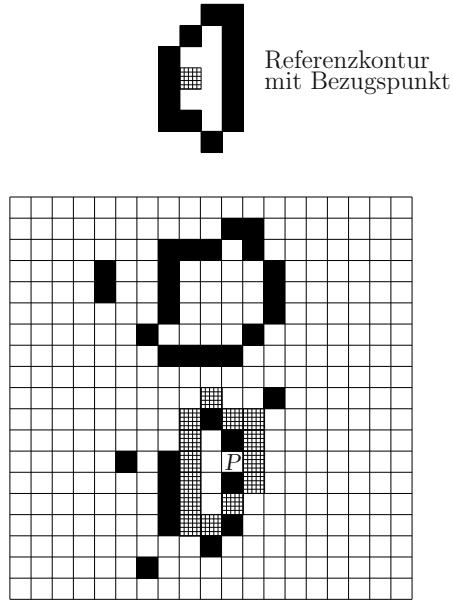


Bild 7.32: Wenn der Punkt P ein Punkt der Referenzkontur ist, müsste der Referenzpunkt in einer bestimmten relativen Position dazu liegen. Es wird der Reihe nach angenommen, dass der Punkt P ein Punkt der Referenzkontur ist. Jedesmal ergibt sich dann eine andere Position für den Referenzpunkt. Die Lösungsmenge für den markierten Punkt P ist schraffiert. Das zu suchende Segment ist darüber abgebildet. Die Lösungsmenge liegt punktsymmetrisch zur Referenzkontur. Für jeden Vordergrundbildpunkt wird die Lösungsmenge bestimmt und die entsprechenden Elemente im Akkumulator erhöht. Am Ende wird dasjenige Akkumulatorelement einen hohen Wert aufweisen, das die Lage des Referenzpunktes im Eingabebild angibt.

A7.6: Erweiterte Houghtransformation.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y))$ ein Binärbild, in dem nur die Ränder der Segmente und eventuelle Störungen auftreten (z.B. Hintergrundpunkte: Grauwert 0, Vordergrundpunkte: Grauwert 1);

◇ $\mathbf{A} = (a(x, y))$ sei ein Akkumulatorfeld, das anfangs mit null vorbesetzt ist.

◇ Das Referenzsegment mit Bezugspunkt liegt in einer geeigneten Datenstruktur vor.

Algorithmus:

(a) Für alle Vordergrundpunkte $s_e(x, y)$ mit dem Grauwert 1:

(aa) Ermittle die Lösungsmenge für den Bezugspunkt.

(ab) Erhöhe die entsprechenden Zähler $a(x, y)$ im Akkumulatorfeld um eins.

(b) Ermittle das Maximum des Akkumulatorfeldes.

(c) Falls das Maximum einen bestimmten Schwellwert übersteigt, geben die Koordinaten des Maximums die Lage des Bezugspunktes für das gesuchte Segment an.

Ende des Algorithmus

Die Bildfolge 7.33 zeigt ein Beispiel. Bild 7.33-a ist das Originalbild, Bild 7.33-b das vorverarbeitete Binärbild. Bild 7.33-c zeigt die Referenzkontur und Bild 7.33-d das Original mit dem eingeblendeten, detektierten Segment.

Ein großer Nachteil dieses Verfahrens ist, dass es weder größen- noch rotationsinvariant ist. Im Rahmen einer praktischen Anwendung müssten die zu untersuchenden Strukturen so vorverarbeitet werden, dass dieser Nachteil keine Rolle spielt. Ein Möglichkeit dazu wäre es, die Teile bereits ausgerichtet dem Videosensor zuzuführen. Ist das nicht möglich, so könnte man versuchen, eine Skalen- und Rotationsinvarianz durch geometrische Operationen zu erreichen. Auch die Berechnung von Momenten (Abschnitt 24.9) wäre anwendbar.

Wenn das Referenzsegment in diskreten Skalen- und Rotationsstufen vorliegt, wird der obige Algorithmus für jede Größen- und Rotationskombination durchlaufen. Das Verfahren ist dann größen- und rotationsinvariant.

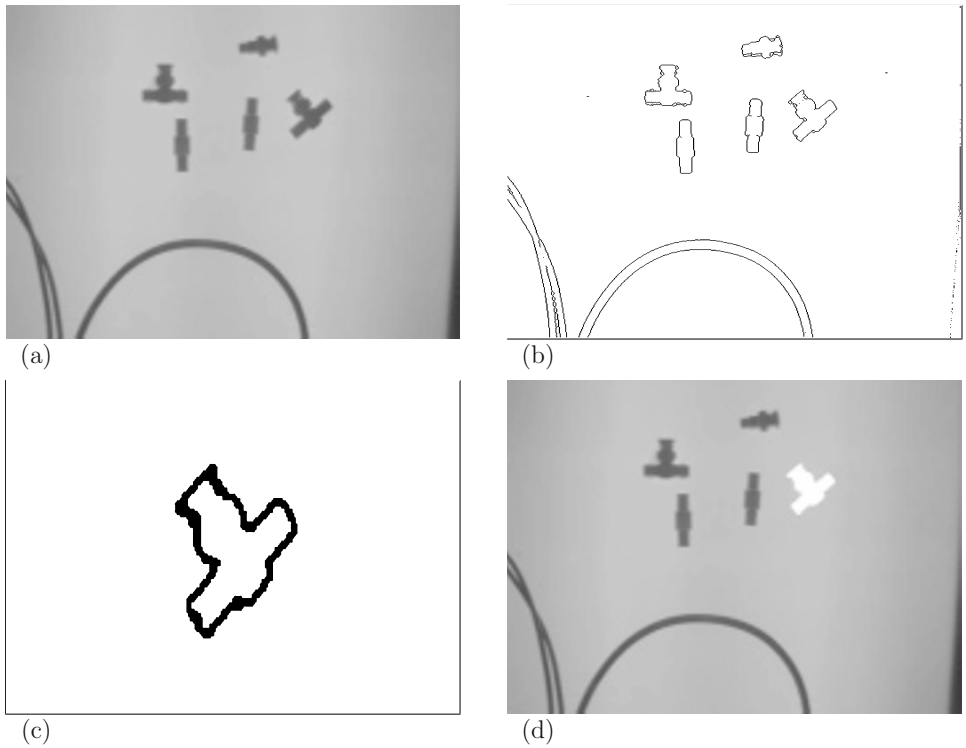


Bild 7.33: (a) Originalbild. (b) Vorverarbeitetes Binärbild. (c) Referenzkontur. (d) Original mit eingeblendetem Segment.

7.13.2 Erweiterung auf flächenhaft gegebene Segmente

Für die Betrachtungen in diesem Abschnitt sei ein Segment flächenhaft durch eine Menge von Punkten (nicht nur Konturpunkten) gegeben, die alle eine bestimmte Eigenschaft besitzen. Beispielsweise kann ein Segment einen bestimmten Grauwert, eine bestimmte Gradientenrichtung oder eine bestimmte Oberflächenstruktur (Textur) besitzen.

In einem Vorverarbeitungsschritt wird für jeden Bildpunkt $s_e(x, y)$ des Eingabebildes $\mathbf{S}_e$ aus einer Umgebung ein (Textur-)Maß $s_t(x, y) = f(U(s_e(x, y))$ berechnet. Dabei wird mit $U(s_e(x, y))$ die Menge der Umgebungsbildpunkte von $s_e(x, y)$ bezeichnet und mit f die Funktion, die zur Berechnung des (Textur-)Maßes auf die Umgebung angewendet wird (Kapitel 15 bis 18).

Das Referenzsegment mit Bezugspunkt besitzt einen bekannten Wert ms_t für das (Textur-)Maß. Für alle Punkte (x, y), für die die Distanz $d(ms_t, s_t(x, y))$ kleiner als ein vorgegebener Schwellwert ist, wird die Lösungsmenge für den Bezugspunkt bestimmt. Die weitere Auswertung erfolgt wie in Abschnitt 7.12.

Da hier im Gegensatz zu Abschnitt 7.12 die Lösungsmenge für wesentlich mehr Bildpunkte bestimmt werden muss, ergeben sich längere Rechenzeiten. Man kann deshalb die Segmente *run-length* codieren (Kapitel 23) und statt des Akkumulatorfeldes $\mathbf{A} = (a(x, y))$ ein Feld $\mathbf{D} = (d(x, y))$ mit

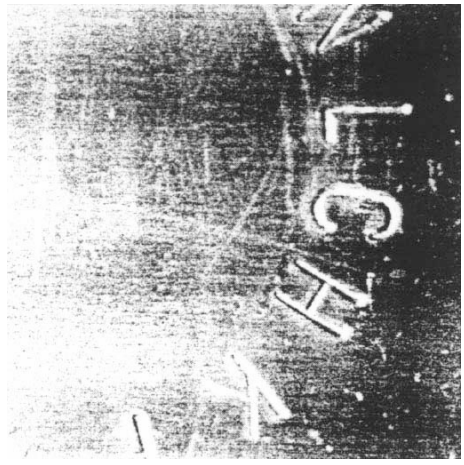
$$d(x, y) = a(x, y) - a(x, y - 1) \tag{7.25}$$

zu verwenden. Tritt nämlich eine bestimmte Zeile des *run-length*-Codes in der Lösungsmenge auf, so müssten alle entsprechenden Felder $a(x, y)$ des Akkumuators um eins erhöht werden. Bei der Verwendung von $d(x, y)$ müssen nur zwei Felder verändert werden: Am linken Rand wird die Differenz um eins größer und am rechten Rand um eins kleiner. Für die Felder dazwischen ändert sich die Differenz nicht. Mit dieser Verbesserung erhöht sich die Rechenzeit nur geringfügig.

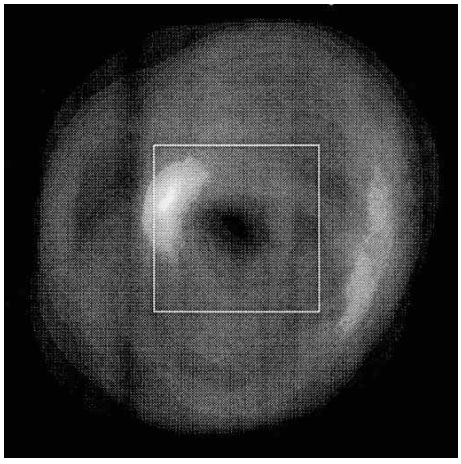
Zur Maximumsuche kann das Akkumulatorfeld $\mathbf{A} = (a(x, y))$ einfach berechnet werden: Für jeden Zeilenanfang wird $a(0, y) = 0$ gesetzt. Die weiteren Zeilenelemente berechnen sich dann wie folgt:

$$a(x, y) = a(x - 1, y) + d(x, y). \tag{7.26}$$

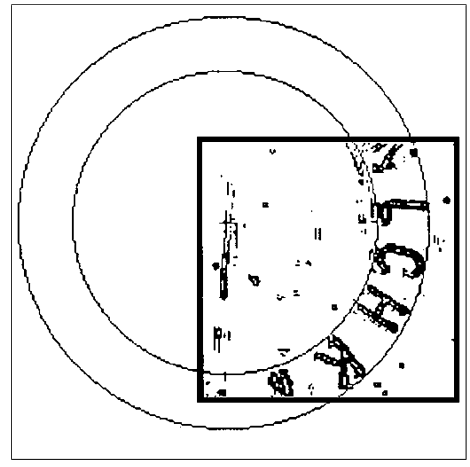
Abschließend zeigt die Bildfolge 7.34 ein Anwendungsbeispiel. Dieses Beispiel und die Bilder wurden [Ever90] entnommen. Zu detektieren sind in Blech eingestanzte Schriftzeichen, die auf einem Kreisbogen angeordnet sind (Bild 7.34-a). Um die Zeichen lokalisieren zu können, ist die Position eines Ringes mit bekanntem Innen- und Außenradius gesucht, in dem die Schriftzeichen liegen. Die Schriftzeichen heben sich durch starke Hell-Dunkel-Übergänge ab. Deshalb wird als Merkmal für die Akkumulation gefordert, dass ein Punkt Vordergrundpunkt im binarisierten Gradientenbild ist. Als Refenenzpunkt für den gesuchten Ring wird sein Mittelpunkt verwendet. Der Akkumulatorwert $a(x, y)$ enthält somit die Anzahl der Vordergrundpunkte, die im Ring liegen, wenn sich sein Mittelpunkt in der Position (x, y) befindet. Bild 7.34-b zeigt das Akkumulationsergebnis (helle Grauwerte entsprechen hohen Werten). Bild 7.34-c zeigt das Gradientenbild mit der detektierten optimalen Lage des Rings.



(a)



(b)



(c)

Bild 7.34: (a) In Blech eingestanzte Schriftzeichen, die innerhalb eines Kreisrings angeordnet sind. Aus [Ever90]. (b) Ergebnis der Akkumulation. (c) Gradientenbild mit detektierter optimaler Lage des Rings.

7.14 Sequentielle Kantenextraktion, Linienverfolgung

Der Grundgedanke bei den sequentiellen Verfahren ist, dass, um eine Grauwertkante oder eine Linie zu extrahieren, nicht alle Bildpunkte des Bildes untersucht werden müssen, sondern nur diejenigen, die Kandidaten sind. Damit werden auch gleich die zwei Teilprobleme der Linienverfolgung deutlich: das *Finden von Ansatzstellen für eine Linienverfolgung* und das *Finden von Nachfolgepunkten*, wenn schon ein bestimmter Abschnitt extrahiert wurde.

Bevor aber diese Probleme näher untersucht werden, noch einige generelle Bemerkungen zur Linienverfolgung. Hier treten eine Reihe von Sachverhalten auf, mit denen alle verwendeten Verfahren konfrontiert werden. Was passiert z.B., wenn die verfolgte Linie unterbrochen ist? Bricht das Verfahren ab oder besteht die Möglichkeit, kleinere Unterbrechungen zu überbrücken? Was geschieht bei Verzweigungen? Werden sie überhaupt erkannt? Können die verschiedenen Äste extrahiert werden? Merkt das Verfahren, wenn es einen Ast ein zweitesmal durchläuft? Was passiert bei Helligkeits- und/oder Kontraständerungen entlang des Linienverlaufs? Schließlich: Besteht die Möglichkeit, auch parallel verlaufende Linien zu erkennen? Bei den folgenden Verfahren kann ihre Eignung anhand dieser Bewertungskriterien gemessen werden.

Jetzt zur ersten Fragestellung, dem Finden von Ansatzstellen für eine Linienverfolgung. Die einfachste und sicherste Möglichkeit ist natürlich die interaktive Festlegung eines Ansatzpunktes für eine Linie. Das klingt zunächst nicht sehr praktikabel, aber es gibt Anwendungsbereiche, wo die interaktive Festlegung durchaus eingesetzt werden kann, z.B. im Rahmen eines Systems, in dem digitalisierte Luft- oder Satellitenbilder teilautomatisch ausgewertet werden. Mit den Methoden der multivariaten Klassifizierung können hier die flächenhaften Objekte (Gewässer, Grünland, Ackerland, usw.) extrahiert werden. Ein Bearbeiter wird dann diese Ergebnisse interaktiv beurteilen und, wenn nötig, korrigieren. Bei dieser Nachbearbeitung ist es dann aber leicht möglich, das verarbeitende System zur Linienverfolgung von Straßen, Gewässerläufen, Eisenbahndämmen usw. jeweils auf geeignete Ansatzstellen zu setzen. In vielen Fällen genügen hier schon wenige Punkte pro Bildausschnitt, wenn der Linienverfolger auch Verzweigungen erkennt und richtig verarbeitet.

Eine andere, automatische Möglichkeit ist die Bewertung eines in Frage kommenden Bildpunktes durch Klassifizierung. Dazu kann so vorgegangen werden, wie in Abschnitt 7.4, Bild 7.6 angedeutet wurde. Zunächst werden die Bildpunkte im Merkmalsraum „Grauwert/Betrag des Gradienten" klassifiziert. Wird ein Bildpunkt als „auf einer Kante liegend" erkannt, so kann ausgehend von diesem Punkt die Verfolgung der Linie begonnen werden.

Oft bieten sich aus der Abfolge der Verarbeitungsschritte Bildpunkte als Kandidaten für eine Ansatzstelle an. Dazu kann als Beispiel die Segmentierung eines Bildes mit Hilfe von Baumstrukturen gebracht werden (Kapitel 19). Hier wird die Baumstruktur zunächst nach homogenen (unstrukturierten) Bildbereichen untersucht, dann werden inhomogene (strukturierte) Bildbereiche extrahiert. Diese Bereiche lassen sich, grob gesprochen, in Bildbereiche mit flächenhaft ausgeprägter Struktur und in Strukturbereiche einteilen, die von linienhaften Bildinhalten erzeugt werden. Nach der Segmentierung von den Bildbereichen,

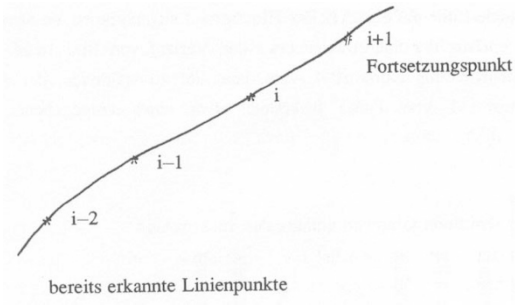


Bild 7.35: Voraussetzungen für die Linienverfolgung: Die Bildpunkte $..., i-2, i-1, i$ sind bereits als Linienpunkte erkannt; der Bildpunkt $i+1$ soll als Fortsetzungspunkt ermittelt werden.

die von der flächenhaften Struktur eingenommen werden, bleiben also im wesentlichen die Linienstrukturen übrig, sodass hier Bildpunkte gezielt als Ansatzstellen ausgewählt werden können.

Weitere Möglichkeiten der Suche von Ansatzstellen sind die Überlagerung mit einem Gitternetz und die Auswertung der einzelnen Flächenstücke oder die Bewertung des Bildinhaltes im Rahmen eines Modells, das im jeweiligen Anwendungsfall bestimmte Bildbereiche als wahrscheinlicher für die Beinhaltung einer Linie erscheinen lässt.

Jetzt zur zweiten Fragestellung, dem Finden von Nachfolgepunkten, wenn schon ein Teil der Linie verarbeitet ist. Die Darstellung der einzelnen Vorgehensweisen geht immer von folgenden Voraussetzungen aus (Bild 7.35): Es wurden bereits die Punkte $..., i-2, i-1, i$ als auf der Linie liegend erkannt. Im nächsten Schritt soll der Punkt $i+1$ als Fortsetzungspunkt ermittelt werden.

Eine Möglichkeit besteht nun darin, vom Punkt i ausgehend Suchstrahlen zu berechnen und entlang dieser Suchstrahlen den mittleren Grauwert zu ermitteln (Bild 7.36).

Wird z.B. eine dunkle Linie auf einem hellen Hintergrund angenommen, so werden die mittleren Grauwerte entlang der Suchstrahlen etwa den Verlauf von Bild 7.37 zeigen. Der Suchstrahl mit der Richtung Nummer 4 wäre dann der zu wählende, der Abstand des Fortsetzungspunktes $i+1$ von Punkt i könnte etwa einer vorgegebenen Konstante entsprechen.

Bei diesem (aber auch bei allen folgenden) Verfahren kann der Suchbereich aus der Vorgeschichte der bereits gefundenen Bildpunkte eingeschränkt werden. Wird zu den letzten k bereits gefundenen Bildpunkten eine Ausgleichsgerade berechnet, so kann ein bestimmtes Winkelfeld (*Suchfeld*) auf beiden Seiten einer Geraden als Suchbereich verwendet werden.

Wie ist dieses Verfahren bezüglich der oben aufgeworfenen Fragestellungen zu bewer-

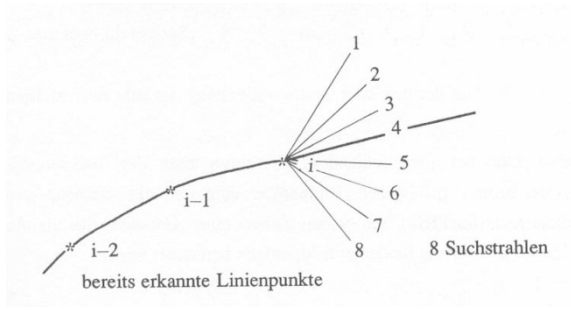


Bild 7.36: Linienverfolgung durch Berechnung von Suchstrahlen.

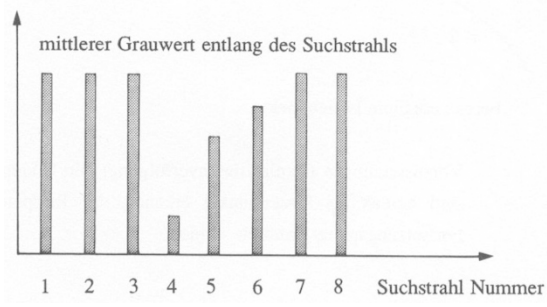


Bild 7.37: Verlauf der mittleren Grauwerte entlang der acht Suchstrahlen.

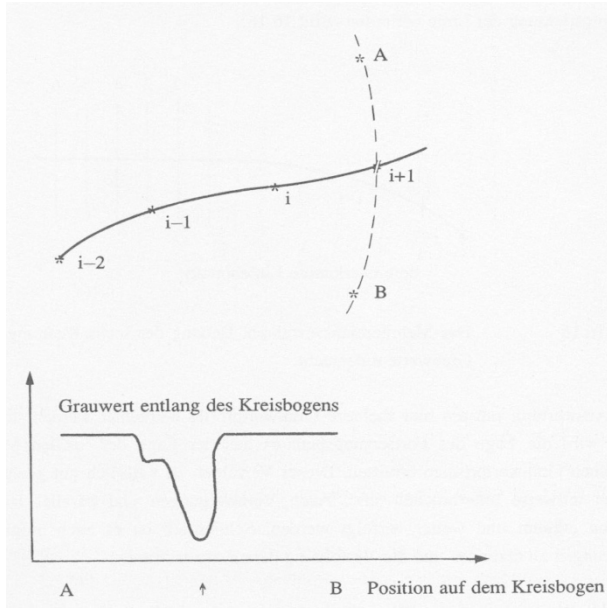


Bild 7.38: Kreisbogenverfahren (a) Kreisbogen vom Punkt A zum Punkt B, begrenzt durch den Suchbereich. (b) Grauwertprofil entlang des Kreisbogens. Die markierte Position wird als Lage des Nachfolgepunktes akzeptiert.

ten? Das Überbrücken von Linien, die in eingeschränkten Bereichen eine Unterbrechung aufweisen, ist möglich, da ja entlang der Suchstrahlen gemittelt wird. Bei starken Kontraständerungen entlang des Linienverlaufs ist dieses Verfahren nicht geeignet. Bei Verzweigungen oder parallel verlaufenden Linien besteht die Gefahr, dass das Suchverfahren an der falschen Stelle den Nachfolgepunkt setzt. Zur Erhöhung der Sicherheit kann hier vom gefundenen Nachfolgepunkt $i+1$ eine Rückverfolgung zum Punkt i angeschlossen werden: Wenn die Rückverfolgung nicht in einer vorgegebenen Kreisumgebung des Linienpunktes i endet, so wird der Punkt $i+1$ nicht als Nachfolger akzeptiert.

Nun eine andere Strategie: Vom Linienpunkt i als Mittelpunkt werden entlang eines Kreisbogens, der durch den Suchbereich definiert ist, die Grauwerte untersucht (Bild 7.38). Lässt sich im Grauwertprofil (im betrachteten Beispiel) ein lokales Minimum ermitteln, so wird die Position des Minimums als Lage des Nachfolgepunktes akzeptiert.

Bei oft unterbrochenen Linien ist dieses Verfahren nicht geeignet. Verbessert kann es dadurch werden, dass mehrere Suchkreisbögen in unterschiedlichen Abständen berechnet

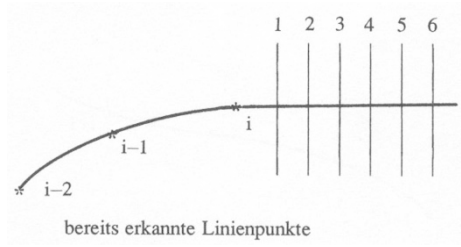


Bild 7.39: Das Mehrbereichsverfahren. Entlang der sechs Richtungen werden die Grauwerte untersucht.

werden. Für die anderen Fragen gelten wohl dieselben Aussagen wie im ersten Verfahren. Eine Verzweigung drückt sich u.U. im Grauwertprofil der Kreisbögen aus. Allerdings kann zunächst nicht zwischen ,,Verzweigung" und ,,paralleler Linie" unterschieden werden.

Die Erweiterung des eben besprochenen Verfahrens, das auch als *Einbereichsverfahren* bezeichnet wird, ist das *Mehrbereichsverfahren*, bei dem, wie schon oben angedeutet, der Suchbereich in unterschiedlichen Abständen untersucht wird. Um Rechenzeit zu sparen, müssen hier nicht unbedingt Kreisbögen berechnet werden, sondern es genügt auch die Überlagerung eines Linienrasters, bei dem die Grauwertprofile etwa senkrecht zur Fortsetzungsrichtung der Linie verlaufen (Bild 7.39).

Zur Auswertung müssen hier mehrere Grauwertprofile betrachtet werden. Im obigen Beispiel wird die Lage des Fortsetzungspunktes aus der Lage der lokalen Minima in den einzelnen Grauwertprofilen ermittelt. Dieses Verfahren ist natürlich gut geeignet, wenn die Linien teilweise unterbrochen sind. Auch Verzweigungen und parallel laufende Linien können erkannt und weiter verfolgt werden. Schließlich ist es auch möglich, Kontraständerungen zu erkennen und die Verfolgung darauf abzustimmen.

Abschließend seien noch stichwortartig einige weitere Möglichkeiten aufgezeigt. Die in Abschnitt 7.11 erläuterte Houghtransformation kann ebenfalls bei sequentiellen Methoden eingesetzt werden. Dazu wird der Suchbereich, etwa ein Rechteck, in dem die zu verfolgende Linie wahrscheinlich verläuft, in den (r, φ)-Raum transformiert und der Verlauf der Linien aus der Auswertung des (r, φ)-Raumes abgeleitet. Für diese Vorgehensweise gelten etwa dieselben Gütekriterien wie für das Mehrbereichsverfahren.

Schließlich kann der weitere Verlauf der Linie auch modelliert und durch Vergleich des Modells mit dem tatsächlichen Verlauf ermittelt werden (Abschnitt 7.4).

7.15 Abschließende Bemerkungen zu Kanten und Linien

Bei vielen Anwendungen ist es notwendig, Segmente in einem Bild zu erkennen und einer bestimmten Klasse zuzuordnen. Die Segmentierung wird in den Kapiteln 11 bis 22 ausführlich erläutert. Wenn Segmente durch ihren Rand oder durch ihr Skelett dargestellt sind, so kann diese Information natürlich auch zur Segmentierung und Erkennung verwendet werden. Hier sei z.B. auf die beiden Ausprägungen des Strahlenverfahrens (Kapitel 25 und 26) hingewiesen.

Ein anderer Gesichtspunkt sei an dieser Stelle ebenfalls kurz angesprochen. Bei der Verarbeitung von Bildfolgen (Bewegungsdetektion, Tracking, Stereobildauswertung) werden oft *Verschiebungsvektorfelder* (*Optischer Fluß*, engl. *optical flow*) berechnet (Kapitel 14). Sie geben an, wie z.B. in einem Bild zum Zeitpunkt t einzelne Bildblöcke verschoben werden müssen, um daraus das Bild zum Zeitpunkt $t + 1$ zu erhalten. Natürlich kann man nicht für alle Bildpunkte einen Verschiebungsvektor berechnen: In homogenen Bildbereichen ist es nicht möglich (und evtl. auch nicht notwendig), Verschiebungsvektoren zu ermitteln. Es werden daher Bildpunkte gesucht, die bevorzugt zur Berechnung von Verschiebungsvektoren geeignet sind. Diese Punkte werden *markante Punkte* genannt. Ein Algorithmus dazu heißt *Punktefinder* oder auch *Deskriptor*.

Welche Kriterien können in einem Punktefinder verwendet werden? Ein markanter Punkt zeichnet sich dadurch aus, dass an dieser Stelle eine starke Grauwertänderung stattfindet, also an dieser Stelle der Gradient einen hohen Wert aufweist. Somit können markante Punkte dadurch gefunden werden, dass in einem Bild die Gradientenbeträge (z.B. Sobeloperator, Abschnitt 5.4) mit einem Schwellwert verarbeitet werden. Alle Punkte, deren Gradientenbetrag über dem Schwellwert liegt, werden als Kandidaten für markante Punkte betrachtet.

Kapitel 8

Operationen im Frequenzbereich

8.1 Anwendungen

Im Rahmen der digitalen Bildverarbeitung werden Operationen im Frequenzbereich häufig eingesetzt. Vor allem die Fouriertransformation wird zur digitalen Filterung verwendet. Mit den Methoden der digitalen Filterung wird versucht, Grauwertstörungen im Bild zu eliminieren. Diese Grauwertstörungen können z.B. durch systematische Fehler bei der Aufzeichnung oder der Digitalisierung des Bildes verursacht werden. Ein anderes Anwendungsgebiet der Fouriertransformation und der Cosinustransformation ist die Bildcodierung. Hier wird z.B. untersucht, ob ein digitalisiertes Bild weniger Bandbreite bei der Übertragung benötigt, wenn man es anstatt der Grauwerte im Ortsbereich anhand der Transformationskoeffizienten (den Linearfaktoren der Fourier- oder Cosinustransformation) im Ortsfrequenzbereich beschreibt. Schließlich wird die Fouriertransformation auch bei der Charakterisierung von Oberflächenstrukturen oder der Beschreibung der Form eines Objektes verwendet.

Es folgen einige Grundlagenabschnitte, die den mathematischen Hintergrund der linearen Transformationen etwas beleuchten. Dann folgt die Darstellung der diskreten zweidimensionalen Fouriertransformation, wie sie in der Bildverarbeitung verwendet wird. Schließlich folgen Bildbeispiele zur digitalen Filterung im Frequenzbereich.

8.2 Lineare Approximation

In der Mathematik, vor allem in der Numerik, wird eine Funktion $f(x)$ häufig durch eine *Approximationsfunktion* (Ersatzfunktion) $\phi(x)$ angenähert. Dies ist notwendig, wenn $f(x)$ entweder mathematisch ungünstig zu handhaben ist (Berechnung der Funktionswerte, Integration) oder nur an diskreten Stützpunkten $(x_i, y_i = f(x_i))$ bekannt ist. Zur Approximation wird ein abgeschlossenes Intervall $[a, b]$ der reellen Zahlen betrachtet. Mit $C[a, b]$ wird die Menge aller stetigen Funktionen auf $[a, b]$ bezeichnet. Es soll $f(x) \in C[a, b]$ durch $\phi(x) \in C[a, b]$ angenähert werden. Die Approximationsfunktion $\phi(x)$ wird in der Regel von Parametern $c_0, c_1, ..., c_n$ abhängen, sodass auch $\phi(x) = \phi(x, c_0, c_1, ..., c_n)$ geschrieben

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_8

werden kann.

Wenn sich die Approximationsfunktion $\phi(x)$ in der Form

$$\phi(x) = \phi(x, c_0, c_1, ..., c_n) = c_0\varphi_0(x) + c_1\varphi_1(x) + ... + c_n\varphi_n(x) \tag{8.1}$$

darstellen lässt, so spricht man von einer *linearen Approximation*. Die Funktionen $\varphi_i(x), i = 0, 1, ..., n$ werden als ein *Funktionssystem* (ein *System von Basisfunktionen*) bezeichnet und die $c_i, i = 0, 1, ..., n$ als *Linearfaktoren*. Bekannte Beispiele sind die Darstellung von Funktionen als endliche oder unendliche Reihen, etwa:

$$f(x) = e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + ... = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + ... + \frac{1}{n!}x^n + R. \tag{8.2}$$

Hier ist das Funktionensystem

$$\varphi_0(x) = 1, \varphi_1(x) = x, \varphi_2(x) = x^2, ..., \varphi_n(x) = x^n. \tag{8.3}$$

Für die Linearfaktoren gilt:

$$c_0 = 1, c_1 = \frac{1}{1!}, c_2 = \frac{1}{2!}, ..., c_n = \frac{1}{n!}. \tag{8.4}$$

R ist ein Restglied, das den Fehler enthält der entsteht, wenn die an sich unendliche Reihenentwicklung zur praktischen Berechnung nach $n + 1$ Komponenten abgebrochen wird.

Die Linearfaktoren $c_i, i = 0, 1, ...n$ werden nach der Methode der *kontinuierlichen* oder *diskreten linearen Approximation im quadratischen Mittel* berechnet. Dies führt zu den Gauß'schen Normalgleichungen, einem System von linearen Gleichungen für die Berechnung [1] der c_i.

Zunächst zum stetigen Fall. Hier wird folgende Norm zugrunde gelegt:

$$\| g \|_2 = \left(\int_a^b w(x)g^2(x)dx \right)^{\frac{1}{2}}. \tag{8.5}$$

Dabei ist $g \in C[a, b]$ und $w(x) > 0$ eine auf $[a, b]$ integrierbare Gewichtsfunktion. Die Gauß'schen Normalgleichungen lauten dann:

$$\sum_{k=0}^{n} c_k \int_a^b w(x)\varphi_j(x)\varphi_k(x)dx = \int_a^b w(x)f(x)\varphi_j(x)dx, \quad j = 0, ..., n. \tag{8.6}$$

[1] Aus numerischen Gründen werden die Linearfaktoren in der Praxis jedoch nicht direkt aus den Gauß'schen Normalgleichungen berechnet, sondern über andere Methoden, z.B. die Householdertransformation.

Im diskreten Fall wird $\phi(x)$ als Approximation für eine Funktion $f(x) \in C[a,b]$ gesucht, die an $N+1$ vorgegebenen Stützpunkten $(x_i, y_i = f(x_i)), \quad i = 0, ..., N$ bekannt ist. Als Norm verwendet man hier:

$$\| g \|_2 = \left(\sum_{i=0}^{N} w_i g^2(x_i) \right)^{\frac{1}{2}}, \tag{8.7}$$

die $w_i > 0$ sind Gewichtskoeffizienten. Hier ergibt sich für die Gauß'schen Normalgleichungen:

$$\sum_{k=0}^{n} c_k \sum_{i=0}^{N} w_i \varphi_j(x_i) \varphi_k(x_i) = \sum_{i=0}^{N} w_i f(x_i) \varphi_j(x_i), \quad j = 0, ..., n, \quad N \geq n. \tag{8.8}$$

Von besonderem Interesse sind solche Systeme von Basisfunktionen, die *orthogonal* oder *orthonormiert* sind. Im kontinuierlichen Fall heißt $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$ ein orthogonales System, wenn gilt:

$$\int_a^b w(x) \varphi_j(x) \varphi_k(x) dx = 0, \text{ für } j, k = 0, 1, ..., n, j \neq k. \tag{8.9}$$

Das System heißt orthonormiert, wenn zusätzlich gilt:

$$\int_a^b w(x) \varphi_j(x) \varphi_j(x) dx = 1, \text{ für } j = 0, 1, ..., n. \tag{8.10}$$

Sinngemäß im diskreten Fall:

$$\sum_{i=0}^{N} w_i \varphi_j(x_i) \varphi_k(x_i) = 0, \text{ für } j, k = 0, 1, ..., n, j \neq k, \quad N \geq n \tag{8.11}$$

und

$$\sum_{i=0}^{N} w_i \varphi_j(x_i) \varphi_j(x_i) = 1, \text{ für } j = 0, 1, ..., n, \quad N \geq n. \tag{8.12}$$

Wenn man in (8.6) oder (8.8) ein Funktionssystem einsetzt, das orthogonal oder orthonormiert ist, gestaltet sich die Berechnung der Linearfaktoren über die Gauß'schen Normalgleichungen einfacher als bei einem nicht orthogonalen System.

Legt man nun eine Menge von Basisfunktionen fest, so kann man eine Funktion $f(x)$ auch durch die Linearfaktoren beschreiben, die in der Approximationsfunktion $\phi(x)$ auftreten. So könnte z.B. zwischen einem Coder (als Sender von Informationen) und einem Decoder (als Empfänger von Informationen) ein bestimmtes Basisfunktionssystem festgelegt werden. Der Coder berechnet die Linearfaktoren c_i der Approximationsfunktion $\phi(x)$

und überträgt sie dem Decoder. Dieser kann aufgrund der bekannten Basis die Approximationsfunktion $\phi(x)$ und damit näherungsweise $f(x)$ rekonstruieren. Beispiele dazu sind die Fourier- oder die Cosinustransformation (Abschnitte 8.4 und 8.9).

Es zeigt sich außerdem, dass manche Eigenschaften der Funktion $f(x)$ anhand der Linearfaktoren der Approximationsfunktion besser zu erkennen sind als in der Originalfunktion. So können z.B. störende Frequenzen eines zeitabhängigen Signals $f(t)$ nach einer Fouriertransformation erkannt und eliminiert (gefiltert) werden.

8.3 Trigonometrische Approximationsfunktionen

Periodische Funktionen mit der Periode 2π ($f(x + 2\pi) = f(x)$, der Übergang zu einer anderen Periode ist möglich) lassen sich durch ihre Fourierreihe darstellen:

$$f(x) = \frac{\alpha_0}{2} + \sum_{k=1}^{\infty} \Big(\alpha_k \cos(kx) + \beta_k \sin(kx)\Big). \tag{8.13}$$

Die Linearfaktoren (Fourierkoeffizienten) α_k und β_k berechnen sich mit den Euler'schen Formeln:

$$\alpha_k = \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x) \cos(kx) dx, k = 0, 1, ... \text{ und } \beta_k = \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x) \sin(kx) dx, k = 1, 2, ... \tag{8.14}$$

Zur praktischen Berechnung wird $f(x)$ durch ein endliches trigonometrisches Polynom $\phi(x)$ approximiert:

$$f(x) \approx \phi(x) = \frac{a_0}{2} + \sum_{k=1}^{n} \Big(a_k \cos(kx) + b_k \sin(kx)\Big). \tag{8.15}$$

Durch einen Vergleich lassen sich leicht die Basisfunktionen und die Linearfaktoren des trigonometrischen Polynoms mit der allgemeinen linearen Approximationsformel (8.1) in Beziehung setzen. Allerdings ist zu beachten, dass in (8.15) der Index n eine andere Bedeutung hat als in (8.1):

$$\begin{array}{ll} \varphi_0(x) = 1 & c_0 = \frac{a_0}{2} \\ \varphi_1(x) = \cos(x) & c_1 = a_1 \\ \varphi_2(x) = \sin(x) & c_2 = b_1 \\ \varphi_3(x) = \cos(2x) & c_3 = a_2 \\ \varphi_4(x) = \sin(2x) & c_4 = b_2 \\ ... & ... \end{array}$$

Es lässt sich zeigen, dass die Basisfunktionen des trigonometrischen Polynoms orthogonal sind. Die Linearfaktoren berechnen sich auch hier aus den Gauß'schen Normalgleichungen. Im Folgenden wird nur mehr der diskrete Fall behandelt. Dazu wird vorausgesetzt, dass von $f(x)$ die $2N$ äquidistanten Stützpunkte $(x_j, y_j = f(x_j))$ bekannt sind, mit $x_j = j(\pi/N)$.

Hier sind $2n+1$ unbekannte Parameter $a_0, a_1, ..., a_n, b_1, b_2, ..., b_n$ zu berechnen. Somit muss für die Anzahl der Stützpunkte gelten: $2n+1 \leq 2N$. Im Fall $2n+1 < 2N$ werden die Parameter wie oben nach der Methode Approximation im quadratischen Mittel berechnet. Es ergibt sich:

$$a_0 = \frac{1}{N}\sum_{j=1}^{2N} y_j, \quad a_k = \frac{1}{N}\sum_{j=1}^{2N} y_j \cos(kx_j), \quad b_k = \frac{1}{N}\sum_{j=1}^{2N} y_j \sin(kx_j), \tag{8.16}$$

für $k = 1, ..., n$.

Falls $n = N$ gilt, ergibt sich für $b_N = 0$. Es sind dann nur $2n$ Parameter zu berechnen und die Anzahl der Parameter ist gleich der Anzahl der Stützpunkte. In diesem Fall liegt eine trigonometrische Interpolation vor, die im Weiteren näher untersucht wird.

Das trigonometrische Interpolationspolynom lautet:

$$\phi(x) = \frac{a_0}{2} + \sum_{k=1}^{N-1} \Big(a_k \cos(kx) + b_k \sin(kx)\Big) + a_N \cos(Nx) \tag{8.17}$$

und für die Koeffizienten gilt:

$$\begin{aligned} a_0 &= \frac{1}{N}\sum_{j=1}^{2N} y_j, \quad a_N = \frac{1}{N}\sum_{j=1}^{2N} (-1)^j y_j, \\ a_k &= \frac{1}{N}\sum_{j=1}^{2N} y_j \cos(kx_j), \quad b_k = \frac{1}{N}\sum_{j=1}^{2N} y_j \sin(kx_j), k = 1, ..., N-1. \end{aligned} \tag{8.18}$$

Dieser Sachverhalt lässt sich auch in vektorieller Form schreiben. Es sei

$$\mathbf{f} = (f(x_1), f(x_2), ..., f(x_{2N}))^T \text{ und } \mathbf{\Phi} = (\phi(x_1), \phi(x_2), ..., \phi(x_{2N}))^T, \tag{8.19}$$

wobei wegen der Periodizität $x_0 = x_{2N}$ gilt. Da $\phi(x)$ die Funktion $f(x)$ interpoliert, gilt $\mathbf{f} = \mathbf{\Phi}$.

Nun lässt sich $\mathbf{f}$ als Linearkombination von $2N$ linear unabhängigen und orthogonalen Basisvektoren $\mathbf{v}_j, j = 0, ..., 2N-1$ schreiben:

$$\mathbf{f} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_{2N}) \end{pmatrix} = \frac{a_0}{2}\begin{pmatrix} 1 \\ 1 \\ ... \\ 1 \end{pmatrix} + a_1 \begin{pmatrix} \cos(x_1) \\ \cos(x_2) \\ ... \\ \cos(x_{2N}) \end{pmatrix} + b_1 \begin{pmatrix} \sin(x_1) \\ \sin(x_2) \\ ... \\ \sin(x_{2N}) \end{pmatrix} + \tag{8.20}$$

$$+ \quad a_2 \begin{pmatrix} \cos(2x_1) \\ \cos(2x_2) \\ ... \\ \cos(2x_{2N}) \end{pmatrix} + b_2 \begin{pmatrix} \sin(2x_1) \\ \sin(2x_2) \\ ... \\ \sin(2x_{2N}) \end{pmatrix} + \quad ... \quad + a_{N-1} \begin{pmatrix} \cos((N-1)x_1) \\ \cos((N-1)x_2) \\ ... \\ \cos((N-1)x_{2N}) \end{pmatrix} +$$

$$+ \quad b_{N-1}\begin{pmatrix} \sin((N-1)x_1) \\ \sin((N-1)x_2) \\ ... \\ \sin((N-1)x_{2N}) \end{pmatrix} + a_N \begin{pmatrix} \cos(Nx_1) \\ \cos(Nx_2) \\ ... \\ \cos(Nx_{2N}) \end{pmatrix} =$$

$$= \quad \frac{a_0}{2}\mathbf{v}_0 + a_1\mathbf{v}_1 + b_1\mathbf{v}_2 + ... + a_{N-1}\mathbf{v}_{2N-3} + b_{N-1}\mathbf{v}_{2N-2} + a_N\mathbf{v}_{2N-1}.$$

Aufbauend auf dieser Darstellung wird in Abschnitt 8.4 die diskrete zweidimensionale Fouriertransformation erläutert. An die Stelle des Vektors $\mathbf{f}$ tritt eine Bildmatrix $\mathbf{S} = (s(x,y))$ die im allgemeinen Fall auch komplex sein kann. Statt der Basisvektoren $\mathbf{v}_j$ treten Basismatrizen $\mathbf{B}_{(u,v)}$ auf.

8.4 Diskrete zweidimensionale Fouriertransformation

Mit $Q_{L,R}$ wird die Menge aller $L \cdot R$-Matrizen bezeichnet, wobei die Elemente einer Matrix auch komplexe Zahlen sein können. $Q_{L,R}$ wird zu einem linearen Raum über dem Körper C der komplexen Zahlen, wenn die Addition von zwei Matrizen $\mathbf{S}$ und $\mathbf{T}$ und die linearen Operationen mit Elementen a, b aus C definiert werden:

$$\begin{aligned} &\mathbf{S} + \mathbf{T} = \Big(s(x,y)\Big) + \Big(t(x,y)\Big) = \Big(s(x,y) + t(x,y)\Big). \\ &(ab)\mathbf{S} = a(b\mathbf{S}) = a\Big(b \cdot s(x,y)\Big) = \Big(ab \cdot s(x,y)\Big). \\ &(a+b)\mathbf{S} = a\mathbf{S} + b\mathbf{S}. \\ &a(\mathbf{S} + \mathbf{T}) = a\mathbf{S} + a\mathbf{T}. \end{aligned} \tag{8.21}$$

Schließlich wird noch ein Skalarprodukt $< \mathbf{S}, \mathbf{T} >$ definiert:

$$< \mathbf{S}, \mathbf{T} >= \frac{1}{L \cdot R} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s(x,y) \cdot t^*(x,y), \tag{8.22}$$

wobei mit $t^*(x,y)$ die konjugiert komplexe Zahl zu $t(x,y)$ bezeichnet wird. Damit ist $Q_{L,R}$ ein unitärer Raum über dem Körper C der komplexen Zahlen. Die Elemente $\mathbf{S}$ von $Q_{L,R}$ sind als Linearkombinationen von $M = L \cdot R$ linear unabhängigen Basismatrizen (einer Basis) darstellbar:

$$\begin{aligned} &\mathbf{B}_{u,v} = \Big(b(x,y)\Big)_{u,v}; \quad u = 0, ..., L-1; v = 0, ..., R-1. \\ &s(x,y) = \sum_{u=0}^{L-1} \sum_{v=0}^{R-1} f(u,v) b(x,y)_{u,v}; \quad x = 0, ..., L-1; y = 0, ..., R-1. \end{aligned} \tag{8.23}$$

Für eine orthonormierte Basis muss gelten:

$$< \mathbf{B}_{u_1,v_1}, \mathbf{B}_{u_2,v_2} >= 0, \text{ falls } (u_1,v_1) \neq (u_2,v_2) \text{ und} \\ < \mathbf{B}_{u_1,v_1}, \mathbf{B}_{u_2,v_2} >= 1, \text{ falls } (u_1,v_1) = (u_2,v_2). \tag{8.24}$$

Es lässt sich zeigen, dass die Basismatrizen

$$\mathbf{B}_{u,v} = \left(\exp\left(i2\pi(\frac{x \cdot u}{L} + \frac{y \cdot v}{R})\right)\right); \quad u = 0, ..., L-1; v = 0, ..., R-1 \tag{8.25}$$

linear unabhängig und orthonormiert sind. Somit kann eine $(L \cdot R)$-Matrix $\mathbf{S} = (s(x,y))$ als Linearkombination dieser Basismatrizen geschrieben werden:

$$s(x,y) = \sum_{u=0}^{L-1} \sum_{v=0}^{R-1} f(u,v) \exp\left(i2\pi(\frac{x \cdot u}{L} + \frac{y \cdot v}{R})\right); \\ x = 0, ..., L-1; y = 0, ..., R-1. \tag{8.26}$$

Da die Basismatrizen orthonormiert sind, berechnen sich die Linearfaktoren $f(u,v)$ gemäß:

$$f(u,v) = \frac{1}{L \cdot R} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} s(x,y) \exp\left(- i2\pi(\frac{x \cdot u}{L} + \frac{y \cdot v}{R})\right); \\ u = 0, ..., v-1; y = 0, ..., R-1. \tag{8.27}$$

Die in (8.25) definierten Matrizen sind die Basis der *diskreten, zweidimensionalen Fouriertransformation*, (8.27) wird als *diskrete, zweidimensionale Fouriertransformation* und (8.26) als *inverse, diskrete, zweidimensionale Fouriertransformation* bezeichnet. $\mathbf{S} = (s(x,y))$ und $\mathbf{F} = (f(u,v))$ sind ein *Fouriertransformationspaar* und $\mathbf{F}$ heißt die *Fouriertransformierte* von $\mathbf{S}$.

Es ist nun der Bezug zu digitalisierten Bildern herzustellen. Eine Bildmatrix $\mathbf{S} = (s(x,y))$ mit L Bildzeilen und R Bildspalten ist ein Element der oben definierten Menge $Q_{L,R}$. Damit können auf Bilder die Transformationen (8.27) und (8.26) angewendet werden. Wenn man die obige Ableitung der diskreten, zweidimensionalen Fouriertransformation nicht über einem unitären Raum von $L \cdot R$ Matrizen, sondern über einem unitären Raum von periodischen Funktionen $s(x,y)$ der diskreten Variablen x und y durchführt, stellt man sich das Bild $\mathbf{S}$, wie in Kapitel 5, Bild 5.2 gezeigt, periodisch in $x-$ und $y-$Richtung fortgesetzt vor. Auch die Fouriertransformierte $f(u,v)$ ist dann eine periodische Funktion der diskreten Varibalen u und v, die dann als *Ortsfrequenzen* bezeichnet werden.

Die Fouriertransformierte von $\mathbf{S}$ ist in der Regel komplex: $\mathbf{F} = \mathbf{P} + i\mathbf{T}$, wobei $\mathbf{P} = (p(u,v))$ und $\mathbf{T} = (t(u,v))$ der Realteil und der Imaginärteil von $\mathbf{F} = (f(u,v))$ sind. Die Beträge

$$\Big(\mid f(u,v) \mid \Big) = \Big(\sqrt{p^2(u,v) + t^2(u,v)} \Big) \tag{8.28}$$

werden als *Fourierspektrum* oder *Spektrum der Ortsfrequenzen* von $\mathbf{S}$ bezeichnet. Die höchste auf einem bestimmten Trägermedium darstellbare Ortsfrequenz bezeichnet man auch als *Auflösung*. Man versteht darunter die durch die physikalischen Eigenschaften des Trägermediums bedingte dichteste Folge von noch unterscheidbaren hellen und dunklen Linien.

Für die qualitative Beurteilung der in einem Bild $\mathbf{S} = (s(x,y))$ vorhandenen Ortsfrequenzen wird für das Spektrum ($\mid f(u,v) \mid$) häufig eine bildhafte Darstellung angestrebt. Die Grauwerte werden z.B. so gewählt, dass sie zum Betrag $\mid f(u,v) \mid$ der jeweiligen, komplexen Spektralkomponente $f(u,v)$ proportional sind. Sollen in der bildlichen Darstellung Spektralkomponenten mit kleinem Betrag stärker betont werden als solche mit großem Betrag, so können die Grauwerte $p(u,v)$ einer bildlichen Darstellung von $\mid f(u,v) \mid$ berechnet werden gemäß

$$p(u,v) = c \cdot \log \Big(1 + \mid f(u,v) \mid \Big), \tag{8.29}$$

wobei c ein Normierungsfaktor ist, der so gewählt werden muss, dass die Grauwerte in der Grauwertmenge G liegen. Bild 8.1 zeigt ein Testbild und daneben die bildliche Darstellung des Spektrums. Andere diskrete (unitäre) Transformationen von $\mathbf{S} = (s(x,y))$ erhält man, wenn die Basismatrizen in (8.25) anders gewählt werden. Beispiele dazu sind die Cosinus-, die Hadamar- oder die Haartransformation.

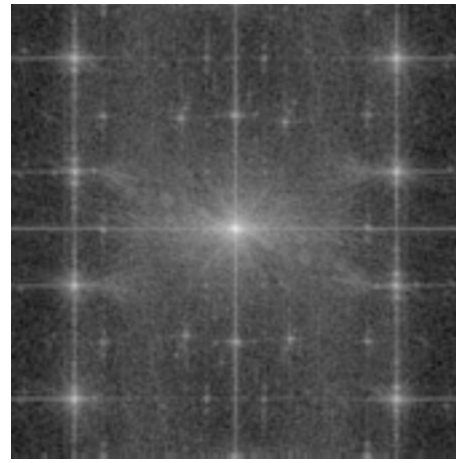
8.5 Digitale Filterung im Ortsfrequenzbereich

Eine digitale Filterung im Bereich der Ortsfrequenzen eines Bildes $\mathbf{S}_e$ besteht aus der Veränderung der Fouriertransformierten $(f_e(u,v))$ durch die Multiplikation mit einer Übertragungsfunktion $g(u,v)$. Man erhält dann die gefilterte Fouriertransformierte $f_a(u,v)$ des Ausgabebildes $\mathbf{S}_a$:

$$f_a(u,v) = g(u,v) \cdot f_e(u,v); u = 0, ..., L-1; v = 0, ..., R-1. \tag{8.30}$$

Je nach der Beschaffenheit der Übertragungsfunktion $g(u,v)$ spricht man von verschiedenen Filtertypen. Werden durch die Übertragungsfunktion hohe Ortsfrequenzanteile abgeschwächt oder sogar unterdrückt, so spricht man von einer *Tiefpassfilterung*. Im Gegensatz dazu werden bei einer *Hochpassfilterung* niedere Ortsfrequenzanteile abgeschwächt oder unterdrückt. Bei *Bandpassfiltern* wird die Übertragungfunktion so gewählt, dass nur die Ortsfrequenzanteile eines bestimmten Ausschnittes unverändert bleiben, während bei der

(a)



(b)

Bild 8.1: Testbild zur Fouriertransformation. (a) Original. (b) Bildliche Darstellung des Spektrums.

Bandsperre die Ortsfrequenzanteile nur des Ausschnittes unterdrückt werden. Das gefilterte Bild $\mathbf{S}_a$ im Ortsbereich des (x, y)-Koordinatensystems berechnet sich mit der inversen Fouriertransformation gemäß (8.26):

$$s_a(x,y) = \sum_{u=0}^{L-1} \sum_{v=0}^{R-1} f_a(u,v) \exp\left(i2\pi(\frac{x \cdot u}{L} + \frac{y \cdot v}{R})\right); \qquad (8.31)$$
$$x = 0, ..., L-1; y = 0, ..., R-1.$$

Da bei einem Tiefpassfilter hohe Ortsfrequenzanteile abgeschwächt werden, erscheint das gefilterte Bild $\mathbf{S}_a$ im Vergleich zum Original unschärfer. Die hohen Ortsfrequenzen treten im Originalbild $\mathbf{S}_e$ vor allem in den Grauwertkanten auf. Werden durch ein Hochpassfilter die niederen Ortsfrequenzen unterdrückt, so werden im gefilterten Bild $\mathbf{S}_a$ vor allem die Grauwertkanten deutlich hervortreten. An dieser Stelle sei auf den Zusammenhang zwischen der Tiefpassfilterung im Ortsfrequenzbereich mit den Summenoperatoren von Abschnitt 5.3 einerseits und zwischen der Hochpassfilterung im Frequenzbereich und den Differenzenoperatoren von Abschnitt 5.4 hingewiesen. Dieser Zusammenhang wird im folgenden Abschnitt noch näher erläutert.

Zunächst aber noch zwei Beispiele: Eine Übertragungsfunktion mit Tiefpass- und eine mit Hochpasscharakter. Für das Folgende wird angenommen, dass das Bild $\mathbf{S}_e = (s_e(x, y))$ quadratisch ist, also $L = R = N$. Es lässt sich zeigen, dass die Fouriertransformierte von $\mathbf{S}_e$

punktsymmetrisch zu $(N/2, N/2)$ ist, wenn vor der Fouriertransformation die Originalwerte mit $(-1)^{(x+y)}$ multipliziert werden. Diese Form wird auch bei der bildlichen Darstellung der Beträge Fouriertransformierten (des Betragsspektrums) gewählt.

Ein Tiefpassfilter ist dann z.B.:

$$g_T(u,v) = \frac{1}{2} + \frac{1}{2}\left\{ \cos\left(\frac{2\pi}{N\sqrt{2}} \sqrt{(u-\frac{N}{2})^2 + (v-\frac{N}{2})^2} \right) \right\}; \quad (8.32)$$
$$u = 0, ..., N-1; v = 0, ..., N-1$$

und ein Hochpassfilter z.B.:

$$g_T(u,v) = \frac{1}{2} - \frac{1}{2}\left\{ \cos\left(\frac{2\pi}{N\sqrt{2}} \sqrt{(u-\frac{N}{2})^2 + (v-\frac{N}{2})^2} \right) \right\}; \quad (8.33)$$
$$u = 0, ..., N-1; v = 0, ..., N-1.$$

Zu weiteren Fragen über den Entwurf von digitalen Filtern im Frequenzbereich, also zur Berechnung von geeigneten Übertragungsfunktionen, sei z.B. auf [Abma94] verwiesen.

Auf den folgenden Seiten sind als Abschluss zu diesem Abschnitt Bildbeispiele zu verschiedenen Filterungen zusammengestellt. Sie zeigen jeweils das Original, daneben das jeweils verwendete Filter und rechts oben das Ergebnis der Filterung. Unter dem Original ist die bildliche Darstellung der Beträge der Fouriertransformierten zu sehen und unter dem Ergebnis die bildliche Darstellung der Beträge der gefilterten Fouriertransformierten.

8.6 Zusammenhang mit dem Ortsbereich

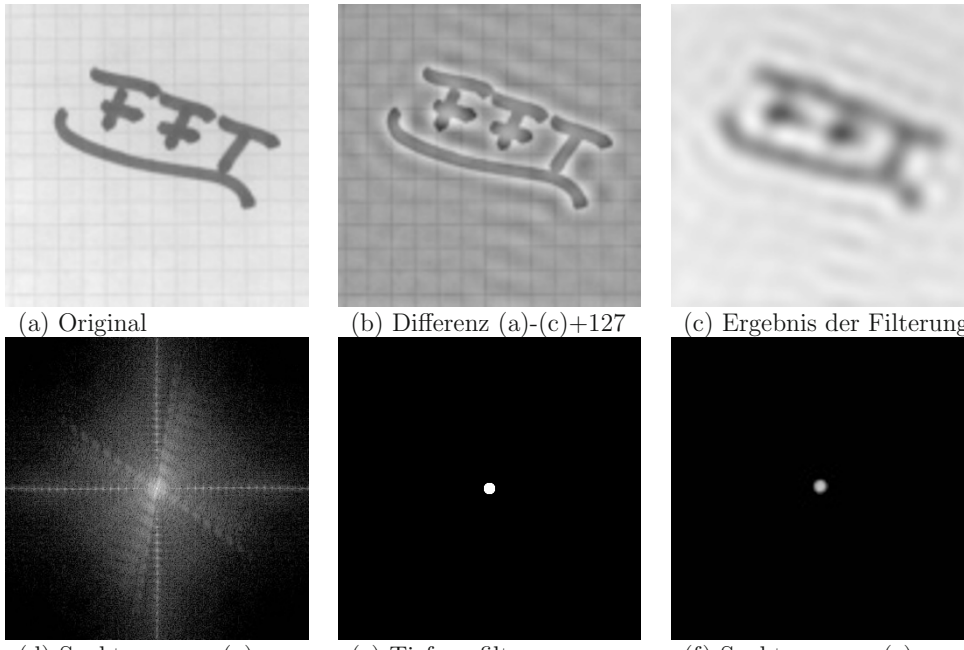
Der Zusammenhang zwischen Ortsfrequenzbereich und Ortsbereich wird durch den *Faltungssatz* vermittelt. Die diskrete Faltung zweier Funktionen $s(x,y)$ und $h(x,y)$ ist definiert als

$$\sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} s(k,l) \cdot h(x-k, y-l) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} s(x-k, y-l) \cdot h(k,l); \quad (8.34)$$
$$x, y = ..., -2, -1, 0, 1, 2...$$

Abkürzend wird dafür oft $s(x,y) \star h(x,y)$ geschrieben. Im Folgenden wird mit FT bzw. mit FT^{-1} die diskrete, bzw. die inverse, diskrete, zweidimensionale Fouriertransformation bezeichnet. Dann lautet der Faltungssatz:

$$\mathrm{FT}\Big(s(x,y)\Big) \cdot \mathrm{FT}\Big(h(x,y)\Big) = \mathrm{FT}\Big(s(x,y) \star h(x,y)\Big). \quad (8.35)$$

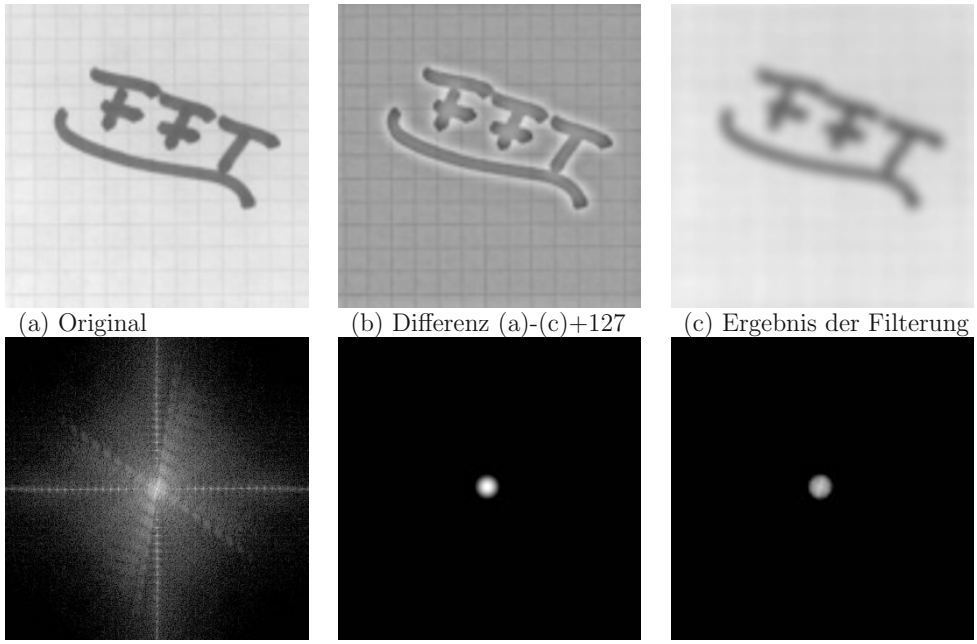
Das bedeutet, dass das Produkt der Fouriertransformierten von $s(x,y)$ und $h(x,y)$ gleich der Fouriertransformierten der Faltung dieser beiden Funktionen ist. In (8.30) und



(a) Original (b) Differenz (a)-(c)+127 (c) Ergebnis der Filterung

(d) Spektrum von (a) (e) Tiefpassfilter (f) Spektrum von (c)

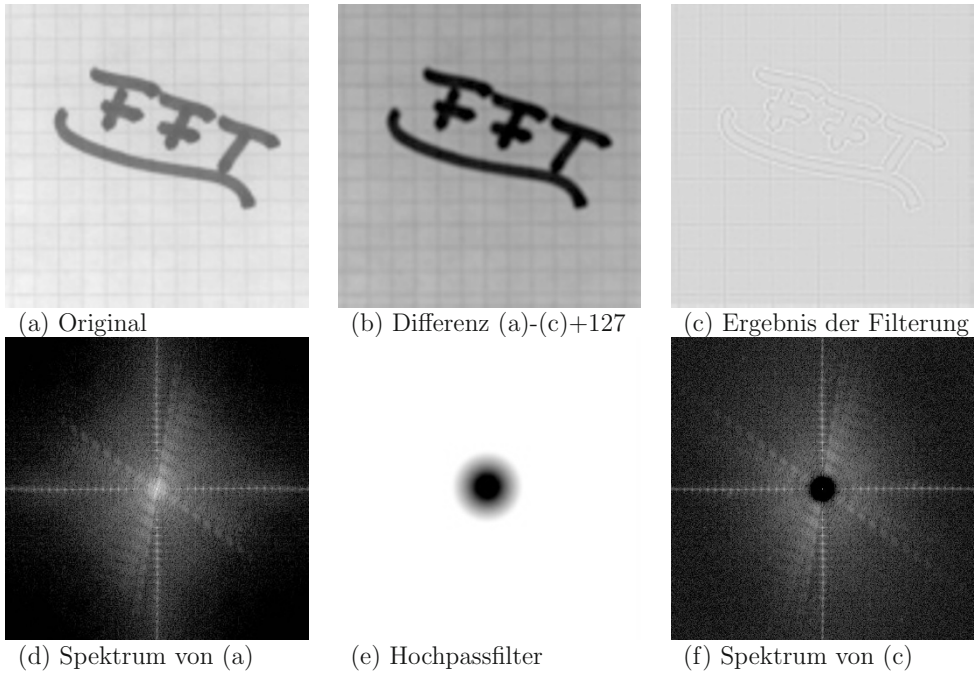
Bild 8.2: Beispiel zu einer Tiefpassfilterung. (a) Original. (b) Differenzbild: (Original - gefiltertes Bild + 127). (c) Ergebnis der Filterung: Das Bild ist deutlich unschärfer geworden. Die Wellenstruktur im Hintergrund wird durch das scharfkantige. Filter erzeugt. (d) Bildliche Darstellung des Spektrums des Originals. (e) Verwendetes Tiefpassfilter: Die Frequenzanteile im schwarzen Bereich werden auf null gesetzt, die im weißen Bereich werden unverändert übernommen. (f) Bildliche Darstellung des gefilterten Spektrums.



(a) Original (b) Differenz (a)-(c)+127 (c) Ergebnis der Filterung

(d) Spektrum von (a) (e) Tiefpassfilter verlaufend (f) Spektrum von (c)

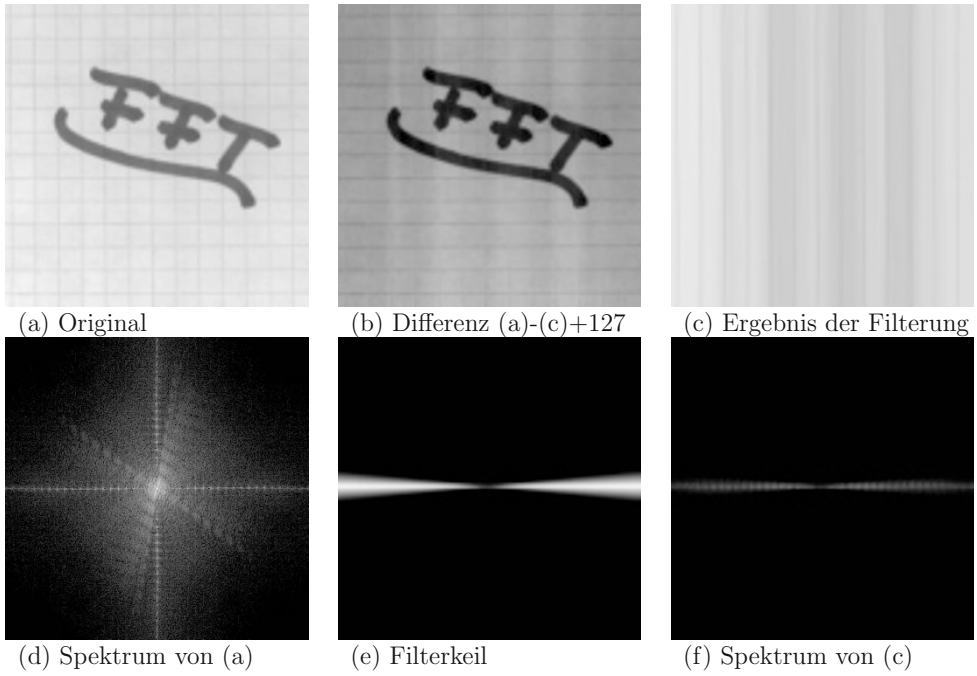
Bild 8.3: Beispiel zu einer Tiefpassfilterung. (a) Original. (b) Differenzbild: (Original - gefiltertes Bild + 127). (c) Ergebnis der Filterung: Das Bild ist deutlich unschärfer geworden. Die Wellenstruktur im Hintergrund tritt hier wegen der Verwendung eines verlaufenden Filters nicht auf. (d) Bildliche Darstellung des Spektrums des Originals. (e) Verwendetes Tiefpassfilter: Die Frequenzanteile im schwarzen Bereich werden auf null gesetzt, die im weißen Bereich werden unverändert übernommen, die Anteile im Übergangsbereich werden abgedämpft. (f) Bildliche Darstellung des gefilterten Spektrums.



(a) Original (b) Differenz (a)-(c)+127 (c) Ergebnis der Filterung

(d) Spektrum von (a) (e) Hochpassfilter (f) Spektrum von (c)

Bild 8.4: Beispiel zu einer Hochpassfilterung. (a) Original. (b) Differenzbild: (Original - gefiltertes Bild + 127). (c) Ergebnis der Filterung: Es sind im Wesentlichen nur mehr die Grauwertübergänge zu sehen, also die Bildbereiche mit hohen Ortsfrequenzen. (d) Bildliche Darstellung des Spektrums des Originals. (e) Verwendetes Hochpassfilter: Die Frequenzanteile im schwarzen Bereich werden auf null gesetzt, die im weißen Bereich werden unverändert übernommen, die Anteile im Übergangsbereich werden abgedämpft. (f) Bildliche Darstellung des gefilterten Spektrums.



(a) Original (b) Differenz (a)-(c)+127 (c) Ergebnis der Filterung

(d) Spektrum von (a) (e) Filterkeil (f) Spektrum von (c)

Bild 8.5: Beispiel zu einer Filterung mit einem keilförmigen Filter. (a) Original. (b) Differenzbild: (Original - gefiltertes Bild + 127). (c) Ergebnis der Filterung: Es sind im Wesentlichen nur mehr die vertikalen Linien zu sehen. (d) Bildliche Darstellung des Spektrums des Originals. (e) Verwendetes Keilfilter: Die Frequenzanteile im schwarzen Bereich werden auf null gesetzt, die im weißen Bereich werden unverändert übernommen, die Anteile im Übergangsbereich werden abgedämpft. (f) Bildliche Darstellung des gefilterten Spektrums.

Ortsbereich		Ortsfrequenzbereich
$s(x,y)$	FT $\longrightarrow$	$f(u,v)$
$\downarrow$		$\downarrow$
$s \star h$ Faltung mit der Implusantwort	$h \rightarrow \mathrm{FT} \rightarrow g$	$f \cdot g$ Multiplikation mit der Übertragungsfunktion
$\downarrow$	$h \leftarrow \mathrm{FT}^{-1} \leftarrow g$	$\downarrow$
$s'(x,y)$ Ergebnis der Filterung	FT^{-1} $\longleftarrow$	$f'(u,v)$ gefilterte Fourier-transformierte

Bild 8.6: Zusammenhänge zwischen dem Ortsbereich und dem Ortsfrequenzbereich.

(8.31) wurde dargestellt, dass die digitale Filterung im Ortsfrequenzbereich durch die Multiplikation der Fouriertransformierten $f(u,v)$ von $s(x,y)$ mit einer Übertragungsfunktion $g(u,v)$ und die anschließende inverse Fouriertransformation erreicht werden kann. Aus dem Faltungssatz kann der Bezug zum Ortsbereich abgeleitet werden: Die Filterung kann auch im Ortsbereich durchgeführt werden, wenn $s(x,y)$ mit einer Impulsantwortfunktion $h(x,y)$ gefaltet wird. Die Impulsantwort- und die Übertragungsfunktion bilden dabei ein Fouriertransformationspaar. Diese Zusammenhänge sind in Bild 8.6 grafisch dargestellt.

Digitalisierte Bilder, interpretiert als Funktionen der diskreten Variablen x und y, sind nur in einem endlichen Rechteckbereich mit der Seitenlänge $L \cdot R$ definiert. Außerhalb dieses Bereichs kann man sich vorstellen, dass sie nur den Wert Null annehmen. Dadurch reduzieren sich die unendlichen Summen auf endliche und (8.34) lautet dann:

$$\sum_{k=0}^{L-1} \sum_{l=0}^{R-1} s(x-k, y-l) \cdot h(k,l). \tag{8.36}$$

Bei praktischen Anwendungen wird die Impulsantwort auf kleinere Bereiche als das Gesamtbild begrenzt, z.B. auf $3 \cdot 3$ Bildpunkte. Der Vergleich mit (5.8) zeigt dann, dass man die in Kapitel 5 erläuterten Summen- oder Differenzenoperatoren erhält.

8.7 Logarithmische Filterung

Die logarithmische Filterung ist ein Beispiel einer digitalen Filterung im Orts- oder Frequenzbereich, verbunden mit einer nicht linearen Transformation der Grauwerte. Ein Anwendungsbeispiel ist die Elimination von ungleichmäßiger Beleuchtung, ohne dabei auf die Bildqualität bei feinen Details verzichten zu müssen.

In Bildern treten unterschiedliche Einflüsse der beleuchtenden Lichtquellen bei der Aufnahme auf, so z.B. Schatteneffekte oder Reflexionen. Die Reflexionen sind dabei oft erwünscht, da sie Details oft deutlicher hervorheben, während sich die Schatten aufgrund von unterschiedlicher Beleuchtung in der Regel nicht positiv auswirken. Die Unterschiede in der Beleuchtung treten in einem Bild meistens nicht so abrupt auf wie die Lichtreflexionen. Das bedeutet also vor dem Hintergrund der Fouriertransformation, dass sich die Lichtreflexionen mehr in den hohen Frequenzanteilen und die Beleuchtungseffekte mehr in den niederen Frequenzanteilen ausdrücken. Um nun solche Beleuchtungseffekte zu eliminieren oder zu mindestens abzudämpfen, ist eine Filterung mit Betonung der hohen Frequenzanteile angebracht.

Es wird angenommen, dass der Beleuchtungs- und der Reflexionsanteil sich multiplikativ zum Grauwert eines Bildpunktes ergänzen:

$$s(x,y) = s_B(x,y) \cdot s_R(x,y), \tag{8.37}$$

wobei $s_B(x,y)$ der Beleuchtungsanteil und $s_R(x,y)$ der Reflexionsanteil ist. Durch Logarithmieren von (8.37) erhält man:

$$\log s(x,y) = \log s_B(x,y) + \log s_R(x,y). \tag{8.38}$$

Da die Fouriertransformation eine lineare Operation ist, können die Summanden in (8.38) gezielt durch digitale Filter beeinflusst werden. Man wird hier also ein Filter im Orts- oder Frequenzbereich einsetzen, das die hohen Bildfrequenzen beeinflusst. Durch die Logarithmierung werden außerdem die hellen Bildteile mehr angesprochen als die dunklen.

Praktisch sind also folgende Schritte durchzuführen:

- nicht lineare Skalierung durch die Logarithmierung der Grauwerte,
- digitale Filterung mit Betonung hoher Bildfrequenzen,
- Rücktransformation in eine lineare Grauskala durch Exponentiation.

8.8 Inverse und Wiener Filterung

Bei der inversen Filterung wird angenommem, dass ein Bild $\mathbf{S}_a = (s_a(x,y))$ durch eine Überlagerung gestört ist, deren Impulsantwort- bzw. Übertragungsfunktion bekannt ist:

$$s_e(x,y) = \frac{1}{L \cdot R} \sum_{k=0}^{L-1} \sum_{l=0}^{R-1} s_a(x-k, y-l) \cdot h(k,l). \tag{8.39}$$

Im Frequenzbereich gilt dann sinngemäß:

$$f_e(u,v) = f_a(u,v) \cdot g(u,v), \tag{8.40}$$

wobei $f_e(u,v)$, $f_a(u,v)$ und $g(u,v)$ die Fouriertransformierten des gestörten Bildes $(s_e(x,y))$, des korrekten Bildes $(s_a(x,y))$ und der Impulsantwortfunktion $h(x,y)$ sind.

Aus (8.40) folgt, dass rein rechnerisch die Fouriertransformierte des korrekten Bildes $(s_a(x,y))$ aus dem Quotienten von $f_e(u,v)$ und $g(u,v)$ gewonnen werden kann:

$$f_a(u,v) = \frac{f_e(u,v)}{g(u,v)} \tag{8.41}$$

und $s_a(x,y)$ durch inverse Fouriertransformation in den Ortsbereich.

Bei praktischen Anwendungen von (8.41) treten jedoch Schwierigkeiten auf und zwar an den Stellen, an denen $g(u,v)$ den Wert null annimmt. Um dies zu umgehen, wird statt des korrekten inversen Filters $1/g(u,v)$ eine Näherung $m(u,v)$ verwendet, die nur diejenigen Frequenzen rekonstruiert, die ein hohes Signal-Rauschverhältnis haben. Dabei wird $m(u,v)$ so gewählt, dass an Stellen, an denen $g(u,v) \neq 0$ ist, das exakte Filter $1/g(u,v)$ verwendet wird.

Bei der Wiener Filterung wird der Fehler zwischen dem korrekten und dem gestörten Bild minimiert und zwar durch einen statistischen Ansatz. Das zu berechnende Bild $(s_a(x,y))$ wird als stochastischer Prozess beschrieben, der durch die Überlagerung der stochastischen Prozesse des korrekten Bildes $(s_a(x,y))$ mit einem Rauschanteil $(z(x,y))$ zustande kommt. Das Bild $(s_a(x,y))$ wird dabei so bestimmt, dass Standardabweichungen der Differenz des korrekten Bildes und des berechneten Bildes minimiert werden.

8.9 Diskrete, zweidimensionale Cosinustransformation

Die unitären (orthogonalen) Transformationen, zu denen auch die Fouriertransformation gehört, haben Vor- und Nachteile, abhängig vom jeweiligen Einsatzgebiet. In der *Bilddatencodierung* sucht man orthogonale Transformationen, bei denen sich die Transformationskoeffizienten möglichst dicht zusammen konzentrieren. Außerdem soll durch das Weglassen von Transformationskoeffizienten, die im Frequenzspektrum ,,weiter weg liegen" die Bildqualität nach der Rücktransformation nur gering verschlechtert werden.

Als erstes kann die *Hadamartransformation* erwähnt werden: Sie ist eine orthogonale lineare Transformation, die auf den Hadamar-Matrizen als Basis-Matrizen aufbaut. Hadamar-Matrizen sind quadratisch und haben in den Zeilen und Spalten nur die Werte 1 oder -1 (ein gemeinsamer Faktor wird vor die Matrix geschrieben). Hadamar-Matrizen haben die Eigenschaft $\mathbf{H}\mathbf{H}^T = \mathbf{I}$. Die kleinste Hadamartransformations-Matrix ist die $(2 \cdot 2)$-Matrix $\mathbf{H}_2$. Es gilt: Falls eine Hadamar-Matrix der Größe $(L \cdot L)$ existiert, wobei L

eine Potenz von 2 ist, so gibt es auch eine der Größe $(2L \cdot 2L)$, die nach folgender Vorschrift erhalten wird:

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{8.42}$$

$$\mathbf{H}_{2L} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{H}_L & \mathbf{H}_L \\ \mathbf{H}_L & -\mathbf{H}_L \end{pmatrix}, \tag{8.43}$$

Ein Beispiel:

$$\mathbf{H}_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \tag{8.44}$$

Ob für beliebige Werte für L Hadamar-Matrizen existieren, ist nicht bewiesen. Es gibt aber für fast alle Werte bis $L = 200$ Konstruktionsmöglichkeiten. Die Hadamartransformation führt multidimensionale 45°-Drehungen durch. Die Annahme, dass diese Drehungen für die Zwecke der Bilddatencodierung immer passend sind, trifft häufig nicht zu.

Die *Hauptkomponententransformation* (Abschnitt 13.4) (*Karhunen-Loeve-Transformation*) liefert zwar immer optimale Ergebnisse, da die Transformationsmatrizen und damit die Drehwinkel aus dem Bild ermittelt werden. Aber die dazu notwendige Berechnung der Kovarianzmatrix, der Eigenwerte und der Eigenvektoren ist für eine schnelle Bilddatencodierung zu aufwendig.

Die Fouriertransformation liefert ein anschaulich interpretierbares Spektrum eines Bildes, das bei anderen Anwendungen gut zu verwenden ist. Ein wesentlicher Nachteil für die Bilddatencodierung sind die komplexen Transformationskoeffizienten. Ein weiterer Nachteil ist die schlechte Konvergenz bei der Rekonstruktion des Bildes, wenn hochfrequente Transformationskoeffizienten weggelassen werden. Dies erklärt sich aus der periodischen Fortsetzung des Eingabebildes, die bei der Fouriertransformation angenommen werden muss.

Wenn die Eingabefunktion reell und symmetrisch ist, sind die Koeffizienten der Fouriertransformation reell. Sie resultieren aus den Cosinus-Anteilen der Basisfunktionen. Man erhält somit die *Cosinustransformation*, wenn man die Eingabefunktion reell wählt und sie symmetrisch fortsetzt. Theoretische und praktische Untersuchungen haben gezeigt, dass sie für die Bilddatencodierung gut geeignet ist.

Die Cosinustransformation lautet:

$$f(u,v) = \frac{2}{L} c(u)c(v) \sum_{x=0}^{L-1} \sum_{y=0}^{L-1} s(x,y) \cos\Big(\frac{\pi(2x+1)}{2L}u\Big) \cos\Big(\frac{\pi(2y+1)}{2L}v\Big) \tag{8.45}$$

$$s(x,y) = \frac{2}{L} \sum_{u=0}^{L-1} \sum_{v=0}^{L-1} c(u)c(v) f(u,v) \cos\Big(\frac{\pi(2x+1)}{2L}u\Big) \cos\Big(\frac{\pi(2y+1)}{2L}v\Big), \tag{8.46}$$

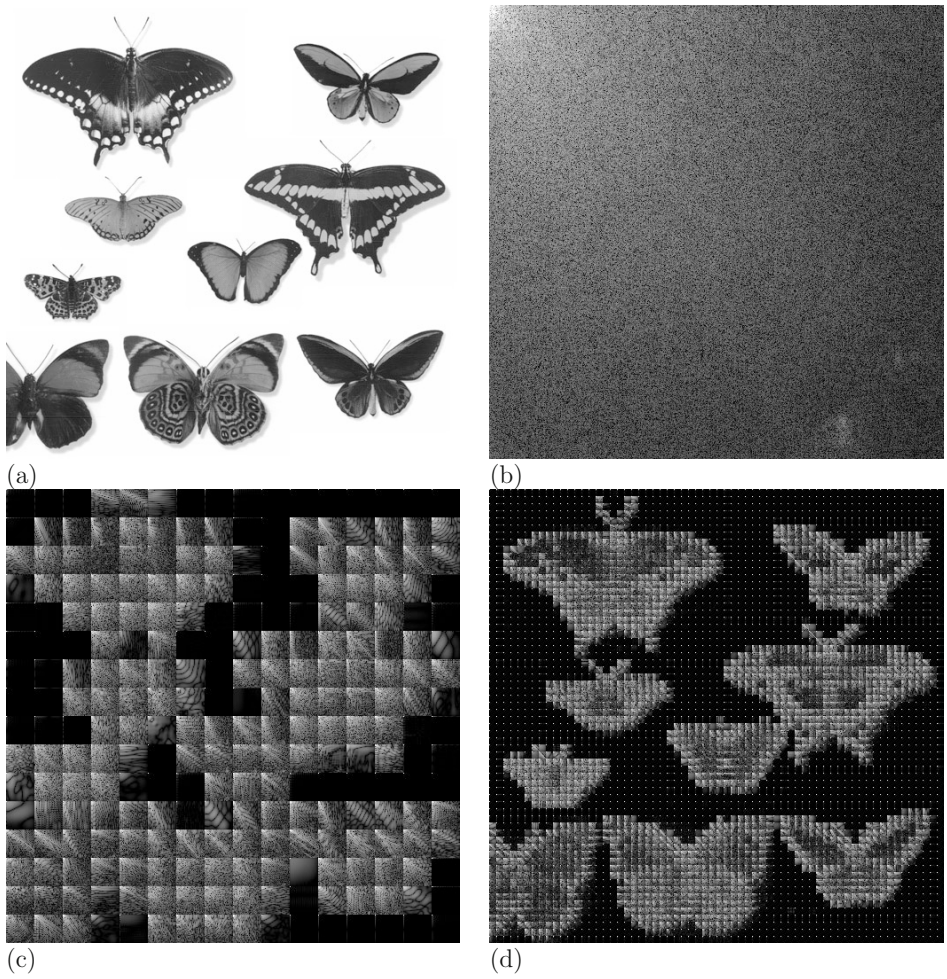


Bild 8.7: Beispiele zur Cosinustransformation. (a) Original. (b) Bildliche Darstellung des Spektrums. Die Cosinustransformation wurde über das gesamte Bild erstreckt (ein $512 \cdot 512$ Block). (c) Bildliche Darstellung des Spektrums. Die Cosinustransformation wurde über $32 \cdot 32$ Blöcke erstreckt. (d) Bildliche Darstellung des Spektrums. Die Cosinustransformation wurde über $8 \cdot 8$ Blöcke durchgeführt. Das entspricht der Vorgehensweise beim JPEG-Verfahren.

wobei $c(0) = \frac{1}{\sqrt{2}}$ und $c(k) = 1, k = 1, 2, ..., L-1$.

In der Bilddatencodierung wird die Cosinustransformation z.B. im JPEG-Verfahren eingesetzt. Hier werden jeweils Blöcke von $8 \cdot 8$ Bildpunkten transformiert. Die Transformationskoeffizienten der Cosinustransformation werden dann quantisiert und mit *run length Codierung* (Abschnitt 23.2) weiter verdichtet. Wenn man die Cosinustransformation auf $8 \cdot 8$-Blöcke beschränkt, ergeben sich aus (8.45) und (8.46) folgende Spezialisierungen:

$$f(u,v) = \frac{c(u)c(v)}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} s(x,y) \cos\Big(\frac{\pi(2x+1)}{16}u\Big) \cos\Big(\frac{\pi(2y+1)}{16}v\Big) \tag{8.47}$$

$$s(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c(u)c(v)f(u,v) \cos\Big(\frac{\pi(2x+1)}{16}u\Big) \cos\Big(\frac{\pi(2y+1)}{16}v\Big), \tag{8.48}$$

wobei wie oben $c(0) = \frac{1}{\sqrt{2}}$ und $c(k) = 1, k = 1, 2, ..., 7$.

Die Bilder 8.7 zeigen Beispiele zur Cosinustransformation. Bild 8.7-a ist das Original ($512 \cdot 512$ Bildpunkte). Bild 8.7-b zeigt die als Grauwerte codierten Transformationskoeffizienten, wenn die Cosinustransformation über das gesamte Bild erstreckt wird (ein $512 \cdot 512$ Block). Bei den beiden folgenden Teilbildern 8.7-c und 8.7-d wurden Blöcke der Größe $32 \cdot 32$ und $8 \cdot 8$ verwendet.

Kapitel 9

Modifikation der Ortskoordinaten

9.1 Anwendungen

Geometrische Transformationen werden bei vielen Problemkreisen der digitalen Bildverarbeitung und Mustererkennung benötigt. Einfache Operationen sind das Vergrößern, das Verkleinern, das Verschieben oder das Drehen von Bildern. Auch Maßstabsänderungen, evtl. unterschiedlich in Zeilen- und in Spaltenrichtung, sind möglich. Etwas allgemeiner sind die affinen Transformationen von Bilddaten.

Eine klassische Anwendung ist die geometrische Entzerrung von Luft- und Satellitenbildern, so dass sie mit Kartenkoordinatensystemen (z.B. UTM oder Gauß-Krüger) deckungsgleich sind, um so z.B. thematische Karten erzeugen zu können. Ein anderes Beispiel in dieser Richtung ist die geometrische Angleichung von zwei Bildern des gleichen Beobachtungsgebiets, die jedoch zu unterschiedlichen Zeitpunkten und damit von unterschiedlichen Standorten aufgezeichnet wurden. Als ähnliche Anwendung ist die Elimination von Bildverzerrungen, die durch den Einfluss der verwendeten Aufnahmetechnik bedingt sind (Optik, Abtastart), zu erwähnen.

Bei anderen Anwendungen werden Bildausschnitte gezielt geometrisch verändert und so an andere Bildinhalte angeglichen. Diese Techniken werden auch als *morphing* bezeichnet.

9.2 Grundlegende Problemstellung

Bei allen bisher besprochenen Verfahren wurde die Anordnung der Bildpunkte in der Bildmatrix $\mathbf{S}$ nicht geändert, d.h. die Ortskoordinaten (x, y) der Bildpunkte wurden von den Operationen nicht betroffen. Bei vielen Bildern besteht aber die Notwendigkeit, sie vor weiteren Verarbeitungsschritten einer Transformation der Ortskoordinaten zu unterziehen. Das ist z.B. der Fall, wenn zwei Bilder desselben Objekts, die jedoch unter verschiedenen Aufnahmebedingungen gemacht wurden, zueinander in Beziehung gesetzt werden sollen. Zur Gewährleistung der Deckungsgleichheit ist hier dann in der Regel eine Anpassung der Ortskoordinaten der beiden Bilder notwendig.

Die grundsätzliche Problemstellung kann wie folgt beschrieben werden: Es sei $\mathbf{S}_e =$

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_9

$(s_e(x_e, y_e))$ die Bildmatrix des Originalbildes mit den Ortskoordinaten x_e und y_e. Eine Transformation der Ortskoordinaten $\mathbf{S}_e \rightarrow \mathbf{S}_a$ wird beschrieben durch

$$s_a(x_a, y_a) = s_e(x_e, y_e), \tag{9.1}$$

wobei

$$x_a = f_1(x_e, y_e) \text{ und } y_a = f_2(x_e, y_e) \tag{9.2}$$

die *Abbildungen der Transformation* (*mapping functions*) sind. Eine Transformation dieser Art bewirkt also, dass der Grauwert $g = s_e(x_e, y_e)$ des Originalbildes $\mathbf{S}_e$ (des verzerrten Bildes) im transformierten (entzerrten) Bild $\mathbf{S}_a$ in der Position mit den Ortskoordinaten x_a und y_a erscheint. Je nach Art der Transformationsfunktionen ergeben sich verschiedene Arten der Modifikation der Ortskoordinaten.

9.3 Vergrößerung, Verkleinerung

Bei der *Vergrößerung* wird das Originalbild $\mathbf{S}_e$ mit L_e Bildzeilen und R_e Bildspalten in ein Bild $\mathbf{S}_a$ übergeführt, das $L_a > L_e$ und $R_a > R_e$ Bildzeilen bzw. Bildspalten besitzt. Ist der Vergrößerungsmaßstab in beiden Richtungen gleich, so haben die Funktionen der Transformation die einfache Form

$$f_1(x_e, y_e) = ax_e \text{ und } f_2(x_e, y_e) = ay_e. \tag{9.3}$$

Dabei ist $a > 1$ der Vergrößerungsfaktor. Durch Einsetzen der Beziehung (9.3) in (9.1) sieht man den Zusammenhang:

$$s_a(ax_e, ay_e) = s_e(x_e, y_e). \tag{9.4}$$

Mit dieser Zuordnungsvorschrift kann die Bildmatrix $\mathbf{S}_a$ gefüllt werden, etwa durch:

```
FOR x_e:=0 TO L_e-1
FOR y_e:=0 TO R_e-1
BEGIN
 x_a:= a*x_e;
 y_a:= a*y_e;
 s_a(x_a,y_a) := s_e(x_e,y_e)
END;
```

Diese Vorgehensweise bezeichnet man als *direkte Methode*: Ausgehend von den Koordinaten des verzerrten Bildes wird die Position des Grauwertes im entzerrten Bild berechnet.

Die Umrechnung der Bildkoordinaten mit der direkten Methode hat in dieser Form allerdings den Nachteil, dass bestimmte Positionen in $\mathbf{S}_a$ nicht besetzt werden, so bei $a = 3$ z.B. die Positionen

$$(x_a, y_a) = (k, l) \text{ und } (x_a, y_a) = (k+1, l+1) \text{ mit } k, l = 1, 4, 7, 10, ...$$

Dieser Nachteil wird vermieden, wenn der Laufindex in Zeilen und Spaltenrichtung durch eine Größe `delta` inkrementiert wird:

```
delta := 1/a;
x_e  := 0;
WHILE x_e < L_e DO
BEGIN
 y_e := 0;
 WHILE y_e < R_e DO
 BEGIN
  x_a:= TRUNC(a*x_e);
  y_a:= TRUNC(a*y_e);
  i := TRUNC(x_e);
  j := TRUNC(y_e);
  s_a(x_a,y_a) := s_e(i,j);
  y_e := y_e + delta
 END;
 x_e := x_e + delta
END;
```

Bei der *indirekten Methode* wird der umgekehrte Weg eingeschlagen. Zu jeder Position des entzerrten Bildes wird im verzerrten Bild der zugehörige Grauwert bestimmt. Dazu werden allerdings die Umkehrfunktionen $f_1^{-1} = f_2^{-1} = 1/a$ benötigt, die auch überall definiert sind, da $a > 1$ vorausgesetzt wurde. Der Algorithmus lautet dann:

```
FOR x_a:=0 TO a*(L_e-1)
FOR y_a:=0 TO a*(R_e-1)
BEGIN
 x_e := ROUND(x_a/a);
 y_e := ROUND(y_a/a);
 s_a(x_a,y_a) := s_e(x_e,y_e)
END;
```

Bei diesen Koordinatenberechnungen tritt der Fall ein, dass die berechneten Werte x_a/a und y_a/a nicht ganzzahlig sind und somit keine Bildpunktposition im verzerrten oder entzerrten Bild bezeichnen. Es ist dann die Frage, welche Position gewählt und wie der zugehörige Grauwert berechnet werden soll. Diese Fragestellungen werden mit dem Schlagwort *resampling* bezeichnet.

Im obigen Algorithmus zur indirekten Methode wird das Problem durch Rundung gelöst, d.h. es wird in der Bildmatrix diejenige Koordinatenposition gewählt, die den Werten x_a/a und y_a/a am nächsten liegt (*nearest neighbor resampling*).

Eine andere Möglichkeit ist die *bilineare Interpolation*, bei der der Grauwert als gewichteter Mittelwert der vier benachbarten Bildpunkte berechnet wird:

$$\begin{aligned} s_a(x_a, y_a) &= a_1 \cdot s\Big(\text{int}(x_e), \text{int}(y_e)\Big) + a_2 \cdot s\Big(\text{int}(x_e), \text{int}(y_e) + 1\Big) + \\ &\quad a_3 \cdot s\Big(\text{int}(x_e) + 1, \text{int}(y_e)\Big) + a_4 \cdot s\Big(\text{int}(x_e) + 1, \text{int}(y_e) + 1\Big). \end{aligned} \tag{9.5}$$

Die Parameter a_i können dabei so gewählt werden, dass sie die Lage der berechneten Position bezüglich der vier Nachbarpunkte berücksichtigen.

Da (9.5) eine Mittelung von Grauwerten beinhaltet, hat die bilineare Interpolation auch grauwertglättende Eigenschaften (vergleichbar mit dem bewegten Mittelwert oder einer Tiefpassfilterung, Kapitel 5). Durch diese Mittelung werden neue Grauwerte erzeugt, die womöglich im Originalbild nicht auftreten. Das begrenzt die Verwendungsmöglichkeit des Verfahrens: Bei logischen Bildern, bei denen die Grauwerte Codes für bestimmte Eigenschaften sind, kann die bilineare Interpolation nicht eingesetzt werden. Im Vergleich mit der nächster-Nachbar-Methode, die z.B. schräg verlaufende Grauwertkanten im entzerrten Bild oft stufig erscheinen lässt, werden bei der bilinearen Interpolation die Grauwertübergänge, bedingt durch das Tiefpassverhalten, etwas gedämpft. Je nach Anwendungsfall mag dieser Effekt mehr oder weniger wünschenswert sein.

Eine weitere Methode zum *resampling* ist die *kubische Faltung* (*cubic convolution*). Sie benutzt eine Umgebung von insgesamt 16 Bildpunkten, die, mit entsprechenden Gewichten versehen, die Berechnung des Grauwertes beeinflussen:

$$\begin{aligned} &i = \text{int}(x_a/a) \text{ und } j = \text{int}(y_a/a) \\ &s_a(x_a, y_a) = \sum_{m=-1}^{2} \sum_{n=-1}^{2} a_{mn} \cdot s_e(i+m, j+n). \end{aligned} \tag{9.6}$$

Die kubische Faltung, bei der durch entsprechende Wahl der Parameter der Tiefpasseffekt vermieden werden kann, liefert im Vergleich mit den beiden anderen Verfahren die besten Ergebnisse. Allerdings ist auch der erhöhte Rechenaufwand zu berücksichtigen.

Die Problemstellung des *resampling* tritt bei allen folgenden Umrechnungen der Ortskoordinaten auf. Es wird jedoch im Weiteren nicht mehr näher darauf eingegangen. Auch ob die direkte oder die indirekte Methode verwendet wird, ist im Folgenden ohne Bedeutung.

Die *Verkleinerung* kann mit der Beziehung (9.3) ausgedrückt werden, wenn für den Parameter $0 < a < 1$ zugelassen wird. Unterschiedliche Maßstabsfaktoren in Zeilen- und Spaltenrichtung können durch die folgende Wahl der Funktionen erreicht werden:

$$f_1(x_e, y_e) = a \cdot x_e \text{ und } f_2(x_e, y_e) = b \cdot y_e \text{ mit } a, b > 0. \tag{9.7}$$

9.4 Affine Abbildungen

Bei der allgemeinen affinen Transformation der Ortskoordinaten sind die Funktionen lineare Polynome in x_e und y_e:

$$\begin{aligned} x_a &= f_1(x_e, y_e) = a_0 + a_1 x_e + a_2 y_e, \\ y_a &= f_2(x_e, y_e) = b_0 + b_1 x_e + b_2 y_e. \end{aligned} \tag{9.8}$$

Die in Abschnitt 9.3 besprochene Vergrößerung und Verkleinerung sind natürlich Spezialfälle von (9.8). In Matrizenschreibweise lautet (9.8):

$$\begin{aligned} &\mathbf{x}_a = \mathbf{A}\mathbf{x}_e + \mathbf{v} \text{ oder} \\ &\begin{pmatrix} x_a \\ y_a \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \end{pmatrix} + \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \end{aligned} \tag{9.9}$$

Durch entsprechende Wahl der Parameter a_i und b_i erhält man verschiedene Transformationen der Ortskoordinaten.

Bei einer *Verschiebung* (*Translation*) ist die Matrix $\mathbf{A}$ die Einheitsmatrix. Der Verschiebungsvektor $\mathbf{v}$ gibt die Verschiebungsanteile in x_e- und y_e-Richtung an:

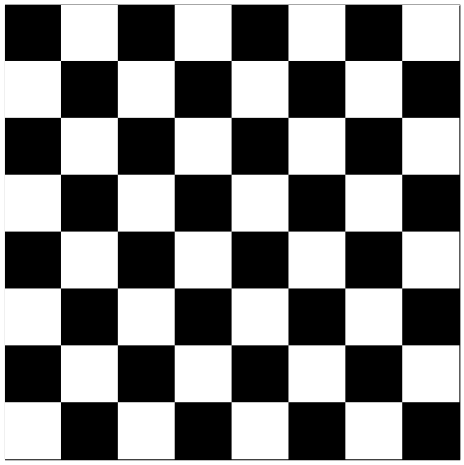
$$\begin{aligned} x_a &= x_e + a_0, \\ y_a &= y_e + b_0. \end{aligned} \tag{9.10}$$

Drehungen (*Rotationen*) können ebenfalls durch (9.8) dargestellt werden. Die Matrix $\mathbf{A}$ hat bei einer Drehung um den Winkel α die Gestalt:

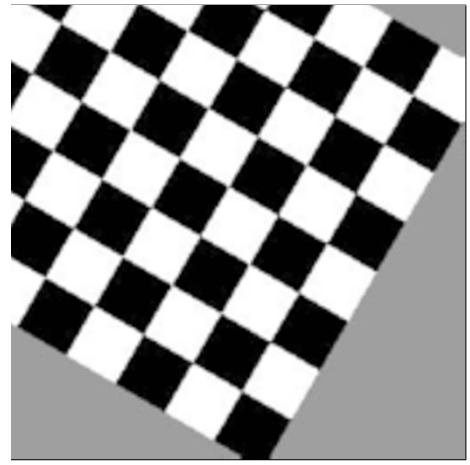
$$\mathbf{A} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \tag{9.11}$$

Ist mit der Drehung auch eine Verschiebung verbunden, so drückt der Verschiebungsvektor $\mathbf{v}$ die Anteile in x_e- und y_e-Richtung aus. Da die Bildmatrix $\mathbf{S}_a$ des gedrehten Bildes wieder eine rechteckige Zahlenanordnung ist, muss unter Umständen ein spezieller Grauwert als Hintergrund festgelegt werden, mit dem diejenigen Bildpunkte codiert werden, die zwar in der Bildmatrix $\mathbf{S}_a$ liegen, aber nicht zum gedrehten Bild gehören (Bild 9.1).

Bei einer allgemeinen Affintransformation werden an die Parameter in (9.8) keine weiteren Forderungen gestellt. Es muss lediglich gewährleistet sein, dass die Determinante der Matrix $\mathbf{A}$ ungleich 0 ist. Es können dann geometrische Transformationen wie *Scherung* oder *Spiegelung* erfasst werden. Auf die Bestimmung der Parameter a_i und b_i wird in den folgenden Abschnitten noch näher eingegangen.



(a)



(b)

Bild 9.1: Koordinatentransformation eines Bildes. (a) Originalbild. (b) Transformiertes Bild (Drehung und Verschiebung) mit Hintergrundgrauwert.

9.5 Interpolation mit Polynomen

9.5.1 Polynome

Sind die Funktionen der geometrischen Transformation $f_1(x_e, y_e)$ und $f_2(x_e, y_e)$ nicht weiter bekannt, so werden Funktionen verwendet, die den tatsächlichen Sachverhalt so gut wie möglich annähern sollen (*Modellbildung*). In der Praxis werden dazu oft Polynome n-ten Grades in den Unbekannten x_e und y_e verwendet:

$$\begin{aligned} x_a &= f_1(x_e, y_e) = a_0 + a_1 x_e + a_2 y_e + a_3 x_e^2 + a_4 x_e y_e + a_5 y_e^2 + ..., \\ y_a &= f_2(x_e, y_e) = b_0 + b_1 x_e + b_2 y_e + b_3 x_e^2 + b_4 x_e y_e + b_5 y_e^2 + ... \end{aligned} \tag{9.12}$$

Ein Sonderfall von (9.12) sind die affinen Abbildungen von Abschnitt 9.4. Auch Polynome zweiten Grades werden in der Praxis oft verwendet. Es kann auch sinnvoll sein, in Zeilen- und in Spaltenrichtung Polynome unterschiedlichen Grades zu verwenden, wenn der jeweilige Anwendungsfall dies rechtfertigt.

Es stellt sich nun die Frage, wie die Parameter a_i und b_i der Transformationspolynome ermittelt werden können. Eine Möglichkeit besteht darin, einen Lösungsansatz über sogenannte *Systemkorrekturen* zu wählen. Hier liegen aufgrund der Aufzeichnungsart eines Bildes Informationen über die geometrischen Verzerrungen, also über die Funktionen der Transformation und deren Parameter vor. Durch einen mathematischen Ansatz können

dann die unbekannten Parameter abgeleitet werden. Als Beispiel hierzu kann ein Bild dienen, das durch eine projektive Abbildung entstanden ist. Der Ansatz mit linearen Polynomen ist dann exakt, und die unbekannten Parameter können z.B. aus der Position des Aufnahmegerätes berechnet werden. Ein weiteres Beispiel ist ein Bild, das mit einem Weitwinkelobjektiv aufgenommen wurde. Liegen vom Hersteller Angaben (Tabellen oder funktionaler Zusammenhang) über die Aufnahmeeigenschaften des Objektivs vor, so kann auch hier, in gewissen Grenzen, eine Systemkorrektur durchgeführt werden.

In allen Fällen, in denen die Parameter nicht über einen Ansatz zur Systemkorrektur berechnet werden können, wird die *Passpunktmethode* verwendet. Zur Erläuterung dieser Methode werden folgende Annahmen gemacht (Bild 9.2):

$\mathbf{S}_e = (s_e(x_e, y_e))$	sei das verzerrte Bild.
$\mathbf{R} = (r(u, v))$	sei ein Referenzbild, an das das verzerrte Bild angeglichen werden soll. Über den funktionalen Zusammenhang der Verzerrung sei weiter nichts bekannt. Es wird jedoch angenommen, dass er durch Polynome n-ten Grades approximiert werden kann.
$\mathbf{S}_a = (s_a(x_a, y_a))$	sei das entzerrte Bild, das dann deckungsgleich mit dem Referenzbild $\mathbf{R}$ ist.

Vereinfachend wird ein linearer Ansatz gemäß (9.8) gewählt. Es sind dann die sechs unbekannten Parameter a_i und b_i mit $i = 0, 1, 2$ zu bestimmen. Zur Ermittlung dieser Parameter sind ingesamt drei *Passpunkte* notwendig. Passpunkte sind dabei einander entsprechende Punkte, die sowohl im Referenzbild $\mathbf{R}$ als auch im verzerrten Bild $\mathbf{S}$ eindeutig ermittelt werden können. Die drei Passpunkte liefern also drei einander entsprechende Koordinatenpaare gemäß:

$$i\text{-ter Passpunkt: } (u_i, v_i) \Longleftrightarrow (x_{e,i}, y_{e,i}), i = 1, 2, 3.$$

Diese Koordinatenpaare werden in die Funktionen (9.8) eingesetzt und liefern zwei lineare Gleichungssysteme:

$$\begin{aligned} u_i &= a_0 + a_1 x_{e,i} + a_2 y_{e,i} \\ v_i &= b_0 + b_1 x_{e,i} + b_2 y_{e,i}, \quad i = 1, 2, 3. \end{aligned} \tag{9.13}$$

In Matrixschreibweise:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & x_{e,1} & y_{e,1} \\ 1 & x_{e,2} & y_{e,2} \\ 1 & x_{e,3} & y_{e,3} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \text{ und } \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 1 & x_{e,1} & y_{e,1} \\ 1 & x_{e,2} & y_{e,2} \\ 1 & x_{e,3} & y_{e,3} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}.$$

Aus diesen Gleichungssystemen können die sechs unbekannten Polynomparameter berechnet und nach Maßgabe der drei Passpunkte kann das verzerrte Bild $\mathbf{S}_a = (s_a(x_a, y_a))$ aufgebaut werden.

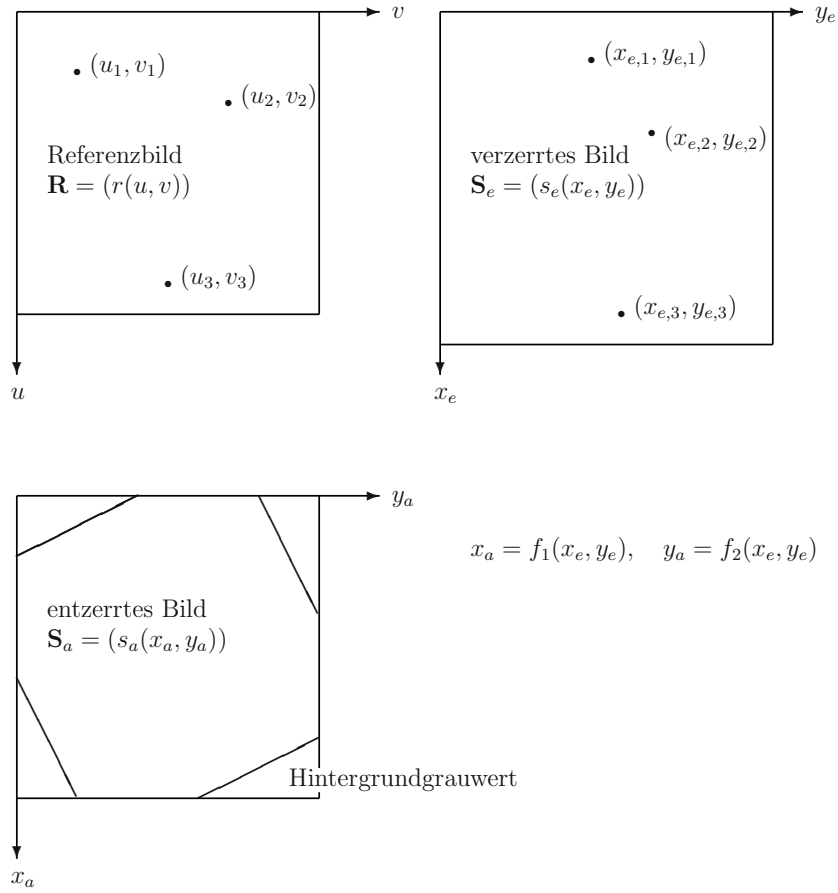


Bild 9.2: Bestimmung der Parameter der Transformationsfunktionen mit der Passpunktmethode. Im Referenzbild $\mathbf{R}$ und im verzerrten Bild $\mathbf{S}_e$ werden die Koordinaten von einander entsprechenden Bildpunkten ermittelt. Anhand dieser Koordinaten werden die unbekannten Parameter der Funktionen der Transformation berechnet.

Die Passpunktmethode kann sinngemäß auf Polynome mit einem höheren Grad oder allgemeinere Abbildungsgleichungen angewendet werden. Bei Polynomen zweiten Grades (12 unbekannte Parameter) sind sechs Passpunkte und bei Polynomen dritten Grades (20 unbekannte Parameter) sind zehn Passpunkte erforderlich.

Bei der Vermessung der Passpunkte müssen in der Praxis immer Messfehler in Kauf genommen werden. Das hat aber zur Folge, dass diese Messfehler in die Berechnung der Parameter eingehen und somit auch die Güte der Entzerrung beeinflussen. So ist es sinnvoll, mehr Passpunkte als für das jeweilige Entzerrungsmodell notwendig sind zu ermitteln und die Parameter so zu bestimmen, dass sie nach der *Methode der kleinsten Quadrate* (*Ausgleichsrechnung*) die Messfehler minimieren. Dies wird im nächsten Abschnitt erläutert.

9.5.2 Ausgleichsrechnung

Zur Berechnung der Funktionen bei einer geometrischen Entzerrung wird die *Ausgleichsrechnung* verwendet, um die Parameter so zu bestimmen, dass die Funktionen eine vorgegebene Fehlerfunktion minimieren. Im Folgenden wird diese Methode am Beispiel von linearen Polynomen dargestellt. Die sinngemäße Übertragung auf andere Funktionen ist möglich.

Es wird vorausgesetzt, dass $p > 3$ Passpunkte vorliegen:

$$(u_i, v_i) \Longleftrightarrow (x_{e,i}, y_{e,i}), \quad i = 1, 2, 3, ..., p$$

Die linearen Gleichungsysteme für die unbekannten Parameter a_i und b_i lauten:

$$\begin{aligned} u_i &= a_0 + a_1 x_{e,i} + a_2 y_{e,i} \\ v_i &= b_0 + b_1 x_{e,i} + b_2 y_{e,i}, \quad i = 1, 2, ..., p \end{aligned} \tag{9.14}$$

In Matrixschreibweise:

$$\mathbf{u} = \mathbf{Pa} \text{ und } \mathbf{v} = \mathbf{Pb}.$$

Dabei sind $\mathbf{u}$ und $\mathbf{v}$ Vektoren der Dimension p, $\mathbf{P}$ eine Matrix mit drei Spalten und p Zeilen und $\mathbf{a}$ und $\mathbf{b}$ Vektoren der Dimension drei (die unbekannten Parameter der Polynome).

Die lineare Ausgleichung wird so durchgeführt, dass die Vektoren $\mathbf{a}$ und $\mathbf{b}$ die folgenden Fehlerfunktionen minimieren:

$$\begin{aligned} \sum_{i=1}^{p} \Big(u_i - (a_0 + a_1 x_{e,i} + a_2 y_{e,i})\Big)^2 &\rightarrow min_a \\ \sum_{i=1}^{p} \Big(v_i - (b_0 + b_1 x_{e,i} + b_2 y_{e,i})\Big)^2 &\rightarrow min_b \end{aligned} \tag{9.15}$$

oder in Matrixschreibweise:

$$(\mathbf{u}-\mathbf{Pa})^T(\mathbf{u}-\mathbf{Pa}) \rightarrow min_a \text{ und } (\mathbf{v}-\mathbf{Pb})^T(\mathbf{v}-\mathbf{Pb}) \rightarrow min_b. \tag{9.16}$$

Da die Funktionen (9.15) stetige partielle Ableitungen nach den a_i und b_i besitzen, lässt sich sofort eine notwendige Bedingung für die Minmimierung angeben:

$$\begin{aligned} \frac{\partial \sum_{i=1}^{p} \Big(u_i - (a_0 + a_1 x_{e,i} + a_2 y_{e,i})\Big)^2}{\partial a_j} &= 0, \quad j = 0,1,2 \\ \frac{\partial \sum_{i=1}^{p} \Big(v_i - (b_0 + b_1 x_{e,i} + b_2 y_{e,i})\Big)^2}{\partial b_j} &= 0, \quad j = 0,1,2 \end{aligned} \tag{9.17}$$

Die Bedingungen (9.17) heißen die *Normalgleichungen*, die bei der linearen Ausgleichung auch durch die folgenden linearen Gleichungssysteme für **a** und **b** ausgedrückt werden können:

$$\mathbf{P}^T\mathbf{Pa} = \mathbf{Pu} \text{ und } \mathbf{P}^T\mathbf{Pb} = \mathbf{Pv}. \tag{9.18}$$

Die Lösungen dieser linearen Gleichungssysteme sind die gesuchten Parameter **a** und **b** der Polynome. Die direkte Lösung der Normalgleichungen (9.18) ist möglicherweise numerisch instabil. Aus diesem Grund werden in der Praxis auch andere Methoden verwendet, so z.B. das *Householder-Verfahren* ([Enge96]).

Im Weiteren ist nun zu untersuchen, wie die Qualität des funktionalen Ansatzes und der Ausgleichung beurteilt werden können.

9.5.3 Beurteilung der Qualität

Die Beurteilung der Qualität sollte bei einer geometrischen Entzerrung immer durchgeführt werden, denn sie gibt Aufschluss, ob der funktionale Ansatz mit den gewählten Funktionen (im vorliegenden Beispiel Polynome ersten Grades) gerechtfertigt ist und wie gut die Funktionen an die gegebenen Passpunkte angepasst wurden. Da bei der Vermessung der Passpunkte manchmal auch größere Fehler auftreten können (z.B. Tippfehler bei den Koordinatenwerten), ist durch die Beurteilung auch eine Möglichkeit gegeben, fehlerhafte Passpunkte zu entdecken.

Zur Beurteilung können mehrere Möglichkeiten kombiniert werden. Als Erstes sind die in (9.15) durch die Ausgleichung erzielten Minima min_a und min_b natürlich Beurteilungsgrößen. Hier empfiehlt es sich, noch eine Normierung der beiden Minima durch die Anzahl p der Passpunkte durchzuführen:

$$min_{a,normiert} = \frac{min_a}{p} \quad \text{und} \quad min_{b,normiert} = \frac{min_b}{p}. \tag{9.19}$$

Besser noch als (9.19) ist eine Normierung über die Überbestimmtheit der Gleichungssysteme. Es sei m die Anzahl der Parameter der Funktionen (im vorliegenden Beispiel: $m = 3$). Als Größen zur Beurteilung der Güte kann dann verwendet werden:

$$min_{a,normiert} = \frac{min_a}{p-m} \quad \text{und} \quad min_{b,normiert} = \frac{min_b}{p-m}. \tag{9.20}$$

Dabei muss natürlich die Anzahl p der Passpunkte größer sein als die Anzahl m der auftretenden Parameter. Für $p = m$ wäre das Fehlermaß (9.20) unendlich groß, was auch plausibel ist, da ja im Falle der exakten Bestimmung der Parameter der Funktionen keine Aussagen über ihre Güte gemacht werden können.

Eine weitere Möglichkeit, die auch gut zur grafischen Darstellung geeignet ist, ist die Berechnung von *Restfehlervektoren*. Dazu werden, wie oben beschrieben, die Parameter der Funktionen ermittelt. Dann werden die Koordinaten der Passpunkte in die Transformationsgleichungen eingesetzt und zu jedem Passpunkt (u_i, v_i) im Referenzbild die Koordinaten $(u_i^{'}, v_i^{'})$ berechnet. Die Differenz der gemessenen und der berechneten Koordinaten $(u_i - u_i^{'}, v_i - v_i^{'})$ wird als Restfehlervektor des i-ten Passpunktes bezeichnet. Zur grafischen Darstellung werden die Restfehlervektoren in ein (u, v)-Koordinatensystem eingetragen (Bild 9.3). Da die Abweichungen oft nur sehr gering sind, empfiehlt es sich, sie um einen geeigneten Maßstabsfaktor c vergrößert darzustellen.

Sind die Richtungen der Restfehlervektoren mehr oder weniger gleichverteilt und die Beträge klein, so kann angenommen werden, dass bei der Vermessung keine groben Fehler gemacht wurden und das gewählte Modell den tatsächlichen funktionalen Zusammenhang zwischen Referenzbild und verzerrtem Bild gut approximiert. Treten in bestimmten Bildbereichen Vorzugsrichtungen auf, so ist das ein Hinweis, dass in diesem Bereich bei der Bestimmung der Passpunkte Fehler gemacht wurden. Allerdings würde sich ein systematischer Fehler, der bei der Vermessung aller Passpunkte in gleicher Weise zum Tragen kam, in der grafischen Darstellung der Restfehlervektoren nicht unbedingt ausdrücken. Um Passpunkte zu entdecken, die, wie oben angedeutet, durch gröbere Koordinatenfehler gestört sind, wird der Betrag der Restfehlervektoren verwendet:

$$\sqrt{(u_i - u_i')^2 + (v_i - v_i')^2} \tag{9.21}$$

Ist der Betrag größer als ein bestimmter Schwellwert c, so wird der zugehörige Passpunkt als möglicherweise falsch vermessen markiert und kann dann noch einmal kontrolliert werden. Der Schwellwert c kann dabei nach Maßgabe der mittleren quadratischen Abweichung der Beträge der Restfehlervektoren aller Passpunkte festgelegt werden.

Eine weitere Variante zur Beurteilung besteht darin, dass zur Berechnung der Restfehlervektoren nicht die Passpunkte verwendet werden, mit denen die Parameter berechnet wurden, sondern eigens dafür vermessene Kontrollpunkte. Allerdings können dann fehlerhafte Passpunkte nicht automatisch erkannt werden. So ist es am besten, wenn alle angebotenen Möglichkeiten kombiniert werden, um ein Höchstmaß an Informationen zur Beurteilung der Güte zu erhalten.

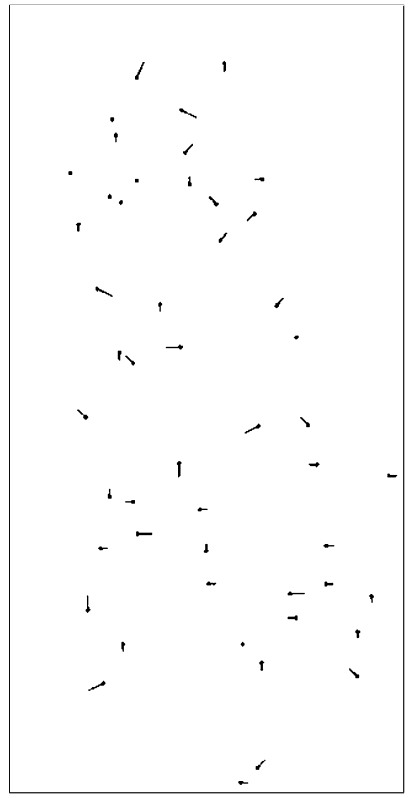


Bild 9.3: Beurteilung der Qualität der Ausgleichung bei der Berechnung der Parameter der Transformationsfunktionen. In einem (u, v)-Koordinatensystem sind die Restfehlervektoren, die sich aus der Differenz der gemessenen und der berechneten Passpunktkoordinaten ergeben, grafisch dargestellt, wobei sie für die Wiedergabe vergrößert wurden.

9.5.4 Vermessung der Passpunkte

Die Vermessung der Koordinaten der Passpunkte kann auf unterschiedliche Weise durchgeführt werden. Liegen das Referenzbild und das verzerrte Bild in digitalisierter Form vor, so können die Koordinaten mit Hilfe der interaktiven Möglichkeiten eines Bildverarbeitungssystems ermittelt werden. Die Koordinaten der Passpunkte werden dabei mit Hilfe einer Maus und eines Fadenkreuzes auf dem Bildschirm festgelegt und im (u, v)-Koordinatensystem des Referenzbildes bzw. im (x_e, y_e)-Koordinatensystem des verzerrten Bildes vermessen. Ist das Referenzbild eine Karte, so werden die (u, v)-Koordinaten im Koordinatensystem der jeweiligen Kartenprojektion (z.B. Gauß-Krüger- oder UTM-Gitter) angegeben. Dabei ist ein Digitalisierungstisch hilfreich, auf den die Kartenvorlage aufgespannt ist. Mit Hilfe des dazugehörigen Fadenkreuzes können dann die Koordinaten relativ genau bestimmt werden. In diesem Fall muss noch eine Zuordnung der (u, v)-Koordinaten des Referenzbildes $\mathbf{R}$ zu den Zeilen- und Spaltenkoordinaten (x_a, y_a) des entzerrten Bildes $\mathbf{S}_a$ hergestellt werden. Einfacher gestaltet sich das Ausmessen der (u, v)-Koordinaten, wenn die Karte schon digitalisiert vorliegt.

Eine aufwändigere Technik ist die Passpunktvermessung mit Hilfe einer *Flächenkorrelation*. Dies sei am Beispiel eines digitalisierten Referenzbildes $\mathbf{R} = (r(u, v))$ und eines verzerrten Bildes $\mathbf{S}_e = (s_e(x_e, y_e))$ erläutert. Zunächst wird im Referenzbild ein markanter Punkt bestimmt, der auch im verzerrten Bild erkannt werden kann und somit als Passpunkt in Frage kommt. Die Koordinaten dieses Punktes im Referenzbild seien (u, v). Im nächsten Schritt wird um den Punkt ein Ausschnitt der Größe $m \cdot m, m = 3, 5, 7, ...$ mit (u, v) als Mittelpunkt definiert. Aufgrund von apriori-Informationen kann in der Regel die ungefähre Lage $(x_{e,0}, y_{e,0})$ des Punktes im verzerrten Bild berechnet oder interaktiv ermittelt werden. Bei apriori-Information über die Art der Verzerrung kann auch der Ausschnitt im Referenzbild an die Geometrie des verzerrten Bildes angepasst werden. Nun wird ein Maß der Nicht-Übereinstimmung zwischen dem Ausschnitt im Referenzbild und dem entsprechenden Ausschnitt im verzerrten Bild berechnet, etwa:

$$\frac{1}{m^2} \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} \Big(s_e(x_{e,i-n+k}, y_{e,i-n+l}) - h(k, l) \Big)^2, \tag{9.22}$$

wobei $n = (m-1)/2$, $(x_{e,i}, y_{e,i})$ mit anfänglich $i = 0$ die berechnete Position im verzerrten Bild und $(h(u, v))$ der evtl. geometrisch angepasste Ausschnitt aus dem Referenzbild ist. Durch Ausquadrieren von (9.22) erhält man:

$$\begin{aligned} &\frac{1}{m^2} \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} s_e(x_{e,i-n+k}, y_{e,i-n+l})^2 - \\ &\frac{2}{m^2} \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} \Big(s_e(x_{e,i-n+k}, y_{e,i-n+l})) \cdot h(k, l) \Big) + \\ &\frac{1}{m^2} \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} h(k, l)^2. \end{aligned} \tag{9.23}$$

Der mittlere Term in (9.23) ist die Korrelation des Ausschnittes des Referenzbildes $(h(u, v))$ mit dem verzerrten Bild im Bereich der Position $(x_{e,i}, y_{e,i})$.

Das Maß der Nicht-Übereinstimmung ist also klein, wenn die Korrelation in (9.23) groß ist. Soll das Maß von Helligkeit und Kontrast unabhängig sein, so kann (9.23) noch durch eine Division durch den ersten Term normiert werden.

Ausgehend von der Berechnung des Maßes in der Position mit dem Index $i = 0$ im verzerrten Bild wird jetzt eine Suche nach dem Korrelationsmaximum durchgeführt. Die gefundene Position des Korrelationsmaximums, die auch auf Bruchteile von Bildpunkten berechnet werden kann, wird dann als Lage des Passpunktes im verzerrten Bild verwendet.

Diese Methode der Passpunktbestimmung über die Korrelation ist gut bei automatischen oder teilautomatischen Entzerrungssystemen geeignet, bei denen die Passflächen der Referenzbilder (z.B. Satellitenbilddaten) in einer Datenbank verwaltet werden. Bei der Integration eines neuen Bildes können dann die Passpunktkoordinaten mit Hilfe der Datenbank automatisch ermittelt werden.

9.6 Abschließende Bemerkungen zur geometrischen Entzerrung

Die Probleme der Transformation der Ortskoordinaten wurden hier am Beispiel der Verwendung von Polynomen erläutert. In der Praxis werden auch andere Funktionen, wie z.B. zweidimensionale Splinefunktionen oder Flächen zweiter Ordnung verwendet.

Bei vielen praktischen Anwendungen findet im zunehmenden Maß die Verarbeitung von dreidimensionalen Koordinaten Bedeutung. Die Stereoluftbildauswertung in der Fotogrammetrie erlaubt es z.B. mit einem Stereobildpaar mit nur fünf Passpunkten, die vollständigen dreidimensionalen Verhältnisse zu rekonstruieren. Auch für Zeilenabtasteraufnahmen sind Entzerrungsmodelle entwickelt worden, die drei verschiedene Blickwinkel auf das zu entzerrende Gebiet verwenden.

Auch bei der Auswertung von Bildfolgen ist die Berechnung der dreidimensionalen Koordinaten wichtig. Hier wird neben dem Biokularstereo auch das Bewegungsstereo verwendet, bei dem z.B. aus aufeinanderfolgenden Zeitreihenbildern räumliche Informationen berechnet werden.

Kapitel 10

Demosaicing

10.1 Anwendungen

Digitale Farbkameras basieren meist auf einem einzigen Flächensensor (CCD- oder CMOS-Chip), dessen Pixel die Menge des einfallenden Lichts über einen breiten Wellenlängenbereich messen. Dadurch erhält man pro Pixel einen Meßwert für die Helligkeit an der entsprechenden Stelle, d.h. man könnte nur Schwarz-Weiß-Bilder aufnehmen. Um nun mit einem einzigen Flächensensor auch Farbbilder aufnehmen zu können, wird vor den Sensor ein sogenanntes *„Bayer"-Farbfilter* (Bayer Color Filter Array, kurz Bayer-CFA, [Baye76]) gestellt, so dass pro Bildpunkt nur einer von den drei Farbwerten R,G, oder B gemessen wird (Algorithmus A10.1 und Bild 10.1). Es fehlen für jeden Bildpunkt somit die Farbinformationen der beiden anderen Farbwerte. Die Aufgabe einer *Demosaicing*-Methode [1] besteht nun darin, aus den umgebenden Messwerten eines Bildpunktes die jeweils fehlenden zwei Farbwerte zu rekonstruieren. Als Ergebnis erhält man ein vollständiges Bild, bei dem an jedem Pixel alle drei Farbwerte R,G,B vorliegen.

Aufgrund des grandiosen Siegeszuges, den digitale Farbkameras seit etwa dem Jahr 2000 am Markt errungen haben, ist auch die wirtschaftliche Bedeutung von Demosaicing-Algorithmen stark gestiegen. Deshalb wurden in diesem Zeitraum eine Reihe von Forschungs- und Entwicklungsvorhaben zur Verbesserung von Demosaicing-Algorithmen gestartet, die in Patenten ([Hami97], [Nisc10]) und Veröffentlichungen ([Malv04], [Hira05]) mündeten. In den folgenden Abschnitten werden wichtige Demosaicing-Algorithmen dargestellt, die aufeinander aufbauen und zunehmend komplexer, aber auch leistungsfähiger werden.

[1] Anstatt des Begriffs *„Demosaicing"* wird manchmal auch der Begriff *„Debayering"* verwendet, weil damit die Wirkung des *„Bayer"-Farbfilters* kompensiert wird.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_10

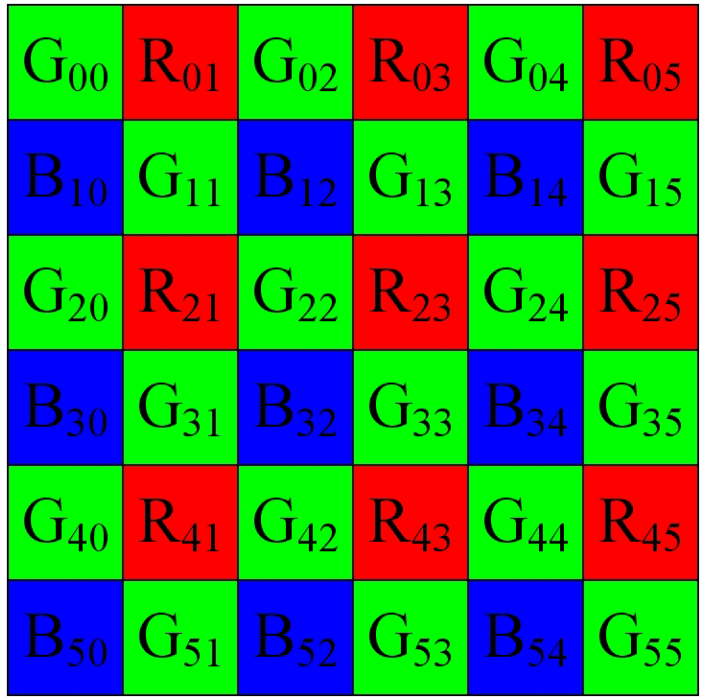


Bild 10.1: Bayer Color Filter Array. Das Grundmuster besteht immer aus einem 2×2 Pixel großen Bereich, in dem diagonal zwei grüne Pixel angeordnet sind und komplementär dazu ein rotes und ein blaues Pixel. Dadurch ergeben sich vier verschiedene Möglichkeiten zur Realisierung eines Bayer Color Filter Arrays, von denen eines hier dargestellt ist. Grün wird doppelt verwendet, weil unser visuelles System in diesem Wellenlängenbereich die höchste Empfindlichkeit aufweist.

A10.1: Bayer Farbfilter.

Voraussetzungen und Bemerkungen:

- ◇ $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1, 2$ ist ein dreikanaliges RGB-Farbbild (Eingabebild).
- ◇ $\mathbf{s}_e(x, y, n) = (R_{xy}, G_{xy}, B_{xy})^T$ ist ein RGB-Farbvektor von $\mathbf{S}_e$ an der Bildzeile x und der Bildspalte y mit $G = \{0, 1, ..., 255\}$ als Grauwertmenge.
- ◇ $\mathbf{S}_b = (s_b(x, y))$ ist ein einkanaliges Grauwertbild mit $G = \{0, 1, ..., 255\}$ als Grauwertmenge (Ausgabebild / Rohbild).

Algorithmus:

(a) Für alle geraden Bildzeilen $x = 0, 2, 4, ..., L-2$ des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:

(aa) Für alle geraden Bildspalten $y = 0, 2, 4, ..., R-2$ des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:

(aaa) Taste den Grünwert des RGB-Farbvektors ab:
$s_b(x, y) = s_e(x, y, 1) = G_{xy}$;

(ab) Für alle ungeraden Bildspalten $y = 1, 3, 5, ..., R-1$ des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:

(aba) Taste den Rotwert des RGB-Farbvektors ab:
$s_b(x, y) = s_e(x, y, 0) = R_{xy}$;

(b) Für alle ungeraden Bildzeilen $x = 1, 3, 5, ..., L-1$ des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:

(ba) Für alle geraden Bildspalten $y = 0, 2, 4, ..., R-2$ des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:

(baa) Taste den Blauwert des RGB-Farbvektors ab:
$s_b(x, y) = s_e(x, y, 2) = B_{xy}$;

(bb) Für alle ungeraden Bildspalten $y = 1, 3, 5, ..., R-1$ des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:

(bba) Taste den Grünwert des RGB-Farbvektors ab:
$s_b(x, y) = s_e(x, y, 1) = G_{xy}$;

Ende des Algorithmus

10.2 Einfaches Demosaicing: bilineare Interpolation

Eine der einfachsten Möglichkeiten zur Rekonstruktion der fehlenden Farbwerte an einem Pixel besteht in der bilinearen Interpolation dieser Werte aus der direkten Nachbarschaft. Um beispielsweise den Grünwert an der Pixelposition 21, an der nur der Rotwert R_{21} direkt gemessen wird, zu berechnen, bildet man den Mittelwert der vier umliegenden Grünwerte ($G_{11}, G_{20}, G_{22}, G_{31}$, Bild 10.2-a), d.h.:

$$G_{21} = \frac{1}{4}(G_{11} + G_{20} + G_{22} + G_{31}). \tag{10.1}$$

Nach dem gleichen Schema berechnet man den Blauwert an der Pixelposition 21, indem man den Mittelwert der vier diagonal benachbarten Blauwerte ($B_{11}, B_{20}, B_{22}, B_{31}$, Bild 10.2-b) berechnet:

$$B_{21} = \frac{1}{4}(B_{10} + B_{12} + B_{30} + B_{32}). \tag{10.2}$$

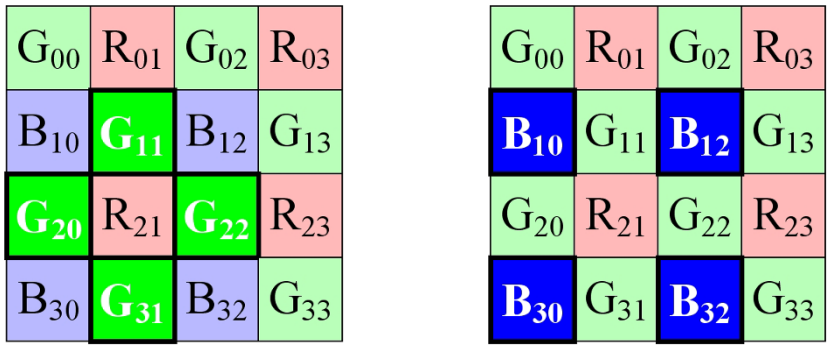


Bild 10.2: Demosaicing-Algorithmus: bilineare Interpolation. (a) Rekonstruktion des Grünwerts an der Pixelposition 21 durch Mittelwertbildung der vier umliegenden Grünwerte $(G_{11}, G_{20}, G_{22}, G_{31})$ (b) Rekonstruktion des Blauwerts an der Pixelposition 21 durch Mittelwertbildung der vier diagonal benachbarten Blauwerte $(B_{11}, B_{20}, B_{22}, B_{31})$

An der Pixelposition 22, an der nur der Grünwert G_{22} direkt gemessen wird, vereinfacht sich die Mittelwertbildung zur Rekonstruktion der fehlenden Rot- und Blauwerte zu (Bild 10.3-a,b):

$$R_{22} = \frac{1}{2}(R_{21} + R_{23}). \tag{10.3}$$

$$B_{22} = \frac{1}{2}(B_{12} + B_{32}). \tag{10.4}$$

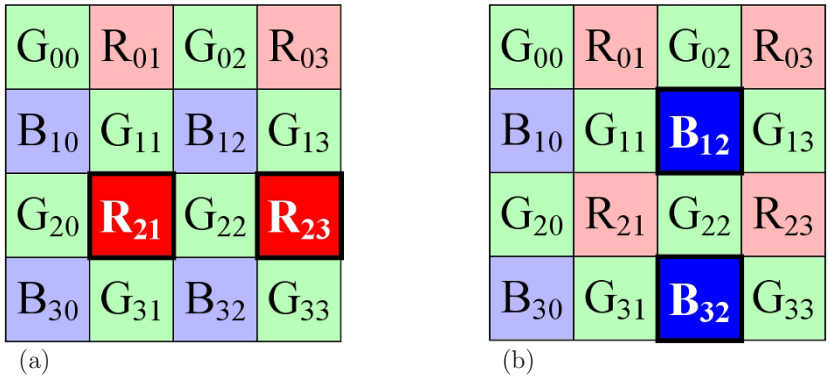
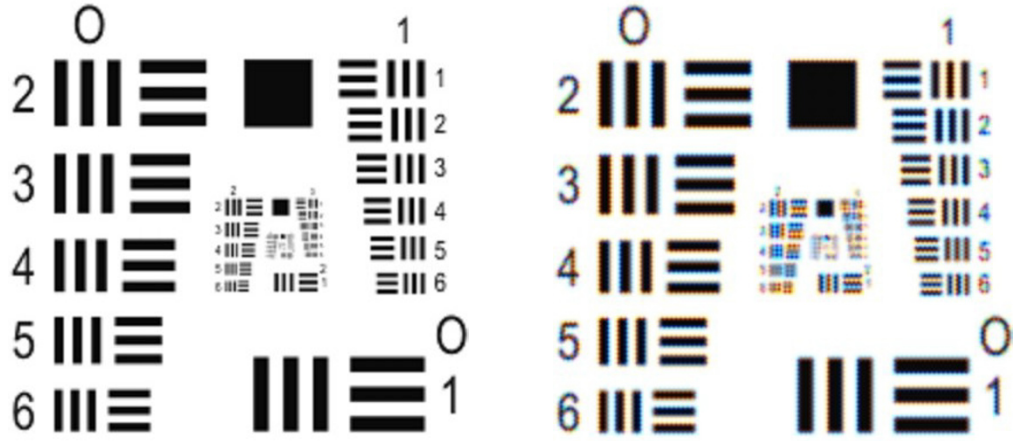


Bild 10.3: Demosaicing-Algorithmus: bilineare Interpolation. (a) Rekonstruktion des Rotwerts an der Pixelposition 22 durch Mittelwertbildung der zwei benachbarten Rotwerte (R_{21}, R_{23}) (b) Rekonstruktion des Blauwerts durch Mittelwertbildung von (B_{12}, B_{32})



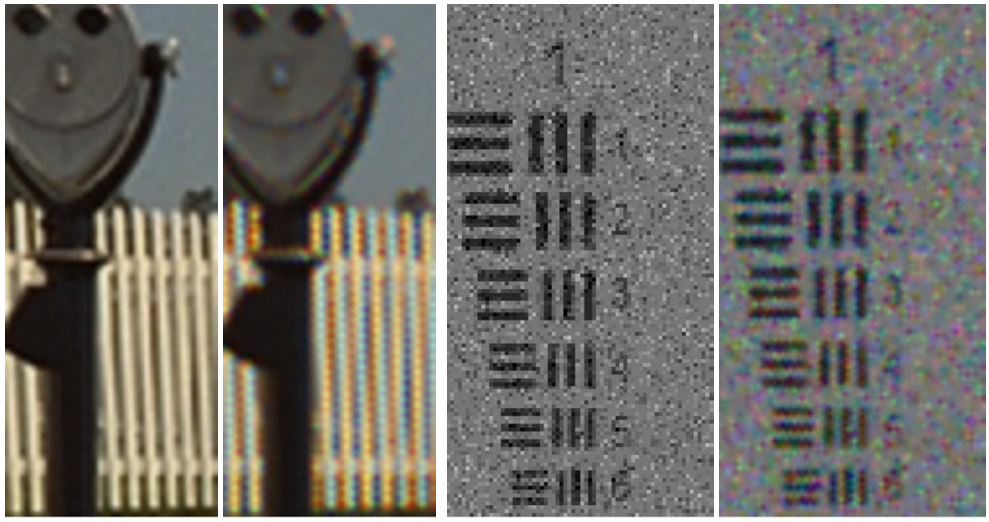
(a) (b)

Bild 10.4: Vergleich (a) Originalbild und (b) durch bilineare Interpolation rekonstruiertes Bild.

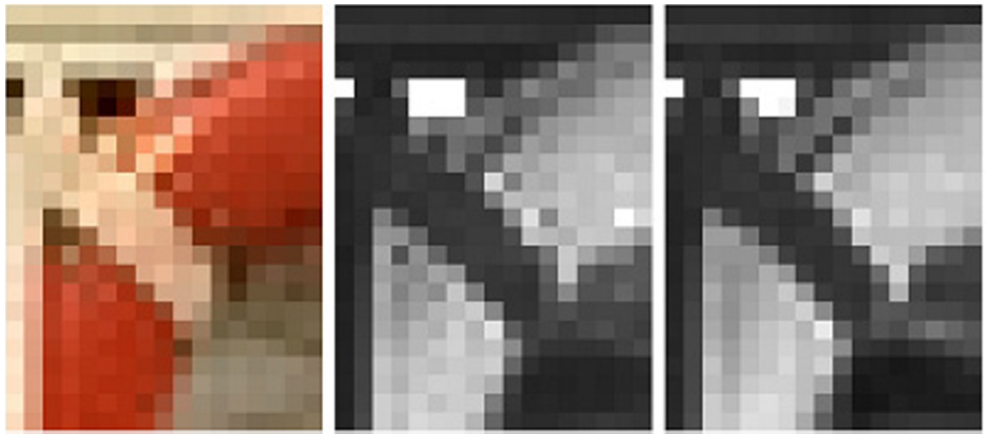
Die bilineare Interpolation der Farbwerte stellt eine Tiefpass-Filterung mit einem gleitenden Mittelwert dar (siehe Abschnitt 5.3), d.h. das rekonstruierte Bild wirkt gegenüber dem Original-Bild geglättet (Bild 10.4). Da bei diesem Demosaicing-Algorithmus keinerlei Rücksicht auf Kanten oder andere Bildstrukturen genommen wird, treten dort starke Artefakte auf. Der wichtigste Vorteil dieses Demosaicing-Algorithmus ist seine Einfachheit und daraus folgend, seine schnelle Implementierbarkeit und hohe Rechengeschwindigkeit.

10.3 Probleme beim Demosaicing

Das Grundproblem beim Demosaicing stellen die in regelmäßigen Abständen fehlenden Farbwerte dar. In jeder zweiten Zeile und jeder zweiten Spalte fehlen die Rot- bzw. Blauwerte völlig, bei den Grünwerten fehlt jedes zweite Pixel. Liegt nun zum Beispiel im Originalbild eine Schwarz-Weiß-Kante genau an einer Spalte, an der die Rotwerte fehlen (in Bild 10.1 z.B. die 2. Spalte), wird das rekonstruierte Bild an den benachbarten Spalten abwechselnd helle und dunkle cyan- bzw. blaustichige Farbsäume aufweisen. Ist die Schwarz-Weiß-Kante um ein Pixel verschoben, so dass die Blauwerte fehlen, ergeben sich abwechselnd helle und dunkle orange- bzw. gelbstichige Farbsäume (Bild 10.4-b und 10.5-b). Diese Artefakte, die immer an Kanten auftreten, werden *„Zipper“* genannt, da sie in der Art eines „Reißverschlusses“ abwechselnd helle und dunkle Falschfarben aufweisen. Falls nicht nur Standbilder betrachtet werden, sondern Videosequenzen, bei denen die Kamera langsam bewegt wird, fallen die Artefakte noch stärker ins Auge, denn dann wandern die *„Zipper“* an den Kanten entlang und ändern ihre Farbe.



(a) (b) (c) (d)



(e) (f) (g)

Bild 10.5: Probleme beim Demosaicing:
(a) Originalbild mit Kanten.
(b) Rekonstruiertes Bild mit *„Zipper“*-Artefakten.
(c) Reines Grauton-Originalbild mit Rauschen.
(d) Rekonstruiertes Bild mit Falschfarben.
(e) RGB-Originalbild mit Farbkante.
(f) Auszug des S-Kanals (Sättigung) des rekonstruierten Bildes.
(g) Auszug des S-Kanals des Originalbildes. Quelle (e-g): [Fail04].

Ein ähnliches Problem tritt bei verrauschten Bildern auf. Ist das Original ein reines Grautonbild mit pixelartigem Rauschen (Bild 10.5-c), wird das rekonstruierte Bild deutlich sichtbare Falschfarben aufweisen (Bild 10.5-d). An Stellen im Bild, an denen ein Sprung im Farbton auftritt (sogenannte Farbkanten, Bild 10.5-e), ergeben sich falsche Sättigungswerte im rekonstruierten Bild (10.5-g). Generell liegt die Ursache für die Artefakte beim Demosaicing in einer Verletzung des Abtasttheorems (Abschnitt 3.2): an den Stellen hoher Ortsfrequenzen im Originalbild (Helligkeits- und Farb-Kanten, Ecken, Rauschen, etc.) gehen durch die Unterabtastung mit dem *„Bayer Color Filter Array“* Informationen verloren, die bei der Rekonstruktion der fehlenden Farben nicht mehr vollständig hergestellt werden können.

10.4 Meßgrößen für die Qualität des Demosaicing

Um die Qualität der jeweiligen Demosaicing-Methode messen zu können, benötigt man eine Meßgröße, die aussagt, wie gut das rekonstruierte Bild mit dem Original übereinstimmt. Dabei geht man davon aus, dass im Originalbild an jedem Pixel alle drei Farbwerte für R, G, B vorliegen und dass man das *„Bayer Color Filter Array“* einfach durch Elimination von jeweils zwei Farbwerten pro Pixel simuliert [2]. Gemessen wird die Summe der mittleren quadratischen Fehler über alle drei Farbkanäle und alle Pixel zwischen Originalbild $s_o(x, y, n)$ und rekonstruiertem Bild $s_r(x, y, n)$ im RGB-Farbraum (Abschnitt 3.5.4) gemäß:

$$q_{\mathbf{MSE}} = \frac{1}{M} \sum_{x=0}^{L-1} \sum_{y=0}^{R-1} \sum_{n=0}^{2} \Big(s_r(x, y, n) - s_o(x, y, n) \Big)^2. \tag{10.5}$$

In anderen Arbeiten ([Fail04]) werden als Meßgrößen auch mittlere quadratische Fehler in anderen Farbmodellen, wie z.B. dem HSI- (Abschnitt 3.5.7), dem YIQ- (Abschnitt 3.5.6), oder dem CIE-Lab-Farbraum (Abschnitt 3.5.8) herangezogen, um die Aspekte der menschlichen Farbwahrnehmung besser zu berücksichtigen. So nützlich diese automatisch generierbaren Maße auch sind, machen sie den menschlichen Beobachter als Qualitätskontrolleur doch nicht vollkommen überflüssig. Denn es gibt Bilder, die zwar den gleichen mittleren quadratischen Fehler des Originals besitzen, aber für den menschlichen Beobachter dennoch sehr unterschiedliche Demosaicing-Qualtiät aufweisen können. Verteilt sich der mittlere quadratische Fehler über alle Pixel und Farbkanäle relativ gleichmäßig, treten

[2] In der physikalischen Realität ist der Unterabtastvorgang durch das Bayer-CFA wesentlich komplexer, denn durch die Apertur (Linsenöffnung) der Kamera und die Modulationstransferfunktionen (MTF) von Linsensystem, Bayer-CFA und Sensor entsteht eine Art Tiefpass-Filterung des ursprünglichen Signals. Dadurch werden einige besonders problematische Situationen, die durch „künstlich“ erzeugte Bildinhalte auftreten, wie z.B. 1-pixel breite Linien mit maximalem Helligkeits- oder Farbkontrast, die unter einem flachen Winkel auf einen Punkt zulaufen, von Natur aus entschärft. Dies wird jedoch nicht weiter vertieft, da alle angegebenen Qualitätsmaße in der Referenzliteratur das Bayer-CFA durch reine Unterabtastung simulieren.

an jedem einzelnen Pixel nur sehr kleine Abweichungen der Farbwerte zwischen Original und Rekonstruktion auf, die für den Menschen als Beobachter kaum wahrnehmbar sind. Konzentriert sich der mittlere quadratische Fehler dagegen auf nur 1% aller Pixel im Bild und dann auch noch an auffälligen Stellen wie z.B. an Kanten und Ecken, sticht es einem Menschen direkt ins Auge. Deshalb wird zur Beurteilung von Demosaicing-Algorithmen immer noch der menschliche Gutachter hinzugezogen, der in farbigen, texturierten und homogenen Bildbereichen einen guten visuellen Eindruck vom Ergebnis erhalten muss.

10.5 Fortgeschrittene Demosaicing-Algorithmen

In den folgenden Abschnitten wird das einfache bilineare Demosaicing (Abschnitt 10.2) schrittweise um immer aufwändigere Berechnungen erweitert und damit qualitativ verbessert. Zunächst wird im *„High Quality Linear Demosacing“* von Malvar et al. [Malv04] die Tiefpassfilter-Wirkung der bilinearen Interpolation durch Ergänzung eines Laplace-Filters (Hochpassfilter, Abschnitt 5.4) der komplementären Farbkanäle kompensiert. Bei der *„Adaptive Color Plane Interpolation“* von Hamilton & Adams [Hami97] wird mit Hilfe von Gradienten die bilineare Interpolation und die Laplace-Korrektur nur entlang von Kanten durchgeführt und nicht senkrecht dazu, so dass Artefakte reduziert werden können. Beim *„Adaptive Homogeneity Directed Demosacing“* von Hirakawa & Parks [Hira05] werden bilineare Interpolation und Laplace-Korrektur ebenfalls nur entlang von Kanten durchgeführt, allerdings wird die Auswahl zwischen horizontalen und vertikalen Kanten auf der Basis eines komplexeren Maßes als dem Gradienten berechnet, nämlich einem speziellen Homogenitätsmaß im CIE-Lab-Farbraum (Abschnitt 3.5.8). Dieses Verfahren wurde durch [Nisc10] im sogenannten *„Improved Adaptive Homogeneity Directed Demosacing“* hinsichtlich Qualität und Rauschempfindlichkeit weiter verbessert.

10.5.1 High Quality Linear Demosacing (HQLin)

Eine gravierende Schwachstelle des einfachen Demosaicing mit der bilinearen Interpolation (Abschnitt 10.2) ist seine tiefpass-filternde Wirkung durch die Mittelwertbildung benachbarten Farbwerte. Im *„High Quality Linear Demosacing“*-Algorithmus von Malvar et al. [Malv04] besteht die Idee zur Beseitigung dieser Schwachstelle darin, die durch den Tiefpass abgeschnittenen hohen Ortsfrequenzen im gesuchten Farbkanal durch einen Wert zu ersetzen, der mit Hilfe eines Hochpass-Filters (hier einer Variante des Laplace-Filters, der in Abschnitt 5.4 beschrieben wird) aus der„ *Hotpixel-Farbe“*[3] an der entsprechenden Pixelposition gewonnen wird. Die Grundannahme dabei ist, dass die Farbkanäle im Bild eine hohe Korrelation aufweisen, denn man ersetzt ja die fehlenden hohen Ortsfrequenzen eines Farbkanals durch die hohen Ortsfrequenzen eines anderen Farbkanals. Die Annahme einer hohen Korrelation zwischen den Farbkanälen ist bei natürlichen Bildern meist sehr gut erfüllt.

[3] Die „ *Hotpixel-Farbe*“ ist die Farbe, die am gerade betrachteten Pixel direkt gemessen wurde. In Bild 10.2 ist die Hotpixel-Farbe an Pixelposition 22 grün und an Pixelposition 23 rot.

Im Detail läuft der *„High Quality Linear Demosacing“*-Algorithmus folgendermaßen ab: um beispielsweise den Grünwert an der Pixelposition 23, an der die Hotpixel-Farbe Rot ist (d.h. nur R_{23} direkt gemessen wird), zu berechnen, bildet man den Mittelwert der vier umliegenden Grünwerte ($G_{13}, G_{22}, G_{24}, G_{33}$, Bild 10.6-a), was der bilinearen Interpolation gemäß 10.1 entspricht, und dazu wird der Laplace-Anteil aus der Hotpixel-Farbe Rot (R_{23}), sowie den vier umliegenden Rotwerten (R_{03}, R_{21}, R_{25}, R_{43}) addiert, d.h.:

$$G_{23} = \underbrace{\frac{1}{4}(G_{13} + G_{22} + G_{24} + G_{33})}_{\text{bilineare Interpolation}} + \underbrace{\frac{1}{8}(4 \cdot R_{23} - R_{03} - R_{21} - R_{25} - R_{43})}_{\text{Laplace-Anteil}}. \qquad (10.6)$$

Analog berechnet man den Blauwert an der Pixelposition 23, indem man den Mittelwert der vier diagonal benachbarten Blauwerte ($B_{11}, B_{20}, B_{22}, B_{31}$, Bild 10.6-b) berechnet, was der bilinearen Interpolation gemäß 10.2 entspricht. Aufgrund der diagonalen Position der benachbarten Blauwerte, die um den Faktor $\sqrt{2}$ weiter vom Hotpixel-Zentrum entfernt liegen als die horizontal und vertikal benachbarten Grünwerte, wird der Laplace-Anteil aus der Hotpixel-Farbe Rot mit einem 50% höheren Vorfaktor als in 10.6 addiert, d.h.:

$$B_{23} = \underbrace{\frac{1}{4}(B_{12} + B_{14} + B_{32} + B_{34})}_{\text{bilineare Interpolation}} + \underbrace{\frac{3}{2} \cdot \frac{1}{8}(4 \cdot R_{23} - R_{03} - R_{21} - R_{25} - R_{43})}_{\text{Laplace-Anteil}}. \qquad (10.7)$$

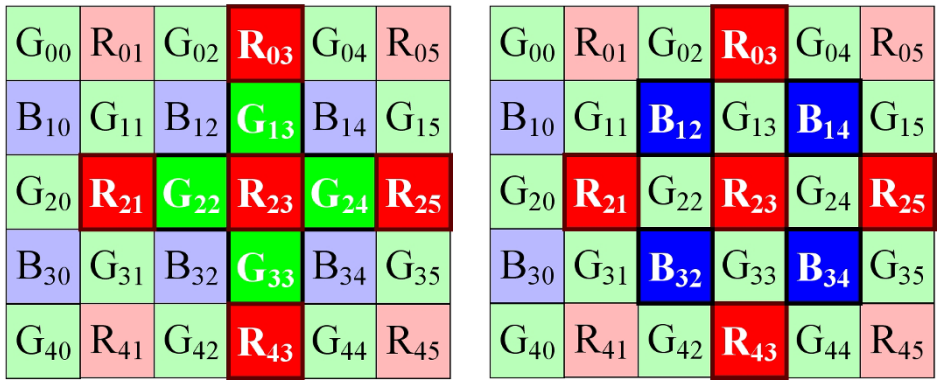


Bild 10.6: High Quality Linear Demosacing. (a) Rekonstruktion des Grünwerts an der Pixelposition 23 durch Mittelwertbildung der vier umliegenden Grünwerte $(G_{13}, G_{22}, G_{24}, G_{33})$ und Addition eines Laplace-Anteils aus den Rotwerten $(4 \cdot R_{23}, -R_{03}, -R_{21}, -R_{43}, -R_{25})$ (b) Rekonstruktion des Blauwerts an der Pixelposition 23 durch Mittelwertbildung der vier diagonal benachbarten Blauwerte $(B_{12}, B_{14}, B_{32}, B_{34})$ und Addition eines Laplace-Anteils aus den Rotwerten $\frac{3}{2} \cdot (4 \cdot R_{23}, -R_{03}, -R_{21}, -R_{43}, -R_{25})$

Der Rotwert an der Pixelposition 22, an der die Hotpixel-Farbe Grün ist, berechnet sich aus dem Mittelwert der zwei horizontal benachbarten Rotwerte (R_{21}, R_{23}, Bild 10.7-a), und einem Laplace-Anteil aus der Hotpixel-Farbe Grün. Da die bilineare Interpolation der benachbarten Rotwerte eine horizontale Vorzugsrichtung aufweist, wird der Laplace-Anteil vertikal modifiziert, d.h.:

$$R_{22} = \underbrace{\frac{1}{2}(R_{21}+R_{23})}_{\text{bilineare Interpolation}} + \underbrace{+\frac{1}{8}(5\cdot G_{22} - G_{20} - G_{24} - G_{11} - G_{13} - G_{31} - G_{33} + \frac{1}{2}G_{02} + \frac{1}{2}G_{42})}_{\text{Laplace-Anteil}}. \tag{10.8}$$

Analog berechnet man den Blauwert an der Pixelposition 22: der Unterschied zu 10.8 besteht nur darin, dass die beiden benachbarten Blauwerte vertikal liegen (Bild 10.7-b) und daher der Faltungskern für den Laplace-Anteil ebenfalls um 90° gedreht wird, d.h.:

$$B_{22} = \underbrace{\frac{1}{2}(B_{12}+B_{32})}_{\text{bilineare Interpolation}} + \underbrace{+\frac{1}{8}(5\cdot G_{22} - G_{02} - G_{42} - G_{11} - G_{13} - G_{31} - G_{33} + \frac{1}{2}G_{20} + \frac{1}{2}G_{24})}_{\text{Laplace-Anteil}}. \tag{10.9}$$

Die Verbesserung des High Quality Linear Demosacing-Algorithmus gegenüber der einfachen bilinearen Interpolation ist in Bild 10.8-c deutlich zu erkennen: es wirkt deutlich schärfer als Bild 10.8-b und der mittlere quadratische Fehler q_{MSE} reduziert sich von 173,03 auf 34,26 gemäß 10.5. Allerdings wird auch bei diesem Demosaicing-Algorithmus keinerlei Rücksicht auf Kanten oder andere Bildstrukturen genommen, so dass dort starke Artefakte (Zipper-Effekte) auftreten. Obwohl beim High Quality Linear Demosacing-Verfahren ein etwas größerer 5×5-Faltungskern benötigt wird, als bei der einfachen bilinearen Interpolation, die mit einem 3×3-Faltungskern auskommt, ist der Algorithmus immer noch relativ einfach implementierbar und schnell. Ein weiterer Vorteil dieses Demosaicing-Verfahrens ist die Möglichkeit, dass er mit Hilfe eines Vorfaktors (z.B. 2 oder 4) vor dem Laplace-Anteil zu einem Schärfefilter erweitert werden kann, wie in Abschnitt 4.8 beschrieben wird. Der große Nachteil des High Quality Linear Demosacing-Algorithmus ist die steigende Rauschempfindlichkeit. Denn durch die Hochpassfilterung mit dem Laplace-Operator werden natürlich nicht nur Kanten verstärkt, sondern auch das Rauschen. So kann der mittlere quadratische Fehler gemäß 10.5 für stark verrauschte Bilder beim High Quality Linear Demosacing mindestens genau so hoch sein, wie bei der einfachen bilinearen Interpolation.

10.5.2 Adaptive Color Plane Interpolation (ACPI)

Der nächste Schritt zur Verbesserung des Demosaicing-Ergebnisses besteht darin, Mittelungen und Gradienten nur entlang von Kanten zu berechnen und nicht senkrecht dazu. Damit lassen sich Zipper-Effekte, wie in Abschnitt 10.3 beschrieben, noch einmal deutlich verringern. Die Umsetzung dieser Idee im *„Adaptive Color Plane Interpolation"*-Verfahren von Hamilton & Adams [Hami97] geschieht in vier Schritten:

1. Berechnung des Gradienten in horizontaler und vertikaler Richtung, z.B. an der Pixelposition 23, wie in Bild 10.9 dargestellt:

$$g\_h_{23} = |G_{22} - G_{24}| + |-R_{21} + 2 \cdot R_{23} - R_{25}|, \tag{10.10}$$

$$g\_v_{23} = |G_{13} - G_{33}| + |-R_{03} + 2 \cdot R_{23} - R_{43}|. \tag{10.11}$$

2. Rekonstruktion des Grünkanals für das gesamte Bild in Abhängigkeit vom Gradienten, wie in Bild 10.9 für die Pixelposition 23 dargestellt:

$$G_{23} = \begin{cases} \frac{1}{2}(G_{22} + G_{24}) + \frac{1}{4}(-R_{21} + 2 \cdot R_{23} - R_{25}), & \text{falls } g\_v_{23} > g\_h_{23} \\ \frac{1}{2}(G_{13} + G_{33}) + \frac{1}{4}(-R_{03} + 2 \cdot R_{23} - R_{43}), & \text{sonst.} \end{cases} \tag{10.12}$$

Der erste Term in 10.12: $\frac{1}{2}(G_{22} + G_{24})$ bzw. $\frac{1}{2}(G_{13} + G_{33})$ entspricht der bilinearen Interpolation in der Kantenrichtung, der zweite Term: $\frac{1}{4}(-R_{21}+2\cdot R_{23}-R_{25})$ bzw. $\frac{1}{4}(-R_{03}+2 \cdot R_{23} - R_{43})$ entspricht einem 1-dimensionalen Laplace-Filter in der Kantenrichtung.

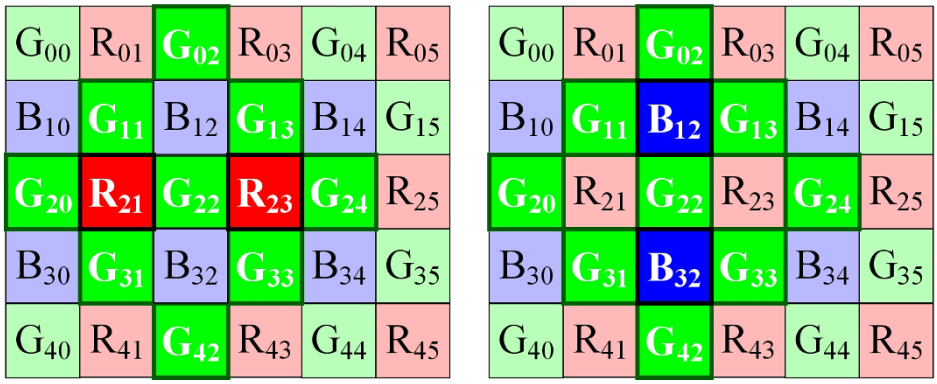
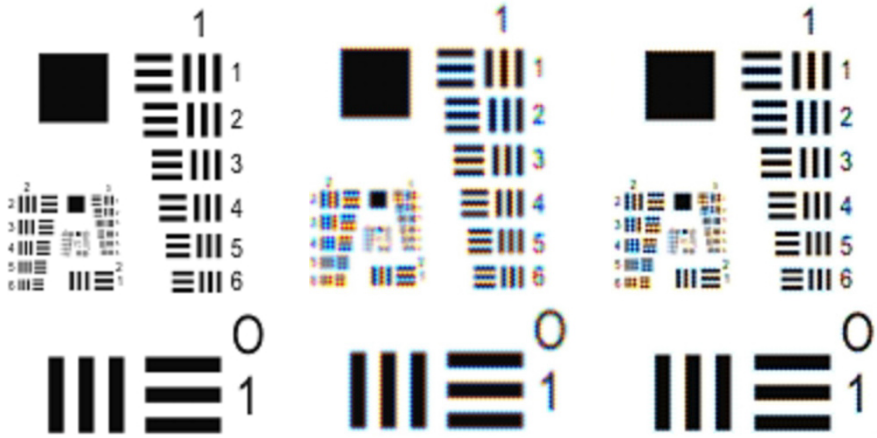


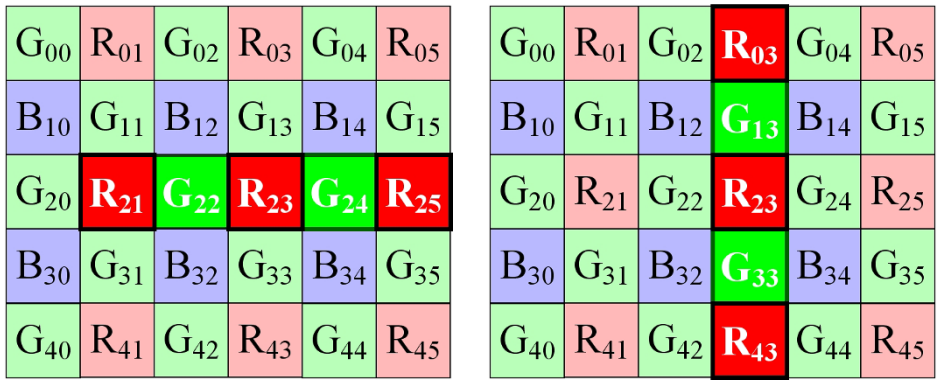
Bild 10.7: High Quality Linear Demosacing. (a) Rekonstruktion des Rotwerts an der Pixelposition 22 durch Mittelwertbildung der zwei benachbarten Rotwerte (R_{21}, R_{23}) und Addition eines Laplace-Anteils aus den Grünwerten $(5 \cdot G_{22}, -G_{20}, -G_{24}, -G_{11}, -G_{13}, -G_{31}, -G_{33}, \frac{1}{2}G_{02}, \frac{1}{2}G_{42})$ (b) Rekonstruktion des Blauwerts an der Pixelposition 22 durch Mittelwertbildung der zwei benachbarten Blauwerte (B_{12}, B_{32}) und Addition eines Laplace-Anteils aus den Grünwerten $(5 \cdot G_{22}, -G_{02}, -G_{42}, -G_{11}, -G_{13}, -G_{31}, -G_{33}, \frac{1}{2}G_{20}, \frac{1}{2}G_{24})$



(a) (b) $q_{MSE} = 173,03$ (c) $q_{MSE} = 34,26$

Bild 10.8: Vergleich (a) Originalbild, (b) durch *„bilineare Interpolation“* rekonstruiertes Bild und (c) durch *„High Quality Linear Demosacing“* rekonstruiertes Bild.

3. Berechnung des Gradienten in diagonaler Richtung mit Hilfe des rekonstruierten



(a) (b)

Bild 10.9: Adaptive Color Plane Interpolation, 1. Schritt: Berechnung der Gradienten in (a) horizontaler und (b) vertikaler Richtung. 2. Schritt: Rekonstruktion des Grünkanals, falls (a) der Gradient in vertikaler Richtung größer ist und falls (b) der Gradient in horizontaler Richtung größer ist.

Grünkanals, z.B. an der Pixelposition 23, wie in Bild 10.10 dargestellt:

$$g\_dn_{23} = |B_{12} - B_{34}| + | - G_{12} + 2 \cdot G_{23} - G_{34}|, \tag{10.13}$$

$$g\_dp_{23} = |B_{14} - B_{32}| + | - G_{14} + 2 \cdot G_{23} - G_{32}|. \tag{10.14}$$

4. Rekonstruktion des Blaukanals in Abhängigkeit vom diagonalen Gradienten, wie in Bild 10.10 für die Pixelposition 23 dargestellt:

$$B_{23} = \begin{cases} \frac{1}{2}(B_{12} + B_{34}) + \frac{1}{4}(-G_{12} + 2 \cdot G_{23} - G_{34}), & \text{falls } g\_dn_{23} > g\_dp_{23} \\ \frac{1}{2}(B_{14} + B_{32}) + \frac{1}{4}(-G_{14} + 2 \cdot G_{23} - G_{32}), & \text{sonst.} \end{cases} \tag{10.15}$$

Die Schritte 3 und 4 laufen für blaue Hotpixel-Positionen (z.B. 34) analog ab. An den grünen Hotpixel-Positionen (z.B. 22) laufen die Schritte 3 und 4 leicht modifiziert ab, wie in Bild 10.11 dargestellt:

$$R_{22} = \frac{1}{2}(R_{21} + R_{23}) + \frac{1}{2}(-G_{21} + 2 \cdot G_{22} - G_{23}), \tag{10.16}$$

$$B_{22} = \frac{1}{2}(B_{12} + B_{32}) + \frac{1}{2}(-G_{12} + 2 \cdot G_{22} - G_{32}). \tag{10.17}$$

Auffällig ist hier, dass bei der Rekonstruktion der Rot- und Blauwerte an den grünen Hotpixel-Positionen keine Fallunterscheidung auf Basis des Gradienten mehr durchgeführt wird, da an diesen Stellen die lineare Interpolation nur in jeweils einer Richtung direkt

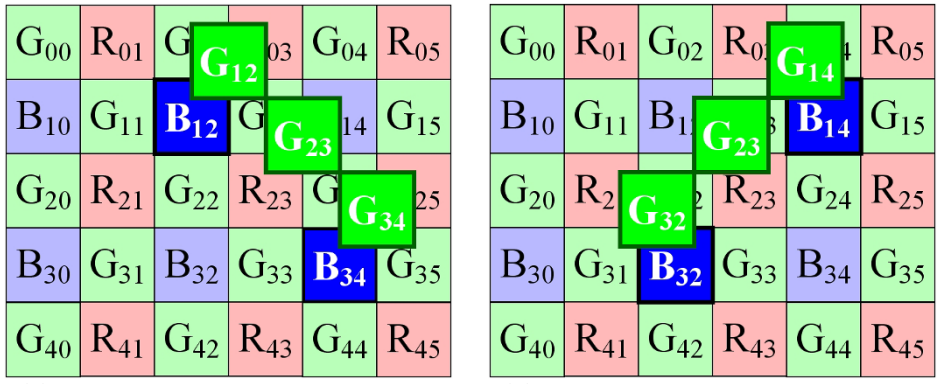


Bild 10.10: Adaptive Color Plane Interpolation, 3. Schritt: Berechnung des Gradienten in (a) diagonal negativer Richtung (d.h von links unten nach rechts oben) und (b) in diagonal positiver Richtung (d.h von links oben nach rechts unten). 4. Schritt: Rekonstruktion des Blaukanals, falls (a) der Gradient in diagonal positiver Richtung größer ist und falls (b) der Gradient in diagonal negativer Richtung größer ist.

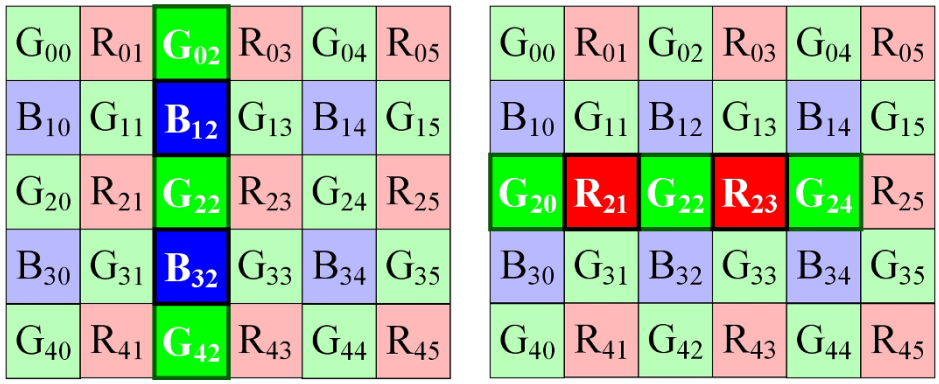


Bild 10.11: Adaptive Color Plane Interpolation, 3. und 4. Schritt an grünem Hotpixel: die Berechnung des Gradienten entfällt. (a) Rekonstruktion des Blaukanals in vertikaler Richtung (b) Rekonstruktion des Rotkanals in horizontaler Richtung

möglich ist (an der Pixelposition 22 stehen rote Hotpixel nur in horizontaler Richtung zur Verfügung und blaue Hotpixel nur in vertikaler Richtung).

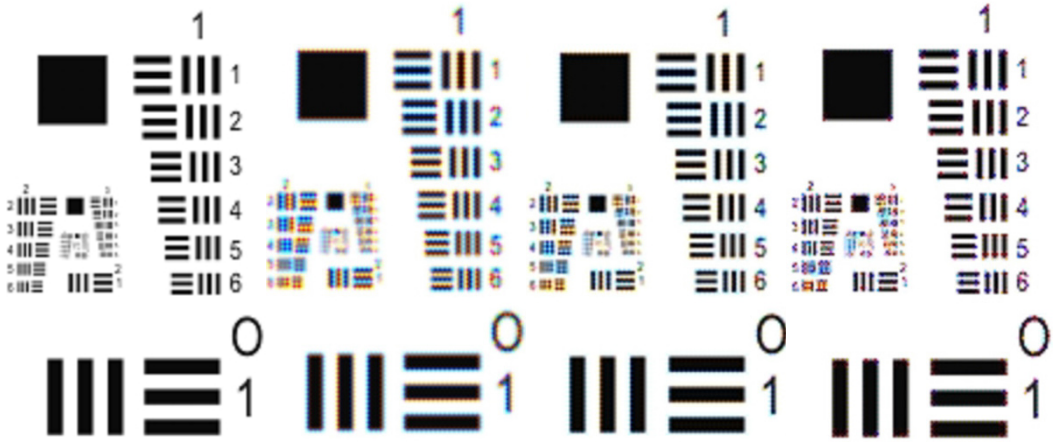
Eine Verbesserung des ACPI-Demosaicing-Verfahrens kann man erreichen, indem man anstatt der Formeln 10.16 und 10.17 auch an den grünen Hotpixel-Positionen die Gradienten berücksichtigt:

$$R_{22} = \begin{cases} \frac{1}{2}(R_{21} + R_{23}) + \frac{1}{2}(-G_{21} + 2 \cdot G_{22} - G_{23}), & \text{falls } g\_v_{22} > g\_h_{22} \\ \frac{1}{2}(R_{12} + R_{32}) + \frac{1}{2}(-G_{12} + 2 \cdot G_{22} - G_{32}), & \text{sonst.} \end{cases} \tag{10.18}$$

$$B_{22} = \begin{cases} \frac{1}{2}(B_{21} + B_{23}) + \frac{1}{2}(-G_{21} + 2 \cdot G_{22} - G_{23}), & \text{falls } g\_v_{22} > g\_h_{22} \\ \frac{1}{2}(B_{12} + B_{32}) + \frac{1}{2}(-G_{12} + 2 \cdot G_{22} - G_{32}), & \text{sonst.} \end{cases} \tag{10.19}$$

Voraussetzung für diese verbesserte Variante des ACPI-Demosaicing-Verfahrens ist allerdings, dass zuerst an allen roten Hotpixel-Positionen die Blauwerte (z.B. $(B_{21} + B_{23})$ in Bild 10.11-a) und an allen blauen Hotpixel-Positionen die Rotwerte (z.B. $(R_{12} + R_{32})$ in Bild 10.11-b gemäß 10.15) für das gesamte Bild berechnet werden.

Beim ACPI-Demosaicing-Verfahren wird erstmals der Bildinhalt in Form von Kanten und Linien mittels Gradienten berücksichtigt, so dass hier die Zipper-Effekte deutlich zurückgehen und die Verbesserung gegenüber dem High Quality Linear Demosacing-Algorithmus deutlich sichtbar (Bild 10.12) und messbar wird (der mittlere quadratische Fehler q_{MSE} reduziert sich von 34,26 auf 11,12 gemäß 10.5). Andererseits produziert das ACPI-Demosaicing-Verfahren an Eckpunkten sehr deutliche Farbartefakte (Bild 10.12-d) und auch die Rauschempfindlichkeit nimmt deutlich zu. Der Rechenaufwand verdoppelt sich in etwa, da jetzt zusätzlich pro Farbwert zwei Gradienten vorab berechnet werden



(a) (b) $q_{MSE} = 173,03$ (c) $q_{MSE} = 34,26$ (d) $q_{MSE} = 11,12$

Bild 10.12: Vergleich (a) Originalbild, (b) durch *„bilineare Interpolation“* rekonstruiertes Bild, (c) durch *„High Quality Linear Demosacing“* rekonstruiertes Bild, (d) durch *„Adaptive Color Plane Interpolation“* rekonstruiertes Bild.

müssen und die Reihenfolge der Berechnungen (Grünkanal zuerst, danach Rot- und Blaukanal) relevant wird.

10.5.3 Adaptive Homogeneity Directed Demosacing (AHD)

Ein weiterer Sprung in der Qualität wurde 2005 durch das *„Adaptive Homogeneity Directed Demosacing“* von Hirakawa & Parks [Hira05] erzielt. Die Grundidee des Verfahrens ist zunächst die gleiche wie beim ACPI-Demosaicing-Verfahren aus dem vorigen Abschnitt 10.5.2: Interpolation und Laplace-Korrektur nur entlang von Kanten. Allerdings werden beim AHD-Demosaicing zunächst zwei vollständige RGB-Bilder rekonstruiert: eines mit horizontaler und eines mit vertikaler Vorzugsrichtung. Um die spätere Entscheidung für den Menschen als Beobachter zu optimieren, werden beide Bilder in den CIE-Lab-Farbraum (Abschnitt 3.5.8) transformiert, denn gleiche Abstände in diesem Farbraum empfindet der Mensch als gleiche Farbunterschiede. Aus den beiden Bildern im CIE-Lab-Farbraum werden anschließend spezielle Homogenitätsmaße abgeleitet, die Auskunft darüber geben, in welcher der beiden Vorzugsrichtungen die Farbwerte gleichmäßiger verlaufen. Die Entscheidung für eine Vorzugsrichtung fällt erst ganz am Ende: ist die Homogenität in horizontaler Richtung größer als in vertikaler, so werden die horizontal rekonstruierten RGB-Farbwerte verwendet, im umgekehrten Fall die vertikal rekonstruierten Farbwerte und bei dem eher seltenen Fall gleicher Homogenitätswerte in horizontaler und vertikaler Richtung werden beide RGB-Farbwerte gemittelt.

Im Detail sieht der *„Adaptive Homogeneity Directed Demosacing“*-Algorithmus folgen-

de Schritte vor:

A10.2: Adaptive Homogeneity Directed Demosacing.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y,n)), n = 0,1,2$ ist ein dreikanaliges RGB-Farbbild (Eingabebild).

◇ $\mathbf{s}_e(x,y,n) = (R_{xy}, G_{xy}, B_{xy})^T$ ist ein RGB-Farbvektor von $\mathbf{S}_e$ an der Bildzeile x und der Bildspalte y mit $G = \{0,1,...,255\}$ als Grauwertmenge.

◇ $\mathbf{S}_b = (s_b(x,y))$ ist ein einkanaliges Grauwertbild, das durch ein Bayer-Farbfilter (Algorithmus **A10.1**) aus dem Eingabebild $\mathbf{S}_e$ hervorgegangen ist (Rohbild).

◇ $\mathbf{S}_a = (s_a(x,y,n)), n = 0,1,2$ ist ein dreikanaliges RGB-Farbbild (Ausgabebild).

◇ Grundannahme: die drei Farbkanäle im Bild weisen eine hohe Korrelation auf (bei natürlichen Bildern ist diese Annahme meist sehr gut erfüllt).

Algorithmus:

(1) Rekonstruktion des Grünkanals, und zwar je einmal für die horizontale und die vertikale Vorzugsrichtung (analog dem ACPI-Demosaicing-Verfahren gemäß 10.12, jedoch ohne Fallunterscheidung):

(1a) Rote Hotpixel (gerade Bildzeilen $x = 0,2,4,...,L-2$, ungerade Bildspalten $y = 1,3,5,...,R-1$), z.B. an der Pixelposition 23 (Bild 10.9):

$$G_{23\_h} = \frac{1}{2}(G_{22} + G_{24}) + \frac{1}{4}(-R_{21} + 2 \cdot R_{23} - R_{25}), \tag{10.20}$$

$$G_{23\_v} = \frac{1}{2}(G_{13} + G_{33}) + \frac{1}{4}(-R_{03} + 2 \cdot R_{23} - R_{43}). \tag{10.21}$$

(1b) Blaue Hotpixel (ungerade Bildzeilen $x = 1,3,5,...,R-1$, gerade Bildspalten $y = 0,2,4,...,L-2$), z.B. an der Pixelposition 32:

$$G_{32\_h} = \frac{1}{2}(G_{31} + G_{33}) + \frac{1}{4}(-B_{30} + 2 \cdot B_{32} - B_{34}), \tag{10.22}$$

$$G_{32\_v} = \frac{1}{2}(G_{22} + G_{42}) + \frac{1}{4}(-B_{12} + 2 \cdot B_{32} - B_{52}). \tag{10.23}$$

(1c) Abschließend erfolgt noch eine Medianbildung (Abschnitt 6.3) in der Vorzugsrichtung, um vereinzelte Ausreißer zu eliminieren.

$$\begin{aligned} G_{23\_h} &= Median(G_{22}, G_{23\_h}, G_{24}), \\ G_{23\_v} &= Median(G_{13}, G_{23\_v}, G_{33}), \\ G_{32\_h} &= Median(G_{31}, G_{32\_h}, G_{33}), \\ G_{32\_v} &= Median(G_{22}, G_{32\_v}, G_{42}). \end{aligned} \tag{10.24}$$

(2) Rekonstruktion von Rot- und Blaukanal, und zwar je einmal für die horizontale und die vertikale Vorzugsrichtung:

(2a) Rote Hotpixel (gerade Bildzeilen $x = 0, 2, 4, ..., L-2$, ungerade Bildspalten $y = 1, 3, 5, ..., R-1$), z.B. an der Pixelposition 23 (Bild 10.13-a):

$$\begin{aligned} B_{23\_h} &= \frac{1}{4}(B_{12} + B_{14} + B_{32} + B_{34}) + \\ &\quad \frac{1}{4}(-G_{12\_h} - G_{14\_h} + 4 \cdot G_{23\_h} - G_{32\_h} - G_{34\_h}), \end{aligned} \tag{10.25}$$

$$\begin{aligned} B_{23\_v} &= \frac{1}{4}(B_{12} + B_{14} + B_{32} + B_{34}) + \\ &\quad \frac{1}{4}(-G_{12\_v} - G_{14\_v} + 4 \cdot G_{23\_v} - G_{32\_v} - G_{34\_v}). \end{aligned} \tag{10.26}$$

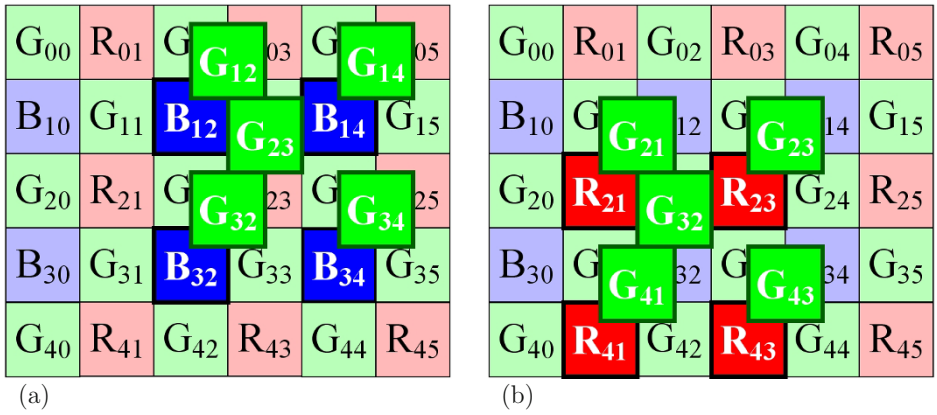


Bild 10.13: Adaptive Homogeneity Directed Demosacing, Schritte 2a und 2b: (a) Rotes Hotpixel: Rekonstruktion des Blaukanals, (b) Blaues Hotpixel: Rekonstruktion des Rotkanals.

(2b) Blaue Hotpixel (ungerade Bildzeilen $x = 1, 3, 5, ..., R-1$, gerade Bildspalten $y = 0, 2, 4, ..., L-2$), z.B. an der Pixelposition 32 (Bild 10.13-b):

$$R_{32\_h} = \frac{1}{4}(R_{21} + R_{23} + R_{41} + R_{43}) + \frac{1}{4}(-G_{21\_h} - G_{23\_h} + 4 \cdot G_{32\_h} - G_{41\_h} - G_{43\_h}), \quad (10.27)$$

$$R_{32\_v} = \frac{1}{4}(R_{21} + R_{23} + R_{41} + R_{43}) + \frac{1}{4}(-G_{21\_v} - G_{23\_v} + 4 \cdot G_{32\_v} - G_{41\_v} - G_{43\_v}). \quad (10.28)$$

(2c) Grüne Hotpixel (gerade Bildzeilen $x = 0, 2, 4, ..., L-2$ und gerade Bildspalten $y = 0, 2, 4, ..., L-2$, oder ungerade Bildzeilen $x = 1, 3, 5, ..., R-1$ und ungerade Bildspalten $y = 1, 3, 5, ..., R-1$), z.B. an der Pixelposition 33 (Bild 10.14):

$$R_{33\_h} = \frac{1}{2}(R_{23} + R_{43}) + \frac{1}{2}(-G_{23\_h} + 2 \cdot G_{33} - G_{43\_h}), \quad (10.29)$$

$$R_{33\_v} = \frac{1}{2}(R_{23} + R_{43}) + \frac{1}{2}(-G_{23\_v} + 2 \cdot G_{33} - G_{43\_v}), \quad (10.30)$$

$$B_{33\_h} = \frac{1}{2}(B_{32} + B_{34}) + \frac{1}{2}(-G_{32\_h} + 2 \cdot G_{33} - G_{34\_h}), \quad (10.31)$$

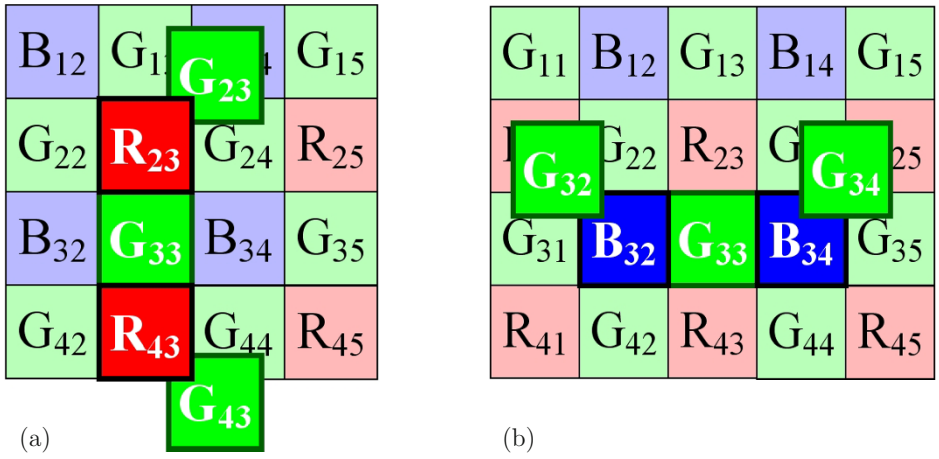


Bild 10.14: Adaptive Homogeneity Directed Demosacing, Schritt 2c an grünem Hotpixel: (a) Rekonstruktion des Rotkanals, (b) Rekonstruktion des Blaukanals.

$$B_{33\_v} = \frac{1}{2}(B_{32} + B_{34}) + \frac{1}{2}(-G_{32\_v} + 2 \cdot G_{33} - G_{34\_v}). \tag{10.32}$$

(3) Umrechnung der RGB-Farbwerte über die (X,Y,Z)-CIE-Normfarbwerte in die CIE-Lab-Farbwerte wie in Abschnitt 3.5.8 beschrieben.

(4) Erstellung der Homogenitätskarten für die horizontale und die vertikale Vorzugsrichtung:

(4a) Berechnung der absoluten Differenzen der L-Werte zu den jeweiligen vier Nachbarpositionen (rechts, links, oben, unten), z.B. an der Pixelposition 33:
horizontale Vorzugsrichtung:

$$\begin{aligned} L_{33\_h\_r} &= |L_{34\_h} - L_{33\_h}|, \quad (rechts) \\ L_{33\_h\_l} &= |L_{32\_h} - L_{33\_h}|, \quad (links) \\ L_{33\_h\_o} &= |L_{43\_h} - L_{33\_h}|, \quad (oben) \\ L_{33\_h\_u} &= |L_{23\_h} - L_{33\_h}|. \quad (unten) \end{aligned} \tag{10.33}$$

vertikale Vorzugsrichtung:

$$\begin{aligned} L_{33\_v\_r} &= |L_{34\_v} - L_{33\_v}|, \quad (rechts) \\ L_{33\_v\_l} &= |L_{32\_v} - L_{33\_v}|, \quad (links) \\ L_{33\_v\_o} &= |L_{43\_v} - L_{33\_v}|, \quad (oben) \\ L_{33\_v\_u} &= |L_{23\_v} - L_{33\_v}|. \quad (unten) \end{aligned} \tag{10.34}$$

(4b) Berechnung der Differenzquadrate der ab-Werte zu den jeweiligen vier Nachbarpositionen (rechts, links, oben, unten), z.B. an der Pixelposition 33:
horizontale Vorzugsrichtung:

$$\begin{aligned} ab_{33\_h\_r} &= (a_{34\_h} - a_{33\_h})^2 + (b_{34\_h} - b_{33\_h})^2, \quad (rechts) \\ ab_{33\_h\_l} &= (a_{32\_h} - a_{33\_h})^2 + (b_{32\_h} - b_{33\_h})^2, \quad (links) \\ ab_{33\_h\_o} &= (a_{43\_h} - a_{33\_h})^2 + (b_{43\_h} - b_{33\_h})^2, \quad (oben) \\ ab_{33\_h\_u} &= (a_{23\_h} - a_{33\_h})^2 + (b_{23\_h} - b_{33\_h})^2. \quad (unten) \end{aligned} \tag{10.35}$$

vertikale Vorzugsrichtung:

$$\begin{aligned} ab_{33\_v\_r} &= (a_{34\_v} - a_{33\_v})^2 + (b_{34\_v} - b_{33\_v})^2, \quad (rechts) \\ ab_{33\_v\_l} &= (a_{32\_v} - a_{33\_v})^2 + (b_{32\_v} - b_{33\_v})^2, \quad (links) \\ ab_{33\_v\_o} &= (a_{43\_v} - a_{33\_v})^2 + (b_{43\_v} - b_{33\_v})^2, \quad (oben) \\ ab_{33\_v\_u} &= (a_{23\_v} - a_{33\_v})^2 + (b_{23\_v} - b_{33\_v})^2. \quad (unten) \end{aligned} \tag{10.36}$$

(4c) Berechnung der Grenzwerte als Vergleichsmaß für die Homogenität, z.B. an der Pixelposition 33:

$$\begin{aligned} L_{33\_eps} &= min(max(L_{33\_h\_r}, L_{33\_h\_l}), max(L_{33\_v\_o}, L_{33\_v\_u})), \\ ab_{33\_eps} &= min(max(ab_{33\_h\_r}, ab_{33\_h\_l}), max(ab_{33\_v\_o}, ab_{33\_v\_u})). \end{aligned} \tag{10.37}$$

(4d) Berechnung der Homogenitätswerte für die horizontale und die vertikale Vorzugsrichtung, z.B. an der Pixelposition 33 (falls die logische Aussage WAHR ist, wird das Ergebnis 1 gesetzt, ansonsten 0):

$$\begin{aligned} Homo_{33\_h} &= \sum_{x=r,l,o,u} (L_{33\_h\_x} \leq L_{33\_eps} \wedge ab_{33\_h\_x} \leq ab_{33\_eps}), \\ Homo_{33\_v} &= \sum_{x=r,l,o,u} (L_{33\_v\_x} \leq L_{33\_eps} \wedge ab_{33\_v\_x} \leq ab_{33\_eps}). \end{aligned} \tag{10.38}$$

(5) Entscheidung, ob die horizontale oder die vertikale Vorzugsrichtung ausgewählt wird, oder ob beide gemittelt werden:

(5a) Tiefpass-Filterung der beiden Homogenitätskarten mit einem 3·3-gleitenden Mittelwert:

$$\begin{aligned} H_{xy\_h} &= \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} Homo_{x+i,y+j\_h}, \\ H_{xy\_v} &= \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} Homo_{x+i,y+j\_v}. \end{aligned} \tag{10.39}$$

(5b) Auswahl des vertikalen oder horizontalen Farbwerts, oder Mittelung:

$$\begin{aligned} \mathbf{s}_a(x,y,n) &= (R_{xy}, G_{xy}, B_{xy})^T = \\ &= \begin{cases} (R_{xy\_h}, G_{xy\_h}, B_{xy\_h})^T, & \text{falls } H_{xy\_h} > H_{xy\_v} \\ \begin{pmatrix} \frac{1}{2}(R_{xy\_h} + R_{xy\_v}) \\ \frac{1}{2}(G_{xy\_h} + G_{xy\_v}) \\ \frac{1}{2}(B_{xy\_h} + B_{xy\_v}) \end{pmatrix}, & \text{falls } H_{xy\_h} = H_{xy\_v} \\ (R_{xy\_v}, G_{xy\_v}, B_{xy\_v})^T, & \text{sonst.} \end{cases} \end{aligned} \tag{10.40}$$

(6) Optional bietet der AHD-Algorithmus noch eine n-fache Median-Filterung (Abschnitt 6.3) der Farbdifferenzen $(R-G)$ und $(B-G)$ als Postprocessing an, wobei

der Filterkern frei wählbar ist, wie z.B. eine 3×3-Umgebung des Bildpunktes:

$$\begin{aligned} \mathbf{s}_a(x,y,n) &= (R_{xy}, G_{xy}, B_{xy})^T = \\ &= \begin{pmatrix} G_{xy} + Median_{i,j=-1,0,+1}(R_{x+i,y+j} - G_{x+i,y+j}) \\ \\ \frac{1}{2}(R_{xy} + B_{xy} \\ -Median_{i,j=-1,0,+1}(R_{x+i,y+j} - G_{x+i,y+j}) \\ -Median_{i,j=-1,0,+1}(B_{x+i,y+j} - G_{x+i,y+j})) \\ \\ G_{xy} + Median_{i,j=-1,0,+1}(B_{x+i,y+j} - G_{x+i,y+j}) \end{pmatrix} \end{aligned} \tag{10.41}$$

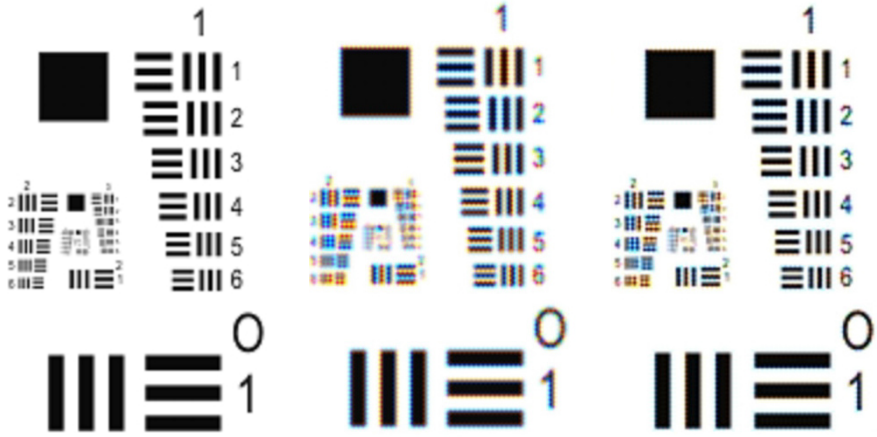
Man beachte, dass beim 6. Schritt auch die ursprünglich gemessenen Farbwerte (die sogenannten Hotpixel) verändert werden.

Ende des Algorithmus

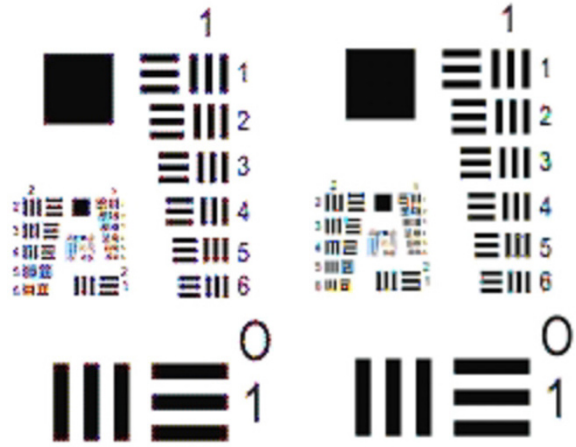
Das positive Ergebnis dieses doch relativ aufwändigen AHD-Demosaicing-Algorithmus kann man in Bild 10.15 sehen und messen: es treten fast keine Zipper-Effekte oder sonstigen Farbfehler an Kanten auf und der mittlere quadratische Fehler q_{MSE} (gemäß 10.5) reduziert sich noch einmal erheblich von $q_{MSE}(ACPI) = 11,12$ auf $q_{MSE}(AHD) = 7,11$. Für den menschlichen Beobachter ergeben sich insbesondere bei natürlichen Bildern, wie in 10.20, sehr gute Resultate. So werden die Farbartefakte an den Kanten (in Bild 10.20-b,c,d die Farbsäume an den weißen Zaunlatten) fast vollständig vermieden.

Leider bringt der AHD-Demosaicing-Algorithmus auch einige Nachteile mit sich. Die Rauschempfindlichkeit ist relativ hoch: bei Bildern, wie 10.17, die mit unkorrelierten, unbunten[4] Zufallswerten verrauscht wurden, treten starke Farbartefakte auf. Die Qualität der Rekonstruktion bei verrauschten Bildern ist beim aufwändigen AHD-Demosaicing-Algorithmus kaum höher als beim einfachen *„High Quality Linear Demosacing“* (Abschnitt 10.5.1), wie man auch am mittleren quadratischen Fehler ($q_{MSE}(AHD) = 51,03$ bzw. $q_{MSE}(HQLin) = 51,12$) sehen kann. Außerdem treten in homogenen Bildbereichen Artefakte in Form von Substrukturen auf, die menschlichen Beobachtern trotz geringer Farbunterschiede ins Auge fallen. Ursache dafür ist die Tiefpass-Filterung der Homogenitätskarten (Schritt 5a in Algorithmus **A10.2**), die eine kantenverfolgende Wirkung hat, was im Fall von Kanten und Linien im Bild sehr vorteilhaft ist, aber in homogenen Bildbereichen zu Substrukturen führt. Der Rechenaufwand steigt gegenüber dem im vorigen Abschnitt 10.5.2 dargestellten ACPI-Verfahren noch einmal um eine Größenordnung an, da die Rekonstruktion parallel auf zwei Bildern, einem mit horizontaler und einem mit vertikaler Vorzugsrichtung, abläuft und dabei aufwändige Filteroperationen (Median und gleitender Mittelwert) sowie eine CIE-Lab-Farbraumkonvertierung stattfindet. Allerdings ist die verfügbare Rechenleistung insbesondere im Sektor der Grafikkarten in den letzten Jahren immens angestiegen, so dass die Rekonstruktion eines Bildes mit einer HDTV-Auflösung von 1920×1080 Bildpunkten in weniger als 10 Millisekunden möglich ist.

[4] d.h. der gleiche Zufallswert wird auf alle drei Farbwerte eines Bildpunktes addiert



(a) (b) $q_{MSE} = 173,03$ (c) $q_{MSE} = 34,26$



(d) $q_{MSE} = 11,12$ (e) $q_{MSE} = 7,11$

Bild 10.15: Demosaicing-Vergleich (a) Originalbild, (b) *„bilineare Interpolation“*, (c) *„High Quality Linear Demosacing“*, (d) *„Adaptive Color Plane Interpolation“*, (e) *„Adaptive Homogeneity Directed Demosaicing“*.

10.5.4 Improved Adaptive Homogeneity Directed Demosacing (IAHD)

Das im vorigen Abschnitt 10.5.3 dargestellte, weit fortgeschrittene AHD-Demosaicing-Verfahren konnte durch [Nisc10] im sogenannten *„Improved Adaptive Homogeneity Directed Demosacing“* weiter verbessert werden, so dass die Rauschempfindlichkeit deutlich geringer wird, Substruktur-Artefakte in homogenen Bildbereichen nicht mehr auftreten und die Demosaicing-Qualität generell weiter erhöht wird.

Die Verbesserungen greifen an vier verschiedenen Stellen des AHD-Demosaicing-Algorithmus **A10.2** an:

- Schritt 2c: Verbesserte Rekonstruktion der Rot-/Blauwerte an grünen Hotpixeln
- nach Schritt 2c: n-fache Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ **vor** der Transformation in den CIE-Lab-Farbraum
- Schritt 5b: Nutung der Homogenitätsinformation für eine weichere Entscheidung zwischen dem Horizontal- und dem Vertikal-Kanal
- Schritt 6: n-fache Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ **nach** der horizontal-/vertikal-Entscheidung, wobei die Hotpixel-Farben unverändert bleiben.

10.5.4.1 Verbesserte Rekonstruktion der Rot- und Blauwerte an grünen Hotpixeln

Die Grundidee des *„Adaptive Homogeneity Directed Demosacing“* von Hirakawa und Parks [Hira05] besteht in der strikt getrennten Berechnung von Farbwerten für die horizontale und die vertikale Vorzugsrichtung. Diese Grundidee wird aber in Schritt 2c des Algorithmus **A10.2** verletzt, indem in grün-blauen (ungeraden) Zeilen an den grünen Hotpixeln (z.B. G_{33}) die Berechnung des Rot-Werts in horizontaler Richtung (R_{33_h}, Formel 10.29) mit Hilfe der Farbdifferenzen $(R_{23}-G_{23\_h}+2\cdot G_{33}+R_{43}-G_{43\_h})$ in vertikaler Richtung erfolgt (Bild 10.14-a), bzw. die Berechnung des Blau-Werts in vertikaler Richtung (B_{33_v}, Formel 10.32) mit Hilfe der Farbdifferenzen $(B_{32}-G_{32\_h}+2\cdot G_{33}+B_{34}-G_{34\_h})$ in horizontaler Richtung erfolgt (Bild 10.14-b). In grün-roten (geraden) Zeilen wird die Grundidee an den grünen Hotpixeln (z.B. G_{22}) verletzt, da die Berechnung des Rot-Werts in vertikaler Richtung ($R_{22\_v}=\frac{1}{2}(R_{21}-G_{21\_v}+2\cdot G_{22}+R_{23}-G_{23\_v})$) mit Hilfe der Farbdifferenzen in horizontaler Richtung erfolgt (Bild 10.16-a), bzw. die Berechnung des Blau-Werts in horizontaler Richtung ($B_{22\_h}=\frac{1}{2}(B_{12}-G_{12\_h}+2\cdot G_{22}+B_{32}-G_{32\_h})$) mit Hilfe der Farbdifferenzen in vertikaler Richtung erfolgt (Bild 10.16-b).

Die Lösung des Problems besteht nun darin, für die Berechnung des horizontalen Rot- bzw. Blau-Werts auch die horizontalen Farbdifferenzen heranzuziehen, sowie für die Berechnung des vertikalen Rot- bzw. Blau-Werts auch die vertikalen Farbdifferenzen, d.h. die Formeln 10.29 und 10.32 in Schritt 2c werden modifiziert zu:

$$R_{33\_h}=\frac{1}{2}(R_{32\_h}+R_{34\_h})+\frac{1}{2}(-G_{32\_h}+2\cdot G_{33}-G_{34\_h}), \tag{10.42}$$

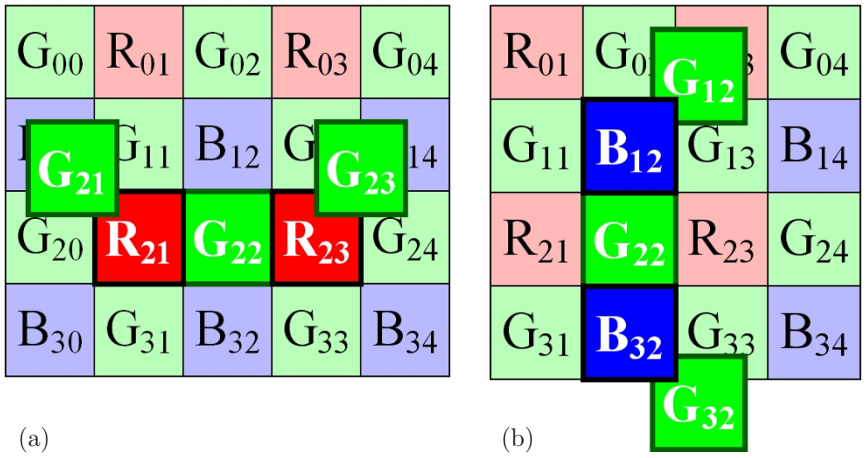


Bild 10.16: Adaptive Homogeneity Directed Demosacing, Schritt 2c an grünem Hotpixel (z.B. G_{22}): (a) Berechnung des vertikalen Rotwerts mit horizontalen Farbdifferenzen, (b) Berechnung des horizontalen Blauwerts mit vertikalen Farbdifferenzen.

$$B_{33\_v} = \frac{1}{2}(B_{23\_v} + B_{43\_v}) + \frac{1}{2}(-G_{23\_v} + 2 \cdot G_{33} - G_{43\_v}), \tag{10.43}$$

$$R_{22\_v} = \frac{1}{2}(R_{12\_v} + R_{32\_v}) + \frac{1}{2}(-G_{12\_v} + 2 \cdot G_{22} - G_{32\_v}), \tag{10.44}$$

$$B_{22\_h} = \frac{1}{2}(B_{21\_h} + B_{23\_h}) + \frac{1}{2}(-G_{21\_h} + 2 \cdot G_{22} - G_{23\_h}). \tag{10.45}$$

Voraussetzung dafür ist allerdings, dass zuerst an roten Hotpixeln (z.B. R_{21}, R_{23}, R_{43}) die fehlenden Blau-Werte (Schritt 2a), bzw. an blauen Hotpixeln (z.B. B_{12}, B_{32}, B_{34}) die fehlenden Rot-Werte berechnet werden (Schritt 2b) und erst danach die fehlenden Rot- und Blau-Werte an den grünen Hotpixeln. Im originalen Algorithmus von Hirakawa und Parks spielt diese Reihenfolge keine Rolle.

Durch die vorgeschlagene Lösung verbessert sich das Demosaicing-Ergebnis deutlich messbar und sichtbar. Die Abweichungen der Rot- und Blau-Werte von den originalen Werten verringern sich deutlich (ca. $10-15\%$ niedrigere mittlere quadratische Abweichungen), wobei sich die Grün-Werte kaum merklich verändern (ca. $\pm$ 2% Veränderungen bei den mittleren quadratischen Abweichungen). Dies gilt ebenso bei stark verrauschten Bildern. Vom visuellen Eindruck her verringern sich die Farbabweichungen durch die vorgeschlagene Lösung deutlich sichtbar.

10.5.4.2 Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ vor der Transformation in den CIE-Lab-Farbraum

Im Schritt 6 des AHD-Demosaicing-Algorithmus **A10.2** wird die Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ **nach** der horizontal-/vertikal-Entscheidung durch-

geführt, und zwar an allen Pixeln. Die Verbesserung des AHD-Demosaicing-Algorithmus besteht nun darin, dass einerseits die Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ **vor** Schritt 3, d.h. der Transformation in den CIE-Lab-Farbraum durchgeführt wird, und andererseits die Farbwerte, die Hotpixel sind (also vom Sensor direkt gemessen wurden, wie z.B.$G_{22}, G_{33}, R_{23}, B_{32}$), bei der Median-Filterung[5] unverändert bleiben:

Neuer und zusätzlicher Schritt 2d im Algorithmus **A10.2** ist die n-fache, modifizierte Median-Filterung für alle Bildpunkte:

$$R_{xy} = \begin{cases} R_{xy} & \text{falls} \quad R_{xy} \text{ ein Hotpixel} \\ G_{xy} + Median_{i,j=-1,0,+1}(R_{x+i,y+j} - G_{x+i,y+j}) & \text{sonst} \end{cases} \tag{10.46}$$

$$G_{xy} = \begin{cases} G_{xy} & \text{falls} \quad G_{xy} \text{ ein Hotpixel} \\ \frac{1}{2}(R_{xy} + B_{xy} & \\ -Median_{i,j=-1,0,+1}(R_{x+i,y+j} - G_{x+i,y+j}) & \\ -Median_{i,j=-1,0,+1}(B_{x+i,y+j} - G_{x+i,y+j})) & \text{sonst} \end{cases} \tag{10.47}$$

$$B_{xy} = \begin{cases} B_{xy} & \text{falls} \quad B_{xy} \text{ ein Hotpixel} \\ G_{xy} + Median_{i,j=-1,0,+1}(B_{x+i,y+j} - G_{x+i,y+j}) & \text{sonst} \end{cases} \tag{10.48}$$

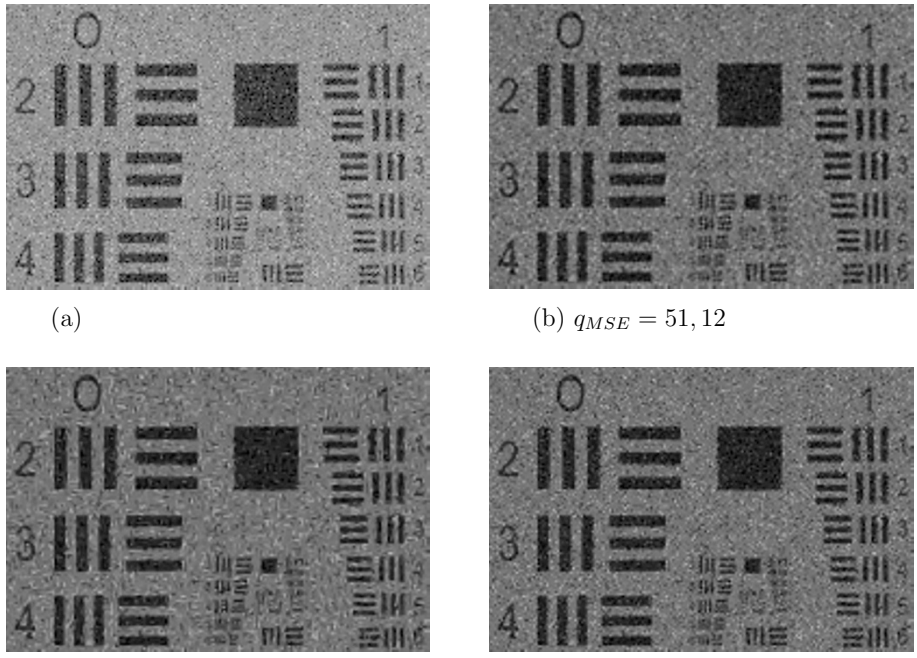
Durch diese n-fache, modifizierte Median-Filterung vor der horizontal/vertikal-Entscheidung verringern sich die Abweichungen aller drei Farbwerte (R, G, B) von den originalen Werten, wodurch auch die horizontal/vertikal-Entscheidung, d.h. der Schritt 5b des Algorithmus **A10.2**, positiv beeinflusst wird.

Voraussetzung für die Median-Filterung[6] vor der horizontal/vertikal-Entscheidung ist, dass alle drei Farbwerte R,G,B für die zu berechnenden Bildpunkte bis zu einem Abstand (nach der Manhattan-Norm) von drei Bildpunkten bereits vorliegen. Falls in einer technischen Realisierung der vorgeschlagenen Demosaicing-Methode die 1. Innovation 10.5.4.1 angewendet wird, liegen die benötigten Farbwerte bereits vor. Ansonsten müssten die benötigten Farbwerte vorher berechnet werden.

Auch die 2. Innovation verbessert das Demosaicing-Ergebnis erheblich: die Abweichungen aller drei Farbwerte (R, G, B) von den originalen Werten verringern sich bei einfacher Anwendung der Median-Filterung schon merklich (ca. $24-42\%$ niedrigere mittlere quadratische Abweichungen), bei dreifacher Anwendung sehr deutlich (ca. $45-60\%$ niedrigere mittlere quadratische Abweichungen), wobei sich die Abweichungen mit zunehmenden Rauschen tendenziell noch stärker verringern als ohne Rauschen (Bild 10.17).

[5] Der Median-Filter kann selbstverständlich auch auf andere Filterkerne, als den in (10.41) erwähnten 3×3-Filterkern der nächsten Nachbarn, erweitert werden (Beispiele dafür sind u.a. in Kapitel 6 und [Fail04] zu finden).

[6] mit 3×3-Filterkern der nächsten Nachbarn



(a) (b) $q_{MSE} = 51,12$

(c) $q_{MSE} = 51,03$ (d) $q_{MSE} = 11,62$

Bild 10.17: Demosaicing-Vergleich (a) verrauschtes Originalbild, (b) *„High Quality Linear Demosaicing“*, (c) *„Adaptive Homogeneity Directed Demosaicing“*, (d) *„Improved Adaptive Homogeneity Directed Demosaicing“*.

10.5.4.3 Nutzung der Homogenitätsinformation für eine weichere Entscheidung zwischen dem Horizontal- und dem Vertikal-Kanal

Im Schritt 5b des AHD-Demosaicing-Algorithmus **A10.2** wird nur im Fall exakt gleicher horizontaler und vertikaler Homogenitätswerte ($H_{xy\_h} = H_{xy\_v}$) eine Mittelung der horizontalen und vertikalen Farbwerte durchgeführt. Ansonsten wird jeweils zu 100% ausschließlich der horizontale oder der vertikale Farbwert ausgewählt. Dies führt in relativ homogenen Bildbereichen (Bereiche niedriger Ortsfrequenzen) zu deutlich sichtbaren blockartigen Artefakten, da der Algorithmus, bedingt durch die Tiefpassfilterung der Homogenitätswerte (Schritt 5a in **A10.2**), eine gewisse Trägheit hinsichtlich der Vorzugsrichtung aufweist. Dies wirkt sich bei hohen Ortsfrequenzen (Kanten, Linien) positiv aus, da kleinere Störungen durch diese Trägheit ausgebügelt werden. Bei niedrigen Ortsfrequenzen (homogene Bildbereiche) wirkt es sich jedoch negativ aus, da häufig über mehrere Pixel hinweg die gleiche Vorzugsrichtung beibehalten wird, bevor die Richtung gewechselt wird. Dadurch entstehen in den homogenen Bildbereichen blockartige Artefakte mit einer Seitenlänge von 4-8 Pixel, die trotz eher geringer Farbunterschiede wegen der homogenen Umgebung sichtbar

werden. Ein weiteres Problem der relativ harten horizontal/vertikal-Entscheidung ist die große Empfindlichkeit gegenüber Rauschen. So schneidet ein relativ einfaches Verfahren, wie z.B. das *„High Quality Linear Demosacing“* (Abschnitt 10.5.1), bei stark verrauschten Bildern genau so gut ab, wie der aufwändige AHD-Demosaicing-Algorithmus.

Die Idee zur Verringerung des Problems besteht nun darin, die in den Homogenitätskarten enthaltenen Informationen besser auszunutzen. Bei kleinen Unterschieden in den Homogenitätswerten wird keine Ja/Nein-Entscheidung getroffen, sondern der endgültige Farbwert wird zu einem vom Homogenitätsunterschied bestimmten Prozentsatz X aus dem horizontalen Farbwert, und dem komplementären Prozentsatz $100 - X$ aus dem vertikalen Farbwert zusammengesetzt. Je größer die Unterschiede in den Homogenitätswerten, desto stärker werden die Farbwerte der homogeneren Vorzugsrichtung gewichtet. Ein möglicher Algorithmus zur Gewichtung der Vorzugsrichtungen ist ein linearer Term[7]: sei S ein Zahlenwert, der die Breite des Übergangsbereichs für die weiche Entscheidung vorgibt. Dann gilt für die Farbmischung beim linearen Ansatz (Bild 10.18):

$$\mathbf{s}_a(x,y,n) = (R_{xy}, G_{xy}, B_{xy})^T =$$

$$= \begin{cases} (R_{xy\_h}, G_{xy\_h}, B_{xy\_h})^T, & \text{falls } H_{xy\_h} > H_{xy\_v} + S \quad , \\ (R_{xy\_v}, G_{xy\_v}, B_{xy\_v})^T, & \text{falls } H_{xy\_h} < H_{xy\_v} - S \quad , \\ \begin{pmatrix} \frac{1}{2}(R_{xy\_h} + R_{xy\_v}) + \frac{1}{2 \cdot S}(R_{xy\_h} - R_{xy\_v}) \cdot (H_{xy\_h} - H_{xy\_v}) \\ \frac{1}{2}(G_{xy\_h} + G_{xy\_v}) + \frac{1}{2 \cdot S}(G_{xy\_h} - G_{xy\_v}) \cdot (H_{xy\_h} - H_{xy\_v}) \\ \frac{1}{2}(B_{xy\_h} + B_{xy\_v} + \frac{1}{2 \cdot S}(B_{xy\_h} - B_{xy\_v}) \cdot (H_{xy\_h} - H_{xy\_v})) \end{pmatrix}, & \text{sonst.} \end{cases} \tag{10.49}$$

[7]Noch besser, allerdings auch rechenintensiver sind die nichtlinearen, komplementären Terme $\frac{1}{2}(1 + \tanh(x))$ und $\frac{1}{2}(1 - \tanh(x))$.

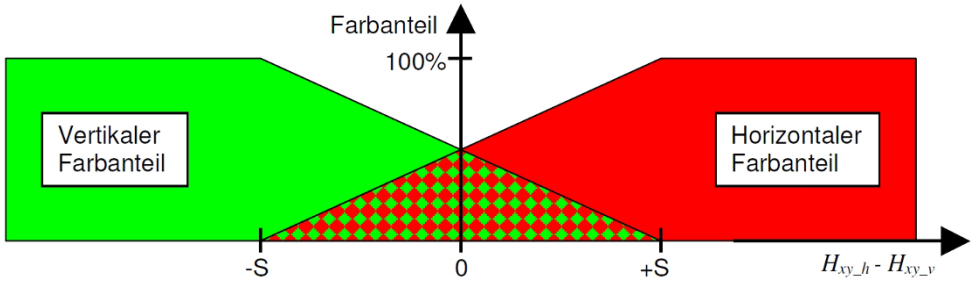


Bild 10.18: Mischung des horizontalen und vertikalen Farbwerts in Abhängigkeit vom Homogenitätsunterschied bei linearer Farbmischung.

Bei einem Mix aus 50% unverrauschten (Bild 10.19 und 10.20) und 50% verrauschten Bildern (Bild 10.17) ergibt sich ein Optimum hinsichtlich der Qualitätsverbesserung bei einer Breite des Übergangsbereichs von $S = 11$. Bei diesem Wert[8] verringern sich die Abweichungen aller drei Farbwerte (R, G, B) von den originalen Werten deutlich (ca. $15 - 25\%$ niedrigere mittlere quadratische Abweichungen), wobei sich die Abweichungen mit zunehmenden Rauschen tendenziell noch stärker verringern als ohne Rauschen. Für den rein visuellen Eindruck bei unverrauschten Bildern (Bild 4 und 5) liegt das Optimum für den Übergangsbereich bei $S = 3$, bei verrauschten Bildern (Bild 6) liegt das Optimum bei $S = 13$. Somit kann über den Parameter S (Breite des Übergangsbereichs) eine Anpassung an die Charakteristik des zu bearbeitenden Bildmaterials vorgenommen werden.

10.5.4.4 Median-Filterung der Farbdifferenzen $(R - G)$ und $(B - G)$ nach der horizontal-/vertikal-Entscheidung

Analog zur Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ **vor** der horizontal-/vertikal-Entscheidung (10.46), wie in Abschnitt 10.5.4.2 dargestellt, kann zusätzlich noch eine n-fache Median-Filterung der Farbdifferenzen **nach** der horizontal- / vertikal-Entscheidung durchgeführt werden. Der Unterschied zum 6. Schritt des AHD-Demosaicing-Algorithmus **A10.2** von Hirakawa und Parks besteht darin, dass die Median-Filterung der Farbdifferenzen $(R-G)$ und $(B-G)$ nicht an allen Pixeln durchgeführt wird, sondern nur solchen, die keine Hotpixel sind (also nicht direkt vom Sensor gemessen wurden). Dadurch lässt sich das Demosaicing-Ergebnis noch einmal deutlich verbessern.

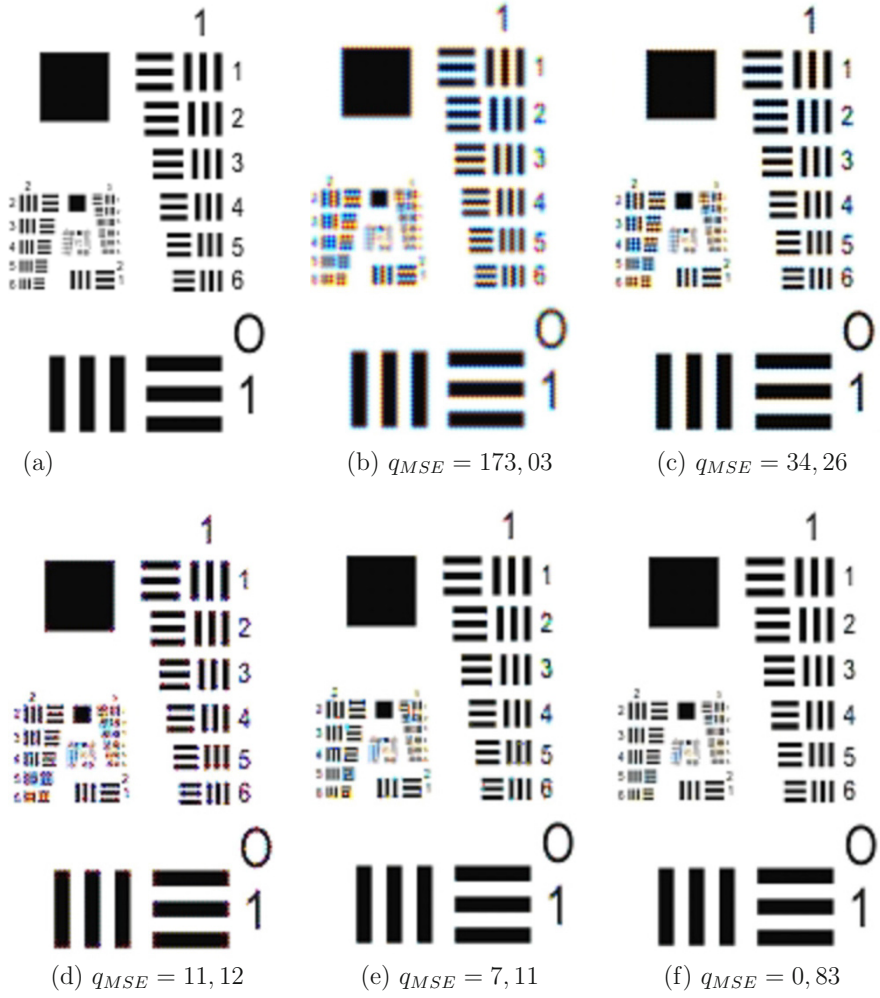
10.5.4.5 Orthogonalität der einzelnen Innovationen

Generell sind alle vier vorgestellten Innovationen auch einzeln anwendbar und leisten jeweils einen Beitrag zur Verbesserung der Bildqualität. Werden alle Punkte innerhalb einer Demosaicing-Methode angewendet, addiert sich näherungsweise ihr jeweiliger Beitrag zur Verbesserung der Bildqualität, d.h. die einzelnen Innovationen sind näherungsweise orthogonal zueinander. Die Abweichungen aller drei Farbwerte (Rot, Grün, Blau) von den originalen Werten (Schritte 1 - 6 des AHD-Demosaicing-Algorithmus von Hirakawa und Parks [Hira05]) verringern sich durch Anwendung aller vier Innovationen drastisch (ca. $58 - 78\%$ niedrigere mittlere quadratische Abweichungen), wobei sich die Abweichungen mit zunehmenden Rauschen tendenziell noch stärker verringern als ohne Rauschen. In den Bildern 10.19, 10.17 und 10.20 sind die Verbesserungen für den visuellen Eindruck dargestellt.

Die Innovationen 2 und 4 (Median-Filterung der Farbdifferenzen) können auch auf andere Demosaicing-Methoden angewendet werden, wie z.B. das *„High Quality Linear Demosaicing“* von Malvar et al. [Malv04]. Alle Punkte sind auch im Rahmen von Echtzeit-

[8] Die optimale Breite des Übergangsbereichs von $S = 11$ ist unabhängig davon, ob die Homogenitätskarten im CIE-Lab- oder im RGB-Farbraum erstellt werden (bei einer vereinfachten Variante des AHD- bzw. des IAHD-Demosaicing-Algorithmus entfällt der 3. Schritt in **A10.2**, die Konvertierung in den CIE-Lab-Farbraum, so dass die Homogenitätskarten im RGB-Farbraum erstellt werden).

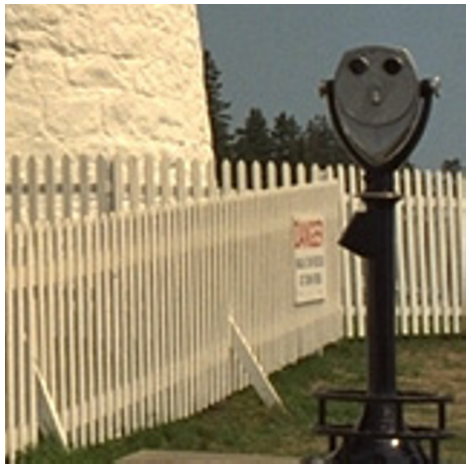
Anwendungen bzw. Hardware-Lösungen anwendbar (wie z.B. in ASICs = Application Specific Integrated Circuits, in FPGAs = Field Programmable Gate Arrays, oder in GPUs = Graphical Processing Units, d.h. programmierbaren Grafikkarten), da sich die Algorithmen als nichtlineare Filterkerne darstellen lassen.



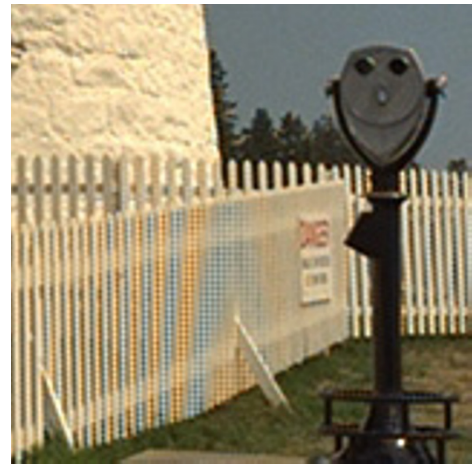
(a) (b) $q_{MSE} = 173,03$ (c) $q_{MSE} = 34,26$

(d) $q_{MSE} = 11,12$ (e) $q_{MSE} = 7,11$ (f) $q_{MSE} = 0,83$

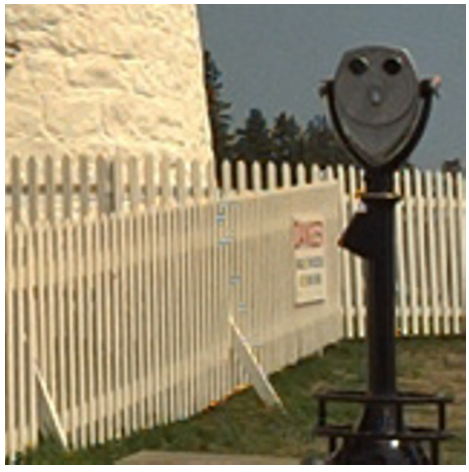
Bild 10.19: Demosaicing-Vergleich (a) Originalbild, (b) *„bilineare Interpolation“*, (c) *„High Quality Linear Demosaicing“*, (d) *„Adaptive Color Plane Interpolation“*, (e) *„Adaptive Homogeneity Directed Demosaicing“*, (f) *„Improved Adaptive Homogeneity Directed Demosaicing“*.



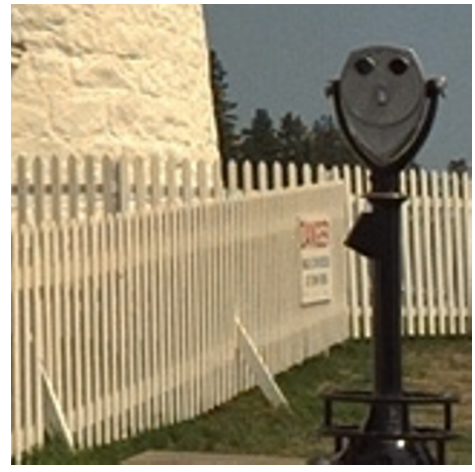
(a)



(b) $q_{MSE} = 82,64$



(c) $q_{MSE} = 31,83$



(d) $q_{MSE} = 15,39$

Bild 10.20: Demosaicing-Vergleich mit einem Ausschnitt des Bildes „*Small Lighthouse*“ (a) Originalbild, (b) „*High Quality Linear Demosaicing*“, (c) „*Adaptive Homogeneity Directed Demosaicing*“, (d) „*Improved Adaptive Homogeneity Directed Demosaicing*“.

Kapitel 11

Szenenanalyse

Die *Szenenanalyse* befasst sich mit der Extraktion von Informationen aus einem Bild oder einer Bildfolge mit automatischen oder halbautomatischen Methoden. Sie kann in die Verarbeitungsschritte der Vorverarbeitung, der Segmentierung und der Interpretation eingeteilt werden.

Bei der *Vorverarbeitung* von digitalisierten Bildern werden Techniken wie Bildverbesserung, geometrische Entzerrung oder digitale Filterung eingesetzt. Auch die Verwendung von speziellen Speicherungstechniken oder die Berechnung von geeigneten Merkmalen zur Bildsegmentierung kann zum Bereich der Bildvorverarbeitung gerechnet werden.

Bei der *Segmentierung* wird der Ortsbereich des Bildes, also der Bereich der Zeilen- und Spaltenkoordinaten x und y, in Bereiche eingeteilt, die nach Maßgabe von *Einheitlichkeitsprädikaten* P zusammengehören. Die resultierenden Bereiche mit einheitlichen Bildeigenschaften werden *Segmente* (oder *Regionen*) genannt. Es wird in der Regel verlangt, dass

- jeder Bildpunkt in genau einer der Regionen erfasst ist,
- das Einheitlichkeitsprädikat, angewendet auf die Bildpunkte der Region, den Wahrheitswert `TRUE` ergibt und
- die Regionen maximal sind, d.h., dass die Hinzunahme von Bildpunkten zu einer Region den Wahrheitswert `FALSE` bewirkt.

Bei der *Interpretation* wird versucht, auf der Basis des segmentierten Bildes einen Zusammenhang zwischen den einzelnen Segmenten abzuleiten, um so letztlich komplexe Objekte erkennen zu können. Dazu einige Beispiele:

In den meisten Anwendungsbereichen der digitalen Bildverarbeitung und Mustererkennung ist es notwendig, aus einem Bild oder einer Bildfolge zusammengehörige Bildstrukturen zu erkennen und diese als Ganzes weiter zu verarbeiten. Beispiele dazu sind in Bild 11.1 zusammengestellt. In der Automobilindustrie werden häufig Teile verklebt. Mit einem Roboter wird dazu eine Kleberaupe auf eines der zu verklebenden Teile aufgebracht. Eine

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_11

Aufgabe der optischen Qualitätskontrolle ist es hier zu überprüfen, ob die Kleberaupe vorhanden ist, ob sie unterbrochen ist und ob sie die richtige Form besitzt. Zunächst muss man somit auf dem zu untersuchenden Bauteil, meistens in einer festgelegten *area-of-interest* (Bild 11.1-a), die Kleberaupe, falls sie vorhanden ist, vom Bildhintergrund trennen. Dies kann, nach einer geeigneten Vorverarbeitung, im einfachsten Fall durch eine Binarisierung geschehen (Bild 11.1-b). Wenn die Binarisierung in der *area-of-interest* nicht eine Mindestanzahl an Vordergrundbildpunkten liefert, könnte man entscheiden, dass die Kleberaupe nicht vorhanden ist. Eine weitere Analyse des Bandes der Kleberaupe kann die Information liefern, ob Unterbrechungen vorliegen.

Wenn auch die Form (der Querschnitt) der Kleberaupe überprüft werden soll, so kann man mit einem Laser und einer Spaltlinse ein schmales Lichtband schräg auf die Kleberaupe projizieren und dieses Lichtband mit einem Videosensor von oben aufzeichnen. Die Verformung des Lichtbandes zeigt dann die Form der Kleberaupe. Hier muss also das Lichtband vom Bildhintergrund separiert werden (Bild 11.1-c). Die Berechnung markanter Punkte auf dem Lichtband und deren relative Lage zueinander liefert die Information, ob die Form des Querschnitts der Kleberaupe im zulässigen Bereich liegt.

Dieses einfache Beispiel enthält die grundlegenden Schritte der Segmentierung, bei der eine Bildstruktur (ein *Objekt*) vom Hintergrund zu trennen ist: Es werden Bildpunkte, die gleiche oder ähnliche Eigenschaften besitzen, zu einer Einheit zusammengefasst. Die Einheitlichkeit ist beim Lichtband durch das Merkmal „Helligkeit“ gegeben, d.h. alle Bildpunkte, deren Grauwert über einem bestimmten Schwellwert liegt, werden als „zum Lichtband gehörig“ gewertet. Die weiteren Auswertungen zur Form des Lichtbandes gehören nicht mehr zum Segmentierungsschritt. Sie können erst dann durchgeführt werden, wenn die Bildpunkte des Lichtbandes gefunden sind.

In diesem Beispiel ist das Objekt (das Lichtband) vom Hintergrund zu trennen. Die Segmentierung wurde hier durch eine Binarisierung durchgeführt. Im Ergebnisbild (11.1-c) sind die weißen Bildpunkte die Kleberaupe und die schwarzen der Hintergrund. Zu dem Objekt „Lichtband“ wurde hier also ein *Segment* erzeugt.

In der Regel werden komplexe Objekte in mehrere Segmente aufgelöst. Erste wichtige Aufgabe, die meistens nicht trivial ist und die weiteren Schritte wesentlich beeinflußt, ist die Festlegung, wie ein komplexes Objekt in einfache Bestandteile zerlegt werden kann. Auch hierzu einige Beispiele (Bildfolge 11.2):

Ein Widerstand (Bild 11.2-a) kann z.B. in die Bestandteile „linker/rechter Anschlussdraht“, „Keramikkörper“ und „Farbcodierung“ zerlegt werden. Man kann sich gut vorstellen, dass bei einer geeigneten Zuführung von Widerständen zu einem Videosensor diese Teile segmentiert werden können. Es werden dann Segmente für den linken und den rechten Anschlussdraht und für den Keramikkörper erzeugt. Die Farbcodierung ist eine zusätzliche Eigenschaft des Keramikkörpers, die erst dann analysiert werden kann, wenn das Segment zum Keramikkörper gefunden ist.

Schwieriger ist es schon bei einem Messgerät (Bild 11.2-b): Es hat ein Gehäuse mit einer bestimmten Abmessung. Im oberen Teil befindet sich die Anzeige mit einer Skaleneinteilung und einem Zeiger, im unteren Teil links und rechts zwei Anschlussbuchsen und in der Mitte ein Drehknopf, der von einer Skala umgeben ist. Bei einer passenden Zuführung eines

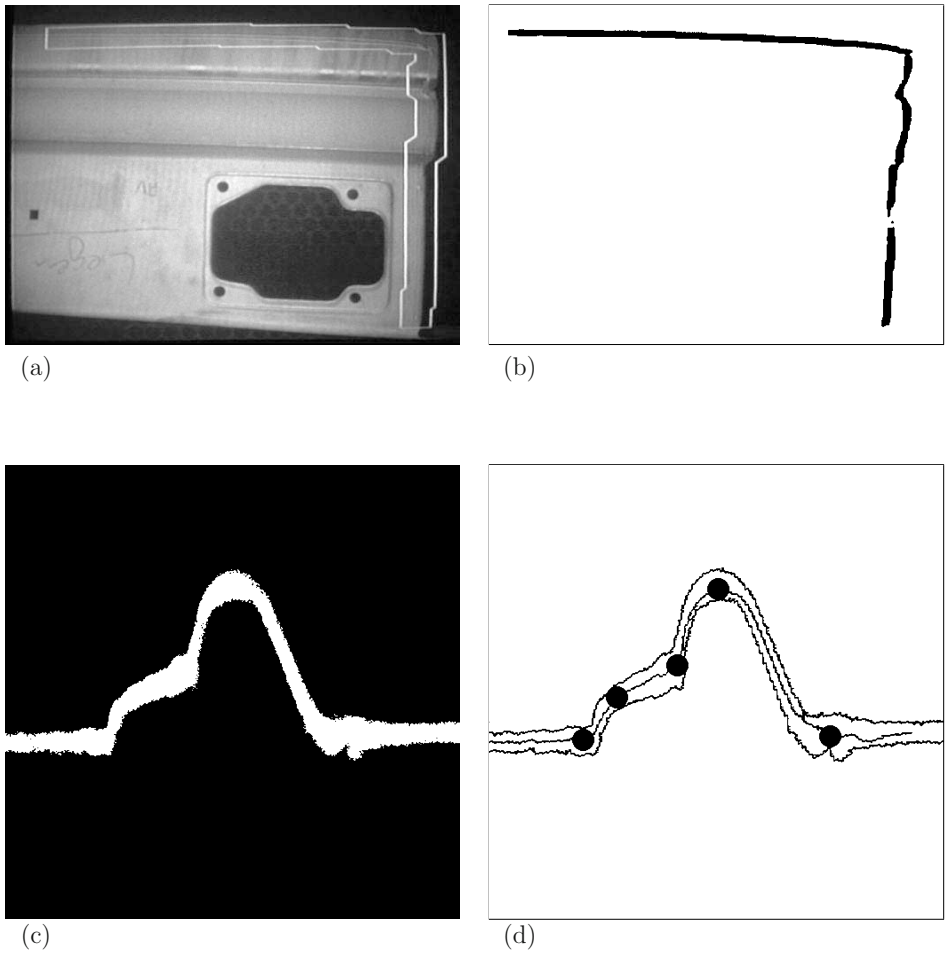
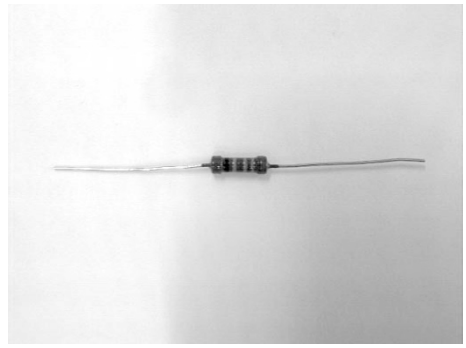


Bild 11.1: Qualitätskontrolle bei Kleberaupen. (a) Blechteil mit eingeblendeter *area-of-interest*. (b) Binarisierte Kleberaupe. Erzeugt die Binarisierung zu wenig Vordergrundbildpunkte in der *area-of-interest*, so kann man entscheiden, dass die Kleberaupe vermutlich nicht vorhanden ist. Eine weitere Analyse des Verlaufs liefert etwaige Unterbrechungen. (c) Zur Untersuchung der Form wird ein schmales Lichtband schräg auf die Kleberaupe projiziert und von oben aufgezeichnet. (d) Die Form des Lichtbandes kann durch die Bestimmung markanter Punkte und deren relativer Lage untersucht werden.



(a)

(b)

(c)

Bild 11.2: Beispiele für Objekte. (a) Widerstand. (b) Messgerät. (c) Fahrzeug.

derartigen Messgeräts, so dass es in einer definierten Lage ist, wäre es sicher möglich, die einzelnen Segmente zu erkennen.

Beim dritten Beispiel (Auto, 11.2-c) ist es nicht mehr ohne weiteres möglich, eine Zerlegung in einzelne erkennbare, einfache Bestandteile anzugeben. Wenn das Auto bei einer gegebenen Problemstellung aus verschiedenen Richtungen beobachtet wird, sieht es für den Betrachter ganz unterschiedlich aus. Bei manchen Anwendungen kann man mit apriori-Wissen, z.B. dass man das Fahrzeug immer nur von der Seite sieht, eine Zerlegung in einfache Bestandteile angeben.

Anhand dieser drei Beispiele sollte gezeigt werden, dass die Beschreibung der Zusammensetzung eines komplexen Objektes aus einfachen Bestandteilen die wichtigste Voraussetzung für eine erfolgreiche Segmentierung und eine daran anschließende Erkennung ist. Es macht sich auf jeden Fall bezahlt, wenn hier sorgfältig analysiert wird. Für die weiteren Betrachtungen wird nun angenommen, dass dieses Problem gelöst ist. Zum Teil wird auch

vereinfachend angenommen, dass ein Objekt nur aus einem Segment besteht, wie das z.B. bei einer Beilagscheibe der Fall ist.

In manchen Anwendungsgebieten ist es schwierig, die einzelnen Objekte, mit denen das System konfrontiert wird, exakt anzugeben. Ein Beispiel dazu ist die Erkennung von Handschriften: Ein Objekt dieses Problemkreises könnte der Buchstabe „A“ sein. Man hat eine Vorstellung, wie ein „A“ aussehen sollte. Aber bei handgeschriebenen Texten wird dieser Buchstabe von Schreiber zu Schreiber und sogar beim selben Schreiber immer unterschiedlich ausfallen. So wird man festlegen müssen, bis zu welcher Grenze man eine Struktur noch als „A“ akzeptieren will. In diesem Fall spricht man von der *Objektklasse* „A“, in der alle zulässigen Ausprägungen zusammengefasst sind.

Die Objekte (oder Objektklassen) werden in unserer Umwelt durch eine Vielzahl von physikalisch messbaren Größen charakterisiert[1]. Im Rahmen eines bestimmten Problemkreises wird man sich auf eine spezielle Sensorik festlegen. Damit hat man sich aber auch auf spezielle physikalische Größen festgelegt, die die Objekte charakterisieren. Falls man mit den gewählten Sensoren gerade diejenigen Eigenschaften nicht erfasst, in denen sich die Objekte unterscheiden, so hat man die falschen Sensoren gewählt (Abschnitt 3.1).

Die aufgezeichneten Originaldaten spannen einen N-dimensionalen *Merkmalsraum* auf, wobei N die Anzahl der Informationsschichten (Kanäle) ist. Wird z.B. mit einer Farbvideokamera ein RGB-Bild aufgezeichnet, so ist der Merkmalsraum der Originaldaten dreidimensional. In diesem Merkmalsraum sind den aufgezeichneten Objekten (Objektklassen) bestimmte Bereiche zugeordnet. Diese Bereiche werden als *Muster* (*Musterklassen*) bezeichnet. Die Lage der Musterklassen im Merkmalsraum ist nicht bekannt. Wenn die aufgezeichneten charakteristischen Messgrößen richtig gewählt sind, so kann man hoffen, dass Musterklassen kompakte Bereiche sind, die für die verschiedenen Klassen getrennt liegen.

Nun kann es aber sein, dass die aufgezeichneten Originaldaten es nicht ermöglichen, die Musterklassen in kompakte, getrennte Cluster zu transformieren. Dann müssen aus den Originaldaten abgeleitete Merkmale berechnet werden. Es wird dazu eine N_e-kanalige Bildfolge $\mathbf{S}_e = (s_e(x, y, n, t))$ als Originaldaten betrachtet. Man kann Merkmale aus den Kanälen oder aus Kanalkombinationen verwenden. Das wird in den Kapiteln 12 und 13 näher untersucht. Auch entlang der Zeitachse können Merkmale berechnet werden. Beispiele dazu sind Verschiebungsvektorfelder. Diese Thematik wird in Kapitel 14 erläutert. Schließlich kann man auch in Umgebungen eines Bildpunktes in der Position (x, y) Merkmale berechnen. Beispiele dazu sind Maßzahlen zur Textur, die in den Kapiteln 15 bis 19 ausführlich behandelt werden. Nach dem Schritt der Merkmalsberechnung kann man das Ergebnis mit einer N_a-kanaligen Szene $\mathbf{S}_a = (s_a(x, y, n))$ beschreiben, bei der jeder Kanal $n = 0, 1, ..., N_a - 1$ Maßzahlen für das jeweilige Merkmal enthält.

Diese Szene $\mathbf{S}_a$, die natürlich auch mit der Originalszene $\mathbf{S}_e$ identisch sein kann (wenn keine aus den Originaldaten abgeleiteten Merkmale berechnet werden), ist die Eingabe für

[1]Natürlich werden Objekte auch noch durch andere Eigenschaften charakterisiert, wie z.B. Tiere oder Menschen durch bestimmte Verhaltensweisen. Diese Eigenschaften sind in der Regel aber nicht physikalisch messbar. Auf die Betrachtung derartiger „Merkmale“ wird im Folgenden verzichtet.

die eigentlichen Segmentierungsverfahren (Klassifikatoren), die ab Kapitel 20 besprochen werden.

In den folgenden Kapiteln werden nun die unterschiedlichsten Möglichkeiten vorgestellt, wie man Merkmale für eine Segmentierung aus den Originaldaten gewinnnen kann. Da einige der vorgestellten Verfahren (z.B. die Gauß- und Laplace-Pyramiden in Kapitel 17) nicht nur zur Merkmalsberechnung eingesetzt werden können, sondern neben diesem Gesichtspunkt auch noch andere Anwendungsmöglichkeiten haben, werden diese ebenfalls dargestellt.

Kapitel 12

Merkmale: Grauwert und Farbe

12.1 Anwendungen

Der Grauwert eines Bildes kann als einfaches Merkmal für eine Bildsegmentierung angesehen werden. In einem mikroskopischen Bild können z.B. dunkle Zellen vor einem hellen Bildhintergrund zu sehen sein. Die Binarisierung eines derartigen Bildes ist ein einfaches Beispiel für eine Segmentierung.

Ein Grauwertbild kann aber auch das Ergebnis einer aufwändigen Merkmalsberechnung sein. So können z.B. die Grauwerte Maßzahlen für die Textur oder die Bewegung sein. Wenn die verschiedenen Klassen in diesem Bild in deutlich getrennten Grauwert- (Merkmals-) Intervallen liegen, können sie mit einfachen Schwellwertverfahren segmentiert werden.

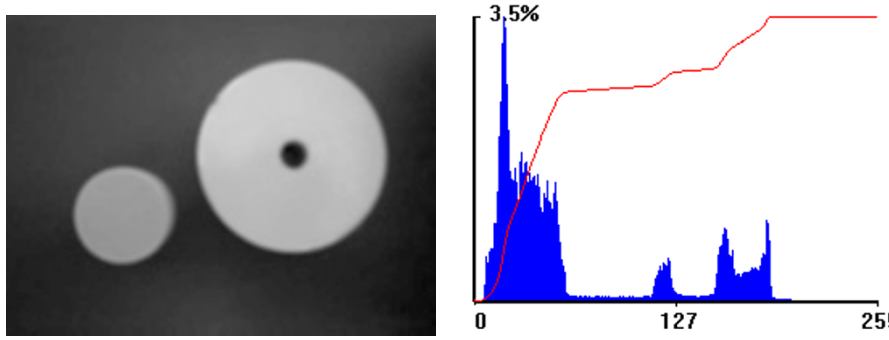
Die Verarbeitung von Multispektral- und Farbbildern war lange Zeit fast ausschließlich eine Domäne der Fernerkundung (Luft- und Satellitenbildauswertung). Hier werden, meistens ohne allzu großen „Echtzeitdruck“, die riesigen Datenmengen, nach geometrischen Korrekturen und weiteren Vorverarbeitungsschritten, mit multivariaten Klassifikatoren verarbeitet, um so z.B. thematische Karten und Landnutzungskartierungen zu erstellen.

Leistungsfähige Prozessoren, großer Hauptspeicherausbau, sowie Hintergrundspeicher mit gewaltigem Fassungsvermögen machen es möglich, die Verarbeitung und Auswertung von Farbbildern in vielen neuen Bereichen einzusetzen. So werden in der Industrie multisensorielle Systeme entwickelt, die u.a. auch Farb- und Multispektralbilder aufzeichnen. Die so gewonnenen Farb- oder Multispektraldaten können unter Umständen direkt als Merkmale für eine Segmentierung verwendet werden.

Aber auch im privaten Bereich sind digitalisierte Farbbilder heute ein Standard. Digitalkameras (mit und ohne Videofunktion) finden sich in fast jedem Mobiltelefon und Notebook und gehören meist zur Grundausstattung eines Haushalts.

In diesem Kapitel werden neben dem Gesichtspunkt des ,,Merkmals Farbe” einige Verfahren beschrieben, die sich bei der Verarbeitung und Darstellung von Farbbildern einsetzen lassen.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_12



(a) (b)

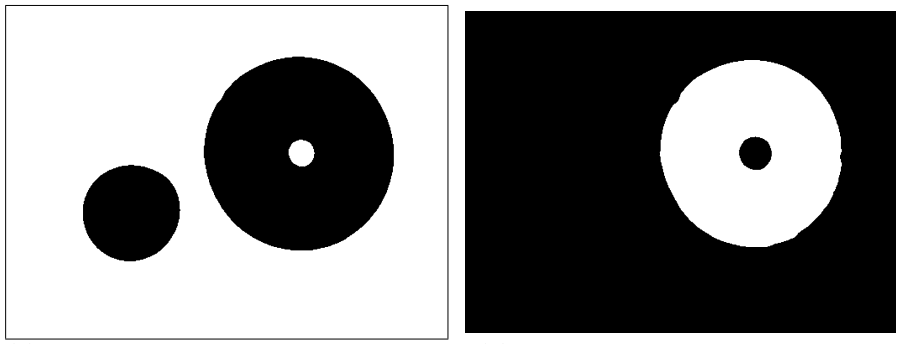
Bild 12.1: (a) Grauwertbild mit zwei hellen Teilen auf einem dunklen Hintergrund. Das Bild enthält somit die drei Klassen ,,Hintergrund", ,,Objekt links unten" und ,,Objekt rechts oben". (b) Histogramm zu Bild (a): Man sieht deutlich die drei lokalen Maxima der drei Klassen.

12.2 Merkmal: Grauwert

Wenn sich in einem Grauwertbild helle Objekte vor einem dunklen Hintergrund oder dunkle Objekte vor einem hellen Hintergrund abzeichnen, kann man sie mit einfachen Schwellwertverfahren segmentieren. Bild 12.1-a zeigt ein Beispiel eines Grauwertbildes mit zwei Objekten vor einem dunklen Hintertgrund. Das Bild enthält somit die drei Klassen ,,Hintergrund", ,,Objekt links unten" und ,,Objekt rechts oben". Rechts daneben ist in Bild 12.1-b das Histogramm gezeigt. Man sieht deutlich die drei lokalen Maxima der drei Klassen: Der Bildhintergrund liegt z.B. etwa im Grauwertbereich von 0 bis 80.

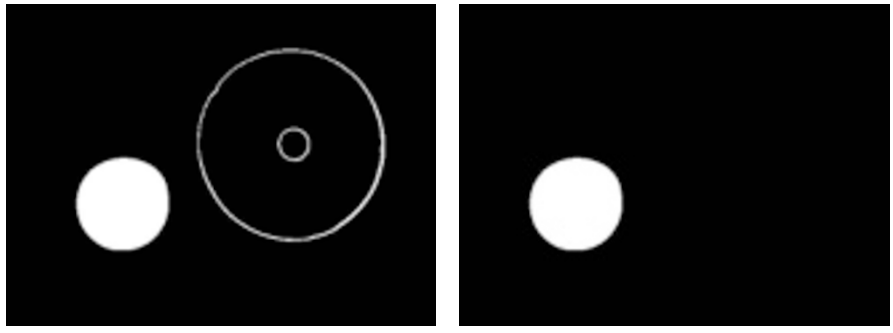
Es ist einfach, den Hintergrund oder das Objekt rechts/oben zu segmentieren: Für den Hintergrund werden alle Grauwerte von 0 bis 80 der Klasse ,,Hintergrund" zugeordnet und alle weiteren Grauwerte einer ,,Zurückweisungsklasse". Für das Objekt rechts/oben werden die Grauwerte 0 bis 150 der Zurückweisungsklasse und die Grauwerte 151 bis 255 der Klasse ,,Objekt rechts oben" zugewiesen. Die Bilder 12.2-a und -b zeigen das Ergebnis.

Probleme gibt es bei der Segmentierung der Klasse ,,Objekt links unten". Aus dem Histogramm in Bild 12.1-b kann man ablesen, dass die Grauwerte dieser Klasse etwa zwischen 110 und 145 liegen. Wenn man das Original mit diesem Grauwertbereich segmentiert und alle anderen Grauwerte der Zurückweisungsklasse zuordnet, erhält man das Ergebnis von Bild 12.3-a. Man sieht deutlich, dass auch Bildpunkte, die zur Klasse ,,Objekt rechts oben" gehören, der Klasse ,,Objekt links unten" zugeordnet wurden. Das liegt daran, dass im Übergangsbereich zwischen dem dunklen Hintergrund und dem hellen Objekt rechts/oben



(a) (b)

Bild 12.2: Segmentierung durch Schwellwertbildung: Die Segmente sind weiß dargestellt, die Zurückweisungsklasse schwarz. (a) Klasse ,,Hintergrund": Grauwertintervall 0 bis 80 im Original. (b) Klasse ,,Objekt rechts oben": Grauwertintervall 151 bis 255 im Original.



(a) (b)

Bild 12.3: Segmentierung durch Schwellwertbildung: Die Segmente sind weiß dargestellt, die Zurückweisungsklasse schwarz. (a) Klasse ,,Objekt links unten": Grauwertintervall 111 bis 145 im Original. Es werden auch Bildpunkte segmentiert, die zur Klasse ,,Objekt rechts oben" gehören. (b) Mit einer morphologischen opening-Operation (Erosionen, gefolgt von Dilatationen) können die falsch segmentieren Bildpunkte eliminiert werden.

auch Grauwerte auftreten, die im Grauwertintervall des Objektes links/unten vorkommen. Unter Umständen kann man einen derartigen Segmentierungsfehler nachträglich korrigieren. In Bild 12.3-b wurde z.B. auf das Ergebnis der Segmentierung eine morphologische opening-Operation (zwei Erosionen gefolgt von zwei Dilatationen, Kapitel 6) angewendet, wodurch der Rand um das Objekt rechts/oben verschwindet.

Abschließend zu diesem Abschnitt ein Beispiel eines Grauwertbildes, das das Ergebnis einer Berechnung von Maßzahlen für die Textur ist. Bild 12.4-a zeigt einen Bildausschnitt mit zwei unterschiedlichen Stoffstrukturen. Durch die Berechnung eines Texturmaßes, hier die Kantendichte (Abschnitt 15.5), ist es gelungen, die beiden Texturen in unterschiedliche Grauwertintervalle (Merkmalsintervalle) abzubilden (Bild 12.4-b). Das Ergebnis einer Segmentierung zeigt Bild 12.4-c.

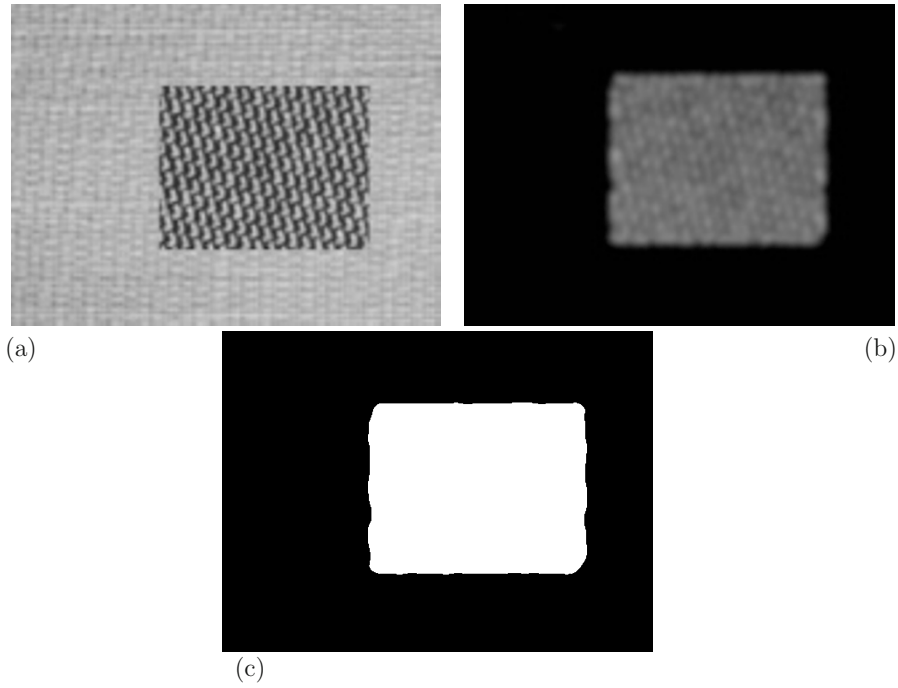


Bild 12.4: (a) Zwei unterschiedlich strukturierte Stoffe. (b) Durch die Berechnung eines Texturmaßes (hier die Kantendichte aus Abschnitt 15.5) ist es gelungen, die beiden Texturen in getrennte Grauwertintervalle (Merkmalsintervalle) abzubilden. (c) Ergebnis der Segmentierung einer der beiden Texturen.

12.3 Merkmal: Farbe

In den folgenden Betrachtungen wird das RGB-Farbmodell verwendet. Ein RGB-Bild ist, wie in Abschnitt 3.5 beschrieben, ein dreikanaliges Bild $\mathbf{S} = (s(x, y, n))$, mit $n = 0, 1, 2$ und der Kanalreihenfolge Rot, Grün und Blau. Der Wertebereich der einzelnen Kanäle ist die Grauwertmenge $G = \{0, 1, ..., 255\}$, so dass ein Farbbildpunkt aus 24 Bit zusammengesetzt ist, was bedeutet, dass $2^{24} = 16777216$ Farben dargestellt werden können.

Die Werte der Farbkomponenten liegen hier nicht, wie im Grundlagenabschnitt 3.5 dargestellt, im Intervall [0,1], sondern in der Grauwertmenge G. Bei Umrechnung in andere Farbmodelle ist das unter Umständen zu beachten. Wegen der diskretisierten Werte können sich bei der Umrechnung Fehler ergeben.

Die Farbe ist ein wichtiges Merkmal für die Segmentierung von Bildern. Ein Beispiel, wie die Muster von Objekten, in diesem Fall die verschiedenfarbigen Köpfe, im 3-dimensionalen RGB-Merkmalsraum liegen, zeigen die Bilder 12.5. Da der dreidimensionale RGB-Merkmalsraum schwer darzustellen ist, wurden hier die drei zweidimensionalen Merkmalsräume RG, RB und GB abgebildet. Bild 12.5-a zeigt das Farbbild. Die Bilder 12.5-b bis 12.5-d zeigen den Rot-, Grün- und Blaukanal. In den Bildern 12.5-e bis 12.5-g sind die zweidimensionalen RB- RG- und GB-Merkmalsräume abgebildet. Man sieht deutlich, wo die verschiedenfarbigen Objekte (Köpfe und Hintergrund) im Merkmalsraum die Punkthäufungen (cluster) bilden.

Bei der Besprechung der Klassifikatoren werden in den Kapiteln 20 bis 22 ebenfalls häufig Beispiele mit dem Merkmal Farbe verwendet.

Die hier folgenden Abschnitte befassen sich mit weiteren Aspekten bei der Digitalisierung und der bildlichen Wiedergabe von Farbbildern.

12.4 Reduktion der Farben durch Vorquantisierung

Bei der Verarbeitung und vor allem bei der bildlichen Darstellung von digitalisierten Farbbildern sind manchmal (z.B. aus Kostengründen in sehr einfachen Umgebungen) Echtfarbsysteme, die 24-Bit (oder mehr) RGB-Bilder darstellen können, nicht verfügbar. Dagegen sind meistens Ausgabesysteme vorhanden, die 256 verschiedene Grau- oder Farbtöne darstellen können (z.B. sehr einfache Grafikkarten oder Farbdrucker).

Um ein RGB-Bild, das ja durch seine 24-Bit-Bildpunkte theoretische $2^{24} = 16777216$ verschiedene Farbtöne besitzen kann, auf einem Ausgabegerät mit nur 256 Farbtönen darstellen zu können, muss die Anzahl der Farbtöne drastisch reduziert werden. Dazu werden im Folgenden einige Beispiele gegeben.

Eine einfache Methode ist die Reduktion der 256 Grauwerte pro Farbauszug durch eine *Vorquantisierung*. In der Praxis hat es sich bewährt, z.B. den Rotanteil auf drei Bit (acht Rottöne), den Grünanteil auf drei Bit (acht Grüntöne) und den Blauanteil auf zwei Bit (vier Blautöne) zu reduzieren. Der so entstehende Farbwert kann dann in acht Bit (einem Byte) dargestellt werden:

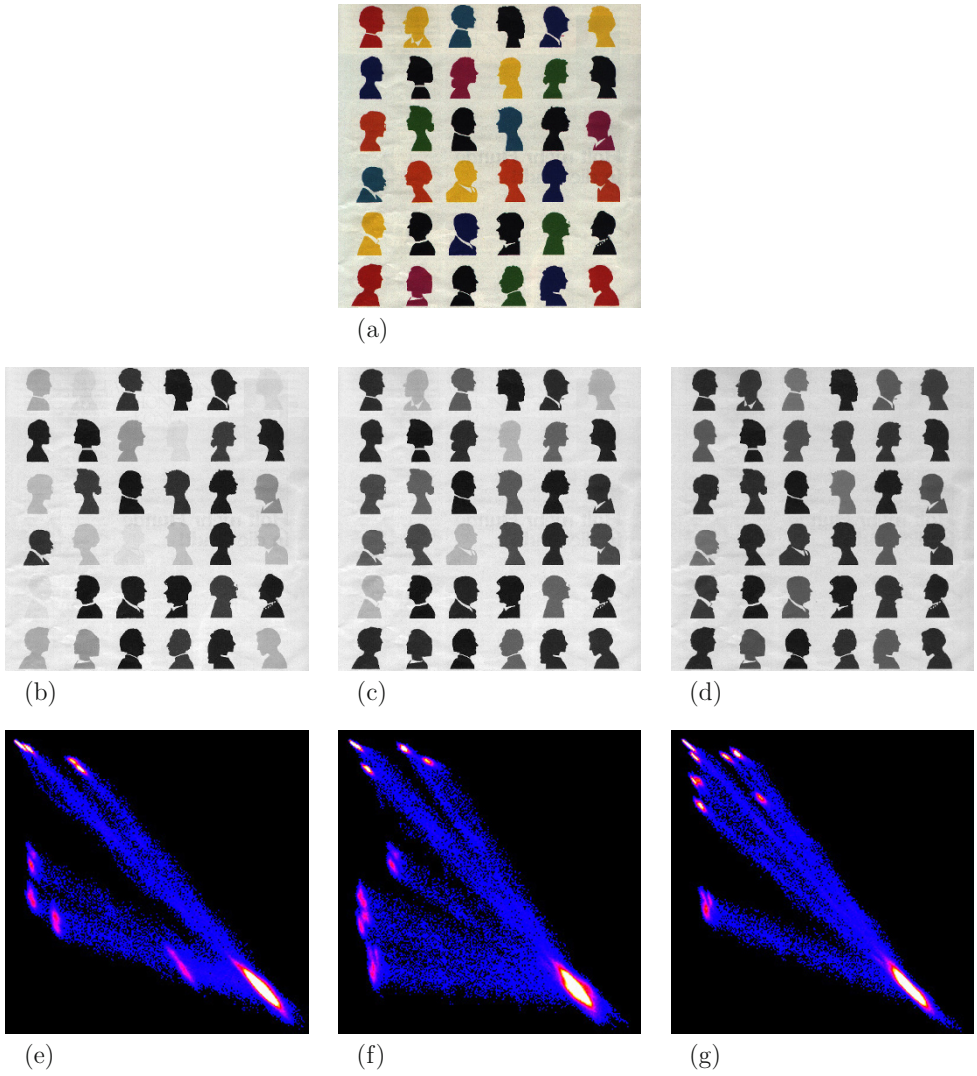


Bild 12.5: (a) RGB-Farbbild. (b) bis (d): Rot-, Grün- und Blaukanal. (e) bis (g): Zweidimensionale RG- RB- und GB-Merkmalsräume. Man sieht deutlich, wo die verschiedenfarbigen Köpfe (Objekte) im Merkmalsraum die Punkthäufungen (cluster) bilden. Zur Orientierung: Der Koordinatenursprung ist in den Merkmalsräumen links oben. Im RG-Raum z.B. ist der Rotkanal nach unten und der Grünkanal nach rechts aufgetragen. Die große Punktwolke rechts unten entspricht dem hellen Hintergrund der Köpfe. Die Zuordnung der weiteren Cluster sei dem Leser als Übung überlassen.

7	6	5	4	3	2	1	0	Bitposition
B	B	G	G	G	R	R	R	Farbanteile

Bei einer einfachen Grafikkarte oder einem 8-Bit-Bildspeicher eines Bildverarbeitungssystems muss nun noch eine Farb-*look-up*-Tabelle ($cLuT$, Abschnitt 4.4) erzeugt werden, die die verwendete RGB-Aufteilung in einem Byte richtig darstellt. Der gesamte Algorithmus ist im folgenden zusammengefasst:

A12.1: 8-Bit-Echtfarben-Darstellung durch Vorquantisierung.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y, n))$ ist ein dreikanaliges RGB-Eingabebild.

◇ $\mathbf{S}_a = (s_a(x, y))$ ist ein einkanaliges Ausgabebild.

◇ Alle Divisionen werden ganzzahlig ausgeführt. | und & bedeuten bitweise logisches ODER bzw. UND, << bedeutet bitweiser Linksshift und >> bitweiser Rechtsshift.

Algorithmus:

(a) Für alle Bildpunkte des Bildes $\mathbf{S}_e = (s_e(x, y, n))$:
$red\_reduced = s_e(x, y, 0) >> 5;$
$green\_reduced = s_e(x, y, 1) >> 5;$
$blue\_reduced = s_e(x, y, 2) >> 6;$
$s_a(x, y) = red\_reduced|(green\_reduced << 3)|(blue\_reduced << 6);$

(b) Berechnung einer passenden $cLuT$ (Beispiel). Für alle $g \epsilon G$:
$cLuT.red[g] = (g\&7) << 5 + 16;$
$cLuT.green[g] = ((g\&56)/8) << 5 + 16;$
$cLuT.blue[g] = ((g\&192)/64) << 6 + 32;$

(c) Abspeichern der $cLuT$ in der Hardware des Displaysystems.

(d) Ausgabe des Bildes $\mathbf{S}_a$ in den Bildwiederholungsspeicher.

Ende des Algorithmus

Bild 12.6-a zeigt ein Beispiel zu diesem Verfahren. Da der Bildinhalt stark strukturiert ist, fällt beim Betrachten der geringere Umfang der Farbskala nicht auf. Bild 12.6-b besitzt dagegen größere homogene Flächen. Hier wird die verminderte Bildqualität deutlich. Diesen

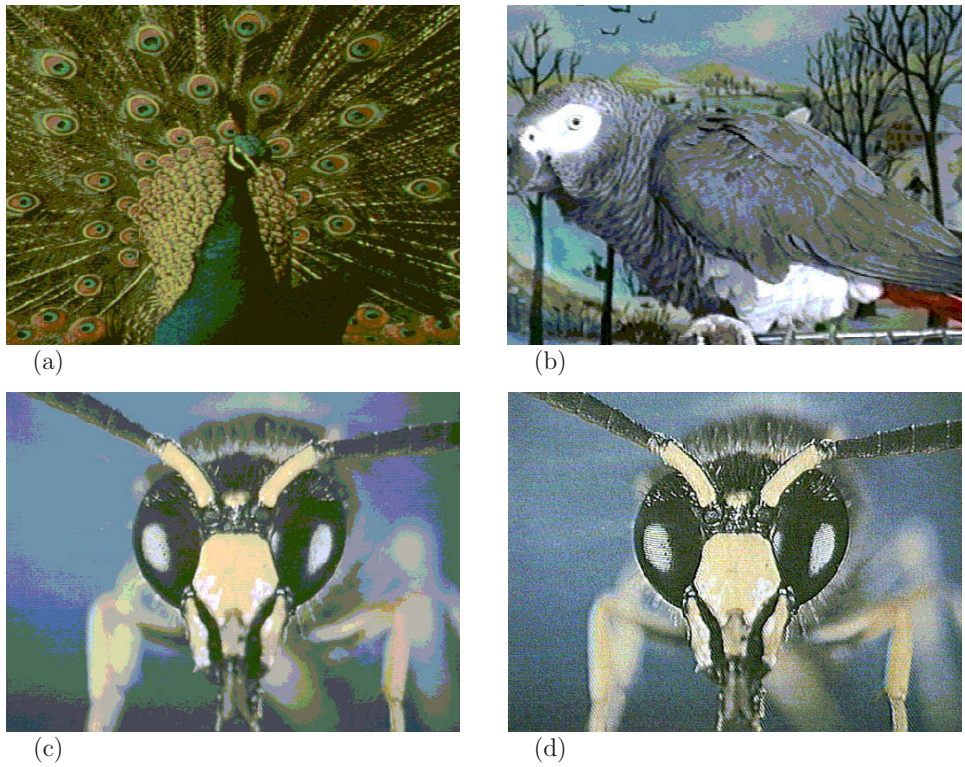


Bild 12.6: Beispiele zur Reduktion der Farbskala. (a) Bei stark strukturierten Bildern fällt der Einfluss der Farbskalareduktion nicht auf. (b) Hier sind neben strukturierten Bereichen auch homogene Bereiche vorhanden. Der Einfluss der Farbquantisierung ist deutlich zu sehen. (c) Besitzt ein Bild größere homogene Flächen, so wird die verminderte Bildqualität sehr deutlich. (d) Hier wurde das Original vor der Quantisierung der Farben mit einem hochpassgefilterten Bild überlagert. Dadurch werden die homogenen Flächen etwas gröber und die Auswirkung der Quantisierung tritt nicht mehr so deutlich hervor.

Effekt kann man etwas abschwächen, wenn man vor der Quantisierung der Farben das Original mit einer leichten Hochpassfilterung überlagert (Abschnitt 5.4). Dadurch werden die homogenen Flächen im Bild etwas gröber, und der störende Effekt von Bild 12.6-c tritt nicht mehr so stark auf (Bild 12.6-d).

12.5 Aufwändigere Verfahren zur Farbreduktion: Indexbilder

Bei aufwändigeren Verfahren zur Reduktion der Farben eines RGB-Bildes wird zunächst die Häufigkeitsverteilung der Farben berechnet. Dann werden die „wichtigsten“ Farben des Bildes ermittelt und für die farbreduzierte Darstellung verwendet. Das farbreduzierte Bild besteht dann aus einem Indexbild, bei dem die Werte der Bildpunkte Zeiger in eine *look-up*-Tabelle sind. Der Rahmen eines derartigen Verfahrens ist im Algorithmus **A12.2** zusammengefasst. Die einzelnen Teilaspekte werden in den folgenden Abschnitten untersucht.

A12.2: Erstellung von Index-Bildern.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y, n))$ ist ein dreikanaliges RGB-Eingabebild.

◇ $\mathbf{S}_a = (s_a(x, y))$ ist ein einkanaliges Ausgabebild (Indexbild).

Algorithmus:

(a) Berechnung der Häufigkeitsverteilung der Farbtöne des Bildes (Histogrammberechnung, Abschnitt 12.5.1).

(b) Aufstellen einer repräsentativen Farb-*look-up*-Tabelle (*cLut*) mit den wichtigsten Farbtönen des Bildes (Abschnitt 12.5.2).

(c) Abbildung der Farben des Originalbildes in die Farb-*look-up*-Tabelle *cLuT* (Abschnitt 12.5.3).

(d) Transformation des dreikanaligen Farbbildes $\mathbf{S}_e$ in ein einkanaliges Index-Bild $\mathbf{S}_a$ unter Verwendung der *cLuT* (Abschnitt 12.5.4).

Ende des Algorithmus

12.5.1 Die Farbhäufigkeitsverteilung eines Farbbildes

Vom mathematischen Standpunkt aus betrachtet können die relativen Häufigkeiten der Farben eines Farbbildes leicht beschrieben werden:

$$\mathbf{s}_e(x, y) = \mathbf{g} = (r, g, b)^T; \quad r, g, b \,\epsilon\, G = \{0, 1, ..., 255\}; \tag{12.1}$$

$$p(\mathbf{g}) = \frac{a_{\mathbf{g}}}{M},$$

wobei $a_{\mathbf{g}}$ die Häufigkeit des Farbvektors $\mathbf{g}$ in $\mathbf{S}_e$ und M die Anzahl der Bildpunkte von $\mathbf{S}_e$ ist.

In der Praxis stößt man schnell auf Schwierigkeiten, wenn man sich überlegt, wieviel Speicherplatz man für das Histogramm $p(\mathbf{g})$ benötigt. Ein Farbvektor $\mathbf{g}$ ist aus 24 Bit (jeweils 8 Bit für Rot, Grün und Blau) zusammengesetzt. Es gibt also 2^{24} verschiedene Farbvektoren, was bedeutet, dass das Histogramm $p_S(\mathbf{g})$ $2^{24} = 16777216$ Einträge vorsehen müsste. Bei 16 Bit pro Eintrag sind das 32 MByte. Da bei einem Bild mit 16 Mio. Pixel eine Farbe maximal 2^{24}-mal auftreten könnte, würde man sogar 24 Bit pro Eintrag benötigen. Das wären dann 48 MByte.

Eine Möglichkeit, die schon in Abschnitt 12.4 angewendet wurde, besteht in der Vorquantisierung der Grauwerte der Farbkanäle. Je nach Anwendungsfall können die Farbpixel auf 8 Bit (3 Bit Rot, 3 Bit Grün und 2 Bit Blau), auf 15 Bit (5 Bit Rot, 5 Bit Grün und 5 Bit Blau) oder 17 Bit (6 Bit Rot, 6 Bit Grün und 5 Bit Blau) reduziert werden. Entsprechend wird die Farb-*look-up*-Tabelle kleiner.

Eine andere Überlegung führt zu den folgenden Verfahren in diesem Abschnitt: Ein Farbbild mit $512 \cdot 512$ Bildpunkten kann natürlich auch maximal $512 \cdot 512 = 2^{18}$ verschiedene Farben enthalten. Das Histogramm muss also maximal für $2^{18} = 262144$ Einträge vorgesehen werden. Bei 16 Bit pro Eintrag sind das nur 0.5 MByte. Bei einem $1024 \cdot 1024$ großen Bild wären es 2 MByte. Da es aber unwahrscheinlich ist, dass in einem Farbbild die maximal mögliche Zahl von Farben auftritt, kann das Histogramm noch weiter verkleinert werden.

Es stellt sich nun die Aufgabe, eine Abbildung H des 24-Bit-Farbraums in ein Histogramm vom Umfang $h\_size < 2^{24}$ zu konstruieren. Diese Abbildung kann natürlich nicht eineindeutig (injektiv) sein. Mit einer *Hash-Codierung* lässt sich dieses Problem lösen.

Zunächst die Datenstruktur des Histogramms (der Häufigkeitstabelle): Man benötigt einen Zähler für die Häufigkeit, außerdem muss der zu jedem Eintrag gehörige RGB-Wert (als „Schlüssel“) mit abgespeichert werden, um vergleichen zu können, ob an dieser Stelle in der Hash-Tabelle auch die richtige Farbe eingetragen ist. Damit ergibt sich für die Hash-Tabelle diese Struktur:

0	*count*	*R*	*G*	*B*
	...			
$h\_size - 1$	*count*	*R*	*G*	*B*

Das Histogramm (die Hash-Tabelle) wird mit h_tab bezeichnet, die einzelnen Datenfelder mit $h\_tab.count$, $h\_tab.R$, $h\_tab.G$ und $h\_tab.B$.

Die Hash-Adresse zu einem Farbvektor $\mathbf{g} = (r, g, b)^T$ wird mit der Modulofunktion berechnet. Treten bei der Einspeicherung in die Hash-Tabelle Kollisionen auf, so werden sie durch eine Fortschaltgröße l und Sondieren eines neuen Speicherplatzes aufgelöst. Es ist zu beachten, dass durch wiederholtes Akkumulieren der Hashadresse h letztlich alle Positionen der Hash-Tabelle erreicht werden.

A12.3: Berechnung des Farbhistogramms eines RGB-Bildes.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1, 2$ ist ein dreikanaliges RGB-Eingabebild.

◇ $\mathbf{g} = \mathbf{s}_e(x, y) = (r, g, b)^T; r, g, b \,\epsilon\, G.$ ist ein RGB-Farbvektor von $\mathbf{S}_e$.

◇ *h_tab* ist das Farbhistogramm mit der Datenstruktur von oben und den Datenfeldern *h_tab.count*, *h_tab.R*, *h_tab.G* und *h_tab.B*.

Algorithmus:

(a) Für alle Bildpunkte $\mathbf{g} = \mathbf{s}_e(x, y)$ von $\mathbf{S}_e$:

(aa) Berechnung der Hashadresse h des Farbvektors $\mathbf{g}$:

$key = (r << 16)\&(g << 8)\&b;$

$h = key \bmod h\_size;$

$i = 1;$

(ab) Wiederhole Folgendes maximal so lange, bis alle Positionen der Hash-Tabelle inspiziert sind:

(aba) Falls der Eintrag in der Position h der Hash-Tabelle leer ist:

Farbvektor $\mathbf{g}$ eintragen;

$h\_tab.count = 1;$

Beende die Schleife (ab);

(abb) Falls der Eintrag in der Position h der Hash-Tabelle nicht leer ist und der Farbvektor $\mathbf{g}$ mit dem eingetragenen Farbvektor identisch ist:

$h\_tab.count = h\_tab.count + 1;$

Beende die Schleife (ab);

(abc) Falls der Eintrag in der Position h der Hash-Tabelle nicht leer ist und der Farbvektor $\mathbf{g}$ mit dem eingetragenen Farbvektor nicht identisch ist (Kollision):

Auflösen der Kollision durch Sondieren eines neuen Speicherplatzes:

$h = h + i;\ i = i + l;$ (z.B. $l = 1$ oder $l = 2$)

falls $h \geq h\_size:\ h = h - h\_size;$

(ac) Falls der Farbvektor $\mathbf{g}$ nicht eingetragen werden konnte, war die Hash-Tabelle zu klein gewählt.

Ende des Algorithmus

12.5.2 Erstellen einer Farb-Look-Up-Tabelle

Zur Erstellung einer Farb-*look-up*-Tabelle $cLuT$ aus dem Histogramm der Farben werden unterschiedliche Verfahren angewendet. Ihr Ziel ist es, aus dem Histogramm die „wichtigsten“ l_size Farbtöne (l_size: Länge der $cLuT$, z.B. $l\_size = 256$) herauszufinden. Zwei Verfahren werden im Folgenden erläutert.

12.5.2.1 Das Popularity-Verfahren

Das Popularity-Verfahren ist sehr einfach: Das Histogramm der Farben (h_tab) wird der Größe nach sortiert, dann werden die ersten l_size Farben, also die häufigsten, ausgewählt.

Es kann hier vorkommen, dass Farben, die zwar weniger häufig, jedoch subjektiv „wichtig“ sind, nicht mit erfasst werden. Sie werden dann bei der reduzierten Darstellung möglicherweise völlig falsch wiedergegeben. Das liegt daran, dass die Farben im RGB-Farbraum *cluster* bilden: Ein bestimmter Farbwert tritt bei natürlichen Bildern nicht alleine häufig auf, sondern es werden auch die Farben seiner Umgebung häufig sein. Als Beispiel erzeuge ein Bild im RGB-Raum fünf *cluster*, deren Farben im Bild alle subjektiv wichtig erscheinen, die jedoch in den Häufigkeiten deutlich abnehmen. Der Popularity- Algorithmus wird dann möglicherweise die Farben der ersten drei „größeren“ *cluster* erfassen, die letzten zwei aber nicht.

12.5.2.2 Das Median-Cut-Verfahren

Das *median-cut*-Verfahren versucht den RGB-Farbraum so aufzuteilen, dass auch kleinere Farbhäufungen (*cluster*) erfasst werden. Zunächst die Darstellung des Verfahrens, dann ein Beispiel.

A12.4: Median-Cut-Verfahren.

Voraussetzungen und Bemerkungen:

◇ h_tab sei das Farbhistogramm gemäß Algorithmus **A12.3** mit der oben angegebenen Datenstruktur und den Datenfeldern $h\_tab.count$, $h\_tab.R$, $h\_tab.G$ und $h\_tab.B$.

◇ l_size sei die vorgegebene Anzahl der zu erzeugenden Farbrepräsentanten in der $cLuT$.

Algorithmus:

(a) Es wird eine Liste von Teiltabellen erstellt. Sie wird mit *list* bezeichnet. Zu Beginn des Verfahrens enthält *list* nur ein Element, nämlich das gesamte Histogramm h_tab.

(b) Wiederhole Folgendes so lange, bis die Liste der Teiltabellen *list* genau *l_size* Elemente besitzt:

(ba) Wiederhole Folgendes für alle Elemente von *list*. Das jeweils aktuell verarbeitete Element von *list* wird mit *act_tab* bezeichnet:

(baa) Ermittle zu den Spalten *act_tab.R*, *act_tab.G* und *act_tab.B* die Minima und Maxima min_R, max_R, min_G, max_G, min_B und max_B.

(bab) *act_col* sei die Spalte von *act_tab*, für die der Wertebereich maximal ist: $max = \max\{(max_R - min_R), (max_G - min_G), (max_B - min_B)\}$;

(bac) Sortiere die *act_tab* nach Maßgabe der Spalte *act_col*.

(bad) Teile die *act_tab* am Medianwert nach Maßgabe der Spalte *act_col*.

Die Teilung am Medianwert wird so durchgeführt, dass etwa gleichviele Farbvektoren in beiden Hälften auftreten.

(bae) Falls *list* jetzt *l_size* Elemente enthält: Beende beide Schleifen (weiter bei (c)).

(bb) Nachdem jetzt eine neue Liste *list* erstellt ist, sortiere die Elemente von *list*, fallend nach der Häufigkeit der Farbvektoren. In jeder Teilliste (jedem Element von *list*) werden die Felder *count* aufsummiert. Am Anfang der sortierten Liste steht dasjenige Element, das die größte Summe ergibt. Bei der Sortierung werden nur solche Elemente berücksichtigt, die mehr als einen Farbvektoreintrag enthalten.

(c) Bestimme die *l_size* Farbrepräsentanten. Falls ein Element von *list* nur aus einem Farbvektor besteht, wird dieser als Farbrepräsentant verwendet. Für Elemente von *list*, die mehr als einen Farbvektor enthalten: Bilde den Mittelwertfarbvektor und verwende ihn als Repräsentanten.

Ende des Algorithmus

Der *median-cut*-Algorithmus ist in verschiedenen Softwareprodukten, die eine Verarbeitung von Farbbildern erlauben, bereits integriert. Als Beispiele seien hier erwähnt:

- *Image Alchemy* von Handmade Software, Inc., Los Gatos, CA. oder das
- PPM-Software-Paket, das in vielen UNIX-Netzen auch als Quellcode zur Verfügung steht.

12.5.3 Abbildung der Farben des Originalbildes in die Farbtabelle

Nachdem die Farb-*look-up*-Tabelle $cLuT$ berechnet ist, müssen die Farben des Originalbildes $\mathbf{S}_e$ in die $cLuT$ abgebildet werden. Zu jedem Farbvektor von $\mathbf{S}_e$ wird ermittelt, zu welchem Farbvektor der $cLuT$ er im RGB-Farbraum den geringsten Abstand hat. Durch diesen wird er dann repräsentiert. Es ist jedoch nicht notwendig, dies für alle Bildpunkte des Originalbildes durchzuführen. Im Farbhistogramm h_tab (der Hash-Tabelle) sind nämlich alle Farbvektoren von $\mathbf{S}_e$ erfasst. Es genügt also, für alle in h_tab auftretenden Vektoren die Zuordnung zur Farb-*look-up*-Tabelle durchzuführen. Dazu kann sogar dieselbe Tabelle verwendet werden, da die Werte von $h\_tab.count$ nicht mehr benötigt werden.

A12.5: Abbildung der Farben des Originals.

Voraussetzungen und Bemerkungen:

◇ h_tab sei das Farbhistogramm gemäß Algorithmus **A12.3** mit der angegebenen Datenstruktur und den Datenfeldern $h\_tab.count$, $h\_tab.R$, $h\_tab.G$ und $h\_tab.B$.

◇ $cLuT$: Farb-*look-up*-Tabelle mit den ausgewählten Farbrepräsentanten des Bildes $\mathbf{S}_e$. Datenfelder von $cLuT$: $cLuT.R$, $cLuT.G$ und $cLuT.B$.

Algorithmus:

(a) Für alle Einträge i von h_tab:

(ab) Für alle Einträge j von $cLuT$:

(aba) Berechne die minimale Distanz von $(h\_tab.R[i],\ h\_tab.G[i],\ h\_tab.B[i])$ zu $(cLuT.R[j],\ cLuT.G[j],\ cLuT.B[j])$.

Dies sei der Eintrag k von $cLuT$.

(abb) Setze $h\_tab.count[i] = k$.

Ende des Algorithmus

12.5.4 Segmentierung des Originalbildes

Nach diesen Vorbereitungen ist die Umsetzung des Originalbildes zu einem Index-Bild einfach: Jedem Farbvektor des Originalbildes wird ein Zeiger (Index) in die $cLut$ zugeordnet. Welche Position in $cLut$ das ist, ist in h_tab gespeichert.

A12.6: Transformation des Originals.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y, n))$ ist ein dreikanaliges RGB-Eingabebild.

◇ $\mathbf{S}_a = (s_a(x, y))$ ist ein einkanaliges Ausgabebild (Indexbild).

◇ *cLuT*: Farb-*look-up*-Tabelle mit den ausgewählten Farbrepräsentanten (wie oben);

Algorithmus:

(a) Für alle Bildpunkte $\mathbf{s}_e(x, y)$ des Bildes $\mathbf{S}_e = (s(x, y, n))$:

(aa) Suche den Farbton $\mathbf{s}_e(x, y) = \mathbf{g} = (r, g, b)^T$ in h_Tab. Dies sei der k-te Eintrag.

(ab) Setze: $s_a(x, y) = h\_tab.count[k]$.

Ende des Algorithmus

Es ist klar, dass zum farbreduzierten, einkanaligen Ergebnisbild $\mathbf{S}_a = (s_a(x, y))$ zusätzlich die Farb-*look-up*-Tabelle *cLuT* gehört, da die „Grau-"Werte $s_a(x, y)$ ja Indizes in die *cLuT* sind.

Auf Bildbeispiele zu diesen Verfahren wird verzichtet, da die oft nur geringen Unterschiede nur auf guten Farbmonitoren zu sehen sind und im Druck verloren gehen würden.

12.5.5 Segmentierung des Originalbildes mit Dithering

Mit der Technik des *Dithering* kann der Segmentierungsschritt zusätzlich verbessert werden. Diese Technik ist vor allem dann geeignet, wenn die Segmentierung auf wenige Farben erfolgte, also eine mit nur wenigen Farben besetzte *cLuT* berechnet wurde.

Das Verfahren verteilt den Fehler, der bei der Darstellung eines Originalfarbvektors $\mathbf{s}_e(x, y)$ durch seinen Repräsentanten aus der *cLuT* gemacht wird, auf die Nachbarvektoren. Um dies in einem einzigen Durchlauf berechnen zu können, wird der Fehler nur auf die rechts und unten vom aktuellen Bildpunkt $\mathbf{s}_e(x, y)$ liegenden Nachbarn übertragen, wobei die Nachbarn unterschiedlich gewichtet werden:

$\mathbf{s}_e(x, y)$	$\mathbf{s}_e(x, y+1) + \frac{3}{8}\mathbf{q}$
$\mathbf{s}_e(x+1, y) + \frac{3}{8}\mathbf{q}$	$\mathbf{s}_e(x+1, y+1) + \frac{1}{8}\mathbf{q}$

Der Algorithmus dazu ist in **A12.7** zusammengestellt.

A12.7: Abbildung der Farben des Originals mit Dithering.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y, n))$ ist ein dreikanaliges RGB-Eingabebild.

◇ $\mathbf{S}_a = (s_a(x, y))$ ist ein einkanaliges Ausgabebild (Indexbild).

◇ *h_tab* das Farbhistogramm (wie oben).

◇ *cLuT*: Farb-*look-up*-Tabelle mit den ausgewählten Farbrepräsentanten (wie oben).

Algorithmus:

(a) Für alle Bildpunkte $\mathbf{s}_e(x, y)$ des Bildes $\mathbf{S}_e$:

(aa) Suche den Farbton $\mathbf{s}_e(x, y) = \mathbf{g} = (r, g, b)^T$ in *h_tab*. Dies sei der j-te Eintrag, der auf den k-ten Eintrag von *cLut* weist.

(ab) Setze: $s_a(x, y) = h\_tab.count[j]$.

(ac) Berechne den Quantisierungsfehler $\mathbf{q}$:

$$\mathbf{q} = \begin{pmatrix} r - cLuT.R[k] \\ g - cLut.G[k] \\ b - cLut.B[k] \end{pmatrix}$$

(ad) Verteile den Quantisierungsfehler auf benachbarte Pixel:

$$\mathbf{s}_e(x, y+1) = \mathbf{s}_e(x, y+1) + \frac{3}{8} * \mathbf{q};$$

$$\mathbf{s}_e(x+1, y) = \mathbf{s}_e(x+1, y) + \frac{3}{8} * \mathbf{q};$$

$$\mathbf{s}_e(x+1, y+1) = \mathbf{s}_e(x+1, y+1) + \frac{1}{8} * \mathbf{q};$$

Ende des Algorithmus

12.5.6 Unüberwachte Klassifikatoren zur Reduktion der Farben

Ein etwas anderer Weg wird beschritten, wenn der RGB-Farbraum mit *multivariaten Klassifikatoren* (Kapitel 20) untersucht und aufgeteilt wird. Hier sind vor allem unüberwachte Klassifikatoren gut geeignet, da sie die Aufteilung des RGB-Merkmalsraums ohne ein vom Bearbeiter zur Verfügung gestelltes Trainingsset (Stichprobe) analysieren.

Dabei geht man folgendermaßen vor: Es wird die Anzahl der zu erzeugenden repräsentativen Farben, also der Umfang *c_size* der Farb-*look-up*-Tabelle *cLuT* vorgegeben. Dann wird ein unüberwachter *cluster*-Algorithmus angewendet, der den RGB-Farbraum in ebenso viele (*c_size*) *cluster* aufteilt. Die Mittelwertvektoren der *cluster* werden dann als Farbrepräsentanten verwendet.

Aus der Vielfalt der möglichen unüberwachten *cluster*-Algorithmen sei der einfache *Minimum-Distance-Cluster-Algorithmus* vorgestellt.

A12.8: Minimum-Distance-Cluster-Algorithmus.

Voraussetzungen und Bemerkungen:

$\diamond$ $\mathbf{S}_e = (s_e(x,y,n)), n = 0,1,2; \mathbf{s}_e(x,y) = \mathbf{g} = (r,g,b)^T$ sei ein RGB-Eingabebild.

Algorithmus:

(a) Wähle einen beliebigen Bildpunkt von $\mathbf{S}_e$ als Zentrum $\vec{z}_0$ des ersten *clusters* aus.

(b) Für alle Bildpunke $\mathbf{g} = \mathbf{s}_e(x,y)$ von $\mathbf{S}_e$:

(ba) Für alle bereits festgelegten *cluster*-Zentren $\vec{z}_i$:

(baa) Ermittle die minimale Distanz $d_i = d(\mathbf{g}, \vec{z}_i)$. Dies sei d_j.

(bb) Falls d_j kleiner als der vorgegebene Schwellwert c ist: Ordne $\mathbf{g}$ dem *cluster* j zu.

(bc) Falls d_j nicht kleiner als der vorgegebene Schwellwert c ist:

(bca) Falls die maximale Anzahl an *cluster*-Zentren noch nicht erreicht ist: Setze $\mathbf{g}$ als neues *cluster*-Zentrum $\vec{z}$.

(bcb) Falls die maximale Anzahl an *cluster*-Zentren erreicht ist: Ordne $\mathbf{g}$ dem *cluster* j zu.

Ende des Algorithmus

Ein großer Vorteil dieses Verfahrens liegt in der schnellen Verarbeitungszeit, da die *cluster*-Bildung in einem einzigen Durchlauf durch das gesamte Bild $\mathbf{S}_e$ berechnet wird.

Ein Nachteil ist sicher, dass das Ergebnis abhängig von der Verarbeitungsreihenfolge ist. Es kann z.B. der Fall auftreten, dass Bildpunkte einem *cluster* j zugeordnet werden, solange

ein neues *cluster k* noch nicht erzeugt wurde. Wird dieses erzeugt, so werden Farbvektoren, die vorher *cluster j* zugewiesen wurden, ab diesem Zeitpunkt dem *cluster k* zugeordnet.

Dieser Effekt wird vermieden, wenn die *cluster*-Bildung iterativ mit mehreren Durchläufen durch die Bilddaten erfolgt. Ein Verfahren dieser Art, das auf der Basis der *fuzzy logic* aufbaut, wird in Kapitel 22 beschrieben.

Abschließend sei noch bemerkt, dass der Minimum-Distance-Cluster-Algorithmus nicht auf dreidimensionale RGB-Farbräume beschränkt ist, sondern auch als unüberwachter Klassifikator bei N-dimensionalen Merkmalsräumen eingesetzt werden kann.

Kapitel 13

Merkmale aus mehrkanaligen Bildern

13.1 Anwendungen

In den vorhergehenden Kapiteln wurden Bildverarbeitungsverfahren beschrieben, die jeweils nur einen Bildkanal verarbeiten. In diesem Kapitel werden nun Verfahren erläutert, die die Informationen von mehreren Bildkanälen verknüpfen. Handelt es sich dabei um Farb- oder Multispektralbilder, so lassen sich mit Differenz- oder Ratiobildung oft einprägsame Darstellungen für die visuelle Weiterverarbeitung erzielen. Durch die Verknüpfung von Kanälen bei logischen Bildern lassen sich viele praktische Effekte, z.B. Einblendungen erzielen. Den letzten Teil dieses Kapitels bildet die Hauptkomponententransformation, mit der versucht wird, nach einer Rotation des Merkmalskoordinatensystems redundante Informationen zu entfernen.

13.2 Summe, Differenz, Ratio

Für die Darstellungen in diesem Kapitel wird zunächst ein mehrkanaliges Bild vorausgesetzt gemäß:

$$\begin{aligned} \mathbf{S}_e = (s_e(x,y,n)) \qquad & x = 0,1,...,L-1, \text{(Bildzeilen)} \\ & y = 0,1,...,R-1, \text{(Bildspalten) und} \\ & n = 0,1,...,N-1, \text{(Kanäle).} \end{aligned} \tag{13.1}$$

Ein einfaches Beispiel einer mehrkanaligen Operation ist die *gewichtete additive Verknüpfung* der einzelnen Kanäle:

$$\begin{aligned} & \mathbf{S}_e \rightarrow \mathbf{S}_a : \\ & s_a(x,y) = \frac{1}{N} \sum_{n=0}^{N-1} a_n \cdot s_e(x,y,n); \quad \sum_{n=0}^{N-1} a_n = 1. \end{aligned} \tag{13.2}$$

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_13

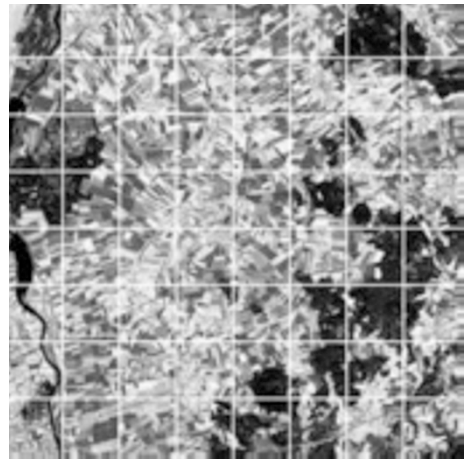


Bild 13.1: Geometrisch entzerrtes Satellitenbild mit überlagertem Gitternetz.

Ist dabei $\mathbf{S}_e$ ein Multispektralbild, so stellt der Grauwert $s_a(x, y)$ den Mittelwert des Bildpunktes in den einzelnen Kanälen dar. Speziell im Fall eines digitalisierten Farbbildes mit Rot-, Grün- und Blauauszug entsprechen dann die Grauwerte des transformierten Bildes in etwa den panchromatischen Grauwerten, die entstehen, wenn das Original ohne Farbfilter, also nur als Grautonbild, digitalisiert würde (HSI-Modell: Intensität).

Ein anderes Beispiel zur additiven Verknüpfung ist das Einblenden von grafischen Informationen in ein digitalisiertes Bild. Ist $s_e(x, y, 0)$ ein beliebiges Grauwertbild und $s_e(x, y, 1)$ ein weiterer Kanal mit grafischen Informationen wie Strichzeichnungen, Gittermuster, Text, usw., so kann durch die Operation (13.2) diese Information in den ersten Kanal eingeblendet werden (Bild 13.1). Dabei wird in dieser Darstellung nur vorausgesetzt, dass die grafische Information weiß (Grauwert 255) auf schwarzem Hintergrund (Grauwert 0) vorliegt.

Durch verschiedene Modifikationen kann (13.2) so geändert werden, dass z.B. die eingeblendete Information transparent oder nicht transparent ist oder dass überlagerte Strichzeichnungen immer einen optimalen Kontrast zum Hintergrund haben.

Die *Differenz* zweier Kanäle k und l sieht Folgendermaßen aus:

$$\mathbf{S}_e \rightarrow \mathbf{S}_a :$$
$$s_a(x, y) = s_e(x, y, k) - s_e(x, y, l) + c. \tag{13.3}$$

Die Konstante c wird verwendet, um negative Grauwerte zu vermeiden. Sie kann z.B. c=127 gesetzt werden. Im Ausgabebild $\mathbf{S}_a$ oszillieren dann die Differenzen um den Grauwert

127. Treten dennoch Differenzen auf, die größer als 127 sind, so werden diese Werte auf 0 bzw. 255 gesetzt. Eine andere Möglichkeit besteht darin, dass zunächst die größte Differenz ermittelt wird und die Werte von $\mathbf{S}_a$ so skaliert werden, dass sie in die Grauwertmenge G passen.

Sind bei der Differenzbildung in (13.3) die Kanäle Multispektralkanäle, so sind im Bild $\mathbf{S}_a$ diejenigen Bereiche hervorgehoben, bei denen spektrale Unterschiede auftreten. Um diese Unterschiede deutlicher herauszuarbeiten, kann eine Kontrastanreicherung des Bildes $\mathbf{S}_a$ sinnvoll sein.

In Kapitel 4 wurden einige Beispiele zur Kalibrierung vorgestellt. Mit der Differenzbildung gemäß (13.3) lassen sich ebenfalls Kalibrierungsprobleme bearbeiten, so z.B. die Elimination von Vignettierungseinflüssen. Wird mit Hilfe einer Videokamera, die über AD-Wandler an ein Rasterbildspeichersystem angeschlossen ist, eine Reihe von Bildvorlagen digitalisiert, so ist durch die Einflüsse des Kameraobjektivs zu erwarten, dass die digitalisierten Bilder mehr oder weniger starke Randabschattungen aufweisen. Sollen diese Bilddaten anschließend zu Binärbildern umgewandelt werden, so ist der Schwellwert ortsabhängig und die Qualität der Binärbilder dementsprechend schlecht. Wird bei der Digitalisierung ein homogen weißes Kalibrierungsbild mit verarbeitet, so kann dieses von allen anderen Bildern abgezogen werden, so dass in den Differenzenbildern die Randabschattungen eliminiert sind. Dabei ist im jeweiligen Anwendungsfall die Konstante c passend zu wählen.

Bei der Verarbeitung von Multispektralbildern wird statt der Differenz zweier Kanäle k und l häufiger das *Ratio* verwendet:

$$\mathbf{S}_e \rightarrow \mathbf{S}_a :$$
$$s_a(x,y) = \frac{s_e(x,y,k) - s_e(x,y,l)}{s_e(x,y,k) + s_e(x,y,l)} \cdot c_1 + c_2. \tag{13.4}$$

Das Ratio ist invariant gegen die Multiplikation der Kanäle des Originalbildes $\mathbf{S}_e$ mit einem konstanten Faktor. Wenn nun proportional zur Leuchtdichte des von der Kamera aufgenommenen Objektes quantisiert wurde, dann ist das Ratiobild $\mathbf{S}_a$ unabhängig von der Beleuchtung des Objektes. Im übrigen liefern Differenz und Ratio sehr ähnliche Bilder (13.2).

Da sich mit Falschfarbencodierung immer nur drei Bildkanäle den drei Grundfarben Rot, Grün und Blau zuordnen lassen, haben die in diesem Abschnitt aufgeführten Beispiele wichtige Anwendungen bei der farbbildlichen Darstellung von Multispektralbildern. Durch die gezeigten Operationen kann die Anzahl der Kanäle reduziert und nicht gewünschte Bildinformation eliminiert werden. Durch eine geschickte farbliche Darstellung können dann Bilder erstellt werden, die in vielen Anwendungsfällen ausgezeichnet für weitere visuelle Interpretationen geeignet sind. Umgekehrt kann man aus wenigen Kanälen durch Verwendung zusätzlicher Kenntnisse über die aufgenommenen Objekte mehrere Kanäle erzeugen. Als Beispiel seien Satellitenaufnahmen genannt, die nur mit zwei Spektralausschnitten vorliegen. Wenn man die Korrelationen der vorhandenen Kanäle bei typischen terrestrischen Objekten (z.B. Wald, Gewässer, Wiese, Wüste, Gestein, usw.) kennt, kann

(a)

(b)

(c)

Bild 13.2: Differenz und Ratio zweier Spektralbereiche eines digitalisierten Farbbildes. (a) Original. (b) Differenz: Rot- und Blauauszug (kontrastverstärkt). (c) Ratio: Rot- und Blauauszug (kontrastverstärkt).

man einen dritten ,,Farbauszug" berechnen und dann eine Echt- oder Falschfarbendarstellung erzeugen.

13.3 Verknüpfung von Kanälen bei logischen Bildern

In diesem Abschnitt wird anhand eines einfachen Beispiels gezeigt, wie einzelne Bildkanäle, mit Grauwertbildern und logischen Bildern miteinander verknüpft werden können. Dazu werden folgende Annahmen gemacht:

Es sei $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1, 2$ ein Bild mit drei Kanälen, die Folgendes enthalten (Bild 13.3-a, -b und -c):

Kanal $s_e(x, y, 0)$:	Ein Grauwertbild.
Kanal $s_e(x, y, 1)$:	Ein Grauwertbild.
Kanal $s_e(x, y, 2)$:	Ein Maskenbild mit den Grauwerten 0 und 255.

In Bild 13.3-d wurden die drei Kanäle des Eingabebildes so kombiniert, dass der Bildhintergrund von Bild 13.3-a durch den Hintergrund von Bild 13.3-b ersetzt wurde.

$$\begin{aligned} s_a(x, y) &= \Big((s_e(x, y, 2).\text{EQUAL.0}) \cdot s_e(x, y, 0)\Big) + \\ &+ \Big((s_e(x, y, 2).\text{EQUAL.255}) \cdot s_e(x, y, 1)\Big). \end{aligned} \tag{13.5}$$

Die Boole'schen Ausdrücke in den inneren Klammern liefern den Wahrheitswert `TRUE`, der rechnerintern mit 1 (manchmal auch mit -1) codiert ist, falls im Kanal 2 ein Bildpunkt den Grauwert 0 bzw. 255 hat. In allen anderen Fällen liefern sie den Wahrheitswert `FALSE`, codiert mit 0. Durch die Multiplikation mit dem jeweiligen Grauwert im Kanal 0 bzw. Kanal 1 haben diese Bildpunkte im Ausgabebild $\mathbf{S}_a$ die gewünschten Grauwerte.

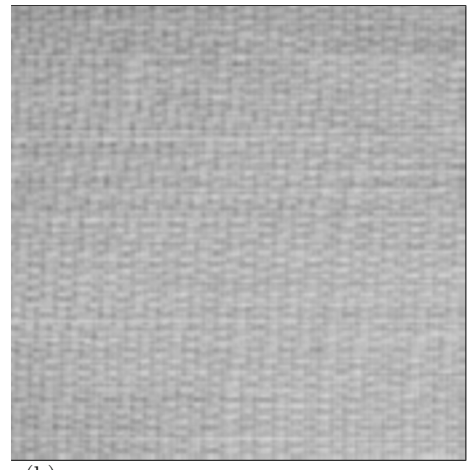
Wenn das Maskenbild kein Zweipegelbild ist, sondern einzelne Flächenstücke mit unterschiedlichen Grauwerten codiert sind, können gezielt bestimmte Flächen kombiniert werden. Die Bilder 13.4 zeigen dazu ein Beispiel.

Diese Beispiele zeigen die vielfältigen Auswertungsmöglichkeiten, wenn mehrkanalige Bilder dieser Art vorliegen. Bei Bilddaten z.B. aus der Fernerkundung lassen sich mit den vorgestellten Techniken hilfreiche Zusatzinformationen für Anwendungen in der Raumplanung, Geologie, Ozeanografie, Ökologie, erzeugen.

13.4 Die Hauptkomponententransformation

Die Hauptkomponententransformation ist eine Standardmethode, die zur Datenreduktion eingesetzt wird. Zur Erläuterung wird zunächst ein zweikanaliges Bild $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1$, betrachtet, dessen Kanäle relativ stark korreliert sind. Die Verteilung der Grauwerte könnte dann etwa durch das zweidimensionale Histogramm von Bild 13.5-d wiedergegeben

(a)

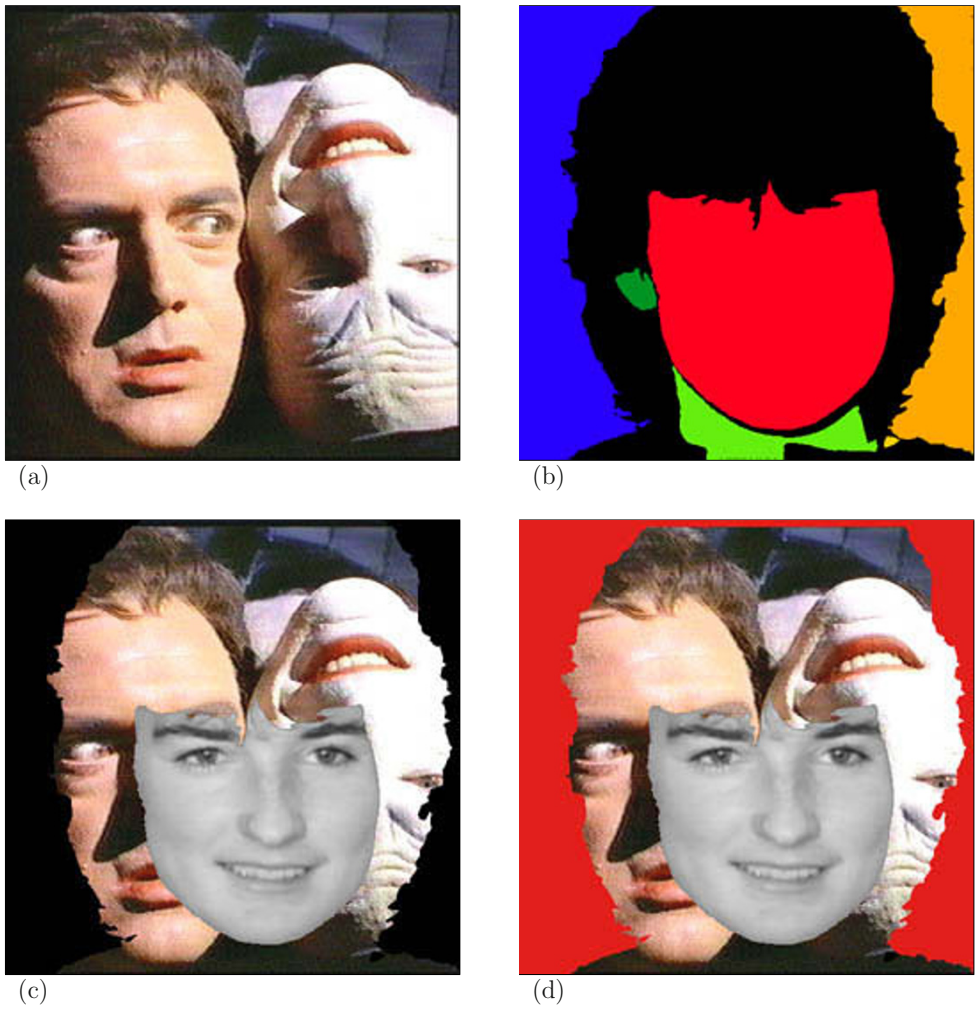


(b)

(c)

(d)

Bild 13.3: Beispiel zur Kombination von Bildkanälen. (a) Grauwertbild im ersten Kanal des Eingabebildes. (b) Grauwertbild im zweiten Kanal des Eingabebildes. (c) Maskenbild (logisches Bild) im dritten Kanal des Eingabebildes mit den Grauwerten, 0 und 255. (d) Ergebnis der Verknüpfung.



(a) (b)

(c) (d)

Bild 13.4: Beispiel zur Kombination von Bildkanälen. (a) Originalbild. (b) Maskenbild, in dem Teilflächen mit unterschiedlichen Grauwerten codiert sind. Diese Grauwerte sind hier mit Pseudofarben dargestellt. (c) Beispiel einer Bildkombination. (d) Hier sind die Hintergrundflächen zusätzlich mit einem roten Farbton aufgefüllt.

werden. Durch die starke Korrelation der Grauwerte der beiden Kanäle kann zu jedem Grauwert im ersten Kanal ein mehr oder weniger breites Grauwertintervall im zweiten Kanal angegeben werden. Der Extremfall wäre die lineare Abhängigkeit der beiden Kanäle. In diesem Fall könnte zu jedem Grauwert im ersten Kanal der zugehörige Grauwert im zweiten Kanal exakt angegeben werden. Der zweite Kanal wäre dann redundant. Aber nicht nur bei linearer Abhängigkeit, sondern auch bei starker Korrelation, beinhalten die beiden Kanäle redundante Grauwertinformationen.

Die Hauptkomponententransformation besteht nun in einer Drehung der Koordinatenachsen der beiden Kanäle, und zwar so, dass die erste Achse in Richtung der größten Streuung der Grauwertkombinationen von Kanal 0 und 1 liegt und die zweite senkrecht dazu. Die neuen Koordinatenachsen heißen die *erste* und die *zweite Hauptkomponente*. In Bild 13.5-d sind sie angedeutet. Die neuen Achsen werden, um negative Grauwerte zu vermeiden, jeweils um den Wert 127 verschoben.

In diesem zweidimensionalen Fall besteht die Hauptkomponententransformation aus einer Drehung (und Verschiebung) des Merkmalskoordinatensystems um einen Winkel α, der die Richtung der stärksten Streuung angibt. Die Transformationsmatrix ist die bekannte Drehmatrix:

$$\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \tag{13.6}$$

Nach der Drehung hat die erste Hauptkomponente die Richtung der maximalen Streuung, während die zweite Hauptkomponente in der Regel eine geringe Streuung aufweist, so dass auf diesen neuen zweiten Kanal verzichtet werden kann, ohne allzuviel Information zu verlieren. Bild 13.5-e zeigt das gedrehte Koordinatensystem der ersten und zweiten Hauptkomponente zu den n Bildkanälen von Bild 13.5-b und 13.5-c. Um negative Grauwerte zu vermeiden, müssen bei praktischen Anwendungen die beiden Hauptkomponenten so gelegt werden, dass der jeweilige mittlere Grauwert gleich 127 ist.

Die Bilder 13.5-f und 13.5-g zeigen die erste und die zweite Hauptkomponente. Man sieht deutlich, dass die zweite Hauptkomponente nur mehr geringe Bildinformation enthält.

Nach dieser einführenden Darstellung am zweidimensionalen Fall soll nun ein N-kanaliges Bild $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1, ..., N-1$ vorausgesetzt werden. Die Ableitung der Hauptkomponententransformation wird jetzt anhand statistischer Methoden durchgeführt. Die Bildpunkte

$$\mathbf{s}_e(x, y) = (g_0, g_1, ..., g_{N-1})^T$$

von $\mathbf{S}_e$ werden als Realisierungen einer N-dimensionalen Zufallsvariablen

$$\mathbf{x} = (X_0, X_1, ..., X_{N-1})^T$$

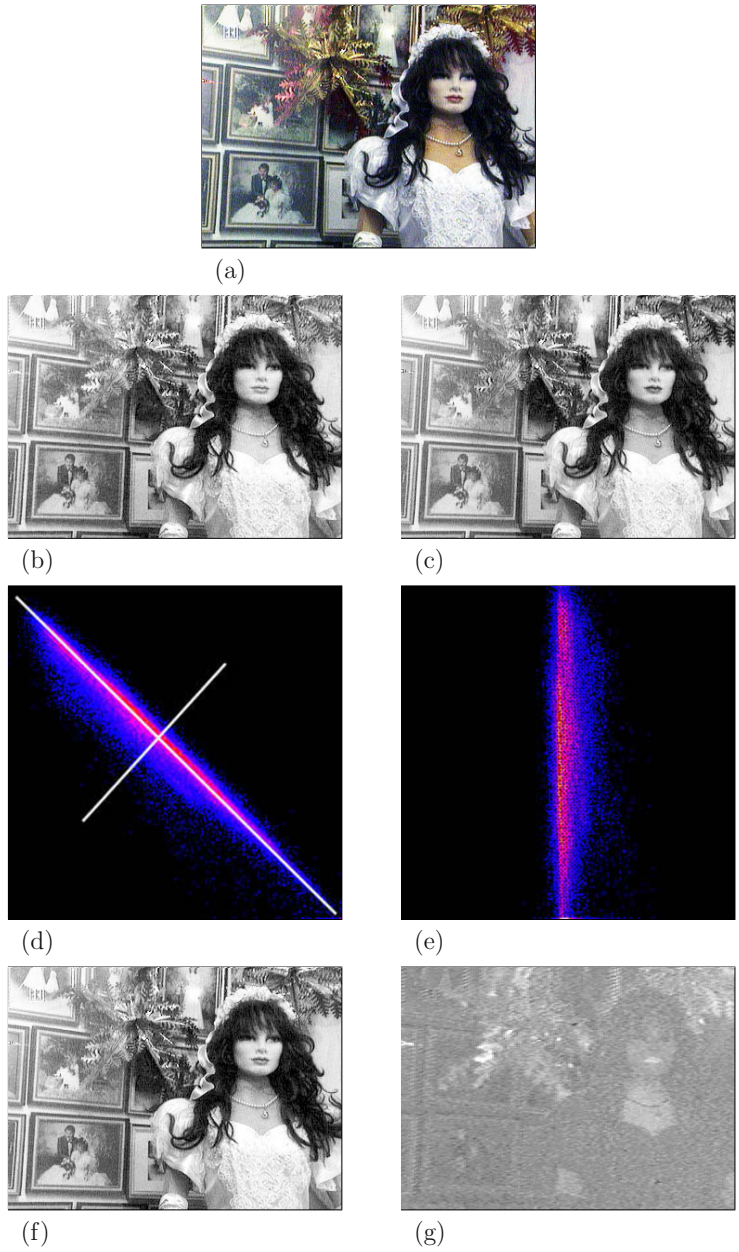


Bild 13.5: Zweidimensionale Grauwertverteilung eines zweikanaligen Bildes, dessen Kanäle stark korreliert sind. (a) Farbbild. (b) Roter Kanal. (c) Blauer Kanal. (d) zweidimensionales Histogramm roter/blauer Kanal. (e) zweidimensionales Histogramm, erste und zweite Hauptkomponente. (f) Erste Hauptkomponente. (g) Zweite Hauptkomponente.

aufgefasst. Zur Herleitung wird o.B.d.A. angenommen, dass der Erwartungswert $E(\mathbf{x}) = \mathbf{0}$ ist und $\mathbf{x}$ die Kovarianzmatrix $\mathbf{C}$ besitzt. Die Verteilung der Zufallsvariablen $\mathbf{x}$ ist im Folgenden nicht wesentlich.

Die Hauptkomponenten sind Linearkombinationen der Zufallsvariablen $X_0, X_1, ..., X_{N-1}$, wobei an die Streuungen spezielle Anforderungen gestellt werden. Die erste Hauptkomponente ist diejenige Linearkombination $\mathbf{a}^T\mathbf{x}$ mit

$$\mathbf{a} = (a_0, a_1, ..., a_{N-1}) \text{ und } \mathbf{a}^T\mathbf{a} = 1 \text{ (normiert)},$$

deren Streuung maximal ist. Für die Streuung von Linearkombination $\mathbf{a}^T\mathbf{x}$ ergibt sich:

$$E\Big((\mathbf{a}^T\mathbf{x})^2\Big) = E\Big(\mathbf{a}^T\mathbf{x}\mathbf{a}^T\mathbf{x}\Big) = E\Big(\mathbf{a}^T\mathbf{x}\mathbf{x}^T\mathbf{a}\Big) = \mathbf{a}^T E\Big(\mathbf{x}\mathbf{x}^T\Big)\mathbf{a} = \mathbf{a}^T\mathbf{C}\mathbf{a}. \tag{13.7}$$

Die Nebenbedingung $\mathbf{a}^T\mathbf{a} = 1$ wird durch einen Ansatz mit einem Lagrange'schen Multiplikator berücksichtigt:

$$\Phi = \mathbf{a}^T\mathbf{C}\mathbf{a} - \lambda(\mathbf{a}^T\mathbf{a} - 1). \tag{13.8}$$

Die Streuung von Φ ist maximal, falls die Ableitung

$$\frac{\partial\Phi}{\partial\mathbf{a}} = 2\mathbf{C}\mathbf{a} - 2\lambda\mathbf{a} = \mathbf{0} \text{ oder } (\mathbf{C} - \lambda\mathbf{I})\mathbf{a} = \mathbf{0}. \tag{13.9}$$

λ ist also ein Eigenwert der Kovarianzmatrix $\mathbf{C}$. Das charakteristische Polynom

$$\det(\mathbf{C} - \lambda\mathbf{I}) = 0 \tag{13.10}$$

ergibt N Eigenwerte $\lambda_0 \geq \lambda_1 \geq ... \geq \lambda_{N-1}$. Die Streuung von $\mathbf{a}^T\mathbf{x}$ kann mit diesen Voraussetzungen durch die Eigenwerte λ_i ausgedrückt werden. Dazu wird (13.9) von links mit $\mathbf{a}^T$ multipliziert:

$$\begin{aligned} &\mathbf{a}^T(\mathbf{C} - \lambda\mathbf{I})\mathbf{a} = 0, \\ &\mathbf{a}^T\mathbf{C}\mathbf{a} - \lambda\mathbf{a}^T\mathbf{a} = 0, \\ &E\Big((\mathbf{a}^T\mathbf{x})^2\Big) = \mathbf{a}^T\mathbf{C}\mathbf{a} = \lambda. \end{aligned} \tag{13.11}$$

Die Streuung von $\mathbf{a}^T\mathbf{x}$ ist also maximal, wenn λ den Wert des größten Eigenwertes λ_0 annimmt und der Koeffizientenvektor $\mathbf{a}$ der zu λ_0 gehörige Eigenvektor $\mathbf{a}_0$ ist. Die erste Hauptkomponente ist dann die Linearkombination $\mathbf{a}_0^T\mathbf{x}$ mit der Streuung λ_0.

Die Bestimmung der zweiten Hauptkomponente ergibt sich aus der Forderung, dass $\mathbf{a}_1^T\mathbf{x}$ maximale Streuung unter all jenen Linearkombinationen besitzt, die mit $\mathbf{a}_0^T\mathbf{x}$ unkorreliert sind. Die weitere Berechnung erfolgt sinngemäß wie oben.

Es zeigt sich letztlich, dass die N Hauptkomponenten der Zufallsvariablen $\mathbf{x}$ die Linearkombinationen

$$\mathbf{a}_0^T\mathbf{x}, \quad \mathbf{a}_1^T\mathbf{x}, \quad ..., \quad \mathbf{a}_{N-1}^T\mathbf{x} \tag{13.12}$$

sind, wobei $\mathbf{a}_i$ der zum Eigenwert $\lambda_i(\lambda_0 \geq \lambda_1 \geq ... \geq \lambda_{N-1})$ gehörige Eigenvektor der Kovarianzmatrix $\mathbf{C}$ ist. Die Streuung von $\mathbf{a}_i^T\mathbf{x}$ ist λ_i.

Die praktische Berechnung wird nun wie folgt durchgeführt:

- Berechnung der Kovarianzmatrix $\mathbf{C}$ des Bildes $\mathbf{S}_e$.
- Berechnung der Eigenwerte und der Eigenvektoren von $\mathbf{C}$.
- Ordnen der Eigenwerte, so dass gilt: $\lambda_0 \geq \lambda_1 \geq ... \geq \lambda_{N-1}$. Die Matrix $\mathbf{V}$ der Eigenvektoren wird sinngemäß umgeordnet.
- Für alle Bildpunkte $\mathbf{s}_e(x,y) = (g_0, g_1, ..., g_{N-1})^T$ von $\mathbf{S}_e$ wird die neue Grauwertkombination $\mathbf{s}_a(x,y)$ berechnet gemäß: $\mathbf{s}_a(x,y) = \mathbf{V} \cdot \mathbf{s}_e(x,y)$.
  $\mathbf{s}_a(x,y)$ ist ein Bildpunkt des hauptkomponententransformierten Bildes $\mathbf{S}_a$.

Die Histogramme von Bild 13.6 sind aus dem Rot-, Grün- und Blau-Kanal des Farbbildes 13.5-a berechnet. Darunter sind die Histogramme der drei Hauptkomponenten abgebildet. Man sieht deutlich die abnehmende Streuung der Histogramme.

Abschließend noch einige Bemerkungen zur Hauptkomponententransformation. In diesem Abschnitt wurde sie anhand eines mehrkanaligen Bildes $\mathbf{S}_e = (s_e(x,y,n), n = 0,1,...,N-1$ erläutert. Sie bringt hier eine Reduktion der Kanäle, da sie das neue Merkmalskoordinatensystem so legt, dass es möglichst redundanzfrei ist. Es ist allerdings zu beachten, dass in manchen Fällen durch diese Drehung des Merkmalskoordinatensystems für spezielle abgebildete Objekte die charakteristischen Eigenschaften verloren gehen oder verschlechtert werden können. Die Kovarianzmatrix $\mathbf{C}$ kann auch nur für Bildausschnitte von besonderem Interesse berechnet werden. In diesem Fall sind die Transformierten an die jeweiligen Bildausschnitte angepasst.

Eine andere Anwendungsmöglichkeit besteht in der Elimination redundanter Bildinformation im Ortsbereich. Hier wird dann die Kovarianzmatrix zu bestimmten Umgebungen (z.B. $8 \cdot 8$ Bildpunkte) berechnet. Allerdings ergeben sich hier sehr große Matrizen, sodass diese Art der Anwendung oft nur von theoretischer Bedeutung ist.

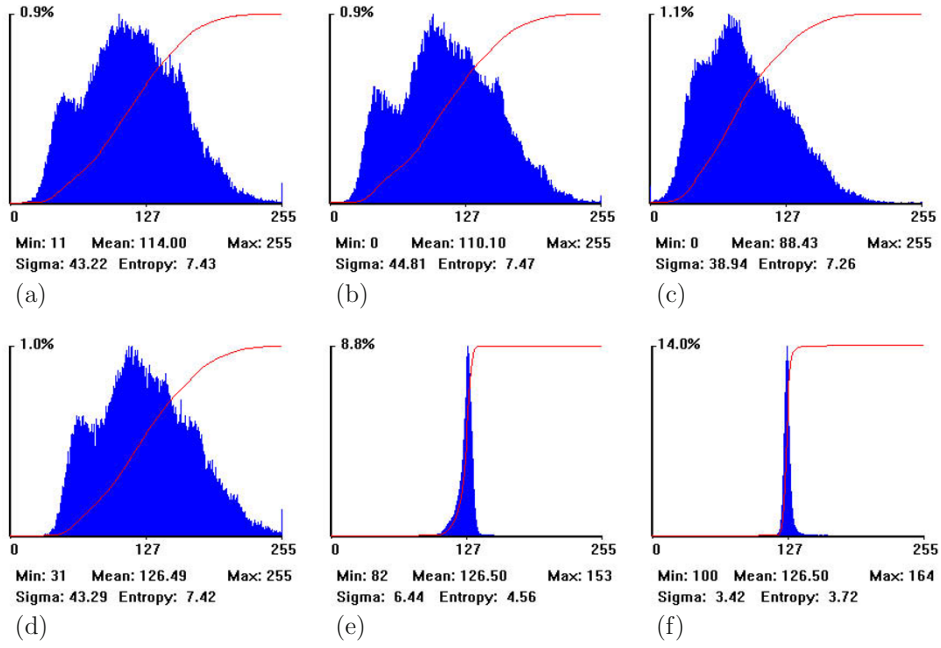


Bild 13.6: Grauwertverteilung eines mehrkanaligen Bildes (Farbbild aus Bild 13.5-a), dessen Kanäle stark korreliert sind. (a) - (c): Histogramm zum Rot- , Grün- und Blau-Kanal des Farbbildes 13.5-a. (d) - (f): Histogramme der drei Hauptkomponenten. Man sieht deutlich die abnehmende Streuung der Histogramme. (a) Histogramm zu Bild 13.5-b, (b) Histogramm zu Bild 13.5-c, (d) Histogramm zu Bild 13.5-f, (e) Histogramm zu Bild 13.5-g.

Kapitel 14

Merkmale aus Bildfolgen

14.1 Anwendungen

In diesem Kapitel werden Bildfolgen $\mathbf{S}_e = (s_e(x, y, t)), t = 0, 1, ..., T-1$, vorausgesetzt. Es kann sich dabei z.B. um eine Bildsequenz handeln, die von einer Videokamera erzeugt wird, die fest installiert ist und einen definierten Bildausschnitt liefert. Wenn es sich um Farbbildfolgen oder Multispektralbildfolgen handelt, kann die Darstellung der Szene $\mathbf{S}_e$ noch um die Kanalparameter erweitert werden: $\mathbf{S}_e = (s_e(x, y, n, t)), n = 0, 1, ..., N-1, t = 0, 1, ..., T-1$. In der folgenden Darstellung werden jedoch nur einkanalige Bildfolgen behandelt.

Bei leicht verrauschten Bildfolgen kann durch eine Mittelung entlang der Zeitachse das Rauschen abgeschwächt werden. Wenn sich in der Bildfolge etwas bewegt, ergeben sich durch die Mittelung Artefakte. Bei manchen Anwendungen, z.B. in der (Radio-)Astronomie können derartige Störungen durch ein Nachführen des Videosensors vermieden werden. Auch die Differenz zwischen einzelnen Teilbildern einer Bildfolge wird in der Praxis, z.B. in der Medizin, verwendet.

Bei medizinischen Anwendungen sind die Bildakkumulation und die Differenzbildung wichtige Verarbeitungsschritte von Bildfolgen. Die Detektion von bewegten Bildausschnitten lässt sich ebenfalls mit den Bildfolgen erzielen.

Ein wichtiger Bereich ist die Erfassung von zeitlichen Veränderungen in einer Bildfolge, die z.B. durch Bewegungen im Beobachtungsgebiet und/oder durch Bewegungen des Videosensors entstanden sind. Hier kann man z.B. zur Überwachung von Sicherheitsbereichen Bilder erzeugen, in denen abgebildet ist, wo sich etwas verändert oder bewegt hat. Ein nächster Schritt ist die Darstellung von Bewegungen und die qualitative Erfassung von Bewegungen. Anwendungsbeispiele gibt es hier aus dem Bereich der Bilddatencodierung bei den MPEG-Verfahren, bei denen Bewegungsschätzung und Bewegungskompensation zur effektiven Codierung von Bildfolgen eingesetzt werden.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_14

14.2 Akkumulation und Differenz

Ist der Videosensor fest installiert und bewegt sich im Bildausschnitt nichts, so sind die Einzelbilder der Bildfolge im Wesentlichen alle gleich, bis auf etwaige Helligkeitsveränderungen und auf das Rauschen, das durch die Aufnahmebedingungen in die Bilddaten integriert wird. Eine Mittelung der zeitlich aufeinanderfolgenden Bilder liefert dann ein Bild, in dem die Rauscheinflüsse geringer sind (Gauß'sches Rauschen mit Erwartungswert 0):

$$s_a(x,y) = \frac{1}{T}\sum_{t=0}^{T-1} s_e(x,y,t). \tag{14.1}$$

Das gemittelte Bild $\mathbf{S}_a$ ist in der Bildqualität verbessert, wobei die Mittelung in (14.1) die Schärfe des Bildes nicht beeinflusst hat.

Auch die Differenz von Teilbildern einer Bildfolge zu den Zeitpunkten t und $t+1$ lässt mehrere Anwendungen zu:

$$s_a(x,y) = s_e(x,y,t) - s_e(x,y,t+1) + c. \tag{14.2}$$

Die Bildfolge 14.1 zeigt ein Beispiel dazu. Die ersten beiden Bilder sind die Teilbilder der Bildfolge zu den Zeitpunkten t und $t+1$. Bild 14.1-c ist das Differenzbild von Bild 14.1-a und Bild 14.1-b. Zu den Werten der Differenz wurde die Konstante $c = 127$ addiert, um negative Grauwerte zu vermeiden. Man sieht deutlich den neu verdeckten und den frei werdenden Hintergrund. Da die Videokamera mit der Hand gehalten wurde, war sie nicht ganz unbewegt. Dadurch ergeben sich auch im Bildhintergrund geringe Differenzsignale. Bild 14.1-d entstand sinngemäß wie Bild 14.1-c, jedoch wurde als zweites Bild der Differenz das Bild zum Zeitpubkt $t+7$ verwendet.

In Kombination mit der zeitlichen Mittelwertbildung wird die Differenzenbildung bei medizinischen Fragestellungen häufig angewendet. So wird z.B. bei der Angiocardiografie (Untersuchung des Herzens) die folgende Versuchsanordnung gewählt:

- Für die Zeitpunkte $t_0, t_1, ..., t_k$ wird eine Bildsequenz ohne Kontrastmittel zur Reduktion des Rauschens aufsummiert:

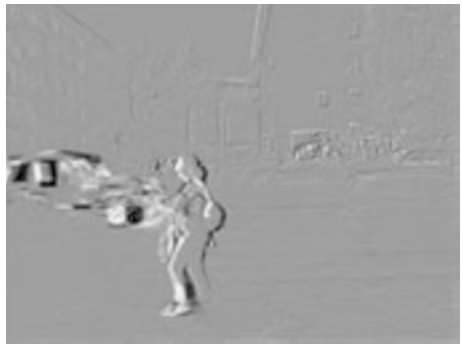
$$s_1(x,y) = \frac{1}{k+1}\sum_{t=0}^{k} s_e(x,y,t). \tag{14.3}$$

- Dann wird ein Kontrastmittel eingespritzt und für die Zeitpunkte $t_{k+1}, t_{k+2}, ..., t_{k+l}$ eine Bildsequenz mit Kontrastmittel aufsummiert:

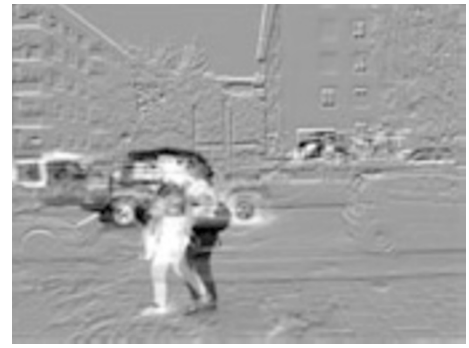
$$s_2(x,y) = \frac{1}{l-k}\sum_{t=k+1}^{l} s_e(x,y,t). \tag{14.4}$$

(a)

(b)



(c)



(d)

Bild 14.1: Differenz von Teilbildern einer Bildfolge. (a) Bild zum Zeitpunkt t. (b) Bild zum Zeitpunkt $t+1$. (c) Differenzbild von (a) und (b). Zu den Werten der Differenz wurde 127 addiert. Man sieht deutlich den neu verdeckten und den frei werdenden Hintergrund. Da die Videokamera mit der Hand gehalten wurde, war sie nicht ganz unbewegt. Dadurch ergeben sich auch im Bildhintergrund geringe Differenzsignale. (d) Wie Bild (c), jedoch wurde als zweites Bild der Differenz das Bild zum Zeitpunkt $t + 7$ verwendet.

- Auch diese Mittelung wird zur Reduktion des Rauschens durchgeführt. Das Differenzbild

$$s_a(x,y) = s_1(x,y) - s_2(x,y) + c \tag{14.5}$$

zeigt dann in der Regel die zu untersuchenden Bildbereiche deutlich (evtl. nach einer abschließenden Kontrastanreicherung), während störender Hintergrund ausgeblendet ist. Allerdings erzeugen bei dieser Vorgehensweise Bewegungen im Untersuchungsgebiet Artefakte.

Eine weitere medizinische Anwendung der Verarbeitung von Bildfolgen ist die bildliche Darstellung des zeitlichen Verlaufs der Ausbreitung eines Kontrastmittels. Voraussetzung ist hier wieder ein unbewegtes Beobachtungsgebiet. Mit der Bildakkumulation wird vor dem Einspritzen des Kontrastmittels begonnen und während der Eingabe des Kontrastmittels und eine bestimmte Zeit anschließend fortgesetzt. Zu bestimmten Zeitpunkten können Zwischenergebnisse abgespeichert werden, sodass schließlich eine Bildfolge mit unterschiedlicher Akkumulationsdauer vorliegt. Wird diese Bildfolge mit einer geeigneten Pseudofarbdarstellung ausgegeben, so ist die unterschiedliche Färbung ein Hinweis auf die zeitliche Ankunft und die Konzentration des Kontrastmittels an bestimmten Stellen.

14.3 Merkmal: Bewegung

Ein weiterer wichtiger Problemkreis bei der Verarbeitung von Bildfolgen ist die *Sichtbarmachung von Bewegungen*, die *Detektion von Bewegungen* und u.U. die *Verfolgung von bewegten Objekten*. Neben militärischen gibt es hier auch viele zivile Anwendungen, etwa die Verfolgung von Werkstücken auf einem Fließband im Rahmen einer industriellen Handhabungsmaschine.

Eine Möglichkeit Bewegungen sichtbar zu machen besteht in der Akkumulation über mehrere Einzelbilder der Bildfolge. In Bild 14.2 wurde z.B. über acht Teilbilder der Bildfolge von Bild 14.1 akkumuliert.

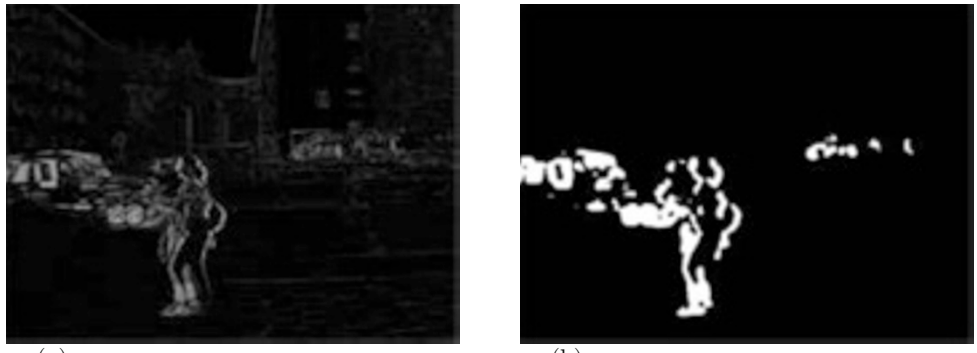
Ein anderer einfacher Lösungsansatz bei Bildfolgen mit einem festen Bildausschnitt, in dem sich ein Objekt bewegt, ist die Berechnung des Differenzbetrags von aufeinanderfolgenden Bildern, etwa zu den Zeitpunkten t und $t+1$, gemäß:

$$s_1(x,y) = |s_e(x,y,t) - s_e(x,y,t+1)|. \tag{14.6}$$

Anschließend kann mit einem passenden Schwellwert z noch eine Binarisierung durchgeführt werden:

$$s_a(x,y) = \begin{cases} 0, & s_1(x,y) < z \\ 1, & \text{sonst.} \end{cases} \tag{14.7}$$

Bild 14.2: Akkumulation über acht Teilbilder der Bildfolge von Bild 14.1.



(a) (b)

Bild 14.3: (a) Differenzbetrag der beiden Teilbilder von Beispiel 14.1. (b) Binarisierung des Differenzbetrags, auf den vor der Schwellwertbildung ein Unschärfeoperator angewendet wurde. Die Bildpunkte mit dem Grauwert 1 sind weiss dargestellt.

In $\mathbf{S}_a$ sind dann Bildbereiche, die von der Bewegung betroffen waren, mit 1 codiert. Damit kann eine Entscheidung getroffen werden, ob sich etwas bewegt hat oder nicht (Bild 14.3).

Die bis jetzt vorgestellten Verfahren zur Verarbeitung von ,,Bewegung" sind nur dazu geeignet, Bewegungen oder bewegte Ausschnitte darzustellen. Will man ,,Bewegung" als Merkmal für eine spätere Segmentierung, so muss die Bewegung zahlenmäßig erfasst werden. Man muss somit einem Bildpunkt ein Maß für seine Bewegung zuordnen. Dabei kann man nicht bildpunktweise vorgehen, sondern muss Umgebungen von Bildpunkten (Blöcke) im (x, y)-Ortsbereich betrachten. Wie groß die Blöcke sind, hängt von der gewünschten Genauigkeit ab. Zusätzlich müssen auch Informationen entlang der Zeitachse t ausgewertet werden.

Bei einem Bewegungsschätzer wird z.B. eine Kombination aus Differenzen entlang der Zeitachse t und im Ortsbereich der (x, y)-Koordinaten verwendet:

$$
\begin{aligned}
&diff_t(x,y) = s_e(x,y,t) - s_e(x,y,t+1). \\
&td = \sum_x \sum_y |diff_t|. \\
&diff_o(x,y) = s_e(x,y,t) - s_e(x,y-1,t). \\
&od = \sum_x \sum_y |diff_o|. \\
&v = \frac{td}{od}
\end{aligned}
\tag{14.8}
$$

Die Summation wird über die Bildpunkte des gewählten Bildausschnitts durchgeführt. Der Wert td ist ungefähr proportional zur Geschwindigkeit der Bewegung in diesem Aus-

schnitt. Allerdings ist der Wert noch von der Größe des Ausschnitts abhängig. Aus diesem Grund wird eine Normalisierung durchgeführt und der Wert *od* berechnet. Der Wert *od* hängt von der Größe des Ausschnitts ab, ist aber unabhängig von der darin auftretenden Bewegung. Als Maß für die Geschwindigkeit wird schließlich der Wert v verwendet. Praktische Erfahrungen haben gezeigt, dass v ein gutes Maß ist, wenn die Objektgeschwindigkeit zwischen dem Bild zum Zeitpunkt $t-1$ und dem Bild zum Zeitpunkt t nicht größer als drei Bildpunkte ist.

Ein Beispiel zu diesem Bewegungsmaß zeigt die Bildfolge 14.4. In den beiden oberen Bildern, die jeweils dasselbe Bild zum Zeitpunkt t darstellen, sind zwei unterschiedliche Bildausschnitte markiert. Für diese Ausschnitte wurde das obige Bewegungsmaß berechnet. Die beiden mittleren Bilder sind die Differenzbilder (+127) zwischen den Zeitpunkten t und $t+1$. Man sieht deutlich die Bewegungen. Für den linken Ausschnitt ergibt sich $v = 0.05$ und für den rechten, in dem sich ein Objekt bewegt hat, $v = 1.39$. Die beiden unteren Bilder sind die Differenzbilder (+127) zwischen den Zeitpunkten t und $t+2$. Hier ergibt sich sinngemäß für den Ausschnitt im linken Bild $v = 0.05$ und für den Ausschnitt im rechten Bild $v = 2.08$. Der Vergleich zeigt, dass sich bei größeren Bewegungen auch größere Werte berechnen. Man könnte nun die Werte von v geeignet skalieren und in das Grauwertintervall G abbilden. Das ergibt ein Bild, in dem die Werte der Bildpunkte Maßzahlen für die Bewegung sind.

14.4 Differentielle Ermittelung von Verschiebungsvektoren

Ein aufwändiger Ansatz zur Erfassung von Bewegungen ist die Ermittlung von *Verschiebungsvektorfeldern*. Der Ansatz beschränkt sich zunächst auf die Erkennung einer reinen Translation parallel zur Bildebene. Genauer gesagt, ist es nur möglich aus einer Bildfolge ohne zusätzliches Wissen die Verschiebung der Grauwertstrukturen abzuschätzen und als Näherung an die Verschiebungsrate anzunehmen. Diese scheinbare momentane Geschwindigkeit einer Grauwertstruktur wird als *optischer Fluß* bezeichnet.

Er kann aber, wenn auch eingeschränkt, auf andere Bewegungssituationen angewendet werden. Als Zielsetzung sind die Verschiebungsvektoren von zwei aufeinanderfolgenden Bildern (Zeitpunkte t und $t+1$) zu bestimmen. Eine vorausgesetzte Annahme ist, dass sich Bildausschnitte gleichmäßig verschieben und somit alle Bildpunkte des Ausschnitts denselben Verschiebungsvektor besitzen. Das trifft bei natürlichen Bildfolgen nicht zu, sodass daraus zum Teil auch die Fehler des Verfahrens zu erklären sind. Trotzdem ist die Berechnung von Verschiebungsvektorfeldern eine sehr effektive Methode, um von Bewegungen sowohl die Richtung als auch den Betrag der Bewegung näherungsweise zu ermitteln. Zur Berechnung der Verschiebungsvektoren werden im Folgenden zwei Verfahren vorgestellt.

In diesem Abschnitt wird die differentielle Methode zur Ermittlung der Verschiebungsvektoren beschrieben. Diese Methode wurde erstmals von [Horn81] vorgestellt.

Zur Berechnung wird zunächst eine Differenzfunktion $d(x, y)$ gebildet:

$$d(x,y) = s(x,y,t+1) - s(x,y,t) = s(x+\Delta x, y+\Delta y, t) - s(x,y,t). \tag{14.9}$$

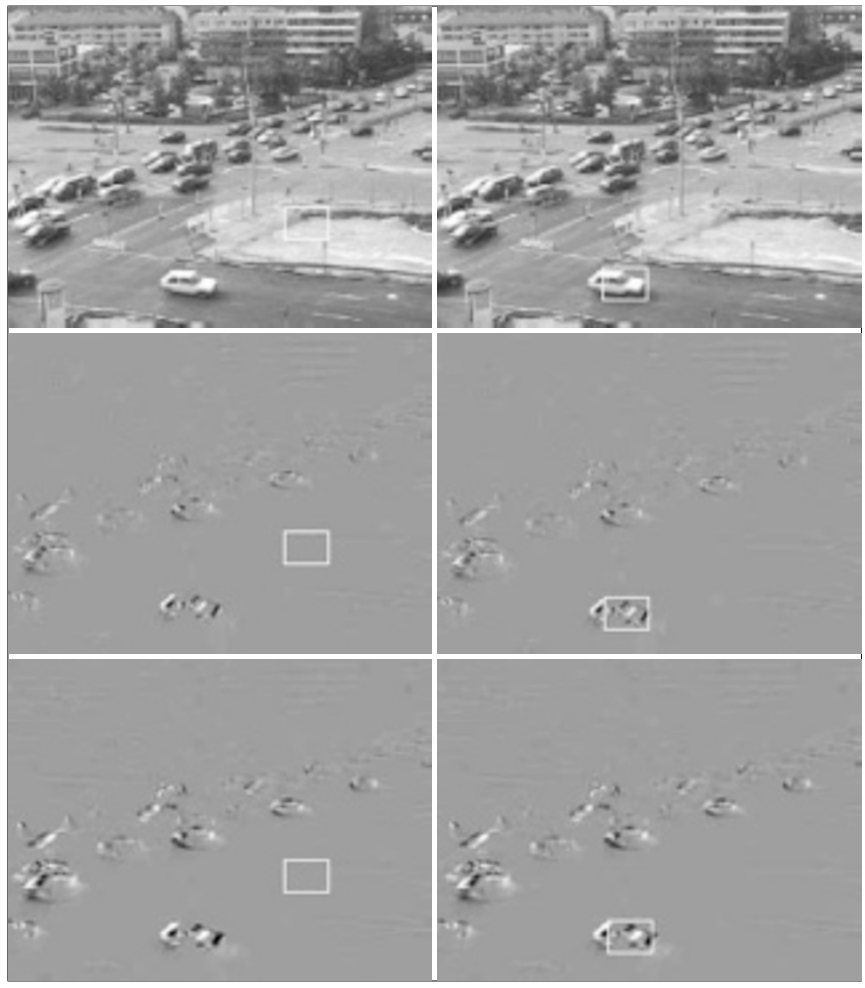


Bild 14.4: Beispiel für ein Maß zur Bewegungsschätzung. In den beiden oberen Bildern sind zwei unterschiedliche Bildausschnitte markiert. Die beiden mittleren Bilder sind die Differenzbilder (+127) zwischen den Zeitpunkten t und $t+1$. Für den linken Ausschnitt ergibt sich $v = 0.05$ und für den rechten, in dem sich ein Objekt bewegt hat, $v = 1.39$. Die beiden unteren Bilder sind die Differenzbilder (+127) zwischen den Zeitpunkten t und $t+2$. Hier ergibt sich sinngemäß $v = 0.05$ und $v = 2.08$.

Diese Umformung ist möglich, da gilt $s(x,y,t+1) = s(x+\Delta x, y+\Delta y, t)$, wenn $(\Delta x, \Delta y)$ der Verschiebungsvektor des bewegten Objektes ist, dessen Helligkeit sich bei der Verschiebung nicht geändert hat. Eine Taylorreihenentwicklung von (14.9) gestattet die weitere Umformung:

$$\begin{aligned} d(x,y) &= s(x,y,t) + \Delta x \frac{\partial s(x,y,t)}{\partial x} + \Delta y \frac{\partial s(x,y,t)}{\partial y} - s(x,y,t) + E = \\ &= \Delta x \frac{\partial s(x,y,t)}{\partial x} + \Delta y \frac{\partial s(x,y,t)}{\partial y} + E. \end{aligned} \tag{14.10}$$

Dabei sind in E Glieder der Taylorreihenentwicklung mit höherer Ordnung zusammengefasst, die weggelassen werden. Es wird nun angenommen, dass für einen Bildausschnitt mit $m \cdot m$ Bildpunkten die Verschiebungsvektoren näherungsweise gleich sind. Dann gilt für alle Bildpunkte in diesem Bereich die folgende Beziehung:

$$\begin{pmatrix} \frac{\partial s(x-k,y-l,t)}{\partial x} & \frac{\partial s(x-k,y-l,t)}{\partial y} \\ \dots & \dots \\ \frac{\partial s(x+k,y+l,t)}{\partial x} & \frac{\partial s(x+k,y+l,t)}{\partial y} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} d(x-k,y-l) \\ \dots \\ d(x+k,y+l) \end{pmatrix}. \tag{14.11}$$

Die Indizes k und l laufen dabei über die Bildpunktpositionen des jeweiligen Ausschnittes. Aus (14.11) können jetzt die Verschiebungskomponenten Δx und Δy für den entsprechenden Bildausschnitt mit Hilfe der Ausgleichsrechnung näherungsweise bestimmt werden und stehen dann z.B. als Eingangsgrößen für eine Nachführelektronik zur Verfügung. Da die Berechnung für jedes Fenster getrennt erfolgt, können mit dieser Methode auch sich unterschiedlich bewegende Objekte erkannt werden, sofern sie in verschiedenen Fenstern liegen. Durch den gewählten Ansatz wird nur eine lineare Variation des Grauwertes mit den Bildpunktkoordinaten zur Berechnung einer Verschiebung berücksichtigt. Dadurch entstehen Ungenauigkeiten an Kanten, die z.B. dann zu falschen Verschiebungsvektoren führen, wenn die Übergangszone zwischen dem Objektinneren und dem Hintergrund schmaler ist als die beobachtete Verschiebung des Objektes (siehe auch [Nage92]).

Die Bildfolge 14.5 zeigt ein Beispiel. Die Bilder 14.5-a und 14.5-b sind zwei aufeinander folgende Bilder einer Bildfolge, die aus dem Frontfenster eines nach vorne fahrenden Autos aufgenommen wurde. Die Bilder 14.5-c und 14.5-d sind die Δx- und Δy- Komponenten der Verschiebungsvektoren als Grauwerte codiert, die für $15 \cdot 15$ Bildpunkte berechnet wurden. Zur besseren Beurteilung wurden in den Bildern 14.6-a und 14.6-b die Verschiebungsvektoren grafisch eingezeichnet. Der weiße Punkt markiert den Fußpunkt eines Verschiebungsvektors, die schwarzen Pixel zeigen die Länge und die Richtung.

14.5 Ermittelung von Verschiebungsvektoren mit Blockmatching

Bei den meisten praktischen Anwendungen (z.B. dem MPEG-Verfahren) werden die Verschiebungsvektoren mit der Methode des *Blockmatching* berechnet [Watk01]. Dazu wird

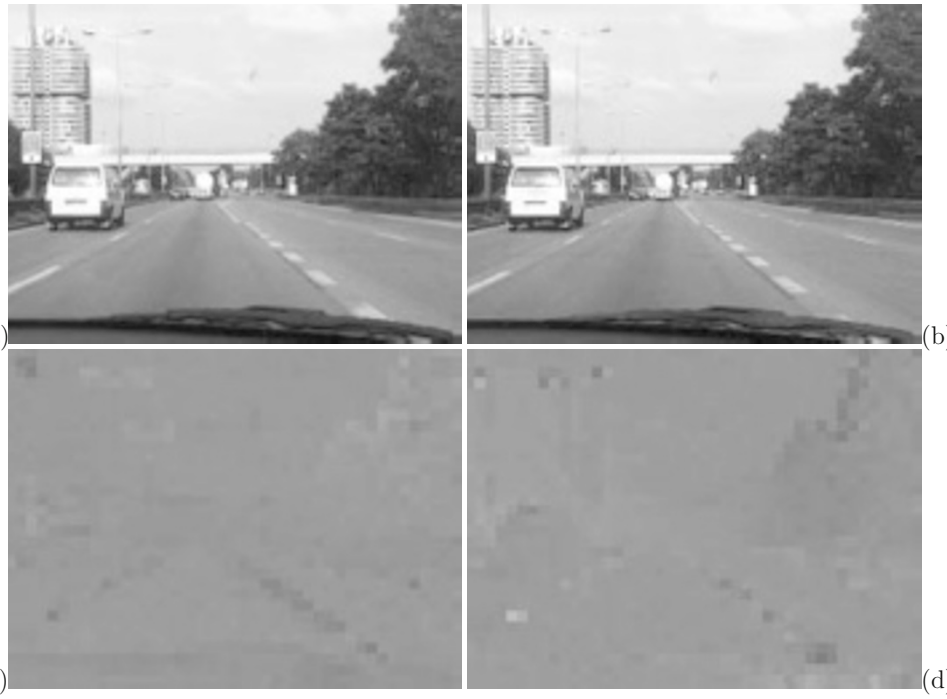


Bild 14.5: Berechnung von Verschiebungsvektorfelderen mit der differentiellen Methode. (a) und (b): Bilder zu den Zeitpunkten t und $t+1$ einer Bildfolge, die aus dem Frontfenster eines nach vorne fahrenden Autos aufgenommen wurde. (c) und (d): Δx- und Δy-Komponenten der Verschiebungsvektoren als Grauwerte codiert, die für $15 \cdot 15$ Bildpunkte berechnet wurden.

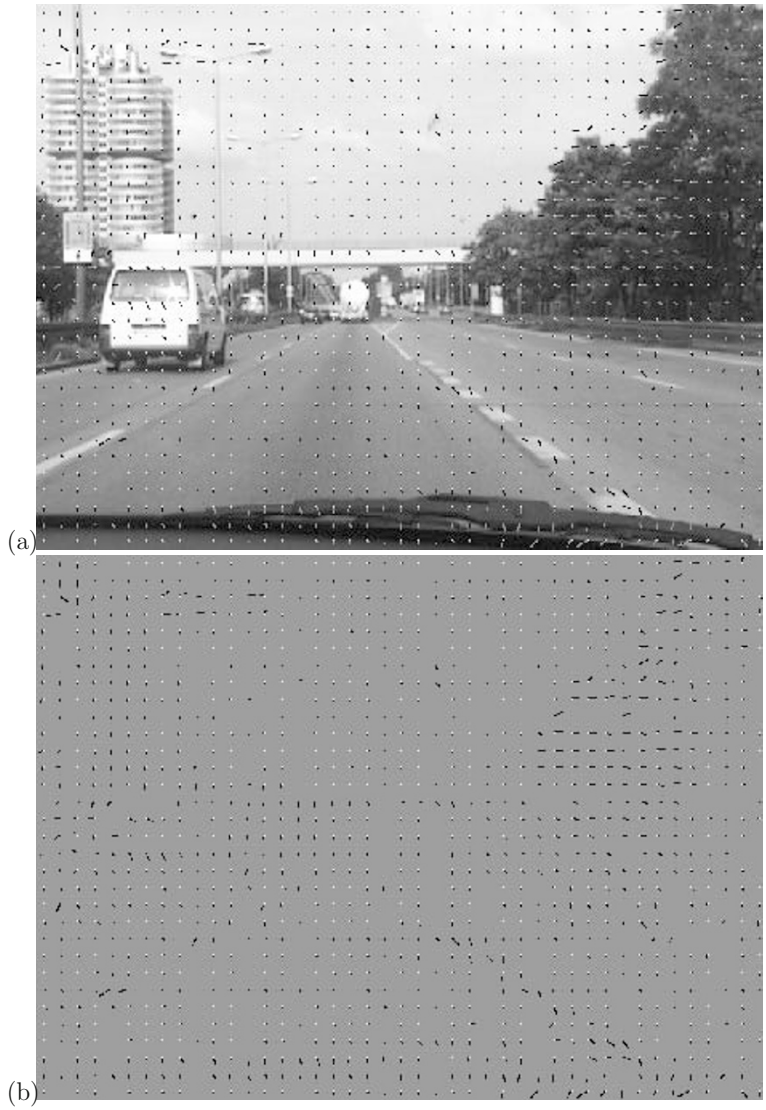


Bild 14.6: Berechnung von Verschiebungsvektorfeldern mit der differentiellen Methode. (a) Die Verschiebungsvektoren wurden hier in das erste Bild eingeblendet. Der weiße Punkt markiert den Fußpunkt eines Verschiebungsvektors, die schwarzen Pixel zeigen die Länge und die Richtung. (b) Hier sind nur die Verschiebungsvektoren gezeigt.

wie folgt vorgegangen:

- Die Verschiebungsvektoren werden zu den Bildern zum Zeitpunkt t (Frame t) und $t+1$ der Bildfolge berechnet.
- Das Bild zum Zeitpunkt t wird blockweise verarbeitet. Es wird versucht, den Bildausschnitt $block(m,n)$ des Frames t im Frame $t+1$ zu lokalisieren. Die Parameter m und n laufen dabei über die Größe p Zeilen und q Spalten der Umgebung (des Blocks, z.B. $p \cdot q = 8 \cdot 8$ Bildpunkte).
- Im Frame $t+1$ wird dazu in einer $umgebung(m,n)$ der $block(m,n)$ verschoben und berechnet, in welche Position er am besten passt. Die $umgebung(m,n)$ hat eine Größe von $(p+2c) \cdot (q+2c)$ Bildpunkten, wobei der Parameter c dabei eine Konstante ist, die die maximal erlaubte Verschiebung festlegt.
- Als Matching-Maß werden verwendet:

$$d_1(i,j) = \frac{1}{pq} \sum_{m=0}^{p-1} \sum_{n=0}^{q-1} \Big(block(m,n) - umgebung(m+i, n+j) \Big)^2 \tag{14.12}$$

  oder

$$d_2(i,j) = \frac{1}{pq} \sum_{m=0}^{p-1} \sum_{n=0}^{q-1} \Big| block(m,n) - umgebung(m+i, n+j) \Big| \tag{14.13}$$

  Die Parameter i und j durchlaufen dabei das Intervall $-c \leq i,j \leq +c$.
- Als passende Position wird die Position (i_{min}, j_{min}) verwendet, für die $d(i_{min}, j_{min})$ minimal ist. Der $block(m,n)$ im Frame t ist somit im Frame $t+1$ um i_{min} Zeilen und j_{min} Spalten verschoben.

Wenn die geschilderte Vorgehensweise verwendet wird, sind bei der Suche $(2c+1)^2$ Positionen zu inspizieren. Man bezeichnet sie als *full search*. Hier ist sichergestellt, dass man ein unter den gegebenen Umständen optimales Ergebnis erzielt. Allerdings muss für jede Position das Matching-Maß (14.12) oder (14.13) berechnet werden, was bei engen Zeitvorgaben zu Problemen führen kann. Aus diesem Grund verwendet man hier auch Verfahren, bei denen zunächst nur an ausgewählten Positionen i,j das Matching-Maß berechnet wird. Dazu wird das Minimum bestimmt. Dann wird um das gefundene Minimum mit kleineren Abständen der Positionen i,j wieder das Matching-Maß berechnet und das Minimum bestimmt. Dieser Vorgang wird mehrstufig durchgeführt. Man hat hier weniger Berechnungen als beim *full search* durchzuführen, es kann jedoch sein, dass man nur ein suboptimales Ergebnis erhält.

Abschließend ein Beispiel: Zu den beiden Teilbildern 14.5-a und 14.5-b wurden die Verschiebungsvektoren mit Blockmatching und *full search* berechnet. Das Ergebnis zeigen die

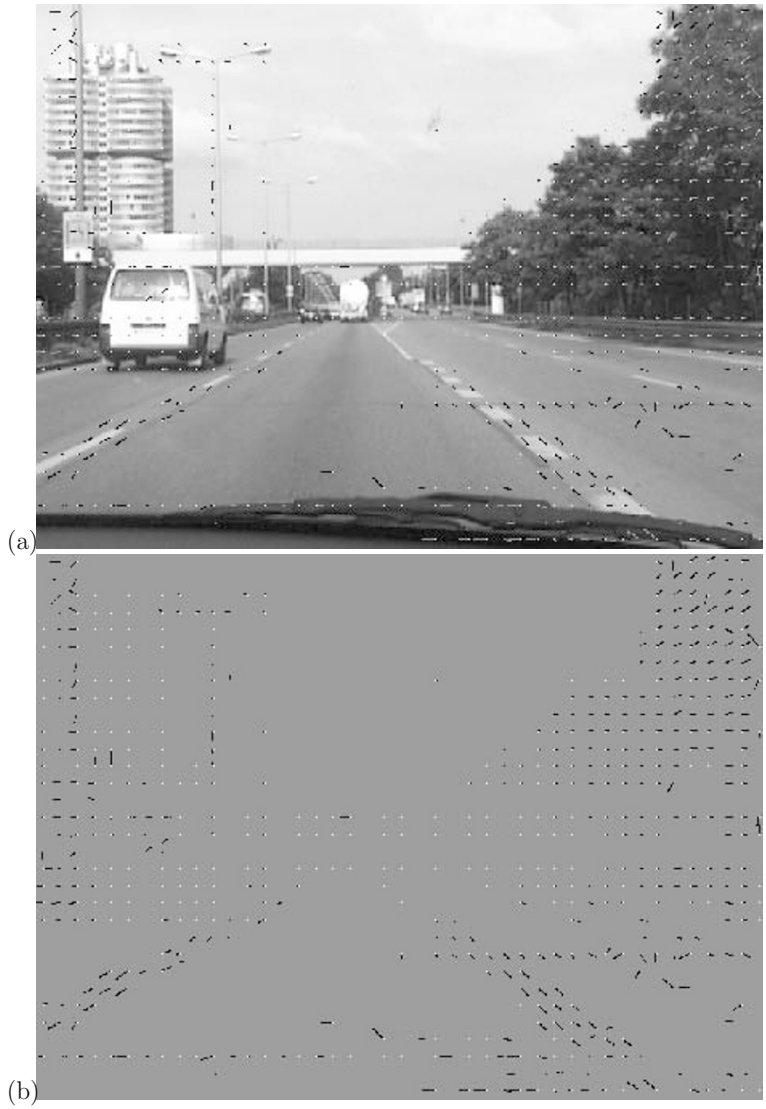


Bild 14.7: Berechnung von Verschiebungsvektorfeldern mit Blockmatching. (a) Die Verschiebungsvektoren wurden hier in das erste Bild eingeblendet. Der weiße Punkt markiert den Fußpunkt eines Verschiebungsvektors, die schwarzen Pixel zeigen die Länge und die Richtung. (b) Hier sind nur die Verschiebungsvektoren gezeigt.

Bilder 14.7-a (Frame t mit Verschiebungsvektoren) und 14.7-b (nur die Verschiebungsvektoren). Als Blockgröße wurden $15 \cdot 15$ Bildpunkte verwendet. Für den Umgebungsparameter wurde $c = 11$ verwendet. Außerdem wurden nur solche Verschiebungsvektoren dargestellt, deren Betrag über einem Schwellwert von 5 liegt. Man sieht recht gut, dass der Sachverhalt beim Blick durch die Frontscheibe eines nach vorne fahrenden Autos auf einer Stadtautobahn richtig wiedergegeben wird. Allerdings haben sich auch einige Fehler eingeschlichen, die sich in der Praxis nie vermeiden lassen.

Die Berechnung von Verschiebungsvektoren zur Bewegungsschätzung wird in der Bilddatencodierung verwendet: Mit Hilfe der Verschiebungsvektoren wird der Frame $t + 1$ aus dem Frame t durch Verschieben von Blöcken angenähert. Beim Codieren kann das Differenzbild zum originalen Frame $t + 1$ berechnet werden. Das Differenzbild kann, bei guter Bewegungsschätzung und weiteren Verarbeitungsschritten, effektiv entropiecodiert werden (z.B. Huffmancode). Beim Decodieren ist es möglich, mit dem Frame t, den Verschiebungsvektoren und dem Differenzbild den Frame $t + 1$ fehlerfrei zu rekonstruieren. Techniken dieser Art werden, mit verschiedenen Modifikationen, bei den MPEG-Codierungsverfahren für Videodaten eingesetzt.

Kapitel 15

Merkmale aus Umgebungen: Texturen

15.1 Anwendungen

Die bis jetzt betrachteten Merkmale wurden für jeden Bildpunkt berechnet, ohne dabei seine Umgebung im Ortsbereich zu berücksichtigen. Häufig ist aber das kennzeichnende Merkmal, das die zu untersuchenden Objekte unterscheidet, eine mehr oder weniger regelmäßige, unterschiedliche Struktur, die als *Textur* bezeichnet wird. Eine exakte Definition des Begriffs „Textur“ ist schwer zu geben, auch wird der Begriff bei verschiedenen Autoren etwas unterschiedlich gebraucht: Manche verwenden den Begriff „Textur“ nur dann, wenn eine gewisse Regelmäßigkeit in der Struktur zu erkennen ist. In der vorliegenden Betrachtung soll „Textur“ gleichbedeutend mit „Oberflächenstruktur“ verwendet werden. Es sind dann künstliche oder natürliche regelmäßige Strukturen, aber auch künstliche oder natürliche unregelmäßige Strukturen gemeint.

15.2 Grundlagen zu Texturmerkmalen

Eine Eigenschaft haben alle Texturen gemeinsam: Wenn man Merkmale für die Textur (*Texturmerkmale*) berechnen will, so kann man sich nicht auf einzelne Bildpunkte beschränken, sondern muss Umgebungen von Bildpunkten betrachten. Man will erreichen, dass die berechneten Texturmerkmale für die verschiedenen Texturen im (Textur-) Merkmalsraum kompakte Bereiche ergeben, die anschließend mit Segmentierungsverfahren getrennt werden können. Soll z.B. für eine bestimmte Textur ein sinnvolles Merkmal berechnet werden, so müssen die Merkmalswerte auf der reellen Zahlenachse (oder in der Grauwertmenge G) in einem eng begrenzten Intervall liegen. Im Idealfall ergibt das Texturmerkmal, abgebildet in die Grauwertmenge, einen homogenen Bereich. Dieser Bereich kann dann mit einem einfachen Schwellwertverfahren segmentiert werden.

Man sieht hier auch den fließenden Übergang zwischen der rein bildpunktorientierten und der umgebungsorientierten Vorgehensweise: Nachdem z.B. ein Texturmerkmal in die

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_15

Grauwertmenge abgebildet ist, ist das entstehende Bild ein Grauwertbild, das mit allen Verfahren der Grauwertbildverarbeitung bearbeitet werden kann. Wird z.B. darauf ein Kantenextraktionsverfahren angewendet, so können die im Kantenbild auftretenden Kanten als *Texturkanten* interpretiert werden. Werden mehrere Texturmerkmale berechnet, so entsteht eine N-dimensionale Szene $\mathbf{S}$, deren Kanäle die einzelnen Texturmerkmale sind. Ein Bildpunkt in der Position (x, y) ist dann ein N-dimensionaler Merkmalsvektor $\mathbf{g} = \mathbf{s}(x, y)$, dessen Komponenten Maßzahlen für das jeweilige Texturmaß an der Stelle (x, y) des Originalbildes sind. Für nachfolgende Segmentierungsverfahren, wie z.B. Klassifikatoren oder neuronale Netze, spielt es keine Rolle, durch welche Vorverarbeitungsschritte die Merkmalsvektoren zustande gekommen sind. Es ist offensichtlich, dass die N-kanalige Merkmalsszene $\mathbf{S}$ auch aus einer Kombination von bildpunktorientierten und umgebungsorientierten Merkmalen aufgebaut sein kann.

Wenn eine Textur eine bestimmte Regelmäßigkeit aufweist, so muss man eine *Grundtexturfläche* finden, die die charakteristischen Eigenschaften der Textur enthält. Die Textur ist dann so beschaffen, dass sich bei vielfacher Wiederholung der Grundtexturfläche die Textur ergibt. Je unregelmäßiger eine Textur ist, desto größer wird die Grundtexturfläche und desto schwieriger ist sie zu bestimmen. Im Extremfall ist die Grundtexturfläche identisch mit der gesamten Oberfläche.

Häufig ist man hier mit zwei entgegengesetzten Sachverhalten konfrontiert: Einerseits soll die Grundtexturfläche so groß sein, dass sie die wesentlichen Struktureigenschaften erfasst, andererseits soll die Grundtexturfläche so klein wie möglich sein. Gründe dafür sind:

- Große Grundtexturflächen verursachen lange Rechenzeiten.
- In einem Bild werden in der Regel unterschiedliche Texturen mit unterschiedlich großen Grundtexturflächen auftreten. Wählt man Umgebungen, für die Texturmerkmale berechnet werden, so müssen sich diese zwangsläufig an den größeren Grundtexturflächen orientieren. In Übergangsbereichen von verschiedenen Texturen wird man dann Mischtexturbereiche erhalten, in denen die gewählten Texturmerkmale falsche Werte liefern.

In [Abma94] wird auch der Gesichtspunkt erwähnt, dass sich eine Textur aus Einzelobjekten zusammensetzt. Demnach kann man versuchen, eine Textur durch die Beschreibung der Einzelobjekte zu charakterisieren. Dazu bieten sich Eigenschaften der Einzelobjekte an, wie die Form, die Größe (und damit die Anzahl), die Grauwertverteilung oder die Anordnung.

Da in der Praxis die Grundtexturflächen nicht oder nur schwer zu ermitteln sind, werden als Kompromiss für jeden Bildpunkt eines Bildes in der Position (x, y) aus einer (u, v)–Umgebung $U = U_{(u,v)}$ (*Texturfenster*) die Werte für die unterschiedlichen Texturparameter berechnet. Dabei werden meistens $m \cdot m$–Umgebungen, mit $m = 3, 5, 7, ...$ verwendet. Es ist klar, dass diese Umgebungen nur eine vage Annäherung an die Grundtexturflächen sind.

Welche Merkmale kann man nun als Texturmerkmale verwenden? In den folgenden Abschnitten werden einige einfache Beispiele zur Beantwortung dieser Frage gegeben.

15.3 Streuung (Varianz)

Die Schätzung der *Streuung* im Texturfenster ist eine erste Möglichkeit, eine Textur zu beschreiben. Intuitiv ist dies auch einleuchtend: Bei einer homogenen Fläche ergibt sich null, je inhomogener die Fläche ist, desto größer werden die Werte. Die Streuung wird über die mittlere quadratische Abweichung im Texturfenster geschätzt:

$$q_U(x,y) = \frac{1}{(M-1)} \sum_{(i,j)\in U} \Big(s(x-i,y-j) - m_U\Big)^2, \tag{15.1}$$

wobei M die Anzahl der Bildpunkte und m_U der Mittelwert des Texturfensters U sind. Bei der Berechnung kann man auch gegen den Rand des Texturfensters die Bildpunkte von q_U geringer gewichten.

In der Bildfolge 15.1 sind Beispiele zur Streuung als Texturmaß gegeben. In Bild 15.1-a ist das Original gezeigt, das auf einem texturierten Hintergrund 16 unterschiedliche Texturen enthält. Die Bilder 15.1-b,c und d zeigen für jeden Bildpunkt die Werte q_U für Umgebungen mit $3 \cdot 3$, $5 \cdot 5$ und $7 \cdot 7$ Bildpunkten. Deutlich ist zu sehen, dass die Textur 6 bereits bei einer $3 \cdot 3$–Umgebung sehr hohe einheitliche Merkmalswerte ergibt, während sich bei den Texturen 5 und 7 einheitlich niedere Werte für q_U berechnen. Dieses Maß wäre somit ausreichend, um die Texturen 5 und 7 von der Textur 6 zu unterscheiden. Die Texturen 9, 10, 11 und 15 tendieren bei den größeren Umgebungen ebenfalls zu homogenen Merkmalswerten. Auffallend ist, dass bei einigen Texturen deutlich die Texturkanten zur Hintergrundtextur hervortreten. In der Textur 13, die sicher keine kleine Grundtexturfläche besitzt, sind Grauwertkanten innerhalb der Textur gut zu sehen. Bei den anderen Texturen sind keine markanten Eigenschaften zu beobachten.

15.4 Gradient

Der *Gradient* kann ebenfalls als Texturmerkmal verwendet werden: In homogenen Bildbereichen werden die Gradientenbeträge gering sein, während in strukturierten Bereichen hohe Gradientenbeträge auftreten. Der Gradient kann mit den in Abschnitt 5.4 angegebenen Formeln berechnet werden. Hinzu kommt hier noch ein weiteres Merkmal, das mit der Berechnung des Gradienten erfasst wird: Die Richtung des Gradienten. Wird die Richtung des Gradienten wieder als Grauwert codiert, so haben in einem entsprechenden Ausgabebild alle Bildpunkte, in deren Umgebung der Gradient dieselbe Richtung besitzt, denselben Grauwert.

Die Bilder 15.2 zeigen Beispiele hierzu. Die Gradienten wurden hier aus 3·3–Umgebungen berechnet. Bild 15.2-a zeigt die Gradientenbeträge. Es ergibt sich kein wesentlicher Unterschied zu den lokalen Streuungen im Texturfenster (Bild 15.2-b). Die Bilder 15.2-b und 15.2-c stellen die Gradientenrichtungen dar. Bei Bild 15.2-b sind deutlich die einheitlichen Richtungen der Gradienten zu sehen (z.B. bei den Texturen 1, 2, 6, 8, 13 und 15). Im Hinblick auf die Forderung, dass Texturmerkmale für die einzelnen Texturen im Merkmalsraum homogene Bereiche ergeben sollen, ist die Eigenschaft der Gradientenrichtungen

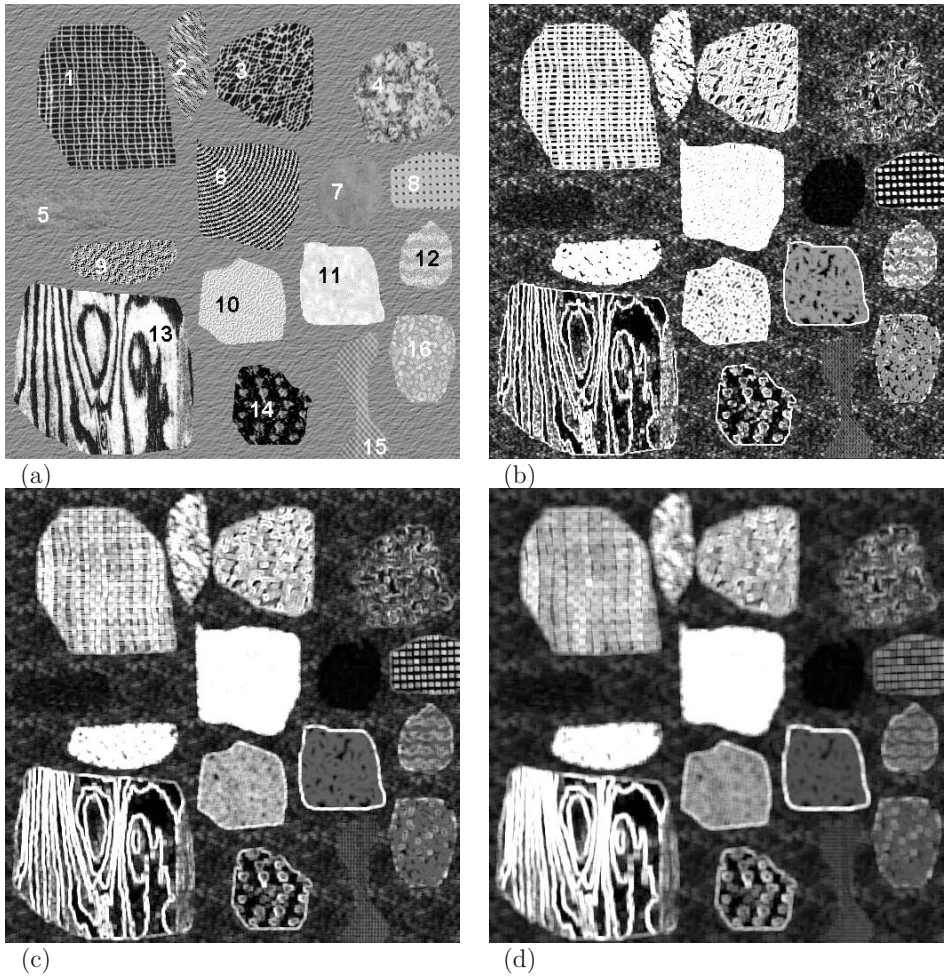


Bild 15.1: Lokale Schätzung der Streuung im Texturfenster. (a) Originalbild mit 16 Texturen vor einem texturierten Hintergrund. (b) Mittlere quadratische Abweichung q_U bei einem Texturfenster von $3 \cdot 3$ Bildpunkten. Die Texturen 5, 6 und 7 ergeben bereits ziemlich einheitliche Bereiche. (c) Mittlere quadratische Abweichung q_U bei einem Texturfenster von $5 \cdot 5$ Bildpunkten. Texturkanten zur Hintergrundtextur sind bei einigen Beispielen deutlich zu sehen. (d) Mittlere quadratische Abweichung q_U bei einem Texturfenster von $7 \cdot 7$ Bildpunkten. Auch die Texturen 9, 10, 11 und 15 tendieren jetzt zu einheitlichen Bereichen.

störend, dass bei Grauwertkanten gegenüberliegende Gradienten (180°) mit unterschiedlichen Grauwerten codiert werden. In Bild 15.2-c wurden aus diesem Grund gegenüberliegende Gradientenrichtungen zu einem Grauwert zusammengefasst. Deutlich hebt sich dann die Textur 15 heraus. In Textur 6 ist der Verlauf der Gradientenrichtungen gut zu verfolgen. Bei Textur 13 ergeben sich durch die ausgeprägte Vorzugsrichtung der Holzmaserung große homogene Bereiche.

In Bild 15.2-d wurde nur für diejenigen Gradienten die Richtung codiert, deren Betrag größer als ein vorgegebener Schwellwert ist. An den Kantenverläufen sind die unterschiedlichen Gradientenrichtungen gut zu erkennen. Der abrupte Übergang von Hellgelb nach Dunkelblau entspricht Stellen, an denen die Gradientenrichtung von 360° auf 0° springt.

15.5 Kantendichte

Aufbauend auf dem Betrag des Gradienten wird in [Erns91] die *Kantendichte* kd_U im Texturfenster U als weiteres Texturmaß vorgeschlagen. Dazu wird das Gradientenbetragsbild binarisiert und die Kantendichte wie folgt berechnet:

$$kd_U(x,y) = \frac{1}{(M-1)} \sum_{(i,j)\in U} e(x-i, y-j), \tag{15.2}$$

$$\text{mit } e(x-i,y-j) = \begin{cases} 1, & \text{Kante,} \\ 0, & \text{keine Kante.} \end{cases}$$

Das Texturfenster U kann hier durch entsprechende Wahl an bestimmte Vorzugsrichtungen der zu untersuchenden Texturen angepasst werden. Der Wert von kd_U liegt zwischen 0 und 1. Durch entsprechende Skalierung kann dieser Wert bei Bedarf in die Grauwertmenge G abgebildet werden (Bild 15.3-a).

Die Berechnung des Parameters kd_U kann auch mit anderen Vorverarbeitungsschritten kombiniert werden. Statt Gradientenbeträge z.B. mit Hilfe eines Sobeloperators zu bilden, können auch die Ergebnisse eines Laplace-Operators (Abschnitt 5.4) als Grundlage für die Berechnung von kd_U verwendet werden (Bild 15.3-b).

Eine andere Möglichkeit besteht darin, das Originalbild $\mathbf{S}_e$ durch ein Medianfilter (Kapitel 6) in ein Bild $\mathbf{S}_{med}$ zu transformieren und dann die Differenz $\mathbf{S}_{diff} = 127 + \mathbf{S}_e - \mathbf{S}_{med}$ zu berechnen. Durch diese Operation bleibt die Struktur der Texturen erhalten, während homogene Bereiche (Gleichanteil) verschwinden (Bild 15.3-c). Dieses Bild $\mathbf{S}_{diff}$ kann jetzt als Ausgangspunkt für die Berechnung des Texturparameters kd_U dienen. Allerdings ist es sinnvoll, auch die Grauwerte, die kleiner als 127 sind, als Kanten zu interpretieren. Das erreicht man z.B. durch

$$\mathbf{S}_{norm} = |\mathbf{S}_{diff} - 127| \cdot 2. \tag{15.3}$$

Bild 15.3-d zeigt das Ergebnis.

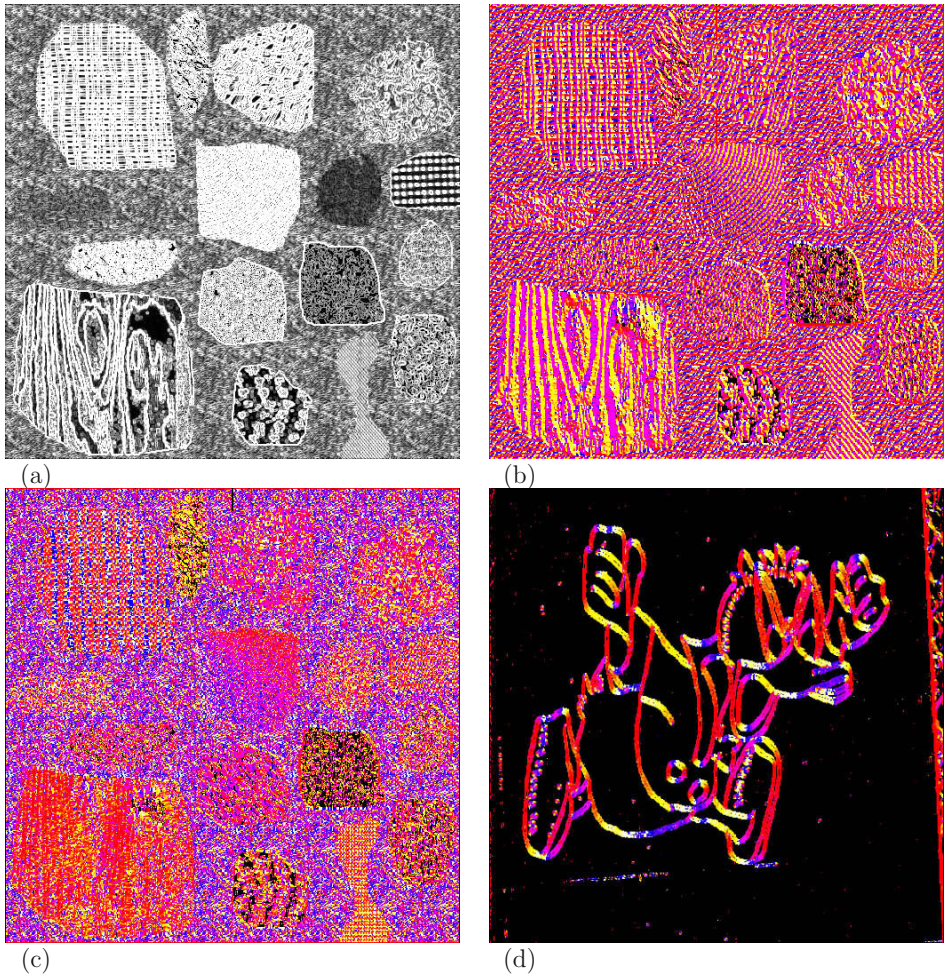


Bild 15.2: Der Gradient als Texturmerkmal. (a) Die Gradientenbeträge zum Testbild 15.1-a. Der Gradient wurde aus $3 \cdot 3$–Umgebungen berechnet. (b) Die Gradientenrichtungen, abgebildet in die Grauwertmenge G. Gleichen Farbtönen entsprechen gleiche Richtungen. (c) Hier wurden um 180° gegenüberliegende Gradientenrichtungen zu einem Grauwert zusammengefasst. (d) Gradientenrichtungen zu Bild 7.18-a. Hier wurden nur für diejenigen Bildpunkte die Gradientenrichtungen dargestellt, deren Gradientenbeträge über einem Schwellwert liegen. Man sieht deutlich, wie sich entlang der Umrandungen der Segmente die Gradientenrichtungen ändern.

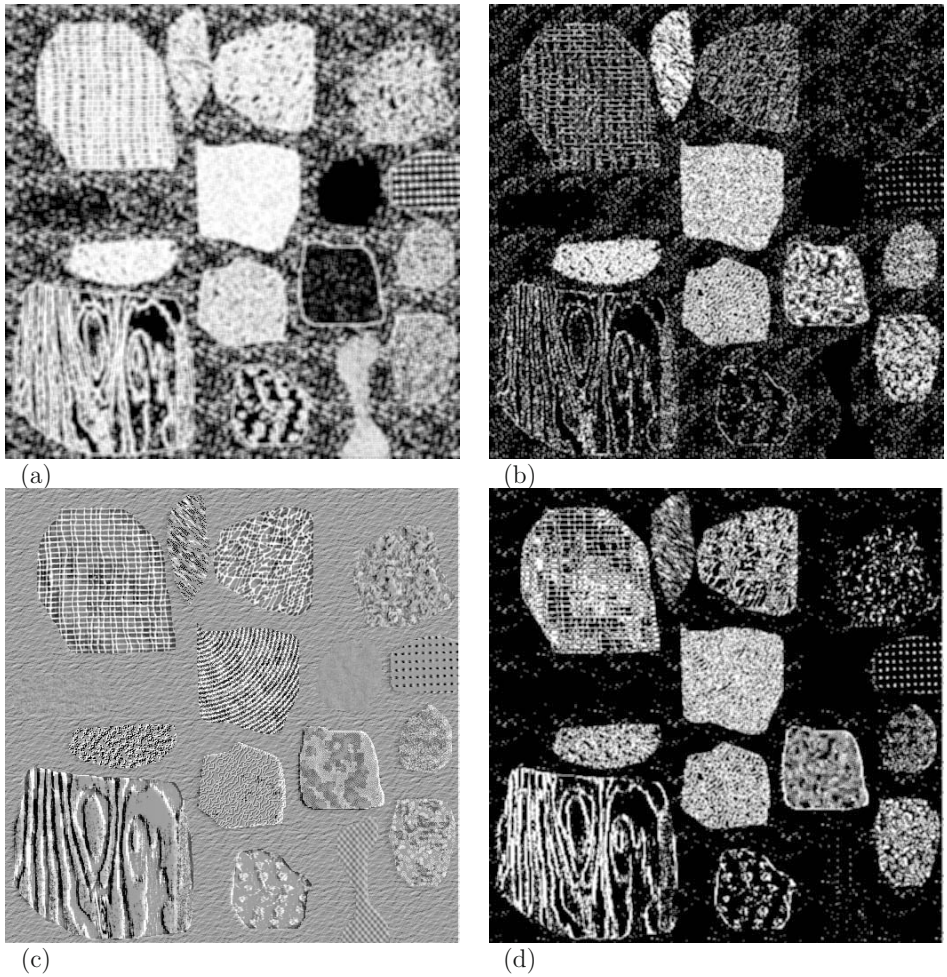


Bild 15.3: (a) Kantendichte der Gradientenbeträge nach einem Sobeloperator mit einem Texturfenster von $5 \cdot 5$ Bildpunkten. (b) Kantendichte nach einem Laplace-Operator. (c) Differenzbild: Original-Medianbild+127. (d) Kantendichte des Differenzbildes.

15.6 Autokorrelation

Die *Autokorrelation* ist ein weiteres Texturmerkmal, das vor allem bei Texturen mit einer Vorzugsrichtung erfolgversprechend ist. Dabei wird die Korrelation der Grauwerte im Texturfenster $U_{(x,y)}$ für den Bildpunkt in der Position (x, y) mit den Grauwerten in einem um einen Vektor $(\Delta x, \Delta y)$ verschobenen Fenster $U' = U_{(x+\Delta x, y+\Delta y)}$ berechnet. Die Korrelation kann man mit folgender Formel berechnen:

$$r_U^{(\Delta x,\Delta y)}(x,y) = \frac{\sum\limits_{(i,j)\in U} \Big(s(x-i,y-j) - m_U\Big)\Big(s(x+\Delta x-i, y+\Delta y-j) - m_{U'}\Big)}{\sqrt{\sum\limits_{(i,j)\in U} \Big(s(x-i,y-j) - m_U\Big)^2 \sum\limits_{(i,j)\in U} \Big(s(x+\Delta x-i, y+\Delta y-j) - m_{U'}\Big)^2}}. \tag{15.4}$$

Dabei ist m_U der Mittelwert der Umgebung U und $m_{U'}$ der Mittelwert der Umgebung U'. Der Korrelationskoeffizient r_U liegt zwischen -1 und $+1$ und kann z.B. durch $255 \cdot |r_U|$ oder $127 \cdot (r_U + 1)$ in die Grauwertmenge G abgebildet werden. Die Bilder 15.4 zeigen Beispiele zur Autokorrelation. Die Bilder wurden mit unterschiedlich großen Texturfenstern und verschiedenen Verschiebungsvektoren berechnet. Durch den Vergleich mit den Originaltexturen von Bild 15.1-a sind die Ergebnisse gut zu interpretieren. In der Praxis muss man in vielen Testläufen die optimalen Parametereinstellungen für die Größe des Texturfensters und den Verschiebungsvektor finden.

15.7 Abschlussbemerkung zu den einfachen Texturmaßen

Die Aufzählung von Texturmerkmalen könnte noch lange fortgesetzt werden. So werden z.B. in [Abma94] weitere Maßzahlen vorgeschlagen, z.B.:

- Merkmale aus der *co-occurrence*-Matrix (Grauwertübergangsmatrix, Abschnitt 3.10).
- Merkmale auf der Basis des Lauflängencodes (Kapitel 23). Hier liegt der Gedanke zugrunde, dass in Bereichen mit ausgeprägter Struktur viele kurze Ketten und in homogenen Bereichen wenige lange Ketten erzeugt werden.
- Merkmale aus der Entropie (Definition in Abschnitt 3.10).
- Merkmale aus dem Frequenzspektrum.

Betrachtet man die Ergebnisse der dargestellten Beispiele, so ist zu sehen, dass nahezu überall das gewählte Texturfenster zu klein ist, um die wesentlichen Eigenschaften der Texturen zu erfassen. Erfolgversprechender sind daher Verfahren, bei denen die Wesensart

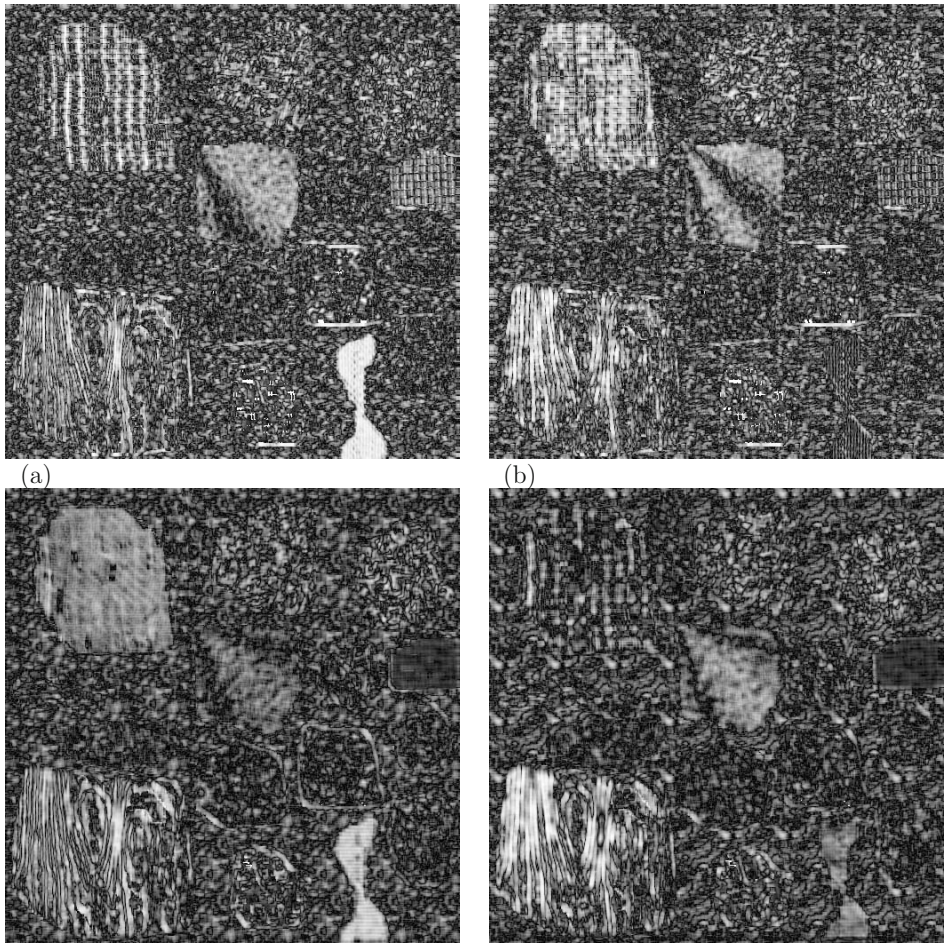


Bild 15.4: Autokorrelation $r_U^{(\Delta x,\Delta y)}$ mit unterschiedlich großen Texturfenstern und verschiedenen Verschiebungsvektoren: (a) $5 \cdot 5$–Umgebung, $(\Delta x, \Delta y) = (0, 3)$. (b) $5 \cdot 5$–Umgebung, $(\Delta x, \Delta y) = (0, -5)$. (c) $7{\cdot}7$–Umgebung, $(\Delta x, \Delta y) = (3, 3)$. (d) $7{\cdot}7$–Umgebung, $(\Delta x, \Delta y) = (5, 5)$.

von Texturen, dass in einer Textur unterschiedliche Frequenzen auftreten, berücksichtigt wird. Wenn eine Textur hierarchisch in Bereiche mit unterschiedlichen Wellenzahlindizes zerlegt wird, können die Eigenschaften der Textur besser erfasst werden.

Beispiele dazu werden im nächsten Kapitel gebracht, wo Texturparameter mittels Gauß- und Laplace-Pyramiden berechnet werden. Im übernächsten Kapitel wird eine weitere Betrachtungsweise erläutert, die auf der fraktalen Geometrie aufbaut.

Kapitel 16

Korrespondenzen in Bildern

In den vorhergehenden Kapiteln wurden Verfahren zum Extrahieren unterschiedlicher Merkmale aus Bilddaten vorgestellt. Diese Reihe von Verfahren soll nun um weitere Merkmale erweitert werden, die sich dazu eignen Korrespondenzen in Bildern zu finden. Diese Merkmale sind sogenannte *invariante Bildmerkmale*, die durch *Detektoren* und *Deskriptoren* beschrieben werden.

Es gibt unterschiedliche Anwendungen, bei denen es interessant oder erforderlich ist, Korrespondenzen in Bildern oder Bildfolgen zu finden und zu erkennen. Beispielsweise sind hier die Stereobildverarbeitung, Kamerakalibrierung, Objekterkennung, das Tracking von Objekten, das Suchen von Bildern in Bilddatenbanken, das Finden von Bildausschnitten in Bildern oder Bildfolgen und das Panoramastitching zu nennen. Bild 16.1 zeigt beispielhaft zwei Bilder, in denen invariante Bildmerkmale (SIFT-Merkmale) detektiert sind, die als Grundlage für die Berechnung der Transformation zum nahtlosen Zusammenfügen zu einem Panoramabild dienen (Quelle: [Brow07]).

Das Finden von Korrespondenzen in Bildern ist ein weites Feld. Es ist hier nicht das Ziel, das gesamte Feld darzustellen. Es werden vielmehr einige aktuelle Verfahren und deren prinzipielle Funktionsweise erklärt. Diese Verfahren sind unabhängig vom Bildinhalt und setzen keine vorhergehende Segmentierung oder Mustererkennung voraus. Sie eignen sich für eine große Anzahl von Bildern und benötigen keine weiteren Informationen über die Bilder oder den Bildinhalt.

16.1 Deskriptive Matching Verfahren

Der Begriff *Matching* bezeichnet in der Bildverarbeitung das Finden übereinstimmender Strukturen in Bildern. Je nach Anwendung kann damit die Übereinstimmung ganzer Bildbereiche oder Objekte gemeint sein. Es kann sich aber auch um die Übereinstimmung von Punkten oder Merkmalen in Bildern handeln.

Hier sollen nun eine Klasse von deskriptiven Verfahren betrachtet werden, die zu den merkmalsbasierten Verfahren gehören und meist lokale, charakteristische Bildmerkmale extrahieren. Deskriptive Matching Verfahren detektieren charakteristische Strukturen in

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_16

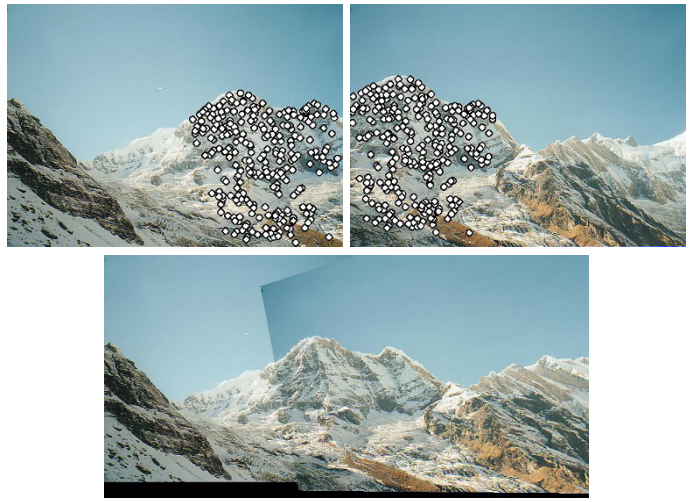


Bild 16.1: Bilder mit eingezeichneten SIFT-Merkmalen, die die Basis für die Bestimmung der Transformation zum Zusammenfügen zu einem Panoramabild bilden. Quelle: [Brow07]

Bildern, beschreiben diese mit sogenannten Deskriptoren und vergleichen dann die ermittelten charakteristischen Strukturen anhand der Deskriptoren in unterschiedlichen Bildern. Ein Deskriptor ist wie der „Fingerabdruck“ eines charakteristischen Merkmals. Entscheidend für die Qualität und Robustheit eines solchen Verfahrens ist die Einzigartigkeit der ermittelten charakteristischen Strukturen oder Bildmerkmale und die Qualität der berechneten Deskriptoren.

Deskriptive Matching Verfahren bestehen aus drei grundlegenden Schritten:

1. Detektion charakteristischer, invarianter Merkmale (*Detektor*)
2. Berechnung von Deskriptoren für die charakteristischen Merkmale (*Deskriptor*)
3. Matching der Deskriptoren (*Matcher*)

Für die einzelnen Schritte können jeweils unterschiedliche Verfahren eingesetzt werden. Charakteristische Merkmale zeichnen sich oft durch besondere Gradienten aus, diese können beispielsweise mit dem Laplace-Operator berechnet werden.

Sind charakteristische Merkmale detektiert worden, so werden diese mit Deskriptoren beschrieben. In diesem Kapitel stellen wir den SIFT-Operator (16.3.1), den SURF-Operator (16.3.2) und den HOG-Operator (16.3.3) ausführlich dar und verweisen auf andere aus diesen Operatoren weiterentwickelte Verfahren.

Deskriptoren haben vergleichsweise große Dimensionen und werden deshalb auch als hoch-dimensionale Deskriptoren bezeichnet [Miko05]. Für das Matching von Deskriptoren hat sich der RANSAC-Algorithmus bewährt, welcher effizient Deskriptoren Paare findet und robust gegenüber Ausreißern ist [Fisc81, Hart04]. Wie ähnlich zwei Deskriptoren sind, bestimmt die Distanzfunktion. Am gebräuchlichsten ist dabei die Euklidische Distanz und die Hamming-Distanz für binäre Deskriptoren.

Für die hier vorgestellten Verfahren gibt es Implementierungen u.a. in OpenCV[1] und Matlab[2], so dass bei Bedarf mit den Verfahren experimentiert werden kann.

16.2 Detektoren

Ein Detektor (auch *Feature-* oder *Merkmalsdetektor* genannt) hat die Aufgabe, charakteristische Merkmale oder markante Punkte in einem Bild zu finden. Dafür werden zahlreiche Verfahren eingesetzt. Für alle Verfahren gilt die Anforderung, möglichst unterscheidbare charakteristische Merkmale zu ermitteln, die auch aus unterschiedlichen Perspektiven wiedererkannt werden können. Ziel ist eine Invarianz gegenüber affinen Transformationen, wie Skalierung und Rotation, sowie gegenüber Beleuchtung, Rauschen und mehr. Meist wird eine Invarianz gegenüber Rotation und Skalierung erreicht [Miko05]. Viele Verfahren betrachten Unterschiede in den Grauwerten bzw. damit einhergehende aussagekräftige Grauwertgradienten, z.B. Grauwertecken (siehe auch Kapitel 5.4). Einen guten Überblick über die verschiedenen Detektionsverfahren erhält man aus [Miko05] sowie online beispielsweise in der Dokumentation zu OpenCV[3].

Charakteristische Merkmale haben den Vorteil, dass sie keine vorhergehende Segmentierung erfordern und deshalb robust gegenüber Verdeckungen oder der Änderung von Aufnahmeparametern sind, da trotz der Verdeckungen oder Änderungen meist eine Teilmenge an Merkmalen sichtbar ist.

Während zum Beispiel FAST (siehe Kapitel 16.2.2) nur Merkmale fester Größe findet, können mit SIFT (siehe Kapitel 16.3.1), der mit verschiedenen Auflösungen des Bildes (Gauß- und Laplace-Pyramide) arbeitet, sowohl größere als auch kleinere Ecken gefunden werden.

16.2.1 Harris Eckendetektor

Der *Harris Eckendetektor* wurde 1988 von Harris & Stephens vorgestellt [Harr88]. Eine anschauliche Beschreibung des Verfahrens findet sich auch in [Burg06]. Als charakteristische Merkmale sind diejenigen Punkte geeignet, die an möglichst komplexen Grauwertübergängen liegen. Das sind Grauwertecken. Der lokale Verlauf der Grauwertfunktion

[1] https://opencv.org/

[2] https://www.mathworks.com/products/image.html

[3] https://docs.opencv.org/master/db/d27/tutorial_py_table_of_contents_feature2d.html (aufgerufen am 10.10.2019)

an dem Bildpunkt $s(x, y)$ wird durch die partiellen Ableitungen $s_x(x, y)$ und $s_y(x, y)$ beschrieben. Die Berechnung der partiellen Ableitungen ist in Kapitel 5.4 zu finden.

Bild 16.2: Harris Ecken. Quelle: [Lowe04a]

Für jeden Bildpunkt $s(x, y)$ wird die *Strukturmatrix*

$$M = \begin{pmatrix} s_x^2 & s_x s_y \\ s_x s_y & s_y^2 \end{pmatrix} = \begin{pmatrix} A & C \\ C & B \end{pmatrix} \tag{16.1}$$

bestimmt. Der Rang der Strukturmatrix gibt die Eigenschaft der Umgebung an. Liegt ein markanter Punkt vor, so ist $rang(M) = 2$. Bei einer geraden Kante ist der Rang der Strukturmatrix $rang(M) = 1$. Für eine homogene, unstrukturierte Fläche hat die Strukturmatrix $rang(M) = 0$. Anders ausgedrückt, geben die Eigenwerte von M die Kantenstärke und die Eigenvektoren die Kantenrichtung an. Die Eigenwerte $\lambda_{1,2}$ berechnen sich als

$$\begin{aligned} \lambda_{1,2} &= \frac{spur(M)}{2} \pm \sqrt{\left(\frac{spur(M)}{2}\right)^2 - \det(M)} \\ &= \frac{1}{2}\left(A + B \pm \sqrt{A^2 - 2AB + B^2 + 4C^2}\right). \end{aligned} \tag{16.2}$$

An einer Ecke treffen sich zwei starke Kanten. Beide Eigenwerte sollten also $\lambda_{1,2} > 0$ sein. Man kann deshalb davon ausgehen, dass $spur(M) > 0$ und daher auch $|\lambda_1| \geq |\lambda_2|$. Für die Differenz der Eigenwerte gilt nach (16.2)

$$\lambda_1 - \lambda_2 = 2 * \sqrt{\left(\frac{spur(M)}{2}\right)^2 - \det(M)}. \tag{16.3}$$

Diese Differenz soll an einem Eckpunkt möglichst klein werden. Aus dieser Differenz wird für den Harris Eckendetektor die Größe

$$\begin{aligned} C(x,y) &= \det(M) - \kappa * spur(M)^2 \\ &= A * B - C^2 - \kappa(A+B)^2 \end{aligned} \tag{16.4}$$

definiert. κ steuert die Empfindlichkeit des Eckendetektors. $C(x,y)$ wird auch als *Eckenstärke* bezeichnet. Ein Punkt $s(x,y)$ ist nun ein Kandidat für eine Ecke, wenn $C(x,y)$ größer als ein Schwellwert τ ist. Allerdings ist ein globaler Schwellwert für ein Bild oftmals wenig sinnvoll. Man unterteilt ein Bild vielmehr in Kacheln und bestimmt pro Kachel in Abhängigkeit eines minimalen Abstands zwischen zwei Ecken und der Anzahl der Ecken pro Kachel einen Schwellwert τ_i. Alternativ kann man einen globalen Schwellwert τ verwenden, wenn man pro Kachel eine nachgeschaltete Auswahl der Eckenkandidaten vornimmt. Man sollte vermeiden, dass eine Häufung von Ecken in stark texturierten Bereichen auftritt.

Die charakteristischen Merkmale des Harris Eckendetektors sind invariant gegenüber Translation, Rotation sowie teilweise gegenüber Helligkeitsänderungen. Es besteht keine Skalierungsinvarianz [Miko05]. Bild 16.2 zeigt mit dem Harris Eckendetektor detektierte charakteristische Punkte.

Der Algorithmus zum Harris Eckendetektor ist folgendermaßen zusammengefasst:

A16.1: Harris Eckendetektor.

Voraussetzungen und Bemerkungen:

$\diamond$ $\mathbf{S} = (s(x,y))$ ist ein Grauwertbild

Algorithmus:

(a) Bestimme s_x und s_y, z.B. mit dem Sobel-Operator (5.23)

(b) Bestimme die Strukturmatrix $M = \begin{pmatrix} A & C \\ C & B \end{pmatrix} = \begin{pmatrix} s_x^2 & s_x s_y \\ s_x s_y & s_y^2 \end{pmatrix}$

(c) Bestimme $C(x,y) = A * B - C^2 - \kappa(A+B)^2$

(d) Bestimme einen Schwellwert τ und sammle alle $s(x,y)$ mit $C(x,y) > \tau$ in einem Array von Eckenkandidaten; Sortiere das Array der Größe nach

(e) Teile das Bild in Kacheln auf; Pro Kachel:

(ea) Bestimme minimalen Abstand von Ecken und maximale Anzahl von Ecken

(eb) Gehe der Reihe nach durch das Array der Eckenkandidaten; entferne diejenigen Eckenkandidaten pro Kachel, die den Kriterien von (ea) nicht genügen

(f) Das verbleibende Array von Eckenkandidaten sind die detektierten Ecken

Ende des Algorithmus

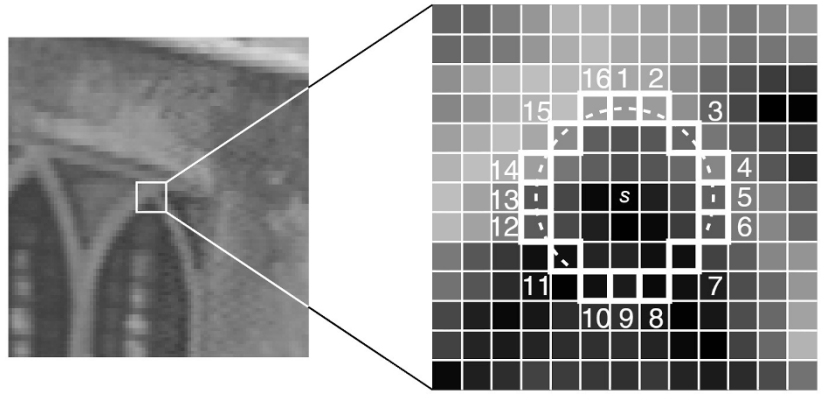


Bild 16.3: FAST Detektion von Ecken in einem Bildausschnitt: Die eingerahmten Bildpunkte im rechten Bildausschnitt werden für die FAST Detektion verwendet. Es werden 16 Bildpunkte mit jeweils einem Abstand von 3 um den Eckenkandidat s betrachtet (Kreis um den Eckenkandidaten). Wenn 12 durchgehend nummerierte Bildpunkte des Kreises um den Eckenkandidat entweder mehr als ein Schwellwert t heller oder dunkler als der Eckenkandidat sind, dann liegt eine Ecke vor. Quelle: [Rost06].

16.2.2 FAST Detektor

Der FAST Detektor (Features from Accelerated Segment Test), vorgeschlagen von Edward Rosten and Tom Drummond [Rost06], findet heuristisch Ecken in einem Bild. Das Ziel des Verfahrens ist eine Detektion, die so schnell ist, dass sie für Echtzeitanwendungen geeignet ist. Dem FAST Detektor liegt die Annahme zu Grunde, dass eine Ecke meist dann vorliegt, wenn sich ein Großteil der Grauwerte umliegender Bildpunkte von dem Grauwert der Ecke unterscheidet. Um Verwechslungen mit einer Kante auszuschließen, wird ein zusammenhängender Kreis von Bildpunkten um eine Eckenhypothese betrachtet. Der Test, ob ein Bildpunkt $s(x, y)$ eine Ecke ist, untersucht einen Pixelkreis mit Radius 3, angenähert durch 16 Bildpunkte (siehe Abbildung 16.3). Werden entlang des Kreises $n = 12$ Bildpunkte in Folge als heller (Grauwert der Eckenhypothese s plus einem Schwellwert t) oder $n = 12$ Bildpunkte in Folge als dunkler (Grauwert der Eckenhypothese s minus einem Schwellwert t) gefunden, wird $s(x, y)$ als Ecke, auch Keypoint genannt, detektiert.

Ein schneller Test, um viele Eckenkandidaten auszuschließen, besteht darin, sich nur jeden vierten Bildpunkt anzusehen. Sollten drei davon nicht heller bzw. dunkler sein, sind auch keine 12 Bildpunkte in Folge heller bzw. dunkler. Diese Idee ist grundlegend und namensgebend für den FAST Detektor, da dadurch schnell viele Bildpunkte als Ecken in einem Bild ausgeschlossen werden können.

Der Algorithmus des FAST Detektors:

A16.2: FAST Detektor.

Voraussetzungen und Bemerkungen:

$\diamond$ $\mathbf{S} = (s(x, y))$ ist ein Grauwertbild

Algorithmus:

(a) Wähle einen Bildpunkt $s(x, y)$ mit $x \in \{4, .., \text{size}_x(\mathbf{S})\}$ und $y \in \{4, .., \text{size}_y(\mathbf{S})\}$, Bildränder werden nicht betrachtet

(b) Betrachte die Bildpunkte
$s_1 = s(x, y-3), s_9 = s(x, y+3), s_5 = s(x+3, y), s_{13} = s(x-3, y)$:

(ba) falls die Grauwerte von 3 Bildpunkten aus s_1, s_9, s_5, s_{13} größer sind als $s(x, y)+t$, dann weiter zu (c)

(bb) falls die Grauwerte von 3 Bildpunkten aus s_1, s_9, s_5, s_{13} kleiner sind als $s(x, y)-t$, dann weiter zu (f)

(bc) sonst gilt: kein Eckenkandidat, gebe `false` zurück $\rightarrow$ Ende des Algorithmus

(c) Prüfung auf Eckenkandidat (in hellerer Umgebung): Setze Zähler auf 3

(d) Teste auf hellere Bildpunkte im Pixelkreis: $\forall i \in \{2-4, 6-8, 10-12, 14-16\}$

(da) Wenn $s_i > s(x, y) + t$, s_i ist der i-te Punkt auf dem Pixelkreis wie in Abbildung 16.3, dann erhöhe den Zähler um 1

(e) Wenn der Zähler < 12,
kein Eckenkandidat, gebe `false` zurück $\rightarrow$ Ende des Algorithmus
Wenn der Zähler ≥ 12, Eckenkandidat weiter zu (i)

(f) Prüfung auf Eckenkandidat (in dunklerer Umgebung): Setze Zähler auf 3

(g) Teste auf dunklere Bildpunkte im Pixelkreis $\forall i \in \{2-4, 6-8, 10-12, 14-16\}$

(ga) Wenn $s_i < s(x, y) - t$, s_i ist der i-te Punkt auf dem Pixelkreis wie in Abbildung 16.3, dann erhöhe den Zähler um 1

(h) Wenn der Zähler < 12,
kein Eckenkandidat, gebe `false` zurück $\rightarrow$ Ende des Algorithmus
Wenn der Zähler ≥ 12, Eckenkandidat weiter zu (i)

(i) Eckenkandidat, gebe `true` zurück $\rightarrow$ Ende des Algorithmus

Ende des Algorithmus

Ein Problem des FAST Detektors ist, dass in texturierten Bildbereichen, verrauschten Bildern oder an starken Grauwertübergängen zu viele Eckenkandidaten gefunden werden. Sinnvoll ist es, hier eine Gütefunktion zu definieren. Eine mögliche Gütefunktion ist die Summe der Beträge zwischen dem Grauwert des Eckenkandidaten $s(x, y)$ und den Grauwerten der Bildpunkte s_i des Kreises um den Eckenkandidaten. In einem kleinen Bildbereich ist dann nur derjenige Eckenkandidat mit der höchsten Gütefunktion eine detektierte Ecke.

16.3 Deskriptoren

Für detektierte charakteristischen Merkmale, auch *Features* genannt, werden Deskriptoren berechnet, die als Grundlage für das Matching von Korrespondenzen in Bildern oder das Matching von Objekten dienen. Ein Deskriptor ist wie der „Fingerabdruck“ eines charakteristischen Merkmals.

16.3.1 SIFT – Scale Invariant Feature Transform

Das von David Lowe 1999 erstmals vorgestellte SIFT-Verfahren stellt einen Meilenstein in der Entwicklung deskriptiver Matching-Verfahren dar [Lowe04a]. Viele andere Verfahren wie GLOH [Miko05], SIFT-PCA oder SURF [Bay06] basieren auf den Ideen des SIFT-Verfahrens. Der SIFT-Operator extrahiert Merkmale, die gegenüber Rotation, Translation und Skalierung invariant sowie partiell invariant gegenüber Helligkeitsänderungen sind. Das SIFT-Verfahren gliedert sich in vier Schritte:

1. Extremwertdetektion im Multiskalenraum
2. Bestimmung von charakteristischen Punkten (sog. Keypoints)
3. Bestimmung einer oder mehrerer Richtungen für jeden Keypoint
4. Berechnung eines Deskriptors für jeden Keypoint

Das SIFT-Verfahren ist unter [Lowe04b] patentiert.

Extremwertdetektion im Multiskalenraum

Für den SIFT-Operator werden Kandidaten für charakteristischen Punkte als Extremwerte im Multiskalenraum ermittelt. Es wird die sog. „Difference of Gaussian“ (DoG) eingesetzt. Grundlage des Verfahrens sind Gauß-Pyramiden (siehe Kapitel 17).

Zu einem Bild werden mehrere Multiskalen-Oktaven erzeugt. Eine Multiskalen-Oktave ist eine Gauß-Pyramide. Sukzessive wird das Bild in der Größe halbiert (jede 2. Zeile und jede 2. Spalte wird weggelassen), und es wird eine Gauß-Pyramide erstellt. Die Anzahl der Schichten einer Gauß-Pyramide hängt von der Bildgröße ab. Es wird von dem Autor des SIFT Verfahrens [Lowe04a] vorgeschlagen, zu einem Bild vier Oktaven mit jeweils fünf Schichten zu erstellen. Gegebenenfalls wird das Originalbild initial vergrößert und Zeilen

sowie Spalten werden interpoliert. Bild 16.4 zeigt vier Oktaven mit je fünf Schichten. Dieser Multiskalenraum ist die Grundlage für die Detektion der Keypoints, der charakteristischen Punkte.

Zu jeder Oktave wird nun die „Difference of Gaussian“ gebildet (Bild 16.4), indem benachbarte Schichten in einer Oktave subtrahiert werden. Das ist deutlich weniger Rechenaufwand als die Erstellung einer Laplace-Pyramide (siehe Kapitel 17) und ausreichend für die Detektion von Extrema im Multiskalenraum, die als Kandidaten für charakteristische Punkte im Bild gelten.

In den Differenzbildern wird nach Extremwerten, d.h. Maxima und Minima, gesucht. Ein Punkt ist dann ein Maximum oder Minimum, wenn es den größten oder kleinsten Wert in einer $9 \times 9 \times 9$ Umgebung im Multiskalenraum hat. Jeder Wert wird also mit 26 benachbarten Werten im Multiskalenraum verglichen. Diese $9 \times 9 \times 9$ Umgebung ist so aufgebaut, dass die 8-Nachbarn in einem Differenzbild sowie der Punkt und die 8-Nachbarn des im Multiskalenraum vorherigen und nachfolgenden Differenzbilds in der jeweiligen Oktave genommen werden (siehe Bild 16.6). Für die Werte des obersten oder untersten Differenzbildes einer Oktave werden keine Maxima bzw. Minima bestimmt. Für diese Differenzbilder sind nicht ausreichend Vergleichspunkte vorhanden. In allen berechneten Oktaven werden die Extrema der Differenzbilder ermittelt. Pro Oktave ergeben sich bei fünf Schichten vier Differenzbilder. Aus vier Differenzbildern erhält man zwei Ergebnisbilder mit Extrema (Differenzbild 1-3 und Differenzbild 2-4). Bei vier Oktaven ergibt dies acht Ergebnisbilder mit Extremwerten. Das sind zuerst einmal viele Extrema, die Kandidaten für charakteristische Punkte darstellen.

Bestimmung von charakteristischen Punkten

Für Extrema wird zuerst die genaue Lage im Originalbild bestimmt. Als Herleitung dazu verwendet [Lowe04a] die Taylorreihenentwicklung des Differenzbildes $D(x, y, \sigma) = D(\mathbf{x})$ von zwei Schichten der Gauß-Pyramide. σ ist der Parameter des Gauß-Filters und gibt die Stärke des Tiefpaßfilters an.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{16.5}$$

Ein Extremwert von D kann durch Bestimmung der Nullstellen der Ableitung gefunden werden. Analytisch bedeutet das:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}. \tag{16.6}$$

Die Lösung von (16.6) ist ein 3×3 lineares Gleichungssystem. Der Vektor $\hat{\mathbf{x}}$ gibt an, wie ein in einer Pyramidenschicht gefundener Extremwert-Kandidat verschoben werden muß, um im Originalbild die Lage eines charakteristischen Punkts (Keypoints) anzugeben.

Bild 16.4: Gauß-Pyramiden für die Detektion von Extremwerten im Multiskalenraum. Es wurden vier Pyramiden, hier Oktaven genannt, zu einem schrittweise halbierten Bild erzeugt. Dieser Multiskalenraum ist die Grundlage der Detektion der charakteristischen Punkte (Keypoints).

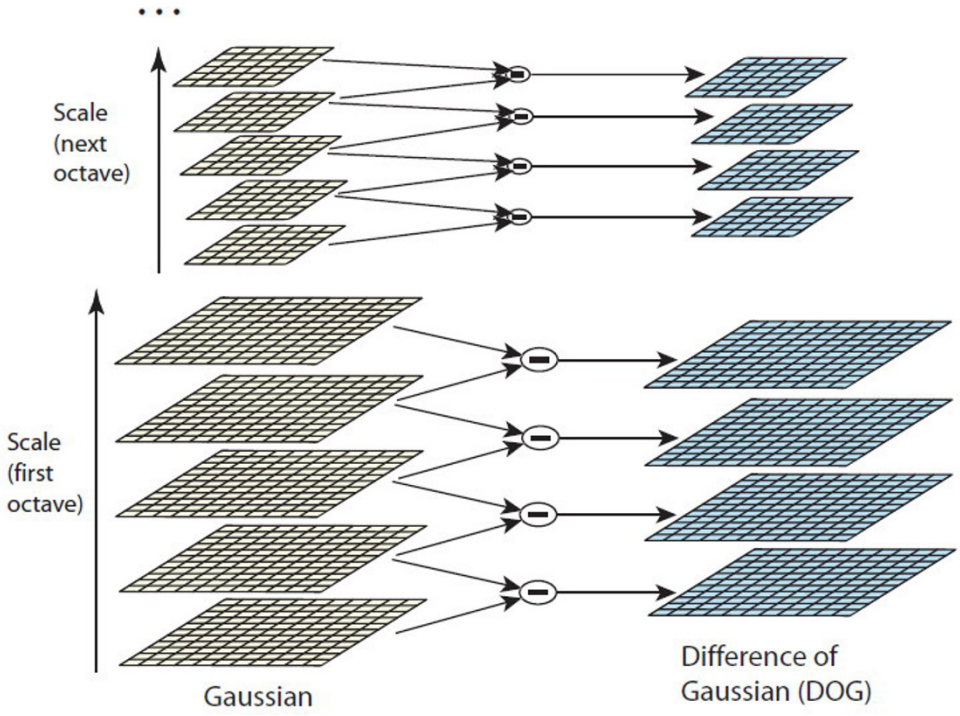


Bild 16.5: Lowes Verfahren zur Detektion lokaler Maxima im Multiskalenraum mittels Gaußfilterung [Lowe04a]. Es werden dabei die Schichten einer Gauß-Pyramide voneinander subtrahiert („Difference of Gaussian“ (DoG)). Diese Differenzbilder werden für alle Oktaven gebildet. Quelle: [Lowe04a]

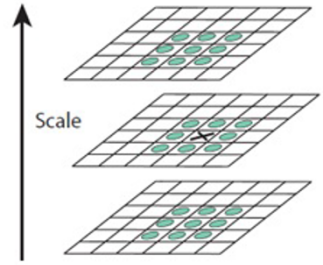


Bild 16.6: Verfahren zur Bestimmung von Minima und Maxima in den DoG-Bildern. Es wird ein Wert im DoG-Bild (X) mit 26 benachbarten Werten im Multiskalenraum verglichen. Quelle: [Lowe04a]

Der Wert von $D(\hat{\mathbf{x}})$ ist geeignet, um instabile Extrema mit geringem Kontrast herauszufiltern.

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}. \tag{16.7}$$

[Lowe04a] schlägt vor, alle Extrema mit $|D(\hat{\mathbf{x}})| < 0,03$ zu verwerfen. Hier ist der Kontrast in der lokalen Grauwertstruktur zu gering. Es wird von normierten Grauwerten im Bereich $[0,1]$ ausgegangen. Bild 16.7 ist [Lowe04a] entnommen und zeigt neben dem Originalbild (16.7-a), in 16.7-b alle Extrema im Multiskalenraum. In 16.7-c sind nur noch die Extrema zu sehen, für die $|D(\hat{\mathbf{x}})| \geq 0,03$.

Als letzter Schritt in der Bestimmung von charakteristischen Punkten wird nun für die verbleibenden Extrema geprüft, ob diese tatsächlich charakteristische Grauwertstrukturen beschreiben. Grauwertübergänge entlang einer Kante sind als charakteristische Punkte ungeeignet. Vergleichbar wie beim Harris Eckendetektor (Kapitel 16.2.1) werden die Eigenwerte der Hessematrix $\mathbf{H}$

$$\mathbf{H} = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix} \tag{16.8}$$

betrachtet. Bei einer Grauwertecke kann davon ausgegangen werden, dass beide Eigenwerte $\lambda_{1,2} > 0$ sind. Interessant sind diejenigen charakteristischen Punkte, für die die Eigengwerte von $\mathbf{H}$ annähernd gleich groß sind. Das ist der Fall wenn,

$$\frac{spur(\mathbf{H})^2}{\det(\mathbf{H})} < \frac{(r+1)^2}{r}. \tag{16.9}$$

Für Bild 16.7-d wurde in [Lowe04a] der Schwellwert für r auf 10 gesetzt.

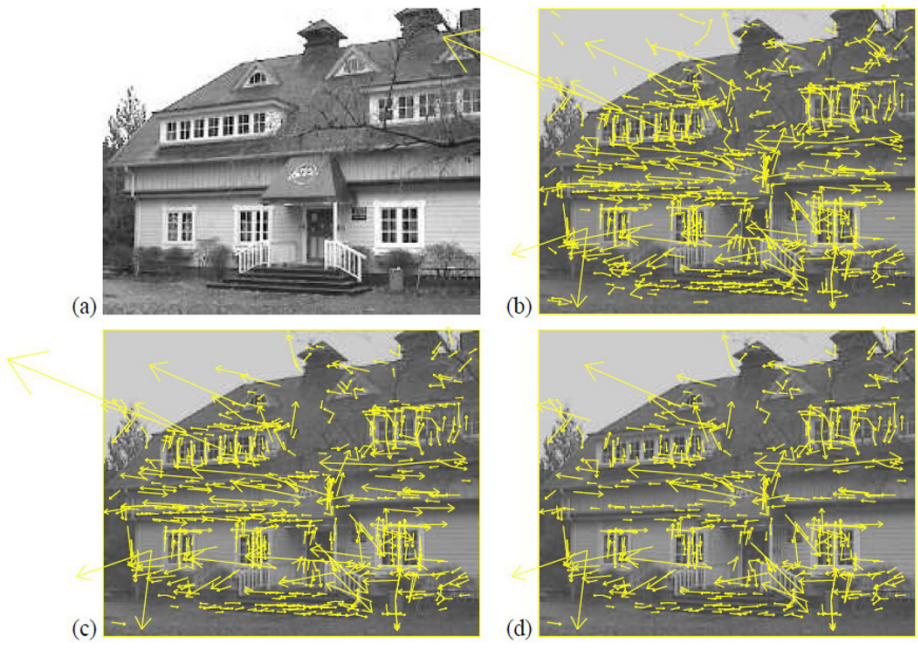


Bild 16.7: Detektion charakteristischer Punkte mit dem DoG-Verfahren als Extrema im Multiskalenraum. (a) Originalbild der Größe 233×189 Pixel, (b) alle 832 Extrema im Multiskalenraum sind dargestellt, (c) Reduktion der Extrema durch $|D(\hat{\mathbf{x}})| < 0,03$. Es verbleiben 729 Extrema, (d) 536 charakteristische Punkte, die an Grauwertecken liegen. Für alle Extrema ist die Hauptorientierung des Grauwertgradienten (Keypoint-Orientierung) als Pfeil dargestellt. Quelle: [Lowe04a]

Keypoint-Orientierung

Im nächsten Schritt wird für jeden durch Extrema im Multiskalenraum detektierten charakteristischen Punkt eine *Keypoint-Orientierung* berechnet. Dazu wird eine Umgebung der Größe $1,5 \cdot \sigma$ um einen charakteristischen Punkt betrachtet. Aus dieser Umgebung wird die dominierende Gradientenrichtung, d.h. Richtung der Grauwertänderung, für den charakteristischen Punkt bestimmt. Alle weiteren Berechnungen erfolgen später relativ zu dieser dominierenden Gradientenrichtung. Dadurch wird eine Rotationsinvarianz erreicht. In einer Umgebung um den charakteristischen Punkt wird nun in der Schicht in der Gauß-Pyramide, in der der Extremwert detektiert wurde ($L(x, y)$), für jeden Grauwert Gradientenbetrag und Gradientenrichtung folgendermaßen berechnet:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \varphi &= \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \end{aligned} \tag{16.10}$$

Gradientenbetrag m und Gradientenrichtung φ werden in ein Histogramm eingetragen. In diesem Histogramm sind die Klassen jeweils 10° Abschnitte der Gradientenrichtung. Die Gradientenrichtung liegt zwischen 0° und 360°. Immer 10° werden zu einer Klasse zusammengefasst. Das Histogramm hat demnach 16 Klassen. Die Häufigkeitsdichte jeder Klasse ergibt sich als die Summe der Gradientenbeträge der Umgebungspunkte, die wegen ihrer Gradientenrichtung in eine Klasse fallen. Die Keypoint-Orientierung ist nun die Gradientenrichtungsklasse für die die Häufigkeitsdichte größer als 80% ist. Gibt es mehr als eine Gradientenrichtungsklasse mit einer Häufigkeitsdichte von über 80%, so wird aus dem charakteristischen Punkt ein weiterer Keypoint mit der zweiten dominierenden Gradientenrichtung generiert.

Berechnung des Deskriptors für jeden Keypoint

Nach der Ermittlung der Ausrichtung des Gradienten, der Keypoint-Orientierung, werden die Deskriptoren berechnet. Dazu wird jeweils ein 16×16 großes Fenster auf der Schicht in der Gauß-Pyramide, in der der Keypoint detektiert wurde, um den Keypoint gelegt. Dieses 16×16 Fenster ist in 4×4 Unter-Fenster unterteilt, die jeweils 4×4 Pixel groß sind. Für jedes Pixel in einem 4×4 Unter-Fenster werden Betrag und Richtung des Grauwertgradienten (16.10) berechnet. Die Gradientenrichtungen werden in Richtung der Keypoint-Orientierung gedreht, um Rotationsinvarianz zu erreichen. Diese Gradientenrichtungen werden dann wieder in ein Histogramm eingetragen. Diesmal hat das Histogramm aber nur 8 Klassen (Abstufung von 45°). Weiterhin werden die Gradientenbeträge für die Häufigkeitsdichte des Histogramms mit dem Abstand vom Keypoint gewichtet. Bildpunkte, die näher an dem Keypoint liegen, zählen mehr als weiter entfernt liegende Bildpunkte. Jedes 4×4 Unter-Fenster liefert ein Orientierungshistogramm aus 8 Werten für Häufigkeitsdichten der Gradientenrichtungen. Aus dem ursprünglichen 16×16 Fenster werden

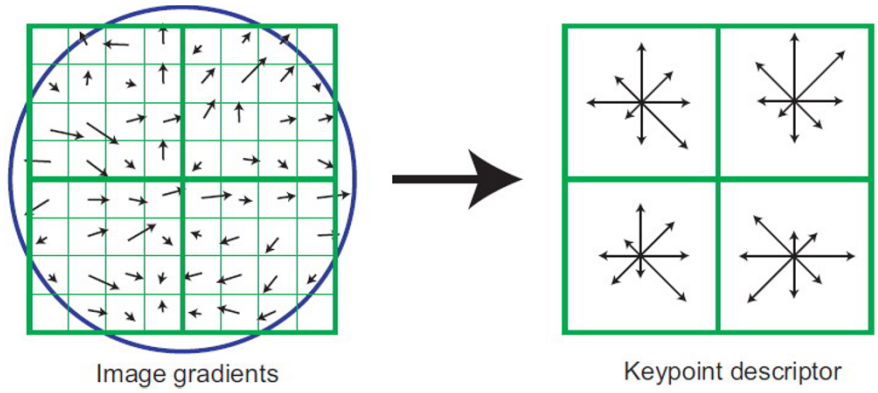


Bild 16.8: Darstellung eines beispielhaften SIFT-Deskriptors. In diesem Bild besteht der Deskriptor nur aus einer 2×2 Matrix und wird für jeweils 4×4 Werte in einer 8×8 Umgebung berechnet. Quelle: [Lowe04a]

4×4 Orientierungshistogramme gewonnen. Schreibt man die Werte der Orientierungshistogramme nun in einen Vektor so erhält man einen Vektor mit $4 \cdot 4 \cdot 8 = 128$ Werten. Dieser Vektor wird normalisiert und ergibt den SIFT-Deskriptor des Keypoints.

Bild 16.8 stellt beispielhaft die Berechnung des Deskriptors für einen Keypoint dar. Aus Gründen der besseren Darstellbarkeit wurde hier statt einem 16×16 großen Fenster nur eine 8×8 Umgebung gewählt. Auch hat der beispielhaft dargestellte Deskriptor in der Abbildung nur 4 statt 16 Orientierungshistogramme.

Das SIFT-Verfahren ist im folgenden Algorithmus zusammengefasst:

A16.3: SIFT: Scale Invariant Feature Transform.

Voraussetzungen und Bemerkungen:

$\diamond$ $\mathbf{S} = (s(x, y))$ ist ein Grauwertbild

Algorithmus:

(a) Generiere vier Oktaven (d.h. Gauß-Pyramiden) mit je fünf Schichten, passe ggf. die Bildgröße an

(b) Berechne die Differenzbilder aus benachbarten Schichten für jede Oktave

(c) Berechne die Extrema der Differenzbilder durch Betrachtung einer $9 \times 9 \times 9$ Umgebung

(d) Bestimme die genaue Lage der Extrema im Originalbild

(e) Verwerfe Extrema mit kontrastarmer lokaler Umgebung

(e) Verwerfe Extrema, die entlang einer Grauwertkante liegen; bevorzuge Grauwertecken

(f) Für alle Grauwertecken:

(fa) Bestimme die Keypoint-Orientierung

(fb) Berechne SIFT-Deskriptor

Ende des Algorithmus

16.3.2 SURF – Speeded-Up Robust Features

Ein Deskriptor mit dem Ziel effizienter zu sein als SIFT ist *SURF - Speeded Up Robust Features* [Bay06]. SURF erreicht bei Tests, beispielsweise nach [Miko05], allerdings nicht die Präzision von SIFT. Das grundlegende Vorgehen von SURF ist wie bei SIFT, aber SURF verwendet Vereinfachungen, um schneller zu sein als SIFT.

SURF *detektiert* skalen- und rotationsinvariante charakteristische Merkmale. SURF arbeitet nicht im Multiskalenraum wie SIFT, sondern auf Integralbildern und wendet Filter unterschiedlicher Größe an, um unterschiedliche Auflösungen, d.h. Skalen, von Bildern zu simulieren.

$$I_\Sigma(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} s(i,j) \tag{16.11}$$

Ein Integralbild (Gleichung 16.11) enthält im Bildpunkt $I_\Sigma(x,y)$ die Summe aller Grauwerte des rechteckigen Bereichs von Bildpunkt $s(0,0)$ bis Bildpunkt $s(x,y)$. Für den SIFT-Operator wird zur Detektion der charakteristischen Punkte die „Difference of Gaussian“ (DoG) eingesetzt. SURF basiert auf der Idee, die Determinante der Hessematrix $\mathbf{H}$ (siehe Gleichung 16.8) für die Bestimmung von Grauwertecken zu verwenden. Diese Idee wird dadurch vereinfacht, dass der in der Hessematrix benötigte Lapace-Operator mit einem Box Filter angenähert wird. Bild 16.9 zeigt die Approximation des Laplace-Operators mit einem Box Filter. Ein großer Vorteil dieser Näherung besteht darin, dass die Faltung mit dem Box Filter auf Integralbildern in konstanter Zeit für unterschiedliche Auflösungen, und zwar mit unterschiedlichen Filtergrößen, berechnet werden kann. Um Skalierungsinvarianz zu erreichen wird statt eines echten Multiskalenraums, wie bei SIFT, dieser nur implizit erstellt. Anstelle von Bildern unterschiedlicher Auflösung wird lediglich die Größe des Box Filters angepasst, um unterschiedlich große Teile des Bildes abzudecken und um so unterschiedliche Skalierungen des Bildes zu simulieren.

Analog zu SIFT wird zur Detektion von charakteristischen Punkten geprüft, ob ein Bildpunkt „nur“ an einer Kante liegt, oder ob es sich tatsächlich um eine Grauwertecke

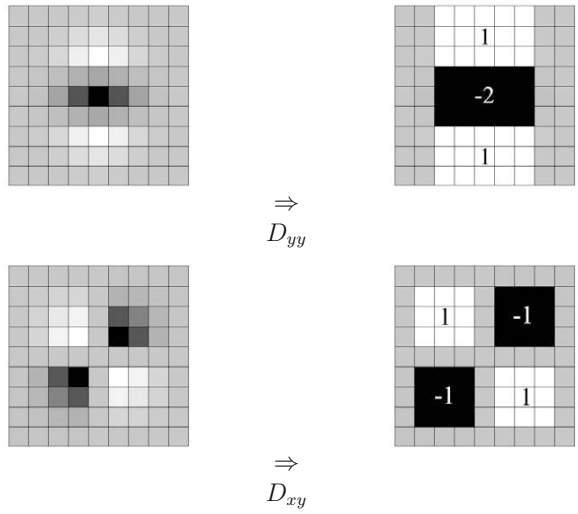


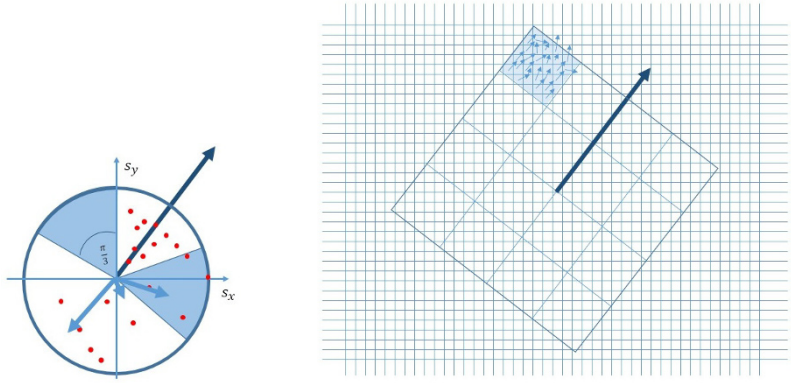
Bild 16.9: Approximation der zweiten partiellen Ableitung des Gaußfilters durch einen Box Filter. Quelle: [Bay06]

handelt. Dafür wird in einer $3 \times 3 \times 3$ Umgebung (in der eigenen Skale (d.h. mit dem Box Filter der eigenen Größe gefiltert) und in benachbarten Skalen (d.h. mit kleinerem oder größeren Box Filter gefiltert) geprüft, ob ein Maximum der Grauwertänderung vorliegt.

Der *SURF Deskriptor* folgt wiederum mit Vereinfachungen der Idee von SIFT. Der SURF Deskriptor ist ein 64-dimensionaler Vektor und damit nur halb so groß wie der SIFT Deskriptor. Der SURF Deskriptor wird für jeden charakteristischen Punkt in zwei Schritten ermittelt.

Der erste Schritt ist die *Bestimmung der Orientierung* des charakteristischen Punktes. Dazu wird um den detektierten charakteristischen Punkt eine kreisförmige Region der Größe $6 \cdot \sigma_i$ betrachtet. σ_i steht für die Skale des charakteristischen Punktes, die sich aus der Box Filter Größe ergibt. In dieser Umgebung werden zufällig Punkte anhand eines Sampling-Abstandes von σ_i ausgewählt. Die in diesen Punkten berechneten Grauwertgradienten werden zusätzlich mit einer Gauß-Funktion gewichtet, die um den charakteristischen Punkt als Mittelpunkt zentriert ist. Mithilfe eines „Sliding Orientation Windows“ der Größe $\pi/3$ der kreisförmigen Region werden die Beträge der berechneten Grauwertgradienten pro Sliding Orientation Window aufsummiert. Das Sliding Orientation Window mit der höchsten Summe gibt die Keypointrichtung an, d.h. die Orientierung des charakteristischen Punktes. Bild 16.10-a zeigt schematisch die Bestimmung der Orientierung des charakteristischen Punktes.

Der zweite Schritt ist die *Bestimmung des 64-dimensionalen Deskriptors*. Über den charakteristischen Punkt wird ein Quadrat (Seitenlänge = $20\sigma_i$) gelegt und ausgerichtet



(a) (b)

Bild 16.10: Bestimmung von Orientierung (a) und Deskiptor (b) für den SURF Operator. Die Orientierung ist die Richtung mit den höchsten Beträgen der Grauwertgradienten in einer Umgebung um den charakteristischen Punkt. Die Orientierung bestimmt die Richtung des Quadrates aus dem aus 16 Regionen der Vektor v aus Gleichung 16.12 bestimmt wird.

nach der Orientierung des charakteristischen Punktes. Bild 16.10-b zeigt schematisch das Quadrat der Größe $20\sigma_i$ in Richtung der Orientierung des charakteristischen Punktes. Das Quadrat wird in $4 \times 4 = 16$ Regionen unterteilt. Für jede Region wird für 25 Bildpunkte, die gleichmäßig über die Region verteilt sind, der Gradient in x- und y-Richtung, s_x und s_y, berechnet. Für jede der 16 Regionen wird dann der Vektor

$$\mathbf{v} = (\sum_{j=1}^{25} s_x(x_j, y_j), \sum_{j=1}^{25} s_y(x_j, y_j), \sum_{j=1}^{25} |s_x(x_j, y_j)|, \sum_{j=1}^{25} |s_x(x_j, y_j)|)^T \tag{16.12}$$

bestimmt. Der Deskriptor ergibt sich als Vektor aus allen 16 Vektoren $\mathbf{v}$ (Gleichung 16.12). Der Deskriptor hat damit die Dimension $16 \times 4 = 64$.

A16.4: SURF – Speeded-Up Robust Features.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S} = (s(x, y))$ ist ein Grauwertbild

Algorithmus:

(a) Bestimme das Integralbild $I_\Sigma(x, y)$ (siehe Gleichung 16.11)

(b) Falte $I_\Sigma(x, y)$ mit Box Filtern der Größe 9×9, 15×15, 21×21, 27×27

(c) Prüfe für jeden Bildpunkt, ob in einer $3 \times 3 \times 3$ Umgebung ein Maximum der Grauwertänderung vorliegt; Bildpunkte, für die über alle Skalen eine maximale Grauwertänderung vorliegt, sind charakteristische Punkte

(d) Für jeden charakteristischen Punkt:

(da) Bestimme die Orientierung:

(daa) Bestimme zufällig ausgewählte Punkte in einer kreisförmigen Region der Größe $6 \cdot \sigma_i$; für jeden Punkt

(daaa) Bestimme den Grauwertgradienten

(db) Summiere die Grauwertgradienten pro Sliding Orientation Window

(dc) Die Orientierung ist die Richtung des Sliding Orientation Window mit der höchsten Summe

(db) Bestimme den Deskriptor:

(dba) Lege über den charakteristischen Punkt ein Quadrat der Größe $20\sigma_i$, unterteile das Quadrat in 4×4 Regionen

(dbb) Wähle 25 gleichmäßig verteilte Bildpunkte in jeder Region und bestimme den Grauwertgradienten s_x, s_y

(dbc) Bestimme für jede Region den Vektor $\mathbf{v}$ (siehe Gleichung 16.12)

(db) Setze die 16 Vektoren $\mathbf{v}$ zu einem 64-dimensionalen Vektor zusammen

Ende des Algorithmus

16.3.3 HOG – Histogram of Oriented Gradients

HOG – Histogram of Oriented Gradients dient der Berechnung von Merkmalsvektoren, die zur Objekterkennung verwendet werden. Das Verfahren wurde von Navneet Dalal und Bill Triggs [Dala05] zur Erkennung von Fußgängern in statischen Bildern vorgestellt. Anders als SIFT ist HOG ein *globaler Feature-Deskriptor*, der die Struktur von Objekten beschreibt. HOG Features werden auch für die Erkennung von Gesichtern [Deni11] oder das Tracking von Objekten [Peso16] eingesetzt (siehe auch Kapitel 16.4).

Als globaler Feature-Deskriptor charakterisieren HOG Features den gesamten Bildinhalt und nicht nur charakteristische Punkte. HOG Features können von Klassifikatoren

(z.B. neuronale Netze (siehe Kapitel 21) oder Support Vector Machines) als Merkmale für die Objekterkennung oder die Objektverfolgung (Tracking) verwendet werden.

Die Berechnung von HOG Features wird in [Dala05] für Bildausschnitte der Größe 64×128 vorgestellt. Die Bildgröße bestimmt die Dimension des Deskriptors. Beliebig große Bildausschnitte sind möglich. Im folgenden wird wie in [Dala05] die Bildgröße 64×128 verwendet.

Gradientenberechnung

Für jeden Bildpunkt wird der Gradient in x- und y-Richtung berechnet. Es wird dazu keine Glättung vorgenommen, sondern es werden die folgenden einfachen Filter verwendet (vgl. Kapitel 5.4):

$$[-1, 0, 1] \qquad \text{und} \qquad [-1, 0, 1]^T \tag{16.13}$$

Berechnung von Gradientenrichtung und Gradientenbetrag

Aus den Gradienten in x- und y-Richtung, s_x und s_y, lassen sich Gradientenrichtung (Gleichung 5.27) und Gradientenbetrag (Gleichung 5.26) bestimmen. Der Betrag der Gradientenrichtung ist die Gradientenorientierung. Für ein Farbbild werden die Gradienten für jeden Farbkanal berechnet. Für jeden Bildpunkt wird im Weiteren nur noch der Gradient aus dem Farbkanal mit dem höchsten Gradientenbetrag betrachtet. Möglich ist auch, HOG Features für jeden Farbkanal separat zu bestimmen.

Erstellen von Histogrammen

Das Bild oder der Bildausschnitt der Größe 64×128 wird in Zellen der Größe 8×8 Bildpunkte unterteilt. Immer 2×2 Zellen werden zu einem Block der Größe von dann 16×16 Bildpunkten zusammengefasst. Die Blöcke werden so angeordnet, dass ein Block und seine benachbarten Blöcke mit 50% der Fläche überlappen. Es entstehen dadurch $7 \times 15 = 105$ Blöcke. Felzenszwalb u.a. zeigen in [Felz10], dass 4×4 Zellen effizienter zu berechnen sind. Bilder bzw. Bildausschnitte anderer Größen werden entsprechend in andere Zellen- und Blockanzahlen aufgeteilt.

Für jede Zelle wird ein Histogramm der Gradientenorientierungen erstellt. Die Gradientenorientierungen von $0 - 180°$ werden in 9 Klassen mit jeweils $20°$ Breite aufgeteilt. Für jeden Bildpunkt einer Zelle wird nun ein Eintrag in der jeweils passenden Gradientenorientierungsklasse des Histogramms vorgenommen. Es werden damit die Werte von $8 \times 8 = 64$ Bildpunkten auf 9 Werte reduziert. Die Histogrammerstellung ist beispielhaft in Bild 16.11 dargestellt. In [Felz10] werden neben Gradientenorientierungen auch Gradientenrichtungen $(0 - 360°)$ mit Gradientenrichtungsklassen bestimmt. Die Grundidee der HOG Features ist die Verwendung von Histogrammen zur Bestimmung der Deskriptoren. Eine Variation der genauen Regeln der Histogrammerstellung kann in konkreten Anwendungsfällen zu charakteristischeren Deskriptoren führen. Die Zuteilung zu den Klassen des Histogramms kann mit dem Gradientenbetrag gewichtet werden. Auch wird in der Literatur vorgeschlagen,

dass das Histogramm geglättet werden kann oder dass die Klassenzuteilung von Gradientenrichtungen an Klassengrenzen gewichtet über die benachbarten Klassen erfolgen kann. Man verwendet den Begriff *Soft-Binning*, wenn man den Eintrag in einen Histogramm-Bin mit der Entfernung zum Zellenmittelpunkt und dem Gradientenbetrag gewichtet.

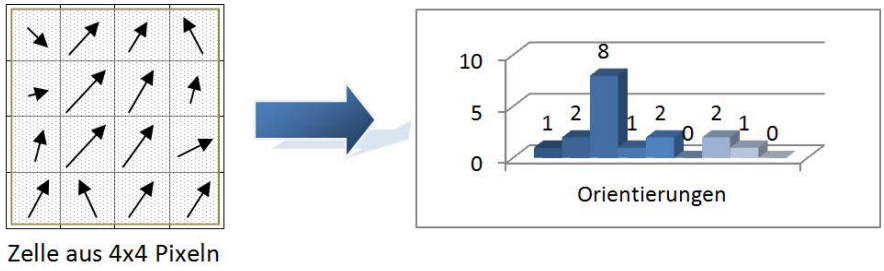


Bild 16.11: Histogramm einer 4×4 großen Zelle. Quelle: Masterarbeit von Danijel Peso, *Lernbasierte Korrelationstracker auf Smartphones*, Hochschule München, 2014

Block-Normalisierung

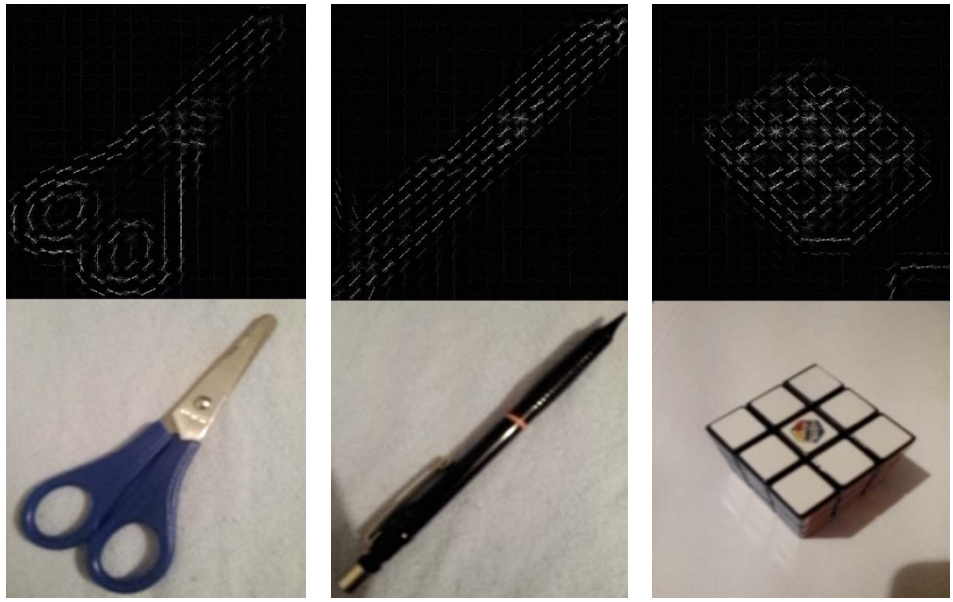
Dalal und Triggs [Dala05] schlagen vier Möglichkeiten vor, die Histogramme innerhalb eines Blocks zu normalisieren. Da sich die Histogramme der einzelnen Zellen deutlich unterscheiden, können die Autoren nachweisen, dass eine Normalisierung der Histogramme der Zellen innerhalb der Blöcke deutlich bessere Ergebnisse liefert als die Verwendung von nicht normalisierten Histogrammen. Eine mögliche Normalisierung für $\mathbf{v}$, den Vektor der Werte des Histogramms aus vier Zellen eines Blocks (d.h. 4×9-dimensional), ist

$$\frac{\mathbf{v}}{(||\mathbf{v}|| + e)}, \tag{16.14}$$

wobei e eine kleine Konstante ist. Es werden alle Histogramme eines Blocks normalisiert. Wegen der Überlappung der Blöcke ist eine Zelle in der Regel Bestandteil von vier Blöcken. Jede Zelle wird damit viermal normalisiert.

Merkmalsreduktion und PCA

Der HOG Deskriptor fügt die Histogramme aller Blöcke eines Bildes zusammen. Für Bilder oder Auschnitte der Größe 64×128 mit 105 Blöcken ergibt sich daraus eine Dimension des Deskriptors von $105 \times 4 \times 9 = 3780$ (105 Blöcke mit jeweils 4 Zellen und einer Histogramm-Dimension von 9). Für andere Bild- oder Zellengrößen ändert sich die Dimension des Deskriptors entsprechend. Um die Größe des Deskriptors zu reduzieren verwenden viele Anwendungen eine Hauptkomponententransformation (engl. Principal Component Analysis (PCA)) (siehe Kapitel 13.4)).



(a) Schere (b) Stift (c) Rubik's Cube

Bild 16.12: Visualisierung der HOG-Features anhand der Histogramme der Gradientenrichtungen. Quelle: Masterarbeit von Danijel Peso, *Lernbasierte Korrelationstracker auf Smartphones*, Hochschule München, 2014

In Bild 16.12 werden für jede Zelle die Histogramme der Gradientenrichtungen visualisiert. Jede Gradientenrichtung ist je nach deren Wert im Histogramm der Zelle als weiße Linie mit entsprechender Helligkeit eingezeichnet.

A16.5: HOG - Histogram of Oriented Gradients.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S} = (s(x, y, n))$ ist ein Farbbild oder $\mathbf{S} = (s(x, y))$ ist ein Grauwertbild der Größe 64×128

Algorithmus:

(a) Bestimme für jeden Bildpunkt den Gradienten in x und y-Richtung

(b) Berechne Gradientenbetrag und Gradientenrichtung für jeden Bildpunkt. Reduziere ein Farbbild auf ein einkanaliges Bild, in dem pro Bildpunkt jeweils nur der Gradient mit dem größten Gradientenbetrag weiter verwendet wird

(c) Teile das Bild in Zellen und Blöcke ein; für jede Zelle:

(ca) Bestimme das Histogramm der Gradientenorientierungen (9 Gradientenorientierungen)

(cb) (optional) gewichte Histogrammeinträge mit dem Gradientenbetrag

(cc) (optional) glätte Histogramm

(d) Normalisiere die Histogramme für jeden Block

(e) Führe die Histogramme aller Zellen eines Bildes/Bildausschnittes zu einem Deskriptor zusammen. Für die hier gewählte Bildgröße gibt es 105×4 Zellen mit jeweils 9-dimensionalen Histogrammen. Der Deskriptor hat daher 3780 Werte.

(f) Reduziere bei Bedarf die Größe des Deskriptors mit Hilfe einer Hauptkomponententransformation

Ende des Algorithmus

16.3.4 Weitere Deskriptoren

Auf dem SIFT-Operator basieren eine Anzahl weiterer Verfahren. Die Forschung versucht hier, effizienter berechenbare Deskriptoren zu finden. Weiterhin sind die Bemühungen derart, spezifischere, besser unterscheidbare und invariantere Deskriptoren zu finden. Alle merkmalsbasierten Verfahren haben das Ziel ohne Domänen- oder Weltwissen Korrespondenzen in Bildern zu finden. Sie sind damit immer dann geeignet Matching-, Tracking-, Erkennungs- oder andere Aufgaben zu lösen, wenn keine zusätzlichen Informationen über die abgebildete Szene vorliegen oder verwendet werden sollen.

Das *PCA-SIFT-Verfahren* [Ke04] basiert auf dem SIFT-Verfahren. Grundlegender Unterschied ist, dass der Deskriptor aus einer Matrix mit 39×39 Elementen besteht, wobei die Grauwertunterschiede nur in den beiden Achsen-Richtungen berechnet werden. Es entsteht ein Vektor mit $39 \cdot 39 \cdot 2 = 3042$ Elementen. Ein effizientes Matching ist damit aufgrund der Vektor-Länge zunächst nicht mehr möglich. Deshalb wird er durch die Methode der Principal Component Analysis (PCA, Hauptkomponententransformation (siehe Kapitel 13.4)) auf nur noch 20 Elemente reduziert. Mit 20 bis 50 Elementen (je nach Implementierung) in den zu vergleichenden Vektoren ist ein sehr effizientes Matching möglich. Kritiker des PCA-SIFT zeigen, dass der Deskriptor nicht so markant und charakteristisch ist wie der SIFT-Operator [Miko05].

2005 verglichen Mikolajczyk und Schmid verschiedene Matching-Verfahren miteinander und stellten im gleichen Paper das *GLOH: Gradient Location and Orientation Histogram* Verfahren vor [Miko05]. Auch hier handelt es sich um eine Anpassung des SIFT-Operators. Der Hauptunterschied liegt bei der Berechnung des Deskriptors, welcher für

ein kreisförmiges, log-polares Gitter berechnet wird. Hinzu kommt, dass statt 8 Orientierungen 16 Orientierungen betrachtet werden. Damit besteht der Deskriptor aus einem Vektor mit $17 \cdot 16 = 272$ Elementen. Auch hier ist das Matching zeitaufwendig, weshalb eine Hauptkomponententransformation (siehe Kapitel 13.4) eingesetzt wird. Der Deskriptor wird damit auf 64 Elemente reduziert.

SIFT verwendet 128-dimensionale-Vektoren als Deskriptoren. Da Gleitkommazahlen verwendet werden, sind 512 Byte pro Deskriptor und charakteristischem Punkt erforderlich. In ähnlicher Weise benötigt SURF auch mindestens 256 Bytes für einen 64-dimensionalen Deskriptor. Diese Deskriptor-Vektoren für Tausende von charakteristischen Punkten erfordern viel Speicher, der für Anwendungen mit eingeschränkten Ressourcen, insbesondere für Embedded Systeme, nicht realisierbar ist. Auch gilt, je größer der Speicherbedarf, desto länger dauert das Matching. Möglicherweise sind jedoch nicht alle Elemente der Deskriptor-Vektoren für das Matching erforderlich. Verschiedene Methoden wie PCA u.a. komprimieren Deskriptoren.

Weiterhin bietet eine Binärdarstellung von Deskriptoren den Vorteil, dass für den Vergleich von Deskriptoren beim Matching einfache XOR Operationen verwendet werden können. Binäre Zeichenfolgen werden verwendet, um Features mithilfe des Hamming-Abstands (*Hamming Distance*) abzugleichen. Dies sorgt für eine Beschleunigung, da beim Ermitteln der Hamming-Distanz nur XOR und Bitanzahl angewendet werden, die in modernen CPUs mit SSE-Anweisungen sehr schnell sind.

BRIEF - Binary Robust Independent Elementary Features [Calo12] ist ein Deskriptor, der binäre 128-512 Bit lange Deskriptor-Zeichenketten generiert. BRIEF verwendet einen geglätteten Bildbereich (Image-Patch) $\mathbf{p}$ um einen charakteristischen Bildpunkt $s_p(x, y)$. Es wird eine Reihe von n_d Bildpunkten $s_i(x, y)$ aus $\mathbf{p}$ nach einem Schema ausgewählt. Bildpunkte werden entweder zufällig oder zum Beispiel nach einem sternförmigen Muster innerhalb des Bildbereichs $\mathbf{p}$ gewählt. Der binäre Deskriptor entsteht dadurch, dass der Grauwert jedes gewählten Bildpunktes $s_i(x, y)$ mit dem Grauwert des charakteristischen Bildpunktes $s_p(x, y)$ verglichen wird. Ist der Grauwert des Bildpunktes $s_i(x, y)$ größer als der Grauwert des charakteristischen Bildpunktes $s_p(x, y)$, so wird eine 1 an die i-te Stelle des Deskriptors geschrieben. Ist der Grauwert kleiner, dann steht an der i-ten Deskriptor-Stelle eine 0. Der BRIEF Deskriptor ist der binäre String

$$f_{n_d}(\mathbf{p}) = \sum_{i=1}^{n_d} 2^{i-1} \tau(s_p, s_i), \tag{16.15}$$

wobei

$$\tau(s_p, s_i) = \begin{cases} 1 & \text{wenn } s_i > s_p, \\ 0 & \text{sonst.} \end{cases} \tag{16.16}$$

n_d kann 128, 256 oder 512 Zeichen lang sein. BRIEF ist ein Feature-Deskriptor und bietet keine Methode zur Detektion von Features. Für die Feature Detektion können die Feature Detektion von SIFT, SURF, FAST oder anderen verwendet werden. Der Vergleich mit dem Grauwert des charakteristischen Bildpunktes $s_p(x, y)$ ist eine stark vereinfachte Form der Gradientenbestimmung. Die Auswahl der Bildpunkte in dem Bildbereich $\mathbf{p}$

(sternförmig, nach Muster oder zufällig) bestimmt stark reduziert die Gradientenrichtung. Man erkennt also eine Ähnlichkeit von BRIEF mit HOG Features.

Ein weiterer binärer Deskriptor ist *BRISK - Binary Robust Invariant Scalable Keypoints* [Leut11]. Die Funktionsweise ist sehr ähnlich zu BRIEF. Der Deskriptor ist schneller, da Vereinfachungen gegenüber BRIEF vorgeschlagen werden. Es wird zum Beispiel immer ein festes Schema zur Auswahl der zu vergleichenden Bildpunkte $s_i(x, y)$ in einem quadratischen Bildbereich verwendet.

Auch im Bereich Detektoren und Deskriptoren von charakteristischen Punkten gibt es mittlerweile Verfahren, die Deep Learning einsetzen. Ein Vertreter ist *LIFT - Learned Invariant Feature Transform* [YiTr16]. Das Verfahren schlägt eine Deep Network-Architektur vor, die aus einer Pipeline für Detektor, Bestimmung der Keypoint Orientierung und Deskriptor besteht.

Viele weitere aktuelle Publikationen beinhalten weitere Vorschläge zur Effizienzverbesserung, die für eine eigene Implementierung recherchiert werden sollten.

16.4 Tracking

Eine Anwendung von Korrespondenzen in Bildern ist das Tracking von Objekten oder Bildbereichen in Bildfolgen. Tracking bedeutet das Verfolgen von Bildinhalten von einem Bild einer Bildfolge $\mathbf{g}_t$ aufgenommen zum Zeitpunkt t zum nächsten Bild der Bildfolge $\mathbf{g}_{t+1}$. Das Tracking oder Verfolgen ist damit ein Matching- oder Korrespondenzproblem.

Traditionelle Tracking Ansätze erreichen das Matching durch Erkennen eines Objektes in $\mathbf{g}_t$ und in $\mathbf{g}_{t+1}$. Man verwendet damit oft ein *Modell-basiertes Tracking*, da für die Objekterkennung Welt- oder Domänenwissen („a priori"-Wissen, d.h. ein Modell) vorhanden sein muss. Ein bewährtes Modell-basiertes Tracking Verfahren ist ein Kalman-Filter (siehe Kapitel 27). Große Herausforderungen sind hierbei vor allem das erfolgreiche Tracking trotz verrauschter Bilddaten, Verdeckungen, Deformationen, Verdrehungen des Objekts, Background-Clutter und Änderungen der Lichtverhältnisse sowie mögliche Echtzeitanforderungen bei der Berechnung.

Für ein Modell-basiertes Tracking eignen sich auch Deskriptoren. Das Modell wird in Form von charakteristischen Punkten und seinen Deskriptoren aus einem Referenzbild oder -objekt vorgegeben. Durch effizientes Matching werden Korrespondenzen von Modell und Abbildung des Objektes in einer Bildfolge gefunden. Bild 16.13 zeigt drei Beispielbilder einer Bildfolge, in der ein Fahrzeug mit einem Smartphone verfolgt wird, das auf einem Quadrocopter montiert ist. Es werden FAST Features detektiert und der BRIEF Deskriptor verwendet. Das Matching wird mit Hilfe des RANSAC-Algorithmus [Fisc81] durchgeführt. Bild 16.13 zeigt das erfolgreiche Tracking.

Ein neuerer Ansatz zur Objektverfolgung heißt *Tracking-By-Detection*. Dabei werden Objekte oder Merkmale von Objekten über die Zeit t in den jeweiligen Bildern der Bildfolge detektiert. Um auf Veränderungen reagieren zu können, werden häufig Machine-Learning Verfahren eingesetzt, durch die bei jeder Detektion neue Informationen über das zu verfolgende Objekt erlernt werden. Diese Vorgehensweisen sehen das Tracking als eine Klas-

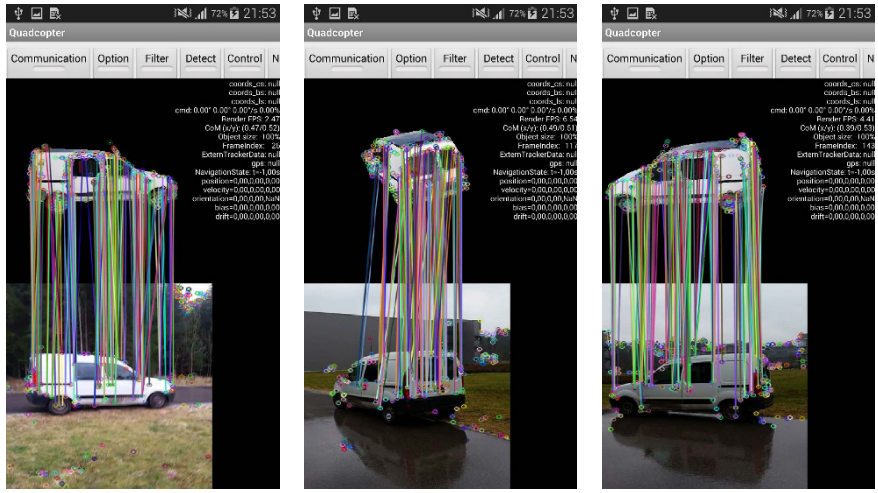


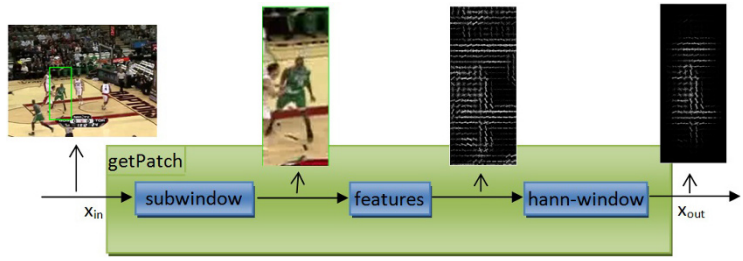
Bild 16.13: Modell-basiertes Tracking mit Hilfe von Merkmalen: Die Merkmale werden mit dem FAST Detektor (Kapitel 16.2.2) detektiert. Es wird der BRIEF Deskriptor (Kapitel 16.3.4) und der RANSAC-Matching Algorithmus verwendet. Das Tracking erfolgt in der Beispielanwendung nicht zwischen Bildern einer Bildfolge sondern zwischen computergenerierten Modellen und realen Abbildungen eines Fahrzeugs. Dadurch wird es möglich neben dem Tracking des Fahrzeugs auch die Pose (d.h. Lage des Fahrzeugs im Raum) zu schätzen. Quelle: Masterarbeit von Siegfried Ippisch, *Modell-basierter Tracker auf Smartphones*, Hochschule München, 2016

sifikationsaufgabe an, um zur Laufzeit ein Objektmodell zu erstellen und zu aktualisieren. Wenn das Objektmodell zur Laufzeit gelernt wird, ist damit kein oder weniger a-priori Wissen erforderlich und es kann „On-The-Fly“ festgelegt werden, welches Objekt verfolgt wird.

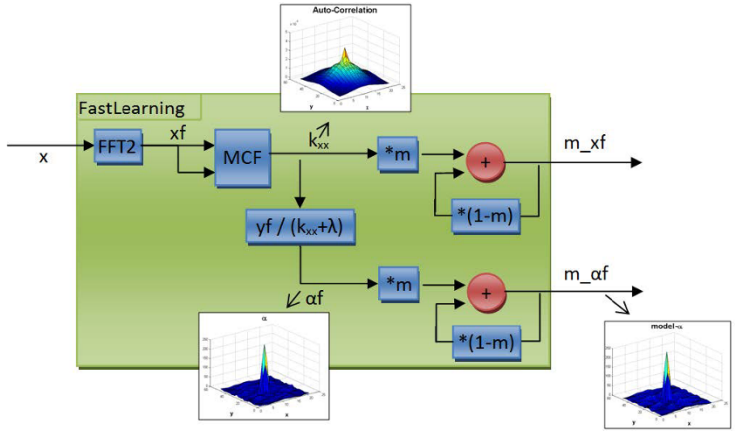
Gerade für mobile Anwendungen (z.B. auf modernen Smartphones) ist ein Tracking in Echtzeit elementar wichtig. Je weniger Modellwissen erforderlich ist und je schlanker die Tracking Verfahren sind, desto eher können Echtzeitanforderungen erfüllt werden.

Deskriptoren lassen sich gut für ein Tracking-By-Detection verwenden. Deskriptoren benötigen kein Domänen- oder Weltwissen um Korrespondenzen in Bildern einer Bildfolge zu finden. Für ein Tracking in Echtzeit sind aber neben guten Deskriptoren effiziente Matching-Algorithmen erforderlich. Peso et al. [Peso16] beschreiben einen *lernbasierten Korrelationstracker* für das Tracking von Objekten in Echtzeit auf Smartphones. Bild 16.14 visualisiert die Verarbeitungsschritte in dem Tracking Algorithmus.

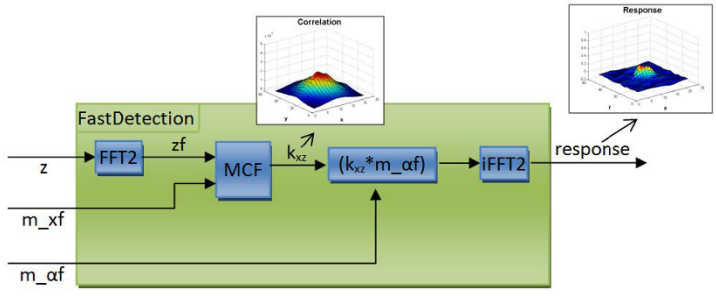
Schritt 16.14-a ist die Festlegung des zu trackenden Objektes bzw. des zu trackenden Bildbereichs. Für den zu trackenden Bildbereich werden die HOG-Deskriptoren berechnet und darauf ein sog. Hann-Window, d.h. eine Cosinus-Funktion angewendet, die die Werte zum Rand absenkt, damit es bei den späteren Verarbeitungsschritten weniger Artefakte bei



(a) Festlegen des zu trackenden Objektes und Berechnung der HOG Deskriptoren



(b) Lernen und Aktualisieren des gelernten Modells



(c) Finden des zu trackenden Objektes und Tracking (d.h. Matching und Tracking)

Bild 16.14: Visualisierung des lernbasierten Korrelationstrackers in Blockschaltbildern mit Zwischenergebnissen. Quelle: [Peso16]

der Transformation in den Frequenzraum gibt. In der Bildverarbeitung werden Korrelationsfilter häufig zur Erkennung und zum Tracking von Objekten verwendet. Hierzu werden Muster, welche die Filter darstellen, mit einem Bild verglichen, in dem sich das Objekt befindet. Das Ergebnis ist ein Peak, welcher die Stelle im Bild mit der größten Ähnlichkeit darstellt. Die Berechnung erfolgt durch eine diskrete Faltung bzw. Kreuzkorrelation mit einer relativen Verschiebung von Muster und Bild. Wie in Kapitel 8 erklärt, vereinfacht sich eine diskrete Faltung im Ortsraum zu einer Multiplikation im Frequenzraum. Der lernbasierte Korrelationstracker von [Peso16] verwendet statt der diskreten Faltung im Ortsraum eine Multiplikation im Frequenzraum (siehe Kapitel 8), um den Echtzeitanforderungen zu genügen.

Für die Verwendung von Merkmalsvektoren wie SIFT oder HOG kommen sogenannte Multi-Channel Korrelationsfilter (Abkürzung MCF aus Bild 16.14-a) zum Einsatz. Die einzelnen Dimensionen des Deskriptor-Vektors stellen hierbei Feature-Channels dar. Die Korrelation wird dabei auf jeden Feature-Channel angewendet. Jede Anwendung eines Korrelationsfilters auf einen Channel ergibt jeweils einen Peak. Durch anschließendes Aufsummieren erhält man einen Gesamtpeak als Endergebnis.

Schritt 16.14-b ist das Lernen und Aktualisieren des Modells. Hierzu werden Deskriptoren (z.B. HOG Features) an der aktuellen Position im Bild berechnet und für den Multi-Channel Korrelationsfilter optimiert. Schritt 16.14-c ist die Erkennung des zu trackenden Objektes im nachfolgenden Bild. Hierzu wird wieder der Multi-Channel Korrelationsfilter im Frequenzraum verwendet. Die größte Impulsantwort liefert die Änderung der Position im nachfolgenden Bild.

Bild 16.15 zeigt Ergebnisse des lernbasierten Korrelationstrackers. Man erkennt, dass

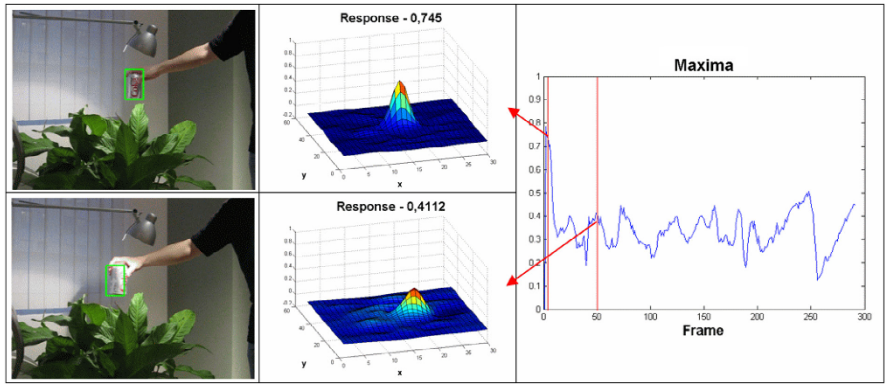


Bild 16.15: Ergebnisse des lernbasierten Korrelationstrackers und Gütefunktion des Matchings: je größer der Peak, desto besser ist das Tracking. Der lernbasierte Korrelationstracker ist robust gegenüber partiellen Verdeckungen und kann auch z.B. bei Glanzlichtern das Objekt tracken. Der Peak fällt geringer aus. Quelle: [Peso16]

die Impulsantwort die Güte der Erkennung zeigt. Man erkennt auch, dass der lernbasierte Korrelationstracker robust gegenüber Beleuchtungsänderungen und Verdeckungen ist. Für das Tracking in Bild 16.15 werden HOG Features verwendet. Durch die effiziente Implementierung ist das Tracking auf einem Smartphone in Echtzeit möglich.

Kapitel 17

Gauß- und Laplace-Pyramiden

17.1 Anwendungen

In diesem Kapitel wird eine interessante Kombination aus Unschärfe und Schärfe erläutert, die bei einigen Anwendungen sinnvoll eingesetzt werden kann. Soll z.B. ein Bild durch eine Filteroperation (Glättungsoperator) sehr stark unscharf gemacht werden, so kann dies durch einen lokalen Summenoperator erzielt werden (Kapitel 5). Allerdings muss dazu der Filterkern entsprechend groß gewählt oder mehrfach wiederholt werden, was einen erhöhten Rechenaufwand bedeutet (er wächst hier quadratisch mit der Größe des Filterkerns). Eine andere Möglichkeit wäre eine Filterung im Ortsfrequenzbereich. Das bedeutet aber unter anderem eine zweidimensionale Fourier-Transformation und eine zweidimensionale, inverse Fourier-Transformation, beides Operationen, die im Rechenaufwand nicht zu unterschätzen sind. In diesem Kapitel wird eine Technik erläutert, die es erlaubt, Bilder mit geringerem Aufwand zu filtern.

Eine andere Anwendung ist die Möglichkeit des *image mosaicing*, also des Zusammensetzens eines Ausgabebildes aus mehreren Eingabebildern. Hier werden, nach Maßgabe einer binären Bildmaske, Ausschnitte zweier Eingabebilder verwendet und zu einem Mosaikbild zusammengesetzt. Würde man nur die Bildausschnitte der beiden Eingabebilder zusammenkopieren, so würde man im Mosaikbild deutliche Ränder sehen. Mit der hier vorgeschlagenen Technik wird dieser Effekt vermieden. Die Bildausschnitte werden wechselseitig interpoliert, so dass in den Übergangsbereichen keine störenden Strukturunterschiede auftreten.

Mit den Gauß- und Laplace-Pyramiden können die Strukturen von Oberflächen (Texturen) untersucht werden. Beispielsweise können einzelne Informationsschichten als Masken bei der Segmentierung dienen. Anwendungsbeispiele dazu sind die Qualitäts- und Vollständigkeitskontrolle. Hier ergibt sich ein Zusammenhang mit der fraktalen Geometrie und dem *scale space filtering* (Kapitel 18).

Schließlich finden Gauß-Pyramiden eine breite Anwendung in der 3D-Computergrafik. Beim Textur Mapping werden häufig verkleinerte Varianten einer Textur benötigt, die in Form einer vorab berechneten Gauß-Pyramide zur Verfügung gestellt werden. In der

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_17

3D-Computergrafik werden Gauß-Pyramiden als *MipMaps* bezeichnet (Band I Abschnitt 13.1.3). Eine weitere Anwendung ist die sogenannte z-Pyramide beim *Occlusion Culling* (Band I Abschnitt 16.2.2).

17.2 Begriffe aus der Signaltheorie

Zum besseren Verständnis der Gauß- und Laplace-Pyramiden sind in diesem Abschnitt in kurzer Form einige Grundlagen der Signaltheorie zusammengestellt. Wenn Strukturen untersucht werden, die sich periodisch wiederholen, so charakterisiert man sie durch ihr Schwingungsverhalten, nämlich durch die Länge einer Periode oder die Anzahl der Schwingungen pro Einheitslänge. In zeitabhängigen Systemen bezeichnet man die Länge einer Periode als *Periodendauer* mit der Einheit *Sekunden* und die Anzahl der Schwingungen pro Sekunde als *Frequenz* mit der Einheit $Sekunden^{-1}$. Bei Bildern verwendet man die Begriffe *Wellenlänge* λ (Einheit: *Länge in Pixel*) und *Ortsfrequenz* f oder *Wellenzahl* k (Einheit: *Anzahl der Wellen pro Pixel*). Naturgemäß ist f bzw. k immer kleiner als eins. Es gilt

$$f = k = \frac{1}{\lambda}. \tag{17.1}$$

Die größtmögliche Wellenlänge wird durch die Bildgröße bestimmt: Bei einem Bild mit z.B. 512 Bildpunkten pro Zeile und Spalte sind die maximale Wellenlänge und die minimale Wellenzahl pro Zeile (Spalte) vorgegeben durch

$$\lambda_{max} = 512 Pixel \text{ und } f_{min} = \frac{1}{\lambda_{max}} = \frac{1}{512} Wellen\ pro\ Pixel. \tag{17.2}$$

Die Größe des Bildes bestimmt somit auch, wie oft sich eine periodische Struktur innerhalb eines Bildes wiederholen kann. Diese Anzahl wird als *Wellenzahlindex* u bezeichnet.

Das Abtasttheorem ([Abma94]) besagt, dass eine periodische Struktur aus den Abtastwerten nur dann richtig rekonstruiert werden kann, wenn die kleinste Wellenlänge mindestens zweimal abgetastet wird. Die maximale Wellenzahl, die bei einer gegebenen Abtastrate fehlerfrei rekonstruiert werden kann, heißt *Grenzwellenzahl*.

Die minimale Wellenlänge und die korrespondierende maximale Wellenzahl in einem digitalen Bild werden somit nicht durch die Bildgröße, sondern durch das Abtasttheorem beeinflusst. Für sie gilt immer:

$$\lambda_{min} = 2 Pixel \text{ und } f_{max} = \frac{1}{\lambda_{min}} = \frac{1}{2} Wellen\ pro\ Pixel. \tag{17.3}$$

In einem $512 \cdot 512$–Bild kann sich somit eine Struktur mit der Wellenzahl $k_{max} = 1/2$ höchstens 256 mal wiederholen, d.h. dass für den maximalen Wellenzahlindex gilt $u_{max} = 256$.

Der kleinstmögliche Wellenzahlindex ist immer $u_{min} = 1$, da die größtmögliche Struktur innerhalb einer Zeile (Spalte) nur einmal auftreten kann.

In einem Bild werden nun, begrenzt durch die minimale und die maximale Wellenlänge, viele verschiedene Wellenlängen auftreten. Die Menge der in einem Bild auftretenden Wellenlängen und den damit verbundenen Wellenzahlen (Ortsfrequenzen) bezeichnet man als *Spektrum* des Bildes. Es kann durch eine Fourier-Transformation berechnet werden (Kapitel 8).

17.3 Motivation für Gauß- und Laplace-Pyramiden

Da die Gauß- und Laplace-Pyramiden Kenntnisse aus der Signaltheorie voraussetzen, werden einige relevante Begriffe kurz zusammengefasst.

Bei vielen Anwendungen der digitalen Bildverarbeitung und Mustererkennung ist es notwendig, Bilder oder Bildausschnitte anhand ihres Frequenzverhaltens zu analysieren und dazu in ihre Frequenzanteile zu zerlegen. Der Begriff *Frequenz* wird in diesem Zusammenhang gleichbedeutend mit *Ortsfrequenz* oder *Wellenzahl* (siehe vorhergehender Abschnitt) verwendet.

Die Fourier-Transformation, mit der eine Frequenzzerlegung berechnet werden kann, ist aber mit einigen Nachteilen verbunden. Hier ist zunächst der beachtliche Rechenaufwand zu erwähnen, der jedoch bei der Verwendung von schneller Hardware bewältigt werden kann. Der zweite Nachteil ist, dass im Frequenzraum die einzelnen Frequenzanteile zwar zu erkennen sind, jedoch ihre direkte Zuordnung zu Strukturen im Ortsbereich fehlt.

Man benötigt somit eine Darstellungsform im Ortsbereich, die ein Bild in mehrere Frequenzbereiche aufspaltet. Die Gauß- und Laplace-Pyramiden führen eine Zerlegung eines Bildes in dieser Form durch.

Die grundlegende Idee ist im Folgenden anhand eines einfachen, eindimensionalen Beispiels erläutert: Angenommen, eine Bildzeile eines Bildes bestehe aus 32 Bildpunkten. Dann können maximal 16 Wellen der Wellenlänge $\lambda_{min} = 2Pixel$ in dieser Zeile auftreten. Das entspricht einer maximalen Wellenzahl von $k_{max} = \frac{1}{2}$ *Wellen pro Pixel*. Die maximale Wellenlänge ist $\lambda_{max} = 32Pixel$, was einer minimalen Wellenzahl von $k_{min} = \frac{1}{32}$ *Wellen pro Pixel* entspricht.

Wenn nun in dieser Bildzeile nur Wellenlängen auftreten, die kleiner sind als $\frac{1}{2}\lambda_{max}$, im gewählten Beispiel also kleiner als 16, so kann diese Zeile auch ohne Informationsverlust mit einem doppelt so großen Raster dargestellt werden. Das Abtasttheorem besagt außerdem, dass die ursprüngliche Bildzeile mit 32 Pixel aus dem gröberen Raster ohne Informationsverlust rekonstruiert werden kann. Ein eindimensionaler Glättungsoperator, etwa der Länge drei, hat aber auf dem gröberen Raster eine wesentlich stärkere Auswirkung als auf dem ursprünglichen Raster mit 32 Bildpunkten. Um Rechenaufwand bei einer Glättung zu sparen, könnte man somit das Raster vergröbern, die Glättung durchführen und anschließend wieder zum ursprünglichen Raster zurückkehren.

Auf diesem Sachverhalt baut die Idee der Gauß- und Laplace-Pyramiden auf: Auf ein Grauwertbild $\mathbf{S}_e = (s_e(x, y))$ wird ein geeigneter Glättungsoperator angewendet, und das so entstehende Bild wird in Zeilen- und Spaltenrichtung um die Hälfte verkleinert. Dieser Vorgang (Tiefpassfilterung und Verkleinerung) wird iterativ fortgesetzt, bis ein Bild mit

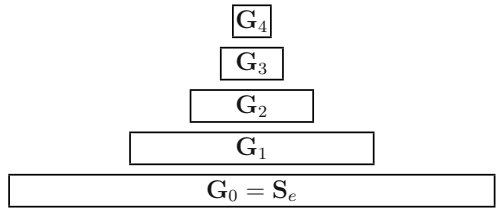


Bild 17.1: Gauß-Pyramide: Ausgehend vom Originalbild in der Schicht (Stufe), die am weitesten unten liegt, entsteht die darüber liegende Schicht durch eine Tiefpassfilterung und eine Halbierung der Zeilen- und Spaltenanzahl.

minimalem Flächeninhalt ($2 \cdot 2$ oder $3 \cdot 3$ Pixel) erreicht ist. Die so entstandene Folge von geglätteten und flächenmäßig jeweils um den Faktor vier reduzierten Bildern bezeichnet man als *Gauß-Pyramide*. Stellt man sich in einer grafischen Darstellung die einzelnen Bilder übereinander geschichtet vor, so entsteht die Form einer Pyramide, woher der Name kommt (Bild 17.1).

Die einzelnen Gauß-Pyramidenstufen unterscheiden sich durch ihren Frequenzanteil. Die unterste Stufe enthält alle Wellenzahlen des Originalbildes, also sein gesamtes Spektrum. Wählt man die Tiefpassfilterung so, dass die Frequenzen des halben Spektrums herausgefiltert werden, so enthält jede Stufe die Frequenzen des halben Spektrums der darunterliegenden Stufe.

Das Abtasttheorem besagt nun, dass ein Signal, bei dem die maximale Frequenz herausgefiltert wurde, durch weniger Abtastwerte dargestellt werden kann, ohne dass ein Informationsverlust auftritt. Da bei der Tiefpassfilterung die Breite des Spektrums halbiert wurde, kann man in der darüberliegenden Schicht die Zeilen- und Spaltenanzahl halbieren.

Die *Laplace-Pyramide* wird durch die Differenzbildung der übereinander liegenden Schichten der Gauß-Pyramide gewonnen. Sie ist somit eine Folge von hochpassgefilterten Bildern.

17.4 Der REDUCE-Operator

In diesem und den folgenden Abschnitten wird der Aufbau von Gauß- und Laplace-Pyramiden näher beschrieben. Dazu werden einige Annahmen gemacht und verschiedene Bezeichnungen eingeführt: Das Originalbild $\mathbf{S}_e = (s_e(x, y))$ sei quadratisch und besitze die Zeilen- und Spaltenlänge $R + 1 = 2^r + 1$. Bei $R = 512$ hat $\mathbf{S}_e$ also die Zeilen und Spalten $0, 1, ..., 512$ $(r = 9)$.

Die unterste Stufe der *Gauß-Pyramide* wird mit $\mathbf{G}_0$ bezeichnet. Sie ist mit dem Origi-

nalbild $\mathbf{S}$ identisch: $\mathbf{G}_0 = \mathbf{S}_e$. In der Praxis treten häufig Bilder der Größe $512 \cdot 512$ oder $256 \cdot 256$ auf. Für das hier beschriebene Verfahren werden aber Bilder der Größe $513 \cdot 513$ oder $257 \cdot 257$ benötigt. Man kann dies erreichen, indem man beim Aufbau der untersten Schicht die erste Bildzeile und die erste Bildspalte des Ausgangsbildes $\mathbf{S}_e$ verdoppelt. In der weiteren Darstellung wird dieser Sachverhalt nicht mehr berücksichtigt. Vielmehr wird abkürzend z.B. von $512 \cdot 512$–Bildern gesprochen.

Die weiteren Schichten sind $\mathbf{G}_1$, $\mathbf{G}_2$ bis $\mathbf{G}_r$. $\mathbf{G}_r$ ist die Spitze der Pyramide mit $2 \cdot 2$ Bildpunkten. In manchen Implementierungen wird als Spitze die darunter liegende Schicht mit $3 \cdot 3$ Bildpunkten verwendet. Die Zählung der Schichten geht dann von $\mathbf{G}_0 = \mathbf{S}_e$ bis $\mathbf{G}_{r-1}$. Wenn die Transformation von $\mathbf{G}_i$ zu $\mathbf{G}_{i+1}$ mit einer Reduktionsfunktion REDUCE bezeichnet wird, so kann der Aufbau einer Gauß-Pyramide formal wie folgt beschrieben werden:

$$\begin{aligned} &\mathbf{G}_0 = \mathbf{S}_e; \\ &\mathbf{G}_{i+1} = \text{REDUCE}(\mathbf{G}_i), \quad i = 0, 1, ..., r-1. \end{aligned} \tag{17.4}$$

In Bild 17.2 ist dieser Formalismus zum Aufbau einer Gauß-Pyramide als Blockdiagramm dargestellt. Bild 17.3 zeigt die Schichten $\mathbf{G}_0$ bis $\mathbf{G}_6$ eines Testbildes.

Aufgrund des Konstruktionsverfahrens der Gauß-Pyramide kann man Aussagen machen über den maximalen Wellenzahlindex und damit über das Spektrum der einzelnen Stufen. Für ein Bild mit z.B. $512 \cdot 512$ Zeilen und Spalten ist das in folgender Tabelle zusammengestellt:

Stufe	Zeilen/Spalten	max. Wellenzahlindex
$\mathbf{G}_0$	$2^9 + 1 = 513$	256
$\mathbf{G}_1$	$2^8 + 1 = 257$	128
$\mathbf{G}_2$	$2^7 + 1 = 129$	64
$\mathbf{G}_3$	$2^6 + 1 = 65$	32
$\mathbf{G}_4$	$2^5 + 1 = 33$	16
$\mathbf{G}_5$	$2^4 + 1 = 17$	8
$\mathbf{G}_6$	$2^3 + 1 = 9$	4
$\mathbf{G}_7$	$2^2 + 1 = 5$	2
$\mathbf{G}_8$	$2^1 + 1 = 3$	1
$\mathbf{G}_9$	$2^0 + 1 = 2$	-

17.5 Der EXPAND-Operator

Um die einzelnen Schichten bei weiteren Verarbeitungsschritten miteinander vergleichen zu können, ist es notwendig, mit einer EXPAND-Operation eine Schicht auf die Größe der darunterliegenden Schicht expandieren zu können. Die fehlenden Zeilen und Spalten werden dabei interpoliert. Die EXPAND-Operation wird mehrmals nacheinander ausgeführt, so

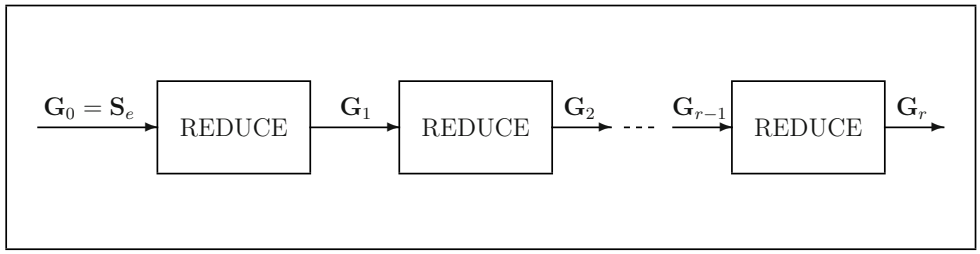


Bild 17.2: Blockdiagramm zum Aufbau einer Gauß-Pyramide.

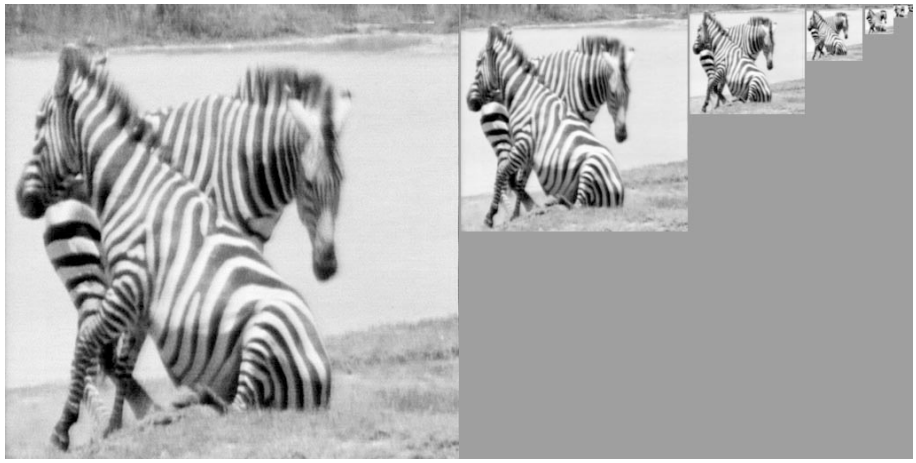


Bild 17.3: Gauß-Pyramide: Die Schichten $\mathbf{G}_0$ bis $\mathbf{G}_6$ eines Testbildes mit einer Größe von $512 \cdot 512$ Bildpunkten.

dass jede Stufe auf die Größe des Originals gebracht werden kann. Dieser Formalismus kann folgendermaßen beschrieben werden:

$$\begin{aligned}
&\mathbf{G}_{i,1} = \text{EXPAND}(\mathbf{G}_{i,0});\ \text{mit}\ \mathbf{G}_{i,0} = \mathbf{G}_i, \quad i = r, r-1, ..., 1; \\
&\mathbf{G}_{i,2} = \text{EXPAND}(\mathbf{G}_{i,1}); \\
&... \\
&\mathbf{G}_{i,i} = \text{EXPAND}(\mathbf{G}_{i,i-1}).
\end{aligned} \tag{17.5}$$

Die Bilder mit den Bezeichnungen $\mathbf{G}_{1,1}$, $\mathbf{G}_{2,2}$ bis $\mathbf{G}_{r,r}$ sind dann auf die Größe des Originalbildes expandiert. Wie der EXPAND-Operator genau funktioniert, wird im nächsten Abschnitt erläutert. In der Bildfolge 17.4 sind die Schichten $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$ und $\mathbf{G}_5$ jeweils auf die Originalgröße expandiert dargestellt. Man sieht, dass in der Schicht $\mathbf{G}_3$ das Streifenmuster bereits undeutlich wird.

In den Schichten der *Laplace-Pyramide* werden nun die Bildanteile gespeichert, die durch die REDUCE-Operation bei der Bildung der Gauß-Pyramide herausgefiltert wurden. Der Schritt von der Gauß-Pyramide zur Laplace-Pyramide besteht somit in einer Differenzbildung der einzelnen Schichten $\mathbf{G}_0$ bis $\mathbf{G}_r$. Dabei tritt allerdings das Problem auf, dass z.B. die Schichten $\mathbf{G}_0$ und $\mathbf{G}_1$ in der Größe nicht zusammenpassen, da ja $\mathbf{G}_1$ flächenmäßig viermal kleiner ist als $\mathbf{G}_0$. $\mathbf{G}_1$ muss somit zuerst durch die bereits oben verwendete EXPAND-Operation auf die Größe von $\mathbf{G}_0$ gebracht werden. Wenn die unterste Schicht der Laplace-Pyramide mit $\mathbf{L}_0$ bezeichnet wird, kann dieser Verarbeitungsschritt durch $\mathbf{L}_0 = \mathbf{G}_0 - \text{EXPAND}(\mathbf{G}_1)$ beschrieben werden. Da $\mathbf{G}_1$ aus $\mathbf{G}_0$ u.a. durch die Anwendung eines Glättungsoperators hervorgegangen ist, wird das Differenzbild $\mathbf{L}_0$ die Information enthalten, die durch die Glättung verloren ging, also die Bildkanten. Die gesamte Laplace-Pyramide mit den Schichten $\mathbf{L}_0$ bis $\mathbf{L}_r$ wird folgendermaßen berechnet:

$$\begin{aligned}
&\mathbf{L}_i = \mathbf{G}_i - \text{EXPAND}(\mathbf{G}_{i+1}) = \mathbf{G}_{i,0} - \mathbf{G}_{i+1,1}; i = 0, 1, ..., r-1; \\
&\mathbf{L}_r = \mathbf{G}_r.
\end{aligned} \tag{17.6}$$

Die Spitze der Laplace-Pyramide $\mathbf{L}_r$ wurde in (17.6) mit der Spitze der Gauß-Pyramide $\mathbf{G}_r$ gleichgesetzt. Damit ist formal der Aufbau der Laplace-Pyramiden beschrieben. Ein Blockdiagramm, das den Aufbau einer Laplace-Pyramide grafisch verdeutlicht, zeigt Bild 17.5.

Wie die Gauß-Pyramide ist die Laplace-Pyramide eine Darstellung von verschiedenen Bereichen der Wellenzahlindizes. Als Beispiel zeigt die Bildfolge 17.6 die zu den Bildern 17.3 und 17.4 gehörige expandierte Laplace-Pyramide. Eine Schicht der Laplace-Pyramide enthält gerade die Frequenzen, die bei der Bildung der Gauß-Pyramide herausgefiltert wurden. In der folgenden Tabelle sind die Bereiche der Wellenzahlindizes für ein $512 \cdot 512$–Bild angegeben:

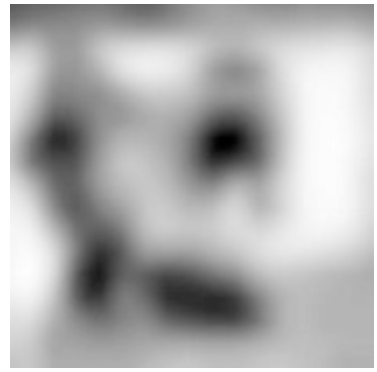
(a) Schicht $\mathbf{G}_0$

(b) Schicht $\mathbf{G}_{1,1}$

(c) Schicht $\mathbf{G}_{2,2}$

(d) Schicht $\mathbf{G}_{3,3}$

(e) Schicht $\mathbf{G}_{4,4}$



(f) Schicht $\mathbf{G}_{5,5}$

Bild 17.4: Expandierte Schichten einer Gauß-Pyramide. (a) bis (f): Schichten $\mathbf{G}_0$ (Original) bis $\mathbf{G}_{5,5}$. In der Schicht $\mathbf{G}_{4,4}$ sind die Streifen der Zebras nicht mehr zu sehen, während grobe Bildstrukturen noch deutlich hervortreten.

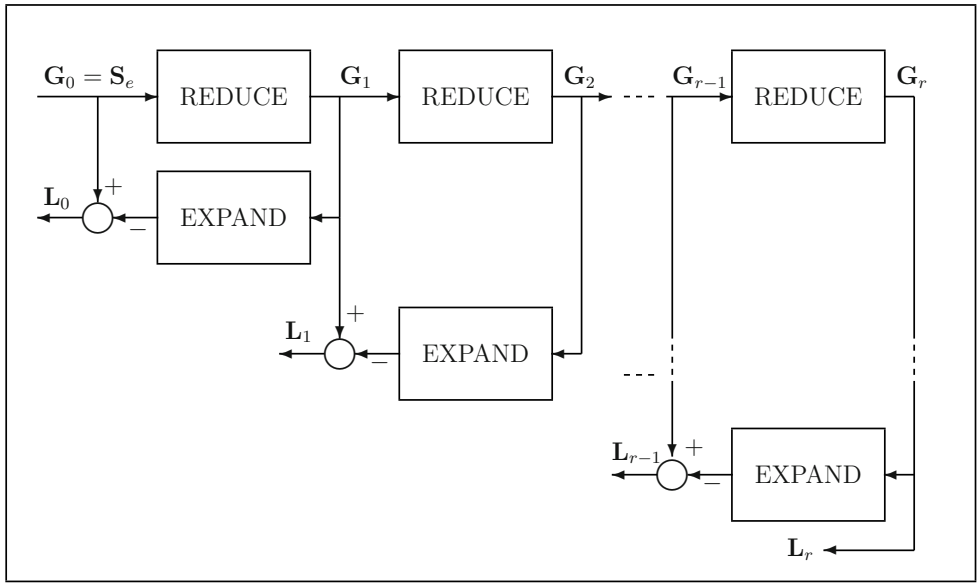


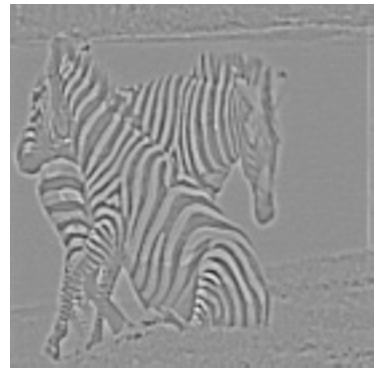
Bild 17.5: Blockdiagramm zum Aufbau einer Laplace-Pyramide.

Stufe	Zeilen/Spalten	Wellenzahlindizes
$\mathbf{L}_0$	$2^9 + 1 = 513$	129-256
$\mathbf{L}_1$	$2^8 + 1 = 257$	65-128
$\mathbf{L}_2$	$2^7 + 1 = 129$	33-64
$\mathbf{L}_3$	$2^6 + 1 = 65$	17-32
$\mathbf{L}_4$	$2^5 + 1 = 33$	9-16
$\mathbf{L}_5$	$2^4 + 1 = 17$	5-8
$\mathbf{L}_6$	$2^3 + 1 = 9$	3-4
$\mathbf{L}_7$	$2^2 + 1 = 5$	1-2
$\mathbf{L}_8$	$2^1 + 1 = 3$	1
$\mathbf{L}_9$	$2^0 + 1 = 2$	-

17.6 Rekonstruktion des Originalbildes

Aus der Laplace-Pyramide kann das Originalbild $\mathbf{S}_e$ wieder rekonstruiert werden. Dies ist bei Anwendungen von Bedeutung, bei denen die einzelnen Schichten der Laplace-Pyramide bestimmten Verarbeitungsschritten unterzogen werden. Das Ergebnis dieser Verarbeitung ist dann das rücktransformierte Bild der möglicherweise veränderten Laplace-Pyramide.

(a) Schicht $\mathbf{L}_0$

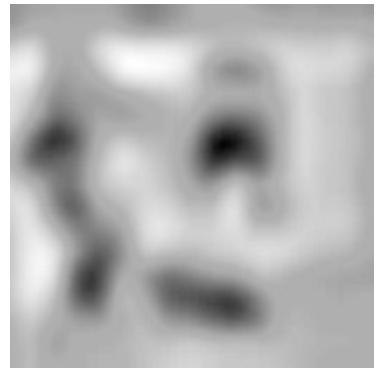


(b) Schicht $\mathbf{L}_{1,1}$

(c) Schicht $\mathbf{L}_{2,2}$

(d) Schicht $\mathbf{L}_{3,3}$

(e) Schicht $\mathbf{L}_{4,4}$



(f) Schicht $\mathbf{L}_{5,5}$

Bild 17.6: Expandierte Schichten einer Laplace-Pyramide. Die Bilder wurden um den Grauwert 127 angehoben und skaliert. (a) - (f) Schichten $\mathbf{L}_0$ bis $\mathbf{L}_{5,5}$. Die Streifen haben einen Wellenzahlindex im Bereich von 36. In Teilbild (c) treten sie am deutlichsten hervor. Die Streifen mit geringerem Wellenzahlindex sind in Teilbild (d) deutlich zu sehen. Niederfrequente Bildstrukturen sind in Teilbild (f) zu sehen.

Beispiele dazu werden z.B. in Abschnitt 17.10.1 gegeben. Im Folgenden wird nur die Rücktransformation einer Laplace-Pyramide zum originalen Grauwertbild beschrieben.

Zur Rücktransformation wird die Spitze der Laplace-Pyramide auf die Größe der darunterliegenden Schicht expandiert und zu dieser Schicht addiert. Dann wird diese Schicht expandiert und zur nächsten Schicht addiert. Das wird fortgesetzt, bis die Größe des Originals erreicht ist. Formal sieht die Rücktransformation folgendermaßen aus:

$$\begin{aligned} &\mathbf{S}_r = \mathbf{L}_r; \\ &\mathbf{S}_{i-1} = \mathbf{L}_{i-1} + \text{EXPAND}(\mathbf{S}_i); \quad i = r, r-1, ..., 1; \\ &\mathbf{S}_e = \mathbf{S}_0. \end{aligned} \tag{17.7}$$

Dieser formale Sachverhalt ist in Bild 17.7 als Blockdiagramm dargestellt.

Bei der Rücktransformation werden alle Bildinformationen, die beim Aufbau der Gauß-Pyramide herausgefiltert wurden, wieder zum Gesamtbild zusammengesetzt. Dies wird am Beispiel einer Laplace-Pyramide mit drei Stufen erläutert. Zunächst die Konstruktion der Laplace-Pyramide:

$$\begin{aligned} &\mathbf{L}_0 = \mathbf{S}_0 - \text{EXPAND}(\mathbf{S}_1); \\ &\mathbf{L}_1 = \mathbf{S}_1 - \text{EXPAND}(\mathbf{S}_2); \\ &\mathbf{L}_2 = \mathbf{S}_2. \end{aligned} \tag{17.8}$$

Die Rücktransformation:

$$\begin{aligned} &\mathbf{S}_e = \mathbf{S}_0; \\ &\mathbf{S}_0 = \mathbf{L}_0 + \text{EXPAND}\Big(\mathbf{L}_1 + \text{EXPAND}(\mathbf{L}_2)\Big). \end{aligned} \tag{17.9}$$

Substitution der einzelnen Ausdrücke:

$$\begin{aligned} &\mathbf{S}_0 = \mathbf{L}_0 + \text{EXPAND}\Big(\mathbf{L}_1 + \text{EXPAND}(\mathbf{S}_2)\Big); \\ &\mathbf{S}_0 = \mathbf{L}_0 + \text{EXPAND}\Big(\Big(\mathbf{S}_1 - \text{EXPAND}(\mathbf{S}_2)\Big) + \text{EXPAND}(\mathbf{S}_2)\Big); \\ &\mathbf{S}_0 = \mathbf{L}_0 + \text{EXPAND}(\mathbf{S}_1); \\ &\mathbf{S}_0 = \Big(\mathbf{S}_0 - \text{EXPAND}(\mathbf{S}_1)\Big) + \text{EXPAND}(\mathbf{S}_1); \\ &\mathbf{S}_0 = \mathbf{S}_0. \end{aligned} \tag{17.10}$$

Mit der Struktur einer Laplace-Pyramide liegt ein Bild in Schichten vor, die jeweils ganz bestimmte Ausschnitte des Spektrums des Originals enthalten. Die Verarbeitung von Bilddatenstrukturen dieser Art wird deshalb im englischen Sprachgebrauch als *multiresolution image processing* bezeichnet. Bildverarbeitungstechniken können gut in der richtigen

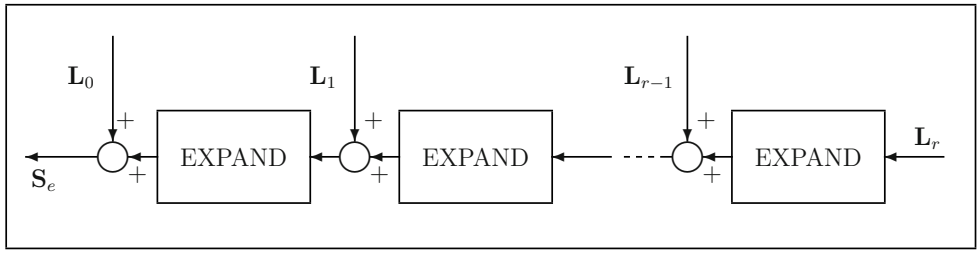


Bild 17.7: Blockdiagramm zur Rücktransformation einer Laplace-Pyramide.

Auflösungsstufe angewendet werden. Das ist diejenige Schicht der Pyramide, in der die interessierende Struktur gerade noch aufgelöst wird. Da die Bildgröße der Schichten in Richtung Spitze der Pyramide stark abnimmt, benötigt die gesamte Pyramide nur etwa $\frac{1}{3}$ mehr Speicherplatz als das Original in herkömmlicher Rasterdarstellung.

Zu bemerken wäre noch, dass bei praktischen Anwendungen nicht immer alle Pyramidenschichten von $\mathbf{L}_0$ bis $\mathbf{L}_r$ berechnet werden müssen. Wenn man z.B. weiß, dass sich die interessierenden Bildstrukturen am stärksten in den Schichten $\mathbf{L}_0$ bis $\mathbf{L}_3$ ausprägen, so kann man auf die weiteren Schichten verzichten. Die Schicht $\mathbf{L}_4$ ist dann identisch mit der Schicht $\mathbf{G}_4$ der Gauß-Pyramide.

17.7 Implementierung des REDUCE-Operators

Nachdem im vorhergehenden Abschnitt die grundlegenden Fragen zum Aufbau der Gauß- und Laplace-Pyramiden erläutert wurden, folgen hier einige Hinweise, die für die Implementierung wichtig sind.

Zunächst wird der in Abschnitt 17.4 verwendete REDUCE-Operator genauer untersucht. Beim Aufbau der Gauß-Pyramide wird eine höhere Schicht aus der darunterliegenden Schicht durch eine Glättung der Grauwerte und eine Verkleinerung gebildet. Der Glättungsoperator verwendet einen Filterkern $\mathbf{H} = (h(u,v))$ mit z.B. $5 \cdot 5$ Elementen. Die Reduzierung der Bildgröße wird dadurch erreicht, dass nur jeder zweite Bildpunkt der unteren Schicht zum Mittelpunkt der Glättungsoperation wird. Dadurch wird die darüberliegende Schicht in Zeilen- und Spaltenrichtung nahezu halbiert.

Mit diesen Voraussetzungen kann die REDUCE-Funktion mit folgendem Formalismus beschrieben werden:

$$\begin{aligned} (g_0(x,y)) &= \mathbf{G}_0 = \mathbf{S}_e = (s_e(x,y)); \\ g_{i+1}(x,y) &= \sum_{u=-2}^{+2} \sum_{v=-2}^{+2} h(2+u,2+v) g_i(2x+u,2y+v); \\ & \quad x = 0,...,2^{r-(i+1)}; \quad y = 0,...,2^{r-(i+1)}; \end{aligned} \tag{17.11}$$

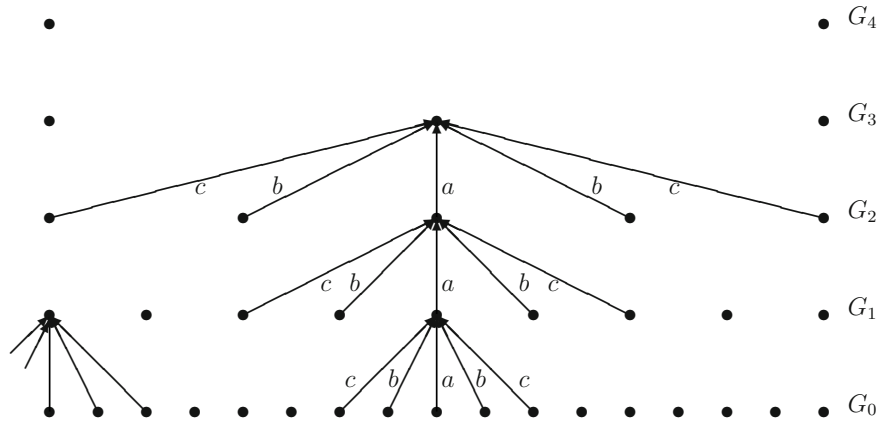


Bild 17.8: Glättungs- und Größenreduktionsoperationen beim Aufbau einer Gauß-Pyramide am Beispiel einer Bildzeile eines Bildes $\mathbf{G}_0 = \mathbf{S}_e$ mit der Zeilen- und Spaltenlänge $17 = (2^4 + 1)$. Am Rand liegende Bildpunkte müssen gesondert berechnet werden.

$$i = 0, 1, ..., r - 1.$$

Der Filterkern $\mathbf{H}$ soll einige Bedingungen erfüllen:

- $(h(u, v))$ ist *separabel*: $(h(u, v)) = (\hat{h}(u)) \cdot (\hat{h}(v))$. Diese Forderung wird vor allem aus programmiertechnischen Gründen aufgestellt, da die Bilder dann zeilen- und spaltenweise getrennt verarbeitet werden können.
- $(h(u, v))$ (und damit auch $(\hat{h})$) ist *symmetrisch*: $(\hat{h}(u)) = (\hat{h}(v)) = (c \quad b \quad a \quad b \quad c)$.
- $(\hat{h})$ ist *normiert*: $a + 2b + 2c = 1$.
- Jeder Bildpunkt einer Schicht i leistet zu den Bildpunkten der Schicht $i + 1$ denselben Beitrag, obwohl nur jeder zweite Bildpunkt zentraler Bildpunkt der Glättungsoperation wird. Diese Forderung hängt mit der Reduzierung der Auflösung bei der REDUCE-Operation zusammen. Da jeder gerade Bildpunkt einmal zentraler Punkt (Gewicht a) und zweimal Randpunkt (Gewicht c) ist und jeder ungerade Punkt zweimal mit dem Gewicht b eingeht, muss gelten: $a + 2c = 2b$.

Die letzte Forderung ist in Bild 17.8 schematisch dargestellt. Hier sind die Glättungs- und Größenreduktionsoperationen beim Aufbau einer Gauß-Pyramide am Beispiel einer Bildzeile eines Bildes $\mathbf{G}_0 = \mathbf{S}_e$ mit der Zeilen- und Spaltenlänge $17 = (2^4 + 1)$ verdeutlicht.

Die Zeilen- und Spaltenlängen der Schichten $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$ und $\mathbf{G}_4$ sind 9, 5, 3 und 2. Am Rand liegende Bildpunkte müssen gesondert berechnet werden. Dies wird weiter unten erläutert.

Aus den Forderungen an $\hat{h}$ können die Beziehungen zwischen den Parametern a, b und c leicht berechnet werden:

$$a = \text{freie Variable}; \quad b = \frac{1}{4}; \quad c = \frac{1}{4} - \frac{1}{2}a. \tag{17.12}$$

Beispiele für Filterkerne sind das *Binomialfilter* mit $a = \frac{3}{8}$

$$(\hat{h}) = \frac{1}{16}\Big(1 \quad 4 \quad 6 \quad 4 \quad 1\Big), \tag{17.13}$$

und das *Gauß-Filter* mit $a = \frac{2}{5}$

$$(\hat{h}) = \frac{1}{20}\Big(1 \quad 5 \quad 8 \quad 5 \quad 1\Big). \tag{17.14}$$

Weitere Bemerkungen zur Wahl des freien Parameters a sind in Abschnitt 17.9 zusammengefasst.

Unter Verwendung der Separabilität von $(h(u,v)) = (\hat{h}(u)) \cdot (\hat{h}(v))$ werden die Glättung und die Reduktion zunächst über alle Bildzeilen und dann über alle Bildspalten durchgeführt. Es ergeben sich folgende Formeln:

$$\begin{aligned}
&\mathbf{G}_0 = \mathbf{S}_e; && && (17.15)\\
&\tilde{g}_{i+1}(x,y) &&= \sum_{v=-2}^{+2} \hat{h}(2+v) g_i(x, 2y+v);\\
& && y = 0, ..., 2^{r-(i+1)};\\
& && x = 0, ..., 2^{r-i};\\
&g_{i+1}(x,y) &&= \sum_{u=-2}^{+2} \hat{h}(2+u) \tilde{g}_{i+1}(2x+u, y);\\
& && x = 0, ..., 2^{r-(i+1)};\\
& && y = 0, ..., 2^{r-(i+1)};\\
& && i = 0, 1, ..., r-1.
\end{aligned}$$

Wie bereits oben erwähnt und in Bild 17.8 angedeutet, müssen die Randpunkte der einzelnen Schichten gesondert behandelt werden. Da in der hier beschriebenen Implementierung $5 \cdot 5$–Filterkerne verwendet werden, ist der Rand jeweils 2 Pixel breit. Die außerhalb des Bildes liegenden Pixel werden extrapoliert und zwar für die Zeilen

$$\begin{aligned} g_i(-1,y) &= 2g_i(0,y) - g_i(1,y), \\ g_i(-2,y) &= 2g_i(0,y) - g_i(2,y) \end{aligned} \tag{17.16}$$

und für die Spalten

$$\begin{aligned} g_i(x,-1) &= 2g_i(x,0) - g_i(x,1), \\ g_i(x,-2) &= 2g_i(x,0) - g_i(x,2). \end{aligned} \tag{17.17}$$

Bei dieser Extrapolation bleibt die erste Ableitung am Bildrand konstant und die zweite Ableitung wird null. Dadurch werden aber Veränderungen im Grauwertverlauf an den Rändern bei höheren Pyramidenstufen verstärkt, was zu Randverfälschungen führt. Dazu wird eine Extrapolation vorgeschlagen, bei der auch die erste Ableitung null ist. Für die Spalten erhält man dann:

$$\begin{aligned} g_i(x,-1) &= g_i(x,0); \\ g_i(x,-2) &= g_i(x,0). \end{aligned} \tag{17.18}$$

Die Extrapolationen für den rechten und den unteren Bildrand lauten sinngemäß.

17.8 Implementierung des EXPAND-Operators

Mit der EXPAND-Operation wird eine Schicht auf die Größe der darunterliegenden Schicht ausgedehnt. Dazu müssen die Zeilen- und die Spaltenanzahl verdoppelt werden, was zur Folge hat, dass Bildpunkte interpoliert werden müssen. Da die EXPAND-Operation als inverse Operation von REDUCE aufgefasst werden kann, wird ebenfalls die Filtermaske $\mathbf{H} = (h(u,v))$, hier allerdings zur Interpolation, verwendet. Da bei der REDUCE-Operation nur die Bildpunkte mit geradzahligem Zeilen-/Spaltenindex zentrale Punkte bei der Filterung werden (siehe Bild 17.8), werden sie in zwei Teilmengen aufgeteilt: Einmal die Bildpunkte, die mit dem Gewicht b in die Glättung eingehen, und zum anderen diejenigen, die mit den Gewichten a und c berücksichtigt werden.

Diesem Sachverhalt wird bei der EXPAND-Operation dadurch Rechnung getragen, dass bei der Interpolation die geradzahligen Positionen einer vergrößerten Zeile (Spalte) aus drei Bildpunkten der Ausgangszeile (-spalte) mit den Gewichten a und c interpoliert werden und die ungeraden Positionen nur aus zwei Bildpunkten mit dem Gewicht b. Dies ist in Bild 17.9 veranschaulicht. Hier wurde bereits berücksichtigt, dass auch die EXPAND-Operation separabel ist.

Durch diese Vorgehensweise wird allerdings die Forderung nicht aufrechterhalten, dass auch die EXPAND-Operation normiert ist, denn es gilt nach Abschnitt 17.4

$$a + 2b + 2c = 1. \tag{17.19}$$

Außerdem gilt wegen der vierten Filterbedingung:

$$a + 2c = 2b. \tag{17.20}$$

Daraus lässt sich leicht ableiten:

$$a + 2c = \frac{1}{2}, \tag{17.21}$$

$$2b = \frac{1}{2}.$$

Das heißt aber, dass man die Gewichtung bei der Interpolation in Zeilen- und Spaltenrichtung verdoppeln muss, was bei dem nicht separierten Filterkern einen Faktor vier und bei der separierten Form jeweils den Faktor zwei bedeutet. Für die folgenden Formeln wird die Notation von (17.5) verwendet:

$$\begin{aligned} g_{i,k+1}(x,y) &= 4 \sum_{u=-2}^{+2} \sum_{v=-2}^{+2} h(2+u, 2+v) g_{i,k}\Big(\frac{x+u}{2}, \frac{y+v}{2}\Big); \\ & \quad x = 0, ..., 2^{r-(i-1)+k}; \\ & \quad y = 0, ..., 2^{r-(i-1)+k}; \\ & \quad i = 1, ..., r; \\ & \quad k = 0, 1, ..., i. \end{aligned} \tag{17.22}$$

Die beiden separierten Formeln lauten:

$$\begin{aligned} \tilde{g}_{i,k+1}(x,y) &= 2 \sum_{v=-2}^{+2} \hat{h}(2+v) g_{i,k}\Big(x, \frac{y+v}{2}\Big); \\ & \quad y = 0, ..., 2^{r-(i-1)+k}; \\ & \quad x = 0, ..., 2^{r-i+k}; \\ g_{i,k+1}(x,y) &= 2 \sum_{u=-2}^{+2} \hat{h}(2+u) \tilde{g}_{i,k+1}\Big(\frac{x+u}{2}, y\Big); \\ & \quad x = 0, ..., 2^{r-(i-1)+k}; \\ & \quad y = 0, ..., 2^{r-(i-1)+k}; \\ & \quad i = 1, ..., r; \\ & \quad k = 0, 1, ..., i. \end{aligned} \tag{17.23}$$

In (17.22) und (17.23) werden nur diejenigen Positionen verwendet, bei denen sich für den Zeilen- und Spaltenzähler ganzzahlige Werte ergeben. Dadurch wird die oben erwähnte Klasseneinteilung der Bildpunkte bei der REDUCE-Operation berücksichtigt.

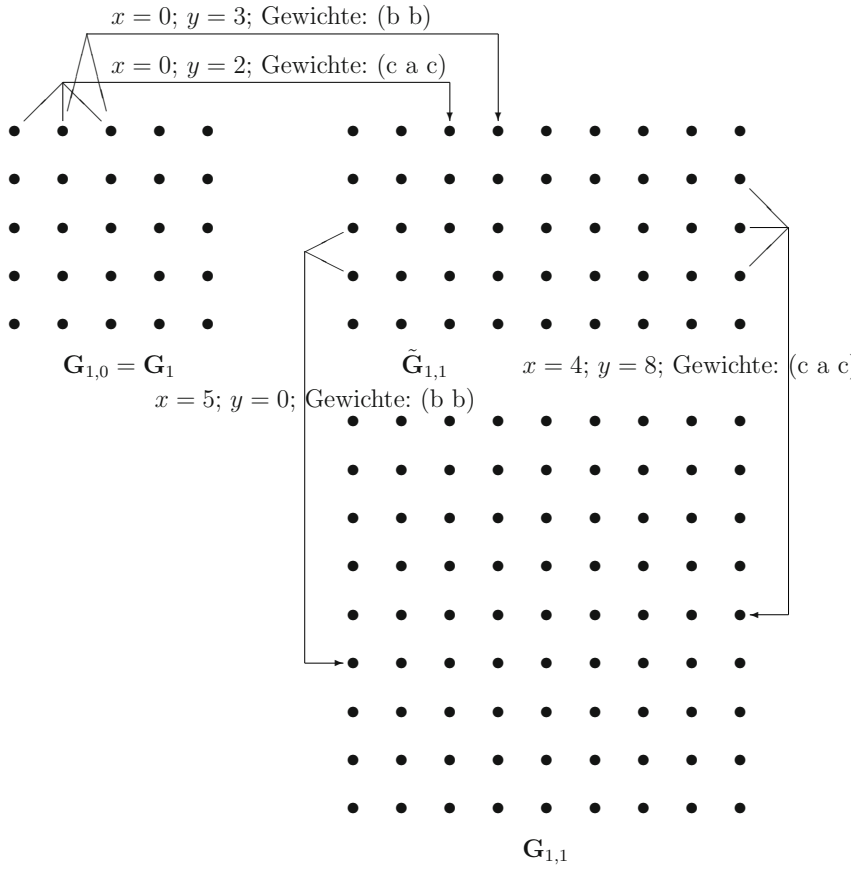


Bild 17.9: EXPAND-Schritt von $\mathbf{G}_1$, über das Hilfsbild $\tilde{\mathbf{G}}_{1,1}$ zur Schicht $\mathbf{G}_{1,1}$. Für das Beispiel wurde $r = 3$ gewählt.

Die Formeln (17.23) werden im Folgenden anhand eines Beispiels erläutert. Schematisch ist dieses Beispiel in Bild 17.9 dargestellt. Es wird ein Eingabebild $\mathbf{S}_e$ der Größe $9 \cdot 9$ verwendet. Also ist $r = 3$. Die Schichten der Gauß-Pyramide sind $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$ und $\mathbf{G}_3$, mit den Zeilen- und Spaltenlängen 9, 5, 3 und 2.

Hier wird nun der EXPAND-Schritt von $\mathbf{G}_{1,0} = \mathbf{G}_1$, über das Hilfsbild $\tilde{\mathbf{G}}_{1,1}$ zum Bild $\mathbf{G}_{1,1}$ erläutert. Gemäß Formel (17.23) wird zur Berechnung der $\tilde{g}_{1,1}(x, y)$-Werte für eine Bildzeile x über die Spalten y interpoliert. So werden beispielsweise für $x = 0$ und $y = 2$ folgende Rasterpositionen berechnet:

$$(0,0),\ (0,\tfrac{1}{2}),\ (0,1),\ (0,\tfrac{3}{2}),\ (0,2)$$

Für $x = 0$ und $y = 3$ ergeben sich:

$$(0,\tfrac{1}{2}),\ (0,1),\ (0,\tfrac{3}{2}),\ (0,2),\ (0,\tfrac{5}{2})$$

Zur Interpolation werden nur die Rasterpositionen mit ganzzahligen Werten verwendet. Somit gehen in die Summation für $x = 0$ und $y = 2$ die ersten drei Pixel von $\mathbf{G}_{1,0}$ mit den Gewichten $(c \quad a \quad c)$ ein. Für $x = 0$ und $y = 3$ werden die Grauwerte in den Positionen $(0, 1)$ und $(0, 2)$ mit den Gewichten $(b \quad b)$ verwendet. Zum Ausgleich der fehlenden Positionen und zur Einhaltung der Normierungsforderung werden die gewichteten Summen mit 2 multipliziert. Sinngemäß wird im zweiten Teil der Formel (17.23) die Interpolation über die Zeilen durchgeführt. Das Ergebnis ist in diesem Beispiel ein $9 \cdot 9$–Bild $\mathbf{G}_{1,1}$, das die Größe von $\mathbf{G}_0$ besitzt.

17.9 Frequenzverhalten und Wahl des freien Parameters a

Die oben angegebenen Frequenzbereiche für die Gauß- und Laplace-Pyramiden beruhen auf theoretischen Überlegungen, basierend auf der Vorgehensweise beim Aufbau der Pyramidenstrukturen. Der tatsächliche Frequenzgehalt der einzelnen Pyramidenstufen hängt jedoch von der Filtermaske $\mathbf{H} = (h(u, v))$ ab, die sowohl bei der REDUCE- als auch bei der EXPAND-Operation verwendet wird. Da dieses Filter in der diskutierten Form ausschließlich vom Parameter a bestimmt wird, ist die richtige Wahl dieses Parameters wesentlich.

Zur Untersuchung des Filters $\mathbf{H}$ berechnet man das Frequenzspektrum der Impulsantwort. Da die Filtermaske symmetrisch ist, ist das Filter nahezu richtungsunabhängig, und es genügt zur Untersuchung ein zweidimensionaler Schnitt durch das zweidimensionale Spektrum der Impulsantwort. In [Burt84] wird für den freien Parameter der Wert $a = \frac{2}{5} = 0.4$ vorgeschlagen. Damit erhält man mit (17.12) das Gauß-Filter:

$$\begin{aligned} a &= \frac{2}{5} = 0.4 \\ b &= \frac{1}{4}; \\ c &= \frac{1}{4} - \frac{1}{2}a = \frac{1}{20} \end{aligned} \tag{17.24}$$

und

$$(\hat{h}(u)) = (\hat{h}(v)) = \frac{1}{20}(1 \quad 5 \quad 8 \quad 5 \quad 1). \tag{17.25}$$

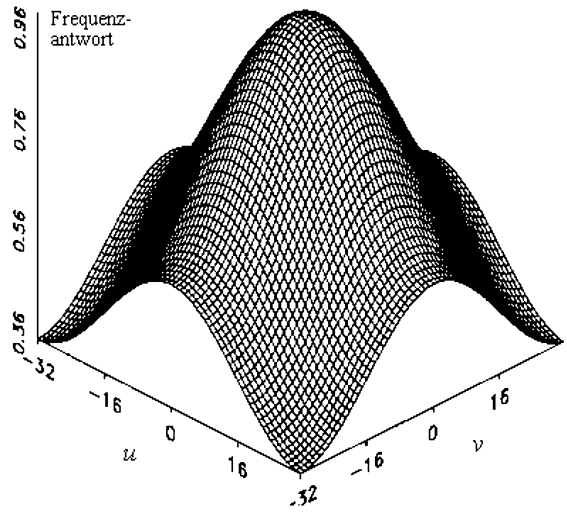
In Bild 17.10-a ist die Frequenzantwort der REDUCE-Funktion ($a = 0.4$, Gauß-Filter) gezeigt. Die Zahlenangaben für die Wellenzahlindizes beziehen sich auf eine Pyramidenstufe der Größe 129·129 Bildpunkte. In Bild 17.10-b ist ein Schnitt durch das dreidimensionale Spektrum der Impulsantwort des Gauß-Filters dargestellt. Man sieht, dass bei höheren Wellenzahlindizes die Signale stärker gedämpft werden. So ist z.B. für $u = 8$ die Durchlassrate ca. 95%, während sie für $u = 16$ nur mehr ca. 75% ist.

Da beim Aufbau der Gauß-Pyramide von Schicht zu Schicht die Zeilen- und Spaltenanzahl halbiert wird, müsste die REDUCE-Operation alle Strukturen mit Wellenzahlindizes $u > \frac{1}{4}M_{\mathbf{G}_i}$ herausfiltern, wobei $M_{\mathbf{G}_i}$ die Zeilen- (Spalten-)länge der zu reduzierenden Schicht ist. Beispielsweise müssten beim Übergang von einer Pyramidenschicht der Größe 129·129 zur nächsten Schicht der Größe 65·65 alle Wellenzahlindizes u im Bereich zwischen 33 und 64 herausgefiltert werden. Ein Vergleich mit der Frequenzantwort des Gauß-Filters zeigt, dass das reale Frequenzverhalten dem nicht entspricht: So ist z.B. bei $u = 32$ im obigen Beispiel eine Durchlassrate von immerhin noch 60% vorhanden. Dieser Sachverhalt kann bei periodischen Bildstrukturen dafür verantwortlich sein, dass bei der Rücktransformation Aliasingeffekte auftreten. Diese Untersuchungen des Frequenzverhaltens kann man nun auf die EXPAND-Funktion, die Gauß-Pyramide und die Laplace-Pyramide ausdehnen.

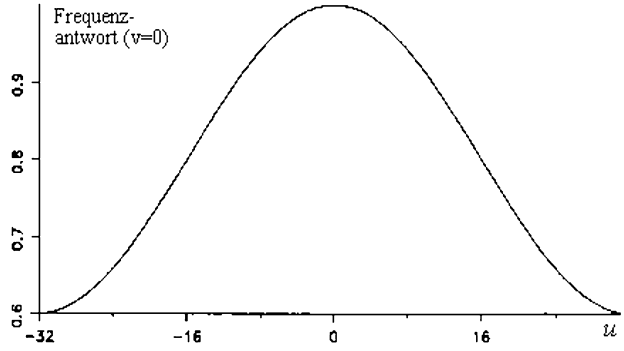
Für die Praxis ist Tabelle 17.1 sehr hilfreich. Sie beschreibt für unterschiedliche Werte von a das Filterverhalten der einzelnen Schichten der Laplace-Pyramiden. Die Tabelle ist für Bilder der Größe 512·512 ausgelegt. Sie kann jedoch leicht auf kleinere Bildgrößen übertragen werden. Die Spalte **max** enthält die maximalen Wellenzahlindizes der einzelnen Schichten. Die Spalte **75%** enthält Wellenzahlindizes, bei denen mindstestens 75% der maximalen Durchlassrate erreicht werden, die Spalten **50%** und **25%** sind sinngemäß zu interpretieren.

Wie kann Tabelle 17.1 genutzt werden? Dazu ein Beispiel, das sich auf Bild 17.6 bezieht. Verwendet wurde ein Gauß-Filter ($a = 0.4$). Die Streifen der Zebras sind etwa 18 Pixel breit. Das entspricht bei einem $512 \cdot 512$–Bild einem Wellenzahlindex von etwa $u = 14$. Das bedeutet, dass die Streifen in der Schicht $\mathbf{L}_2$ deutlich hervortreten und in $\mathbf{L}_3$ bereits gedämpft sind. Ab der Schicht $\mathbf{L}_4$ sind sie nicht mehr zu sehen. Die Bilder bestätigen dies.

Bei Verwendung von Tabelle 17.1 muss man bei einer praktischen Anwendung wissen, in welchem Frequenzbereich sich die zu untersuchende Struktur ausprägt. Anhand der Tabelle kann man dann den passenden Parameter a wählen. Außerdem gestattet es die Tabelle, die richtige Schicht der Pyramide zu finden, in der die Struktur am deutlichsten zu finden ist.



(a)



(b)

Bild 17.10: (a) Frequenzantwort (Impulsantwort) der REDUCE-Operation für das Gauß-Filter ($a = 0.4$). Die Zahlenangaben der Wellenzahlindizes beziehen sich auf eine Pyramidenstufe mit $129 \cdot 129$ Bildpunkten. (b) Schnitt durch das zweidimensionale Spektrum der Impulsantwort des Gauß-Filters. Man sieht, dass z.B. Wellenzahlindizes der Größe $u = 16$ auf etwa 75% ihres ursprünglichen Wertes gedämpft werden. Am Rand sollten die Wellenzahlindizes ganz herausgefiltert werden. Das ist aber nicht der Fall, so dass bei bestimmten periodischen Bildstrukturen Aliasingeffekte zu erwarten sind.

Parameter a	**Schicht**	**max**	**75%**	**50%**	**25%**
0.30	$\mathbf{L}_0$	196	80-256	57-256	37-256
	$\mathbf{L}_1$	49	32- 74	23- 98	16-128
	$\mathbf{L}_2$	24	16- 35	12- 44	8- 56
	$\mathbf{L}_3$	12	8- 17	6- 22	4- 28
	$\mathbf{L}_4$	6	4- 8	3- 11	2- 14
	$\mathbf{L}_5$	3	2- 4	2- 5	1- 7
	$\mathbf{L}_6$	2	1- 2	1- 2	1- 4
0.35	$\mathbf{L}_0$	211	92-256	64-256	41-256
	$\mathbf{L}_1$	55	35- 94	26-128	17-128
	$\mathbf{L}_2$	26	17- 41	13- 55	9- 64
	$\mathbf{L}_3$	13	9- 20	7- 26	5- 32
	$\mathbf{L}_4$	7	5- 10	4- 13	2- 16
	$\mathbf{L}_5$	3	3- 5	2- 6	1- 8
	$\mathbf{L}_6$	2	2- 2	1- 3	1- 4
0.40	$\mathbf{L}_0$	256	112-256	76-256	48-256
	$\mathbf{L}_1$	68	40-128	29-128	19-128
	$\mathbf{L}_2$	31	19- 64	14- 64	9- 64
	$\mathbf{L}_3$	15	10- 30	7- 32	5- 32
	$\mathbf{L}_4$	7	5- 14	4- 16	3- 16
	$\mathbf{L}_5$	4	3- 7	2- 8	2- 8
	$\mathbf{L}_6$	2	2- 3	1- 4	1- 4
0.45	$\mathbf{L}_0$	256	144-256	97-256	61-256
	$\mathbf{L}_1$	128	63-128	42-128	26-128
	$\mathbf{L}_2$	64	29- 64	20- 64	13- 64
	$\mathbf{L}_3$	32	14- 32	10- 32	6- 32
	$\mathbf{L}_4$	16	7- 16	5- 16	3- 16
	$\mathbf{L}_5$	8	4- 8	3- 8	2- 8
	$\mathbf{L}_6$	4	2- 4	2- 4	1- 4

Tabelle 17.1: Filterverhalten der einzelnen Schichten der Laplace-Pyramide für unterschiedliche Werte des Parameters a.

17.10 Anwendungsbeispiele zu den Laplace-Pyramiden

17.10.1 Verwendung einzelner Schichten

Eine einfache Verwendung der Laplace-Pyramiden ist die Verarbeitung einzelner Schichten. Da jede Schicht bestimmte Wellenzahlindexbereiche enthält, kann man gezielt eine Schicht auswählen und versuchen, die interessierenden Strukturen in dieser Schicht zu entdecken und weiterzuverarbeiten. Dabei ist es möglich, entweder in der größenreduzierten Form die weiteren Verarbeitungsschritte anzuwenden oder die Schicht durch die EXPAND-Operation auf die Größe des Originals zu bringen.

Auch die Verwendung mehrerer Schichten kann sinnvoll sein. Dies wird anhand eines Beispiels erläutert: Bei einer konkreten Anwendung könnten die Wellenzahlindexintervalle 5 bis 16 und 33 bis 64 die interessierenden Bildstrukturen enthalten. Man wird dann mit der Schicht $\mathbf{L}_5$ beginnen und sie auf die Größe der darunter liegenden Schicht expandieren:

$$\mathbf{L}_{5,1} = \text{EXPAND}(\mathbf{L}_{5,0}), \text{ mit } \mathbf{L}_{5,0} = \mathbf{L}_5. \tag{17.26}$$

Dazu wird jetzt $\mathbf{L}_{4,0}$ addiert:

$$\tilde{\mathbf{L}}_{4,0} = \mathbf{L}_{4,0} + \mathbf{L}_{5,1}. \tag{17.27}$$

Dieses Ergebnis muss jetzt zweimal expandiert werden, dann wird die Schicht $\mathbf{L}_2$ dazu addiert:

$$\tilde{\mathbf{L}}_{2,0} = \tilde{\mathbf{L}}_{4,2} + \mathbf{L}_{2,0}. \tag{17.28}$$

Das so entstandene Bild enthält nun die gewünschten Wellenzahlindexbereiche. Es kann bei Bedarf abschließend auf die Größe des Originals expandiert werden. Allerdings ist zu beachten, dass durch das reale Frequenzverhalten beim Aufbau der Laplace-Pyramide die Wellenzahlindexbereiche nicht so exakt ausgewählt werden können, wie es in diesem Beispiel gemacht wurde. Hilfreich kann dabei Tabelle 17.1 sein.

17.10.2 Mosaicing

In manchen Anwendungsbereichen ist es notwendig, den Bildinhalt verschiedener Originalbilder in einem Ergebnisbild zusammenzufassen. Dieser Vorgang wird *mosaicing* genannt. Beispiele dazu sind Fotomontagen im grafischen Gewerbe, Phantombilder, bei denen Gesichter aus Teilstücken zusammengesetzt werden oder architektonische Aufnahmen, mit denen die Verträglichkeit von baulichen Veränderungen mit der Umgebung vor der Durchführung der Baumaßnahmen beurteilt werden soll.

Als beispielhafte Problemstellung soll zunächst das Zusammenfügen von zwei Bildhälften, linke Hälfte vom ersten Bild und rechte Hälfte vom zweiten Bild, dienen. Werden die beiden Hälften ohne weitere Verarbeitung nur zusammengefügt, so wird immer eine Nahtstelle zu sehen sein.

Eine Mittelung entlang der beiden Übergangsbereiche wäre ein nächster Lösungsversuch, der sich aber ebenfalls als nicht ausreichend herausstellt, da die Breite des Übergangsbereichs abhängig vom Bildinhalt ist: Bei sehr fein strukturierten Bildinformationen muss der Übergangsbereich kleiner gewählt werden als bei grob strukturierten. Diese Wahl mag bei Einzelproblemen möglich sein, es lässt sich daraus aber kaum ein allgemeines Verfahren ableiten. Außerdem wird bei dieser Vorgehensweise die Schnittkante nur unschärfer und fällt dadurch lediglich etwas weniger auf.

Mit den Laplace-Pyramiden hat man eine Datenstruktur, die hervorragend zur Lösung dieser Problemstellung geeignet ist. Der Lösungsweg sieht wie folgt aus: Zu den beiden Eingabebildern $\mathbf{S}_{e1}$ und $\mathbf{S}_{e2}$ werden die Laplace-Pyramiden $\mathbf{L}^{(e1)}$ und $\mathbf{L}^{(e2)}$ berechnet. Aus diesen beiden Laplace-Pyramiden wird nun ausgehend von der Spitze eine neue Laplace-Pyramide $\mathbf{L}^{(a)}$ aufgebaut, deren linke Hälfte aus den Schichten von $\mathbf{L}^{(e1)}$ gebildet wird und deren rechte Hälfte von $\mathbf{L}^{(e2)}$ stammt. Die mittlere Bildspalte nimmt eine Sonderstellung ein: Ihre Grauwerte ergeben sich durch Mittelung der entsprechenden Bildspalten der beiden Eingabepyramiden. Im Einzelnen sehen diese Operationen wie folgt aus:

$$l_i^{(a)}(x,y) = \begin{cases} l_i^{(e1)}(x,y), & \text{falls } (x,y) \text{ in der linken Bildhälfte liegt,} \\ \frac{l_i^{(e1)}(x,y)+l_i^{(e2)}(x,y)}{2}, & \text{falls } (x,y) \text{ im Übergangsbereich liegt,} \\ l_i^{(e2)}(x,y), & \text{falls } (x,y) \text{ in der rechten Bildhälfte liegt.} \end{cases} \tag{17.29}$$

Der Index i läuft hier über die Schichten der Pyramiden, also von r bis 0. Abschließend wird aus $\mathbf{L}^{(a)}$ mit dem in (17.7) angegebenen Verfahren das Ergebnisbild $\mathbf{S}_a$ erzeugt. Da bei der Umwandlung der Laplace-Pyramide $\mathbf{L}^{(a)}$ in das Ausgabebild $\mathbf{S}_a$ von der Spitze ausgehend die einzelnen Schichten expandiert und akkumuliert werden, werden dadurch hohe und niedere Ortsfrequenzen beeinflusst, sodass im zusammengefügten Bild keine Nahtstelle zu sehen ist.

Die Bildfolge 17.11 zeigt ein Beispiel dazu: Oben sind die beiden Originale abgebildet, und in den unteren Bildern wurde jeweils eine Hälfte des linken oberen Bildes mit einer Hälfte des rechten oberen Bildes zusammengefügt.

Das Zusammenfügen einer linken und rechten Bildhälfte wurde hier nur als einfaches Beispiel zur Erläuterung der Vorgehensweise verwendet. In praktischen Anwendungen tritt dagegen das Problem auf, dass von den beiden Eingabebildern beliebige Bildausschnitte zusammenzufügen sind. Auch diese allgemeinere Problemstellung, die das oben geschilderte, einfache Beispiel beinhaltet, lässt sich mit der Datenstruktur der Laplace-Pyramiden lösen.

Die Vorgehensweise ist wie folgt: Die beiden Eingabebilder werden wieder mit $\mathbf{S}_{e1}$ und $\mathbf{S}_{e2}$ und das Ausgabebild mit $\mathbf{S}_a$ bezeichnet. Zusätzlich wird ein Maskenbinärbild $\mathbf{S}_m$ benötigt, mit dem angegeben wird, welche Bildteile aus $\mathbf{S}_{e1}$ und welche aus $\mathbf{S}_{e2}$ in das Ausgabebild $\mathbf{S}_a$ zu übernehmen sind. Dieses Maskenbild muss vor dem Vorgang des Zusammenfügens von Bildteilen erstellt werden. Man kann dazu handgezeichnete Skizzen digitalisieren oder computerunterstützte Zeichenprogramme verwenden. Auch binarisierte Bilder sind als Masken geeignet. Das Ergebnis muss ein Binärbild sein, in dem z.B. die

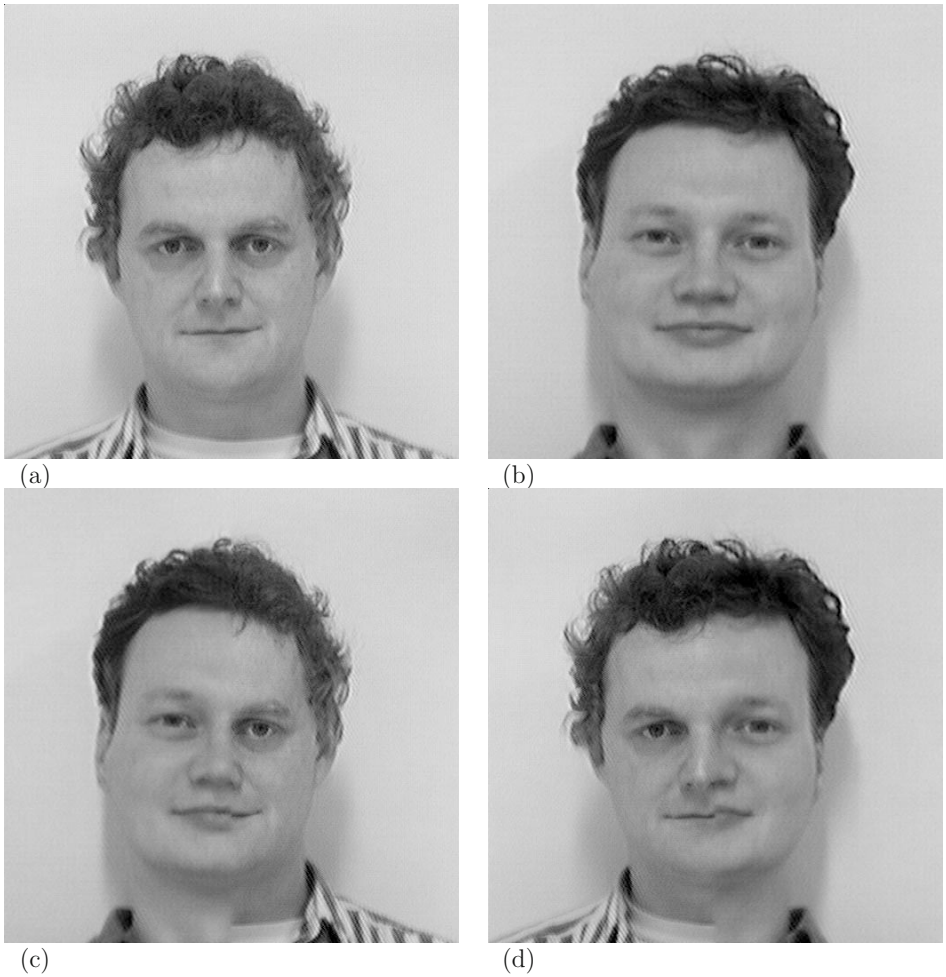


Bild 17.11: Zusammenfügen der linken und rechten Bildhälften mit Hilfe der Laplace-Pyramiden. (a) und (b) Originalbilder. (c) und (d): Über die Laplace-Pyramiden zusammengefügte Bildhälften. Die sichtbaren Ungenauigkeiten im Bereich des Mundes sind durch die unterschiedlichen Mundformen zu erklären. Hier wäre eine geometrische Korrektur vor dem Zusammenfügen sinnvoll.

Bildpunkte mit dem Grauwert 0 bzw. 1 signalisieren, dass an dieser Stelle im Ausgabebild die Information vom Eingabebild $\mathbf{S}_{e1}$ bzw. $\mathbf{S}_{e2}$ zu verwenden ist.

Zu den beiden Eingabebildern werden nun die Laplace-Pyramiden $\mathbf{L}^{(e1)}$ und $\mathbf{L}^{(e2)}$ erzeugt. Beim Aufbau der Pyramide $\mathbf{G}^{(m)}$ des Maskenbildes ist zu beachten, dass hier nur die Größenreduktion durchgeführt wird. Es entfallen im Reduktionsschritt die Tiefpassfilterung (Parameter für die Filtermaske $\mathbf{H} : a = 1, b = 0, c = 0$) und die anschließende Bildung der Laplace-Pyramide. Außerdem müssen in der Pyramide die Ränder zwischen den 0/1-Bereichen markiert werden, etwa mit dem Grauwert 2. Bei der Ermittlung der Ränder kann, je nach Problemstellung, mit 4- oder 8-Nachbarschaft gearbeitet werden.

Als Nächstes wird die Laplace-Pyramide $\mathbf{L}^{(a)}$ des Ausgabebildes berechnet. Wenn die Maskenpyramide an der gerade betrachteten Position den Wert 0 bzw. 1 aufweist, wird in die Ausgabepyramide der Grauwert der Laplace-Pyramide $\mathbf{L}^{(e1)}$ bzw. $\mathbf{L}^{(e2)}$ übernommen. Handelt es sich um einen Randpunkt (Wert 2 in der Maskenpyramide), so wird in die Ausgabepyramide der Mittelwert der beiden Grauwerte der Eingabepyramiden übernommen.

$$l_i^{(a)}(x,y) = \begin{cases} l_i^{(e1)}(x,y), & \text{falls } g_i^{(m)}(x,y) = 0, \\ \frac{l_i^{(e1)}(x,y) + l_i^{(e2)}(x,y)}{2}, & \text{falls } g_i^{(m)}(x,y) = 2, \\ l_i^{(e2)}(x,y), & \text{falls } g_i^{(m)}(x,y) = 1. \end{cases} \tag{17.30}$$

Nachdem die Ausgabepyramide $\mathbf{L}^{(a)}$ berechnet wurde, wird mit dem in (17.7) beschriebenen Verfahren das Ausgabebild $\mathbf{S}_a$ erzeugt. Die Bildfolge 17.12 zeigt Beispiele: Bild 17.12-a und 17.12-b sind die beiden Eingabebilder. Die Maskenbilder 17.12-c und 17.12-d wurden interaktiv mit einem Zeichenprogramm erzeugt. Die Bilder 17.12-e bis 17.12-h zeigen die Ergebnisse des *mosaicing* mit unterschiedlichen Kombinationen. Weitere Beispiele zum *mosaicing* enthält die Bildfolge 17.13.

Für die Behandlung der Übergangsbildpunkte im Maskenbild sind auch noch andere Varianten möglich: Falls sich die beiden Eingabebilder überlappen, kann auch ein Verfahren eingesetzt werden, bei dem die Übergangskanten im Maskenbild nicht mit einem starren Wert markiert sind, sondern stetig verlaufen. Dazu wird zum Maskenbild ebenfalls eine Laplace-Pyramide mit der üblichen Filtermaske $\mathbf{H}$ aufgebaut. Die Berechnungsvorschrift zur Bildung der Ergebnislaplace-Pyramide lautet dann:

$$l_i^{(a)}(x,y) = g_i^{(m)}(x,y) \cdot l_i^{(e1)}(x,y) + \left(1 - g_i^{(m)}(x,y)\right) \cdot l_i^{(e2)}(x,y). \tag{17.31}$$

17.10.3 Multifokus

Bei den bis jetzt beschriebenen *mosaicing*-Anwendungen wurde anhand eines Maskenbildes entschieden, welche Bildinformation in das Ausgabebild zu übernehmen ist. Bei der Multifokus-Anwendung wird anhand der Laplace-Pyramiden der beiden Eingabebilder entschieden, wie das Ausgabebild aufzubauen ist. Ein Beispiel dazu sind zwei Eingabebilder mit identischem Beobachtungsgebiet, die jedoch mit unterschiedlicher Schärfentiefe aufgezeichnet wurden. Im ersten Bild wurde z.B. auf den Vordergrund fokussiert, so dass der Hintergrund unscharf ist, während im zweiten Bild auf den Hintergrund fokussiert wurde.

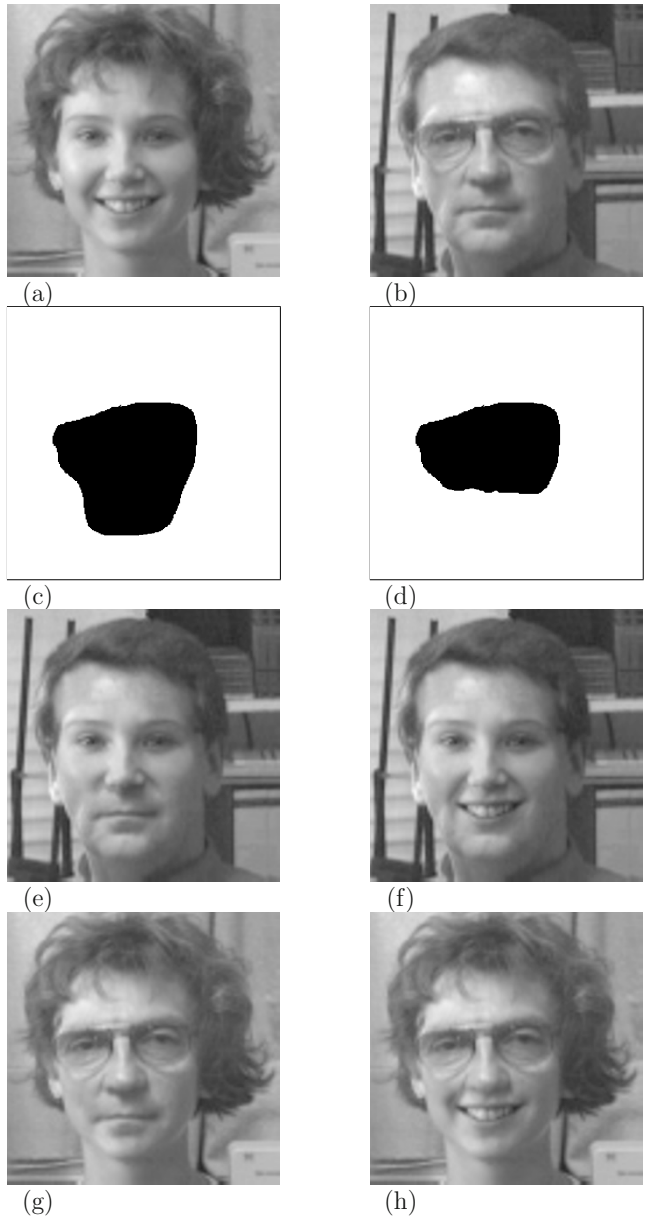


Bild 17.12: *mosaicing* von zwei Eingabebildern (a) und (b). Die Maskenbilder (c) und (d) geben an, welche Bildteile aus dem ersten bzw. zweiten Eingabebild zu übernehmen sind. Unterschiedliche Kombinationen zeigen die Bilder (e) bis (h).

(a) (b)

(c) (d)

Bild 17.13: Beispiele zum *mosaicing*. (a) Zebras eingeblendet in ein Hintergrundbild. In den hellen Bereichen der Zebras erscheint das Hintergrundbild. (b) Gruß aus München? – Jeder, der München kennt, müsste bei dieser Ansicht skeptisch werden. (c) „Sabine hoch 4“. (d) Fachhochschule München. Wer blickt durch den Baum? Als Maske wurde das FHM-Logo verwendet, das rechts oben eingeblendet ist.

Man kann nun ein Mosaikbild erzeugen, bei dem die Schärfentiefe so ausgedehnt ist, dass sowohl Vorder- als auch Hintergrund scharf erscheinen.

Dazu baut man wieder die beiden Eingabepyramiden $\mathbf{L}^{(e1)}$ und $\mathbf{L}^{(e2)}$ auf. Die Schichten der beiden Laplace-Pyramiden, die die niederen Frequenzbereiche abdecken, werden nur geringe Unterschiede aufweisen, da Fokusveränderungen in den niederen Frequenzbereichen nur wenige Auswirkungen zeigen. Dagegen werden Fokusveränderungen die Schichten beeinflussen, die die hohen Ortsfrequenzen enthalten. Korrespondierende Ausschnitte der beiden Eingabebilder werden nahezu dieselben Bereiche des Beobachtungsgebietes enthalten. Sie werden sich aber möglicherweise dadurch unterscheiden, dass sie unterschiedliche Bildschärfe aufweisen und somit unterschiedliche Amplituden in den hochfrequenten Schichten der Laplace-Pyramide besitzen. Für das zu erzeugende Ausgabebild heißt das, dass für die jeweilige Position derjenige Wert verwendet wird, der betragsgrößer ist:

$$l_i^{(a)}(x,y) = \begin{cases} l_i^{(e1)}(x,y), & \text{falls } |l_i^{(e1)}(x,y)| > |l_i^{(e2)}(x,y)|, \\ l_i^{(e2)}(x,y), & \text{falls } |l_i^{(e1)}(x,y)| \leq |l_i^{(e2)}(x,y)|. \end{cases} \tag{17.32}$$

17.10.4 Glättungsoperationen in Laplace-Pyramiden

In der Einleitung zu diesem Kapitel wurde bereits darauf hingewiesen, dass eine Motivation für den Einsatz der Pyramiden die Reduzierung des Rechenaufwandes bei Filteroperationen ist. Soll ein Bild auch im niederfrequenten Bereich geglättet werden, so muss ein großer Filterkern verwendet werden, was einen beachtlichen Rechenaufwand bedeutet.

Da ein Bild als Laplace-Pyramide in verschiedenen Auflösungsschichten vorliegt, haben auch kleine Filterkerne eine globale Auswirkung, je nachdem, in welcher Schicht der Pyramide sie angewendet werden. Bild 17.14 zeigt links oben ein Original. Rechts daneben ist das in der Pyramidenstruktur tiefpassgefilterte Bild zu sehen. Es wurde ein separabler $3 \cdot 3$–Binomialfilterkern (1 2 1) auf alle Schichten der Laplace-Pyramide angewendet. In der Schicht $\mathbf{L}_0$ der Pyramide wird dieser Kern nur geringe Auswirkungen zeigen. Bei Schichten, die näher an der Pyramidenspitze liegen, werden durch diesen Filterkern zunehmend niederfrequente Bildinformationen erfasst. Zur besseren Verdeutlichung wurde dieser Vorgang zweimal wiederholt. Die Ergebnisse sind in den beiden unteren Teilbildern von 17.14 abgebildet. Man sieht deutlich, dass die feinen Bildstrukturen unschärfer werden. Aber auch die niederfrequenten Anteile werden abgeschwächt, was hier am abnehmenden Kontrast zu sehen ist.

17.10.5 Texturen und Segmentierung

Zur Untersuchung von Texturen sind die Laplace-Pyramiden ebenfalls gut geeignet. Da sich eine Textur in verschiedenen Frequenzbereichen ausprägen kann, ist zu erwarten, dass sich diese Eigenschaft auch in den Schichten der Laplace-Pyramide zeigt. Ohne hier im Detail auf die Problematik der Texturanalyse eingehen zu wollen, werden im Folgenden einige Beispiele gebracht, die den Einsatz von Laplace-Pyramiden in der Texturanalyse und bei der Segmentierung von strukturierten Bildbereichen verdeutlichen.

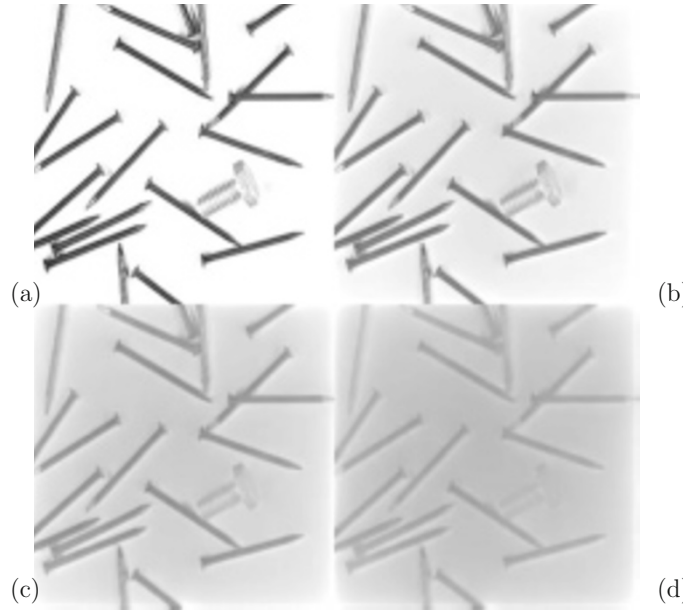


Bild 17.14: Filterung in Laplace-Pyramiden. (a) Original. (b) Das mit einem separablen $3 \cdot 3$–Binomialfilterkern bearbeitete Bild. Der Filterkern wurde auf alle Schichten der Laplace-Pyramide angewendet. In der Schicht $\mathbf{L}_0$ der Pyramide zeigen sich nur geringe Auswirkungen. Bei Schichten, die näher an der Pyramidenspitze liegen, werden durch diesen Filterkern zunehmend niederfrequente Bildinformationen erfasst. (c) Zweimal gefiltertes Bild. (d) Dreimal gefiltertes Bild. Man erkennt, dass feine Bildsturkturen unschärfer werden. Niederfrequente Anteile werden abgeschwächt, was hier am abnehmenden Kontrast zu sehen ist.

Weitere Bemerkungen zur Thematik „Textur“ sind in Kapitel 18 zusammengestellt. Dort ergibt sich ein interessanter Querbezug zur *fraktalen Geometrie* und zum *scale space filtering*.

Bei allen Verfahren zur Texturanalyse werden *Texturmerkmale* berechnet. Die sich dabei ergebenden Maßzahlen sollen so beschaffen sein, dass sie für bestimmte Texturen „homogene Bereiche“ im Merkmalsraum der Texturparameter ergeben. Bei nur einem Texturmerkmal heißt das, dass die Maßzahl, als Grauwert dargestellt, für die Textur einen homogenen Grauwertbereich ergibt. Die so entstandenen Grauwertbilder aus Texturmaßzahlen können dann mit üblichen Methoden der Bildverarbeitung (z.B. Schwellwertbildung, Klassifizierung, Kantenextraktion, Binarisierung, usw.) weiter verarbeitet und z.B. als Masken zur Extraktion bestimmter Texturen verwendet werden. Beispiele für Texturparameter sind:

- Die mittlere quadratische Abweichung von lokalen Umgebungen,
- der Betrag in die Richtung des Gradienten,
- aus der *co-occurrence*-Matrix (Abschnitt 3.10) abgeleitete Maßzahlen,
- die Kettenanzahl und Kettenlänge eines Lauflängencodes,
- Parameter aus den Fourier-Koeffizienten,
- usw.

Die Eignung der Laplace-Pyramiden zur Texturuntersuchung soll zunächst anhand eines Testbildes (Bild 17.15) untersucht werden.

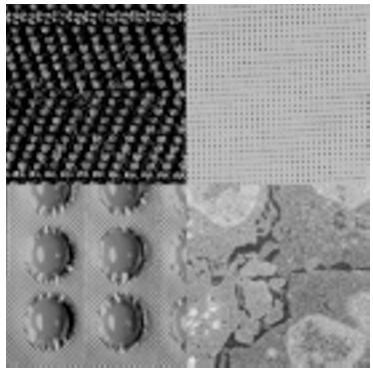
Für die erste Textur, ein Stoffgewebe, liegen die wesentlichen Wellenzahlindizes im Bereich $20 < u < 100$. Dieses breite Frequenzband verteilt sich hauptsächlich über die Schichten $\mathbf{L}_1$, $\mathbf{L}_2$ und $\mathbf{L}_3$. In $\mathbf{L}_0$ sind nur geringe und ab $\mathbf{L}_4$ fast keine Signale mehr zu erkennen.

Die zweite Textur (Gittergewebe) besitzt ebenfalls ein breites Frequenzspektrum: $50 < u < 170$. Diese Frequenzanteile sind im Wesentlichen in $\mathbf{L}_1$ enthalten. Wegen der Überschneidung der Frequenzbereiche sind auch in $\mathbf{L}_0$ und $\mathbf{L}_2$ Schwingungsanteile zu erkennen.

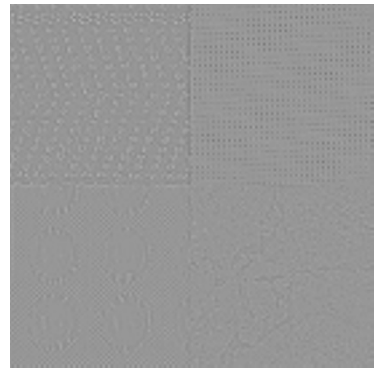
Die markanten Schwingungsanteile der dritten Textur (Blisterpackung) liegen im Bereich von $4 < u < 15$. Das bedeutet, dass sich diese Frequenzanteile hauptsächlich in den Schichten $\mathbf{L}_4$ und $\mathbf{L}_5$ zeigen. Die deutlich ausgeprägten Ränder um die Tablettenmulden sind hochfrequenter und daher in den Schichten $\mathbf{L}_1$ und $\mathbf{L}_2$ zu sehen.

Bei dem mikroskopischen Schnitt durch Melanomzellen (Bild 17.15-a, unten rechts) ist keine regelmäßige Struktur wie bei den anderen Beispielen zu erkennen. Das heißt, dass es auch keine markant hervortretenden Schwingungsanteile gibt und somit auch in keiner Schicht der Laplace-Pyramide diese Oberflächenstruktur signifikant hervortritt.

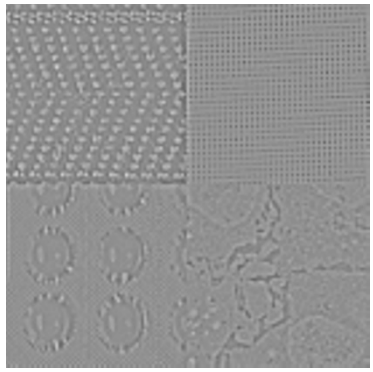
Wenn nun zu einer Laplace-Pyramide Texturmerkmale berechnet werden, so kann dies in jeder Schicht durchgeführt werden. Dabei sind die verschiedensten Vorgehensweisen möglich. Einige Beispiele sind im Folgenden aufgeführt:



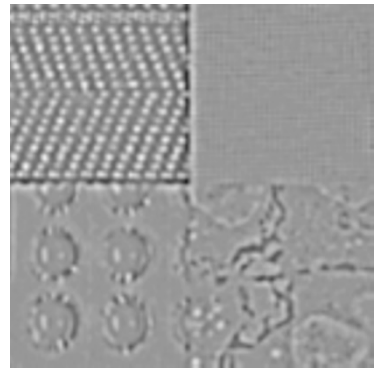
(a) Original



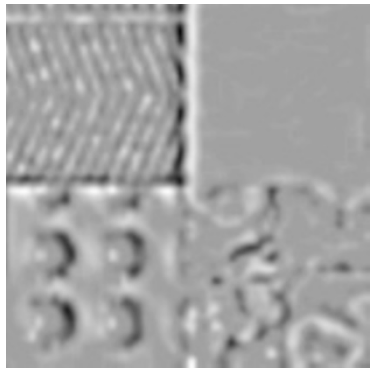
(b) Schicht $\mathbf{L}_0$



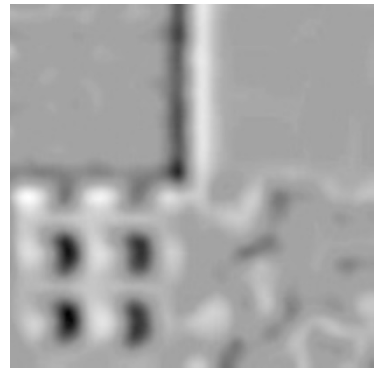
(c) Schicht $\mathbf{L}_{1,1}$



(d) Schicht $\mathbf{L}_{2,2}$



(e) Schicht $\mathbf{L}_{3,3}$



(f) Schicht $\mathbf{L}_{4,4}$

Bild 17.15: Laplace-Pyramiden und Texturanalyse. (a) Originalbild mit vier unterschiedlichen Texturen: Stoffgewebe, Gittergewebe, Blisterpackung, Melanom. (b) - (f) Die Schichten $\mathbf{L}_0$, $\mathbf{L}_{1,1}$ bis $\mathbf{L}_{4,4}$.

- Aufgrund von Voruntersuchungen kann man feststellen, dass die zu interessierende Struktur in einer bestimmten Schicht am deutlichsten hervortritt. Für diese Schicht wird ein Texturmaß berechnet und das daraus resultierende Ergebnis wird weiterverarbeitet.

- Ein Texturmaß zeigt für verschiedene Texturen in mehreren Schichten unterschiedliches Verhalten. Die Texturmaßzahlen der einzelnen Schichten werden zu mehrdimensionalen Merkmalsvektoren zusammengefasst und klassifiziert.

- Es werden verschiedene Texturmaße für die unterschiedlichen Schichten berechnet. Diese werden wie im vorhergehenden Fall als mehrdimensionale Merkmalsvektoren klassifiziert.

In den beiden folgenden Abschnitten werden zwei unterschiedliche Texturansätze in Laplace-Pyramiden vorgestellt und anhand verschiedener Bildbeispiele erläutert.

17.10.5.1 Mittlere quadratische Abweichung

Bei der Verwendung der mittleren quadratischen Abweichung q als lokales Texturmerkmal werden zu begrenzten Umgebungen (z.B. $3\cdot3$, $5\cdot5$,...) die Quadratsummen der Abweichungen vom Mittelwert aufsummiert. Die Resultate sind proportional zu einem Schätzwert für die Streuung der Umgebung. Oft wird deshalb q abkürzend als „Streuung der Umgebung“ bezeichnet. Da q in allen Schichten für dieselbe Umgebung berechnet wird, werden die Streuungen für die verschiedenen Frequenzbereiche abgeschätzt.

Der Formalismus zur Berechnung von q wird im folgenden für eine $3 \cdot 3$–Umgebung wiedergegeben:

$$\begin{aligned} q_i(x,y) &= \frac{1}{9}\sum_{u=-1}^{+1}\sum_{v=-1}^{+1}\Big(l_i(x+u,y+v)-m_i(x,y)\Big)^2; \\ & x=0,1,...,2^{r-i}; \\ & y=0,1,...,2^{r-i}; \\ & i=0,1,...,r. \end{aligned} \tag{17.33}$$

In dieser Formel ist $m_i(x,y)$ der Mittelwert der $3 \cdot 3$–Umgebung:

$$m_i(x,y) = \frac{1}{9}\sum_{u=-1}^{+1}\sum_{v=-1}^{+1} l_i(x+u,y+v). \tag{17.34}$$

Über die binomische Formel lässt sich (17.33) umformen:

$$
\begin{aligned}
q_i(x,y) &= \frac{1}{9}\sum_{u=-1}^{+1}\sum_{v=-1}^{+1} l_i^2(x+u,y+v) \quad - \quad m_i^2(x,y); \qquad (17.35)\\
&\quad x = 0,1,...,2^{r-i};\\
&\quad y = 0,1,...,2^{r-i};\\
&\quad i = 0,1,...,r.
\end{aligned}
$$

Ähnlich wie bei der REDUCE- und der EXPAND-Operation kann (17.35) in Zeilen- und Spaltenoperationen separiert werden:

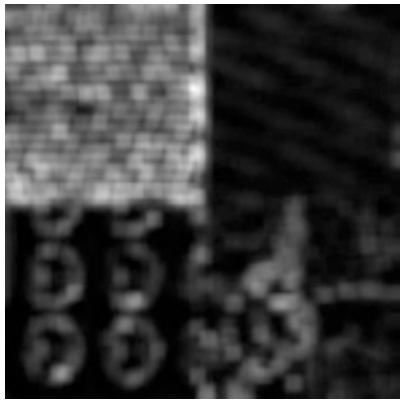
$$
\begin{aligned}
\tilde{q}_i(x,y) &= \sum_{v=-1}^{+1} l_i^2(x,y+v); \qquad (17.36)\\
&\quad x = 0,1,...,2^{r-i};\\
&\quad y = 0,1,...,2^{r-i};\\
&\quad i = 0,1,...,r;\\
q_i(x,y) &= \frac{1}{9}\sum_{u=-1}^{+1} \tilde{q}_i(x+u,y) \quad - \quad m_i^2(x,y);\\
&\quad x = 0,1,...,2^{r-i};\\
&\quad y = 0,1,...,2^{r-i};\\
&\quad i = 0,1,...,r.
\end{aligned}
$$

Für den Mittelwert der Umgebungen wird in den Laplace-Pyramiden null angenommen. Deshalb kann auch auf die Berechnung dieses Terms verzichtet werden, so dass man als Schätzwert für die Streuung der $3 \cdot 3$–Umgebungen erhält:

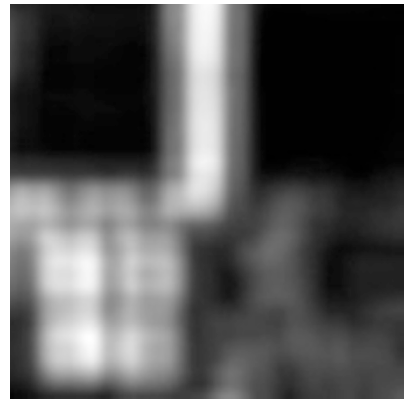
$$
\begin{aligned}
q_i(x,y) &= \frac{1}{9}\sum_{u=-1}^{+1}\sum_{v=-1}^{+1} l_i^2(x+u,y+v); \qquad (17.37)\\
&\quad x = 0,1,...,2^{r-i};\\
&\quad y = 0,1,...,2^{r-i};\\
&\quad i = 0,1,...,r.
\end{aligned}
$$

Auch diese Formel kann wie oben in Zeilen- und Spaltenoperationen separiert werden.

Häufig tritt ferner folgende Problemstellung auf: Oft ist es notwendig, in einem Bild Bereiche mit Objekten vom Bildhintergrund abzugrenzen. Die hier zugrundeliegende Idee ist folgende: Wenn der Bildhintergrund ein charakteristisches Frequenzverhalten zeigt, dann verursachen Objekte vor diesem Hintergrund Störungen im Frequenzverhalten. Diese Störungen müssen sich in den Schichten der Laplace-Pyramide, zu der ein passendes



(a) Schicht $\mathbf{Q}_{2,2}$



(b) Schicht $\mathbf{Q}_{4,4}$

Bild 17.16: Mittlere quadratische Abweichung als Schätzwert für die Streuung der Texturen von Bild 6.15. (a) Die Schicht $\mathbf{Q}_2$ als mittlere quadratische Abweichung der Schicht $\mathbf{L}_2$. (b) Die Schicht $\mathbf{Q}_4$ als mittlere quadratische Abweichung der Schicht $\mathbf{L}_4$. In $\mathbf{Q}_2$ ergibt die erste Textur die größten Werte, die anderen haben Werte, die sehr klein oder sogar null sind. Man könnte diese Schicht zur Extraktion der ersten Textur verwenden. In $\mathbf{Q}_4$ dagegen ist die dritte Textur deutlich markiert. Die breiten Balken in der Mitte sind Störungen im Bereich von Texturkanten.

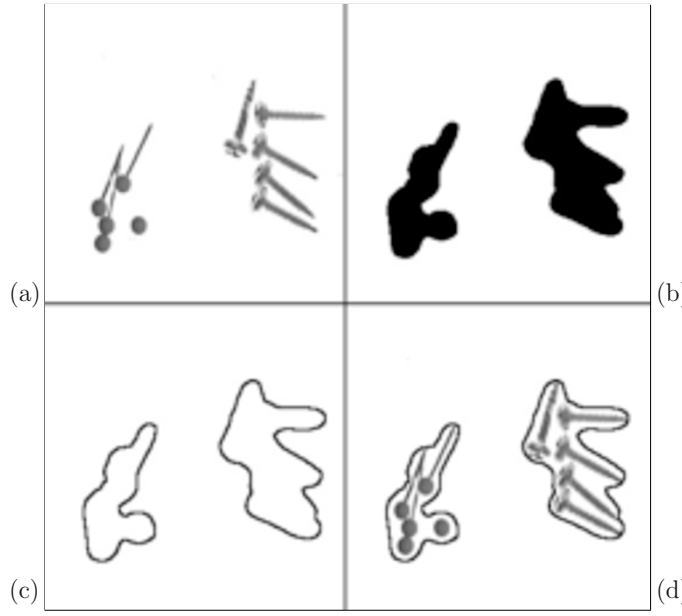


Bild 17.17: Abgrenzung von Bildbereichen mit Objekten vom Hintergrund. Links oben das Original. Durch Segmentierung geeigneter Schichten der Laplace-Pyramide können Masken für die Extraktion der Bildbereiche berechnet werden.

Texturmaß berechnet wurde, ausdrücken. Mit apriori-Wissen kann eine Schicht ermittelt werden, in der sich die Störungen am besten ausprägen. Diese Schicht wird dann zur Segmentierung verwendet.

Bild 17.17 zeigt ein einfaches Beispiel dazu. Das Original links oben enthält zwei Bereiche mit unterschiedlichen Objekten, nämlich Pin-Nadeln und Schrauben. Der Hintergrund ist hier trivialerweise homogen, so dass die Objekte sicherlich deutliche Störungen im Frequenzverhalten des Hintergrundes verursachen werden. Zur Segmentierung der beiden Bereiche wurde hier wie oben das Texturmaß berechnet und die Schicht $\mathbf{L}_2$ auf Originalgröße expandiert. Dieses Bild wurde mit einem Schwellwert (nahe bei 255) segmentiert und invertiert (rechts oben). Die Extraktion des Randes und die Überlagerung mit dem Original zeigen die beiden unteren Teilbilder. Man sieht, dass die beiden Bildbereiche gut erfasst werden. Mit den Informationen der Schichten $\mathbf{L}_0$ und $\mathbf{L}_1$ kann man jetzt versuchen zu unterscheiden, ob es sich um Nadeln oder um Schrauben handelt. Die Berechnung kann dabei auf die segmentierten Bereiche beschränkt werden.

Bild 17.16 zeigt dieses Texturmaß für die Bildbeispiele von Bild 17.15. Das linke Bild enthält die expandierte Schicht $\mathbf{Q}_2$ als mittlere quadratische Abweichung der Schicht $\mathbf{L}_2$ und das rechte Bild die expandierte Schicht $\mathbf{Q}_4$ als mittlere quadratische Abweichung der Schicht $\mathbf{L}_4$. In $\mathbf{Q}_2$ ergibt die erste Textur die größten Werte, die anderen haben Werte, die sehr klein oder sogar null sind. Man könnte diese Schicht zur Extraktion der ersten Textur verwenden. In $\mathbf{Q}_4$ dagegen ist die dritte Textur deutlich markiert. Die breiten Balken in der Mitte sind Störungen im Bereich von Texturkanten.

Schwieriger wird das Problem, wenn der Hintergrund texturiert ist (Bild 17.18). Im Original links oben liegen auf einer Plastikverpackungsfolie eine Schraube und eine Mutter. Zur Segmentierung wurden hier die Texturwerte der Schicht $\mathbf{L}_3$ verwendet (rechts oben). Um die Maske sinnvoll anfertigen zu können, wurde Zusatzinformation herangezogen: Zunächst wurde eine *area-of-interest* definiert, die die Störungen am Bildrand ausblendet. Alternativ dazu können die Störungen am Bildrand auch mit morphologischen Operationen (z.B. *grassfire*, Grundlagen in Kapitel 6) entfernt werden. Sodann wurde die apriori-Information verwendet, dass Störungen eine Fläche von mindestens c Bildpunkten besitzen. Der Bereich der Schraube wurde zufriedenstellend extrahiert. Für die Mutter ist die Maske etwas zu klein geraten. Das liegt u.a. daran, dass die Mutter etwa dieselbe Größe hat wie die Luftblasen der Plastikfolie des Hintergrundes und damit die Störung des charakteristischen Frequenzverhaltens des Hintergrundes nicht so ausgeprägt ist.

17.10.5.2 Spektrale Signalenergie

Aufgrund der Frequenzzerlegung der Laplace-Pyramide können die einzelnen Schichten als spektrale Zerlegung eines Bildes im Ortsbereich aufgefasst werden. Verschiedene Texturen prägen sich in den einzelnen Schichten unterschiedlich aus. Wenn sich nun eine Textur in einer bestimmten Schicht der Laplace-Pyramide weniger stark zeigt, so kann man davon ausgehen, dass die Signalenergie für diese Textur in dieser Schicht geringer ist. Ein Maß für die Signalenergie ist die mittlere quadratische Abweichung, wie sie im letzten Abschnitt verwendet wurde. Allerdings berücksichtigen die Berechnungen dort nicht, dass eine Schicht von verschiedenen Texturen beeinflußt wird.

Um in einzelnen Schichten unterschiedliche Texturen untersuchen zu können, muss man ein Verfahren anwenden, bei dem die Signalenergie in lokalen Bereichen für unterschiedliche Umgebungsgrößen abgeschätzt wird. Dazu wird eine *Energiehierarchie* aufgebaut, die als *hierarchische diskrete Korrelation* (HDC) bezeichnet wird. Begonnen wird mit der untersten Schicht $\mathbf{L}_0$. Hier werden die Signalwerte quadriert, dann wird r-mal die HDC-Operation (genaue Erläuterung der HDC-Operation weiter unten) darauf angewendet. Im nächsten Schritt wird die Schicht $\mathbf{L}_1$ quadriert und dann $(r-1)$-mal die HDC-Operation angewendet. Dies wird bis zur Spitze $\mathbf{L}_r$ fortgesetzt, deren Signalwerte nur quadriert werden. Der Aufbau dieser Energiehierarchie ist im Blockdiagramm in Bild 17.19 dargestellt.

Die HDC-Operation ist der REDUCE-Operation ähnlich. Zur Glättung wird die gleiche Filtermaske $\mathbf{H} = (h(u, v))$ verwendet. Beim Übergang zur nächsten Stufe wird jedoch nicht die Größe um die Hälfte reduziert, sondern die Breite des Ausschnittes vergrößert, in dem die Signalwerte geglättet werden. Durch diese Vorgehensweise kann man Bilder über einen

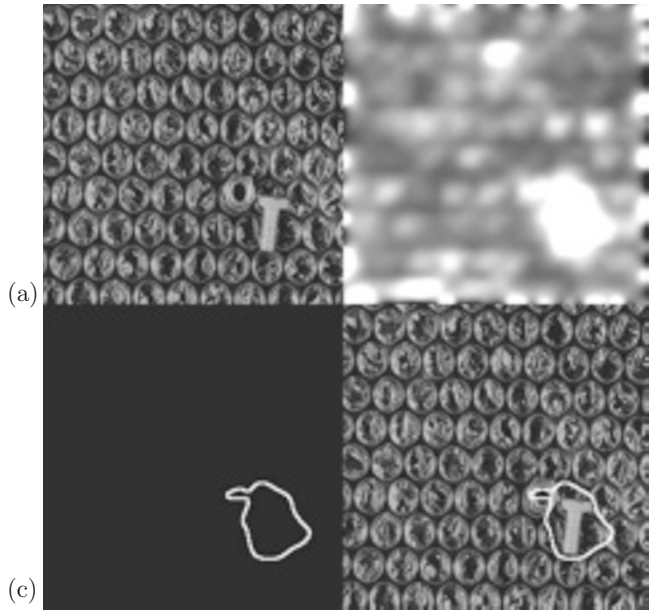


Bild 17.18: Abgrenzung von Bildbereichen mit Objekten vom texturierten Hintergrund. Links oben das Original. Durch Segmentierung geeigneter Schichten der Laplace-Pyramide können auch hier Masken für die Extraktion der Bildbereiche berechnet werden.

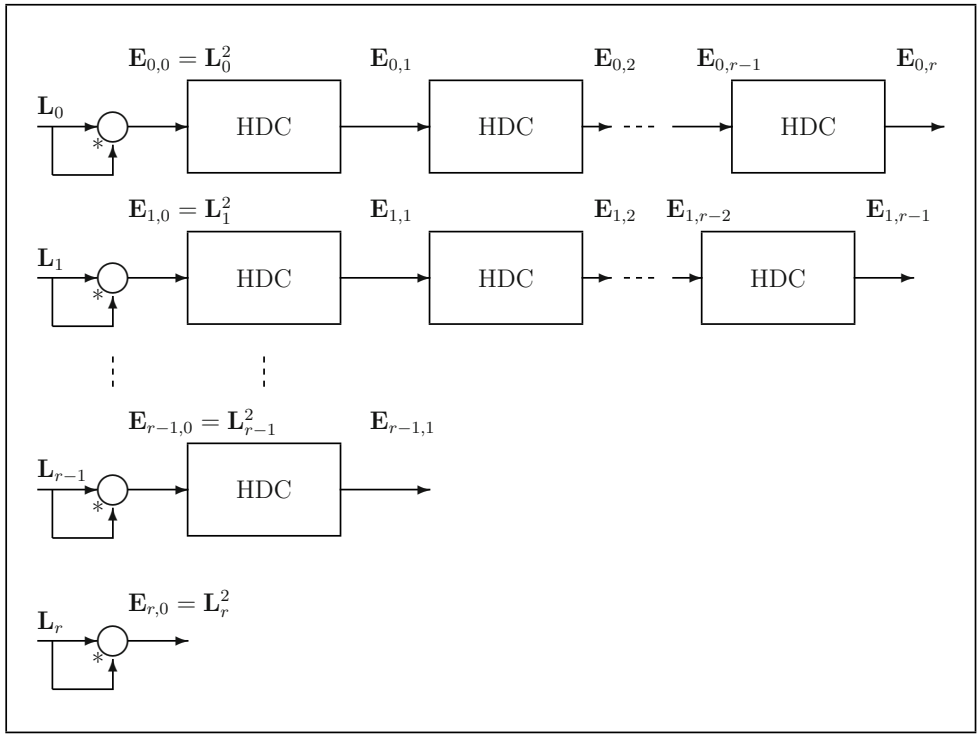


Bild 17.19: Blockdiagramm zum Aufbau der Energiehierarchie.

großen Bereich von Wellenzahlindizes glätten. Die HDC-Operation wird für jede Schicht $\mathbf{L}_i$ der Laplace-Pyramide durchgeführt. Es werden folgende Bezeichnungen verwendet:

$$\begin{aligned} \mathbf{E}_{i,0} &= \mathbf{L}_i^2; \\ \mathbf{E}_{i,k+1} &= \mathrm{HDC}(\mathbf{E}_{i,k}); \\ & \quad i = 0, 1, ..., r; \\ & \quad k = 0, 1, ..., r - (i+1). \end{aligned} \tag{17.38}$$

Die einzelnen Elemente der Energiehierarchiestufen berechnen sich dann gemäß:

$$\begin{aligned} e_{i,k+1}(x,y) &= \sum_{u=-2}^{+2} \sum_{v=-2}^{+2} e_{i,k}(x + u \cdot 2^k, y + v \cdot 2^k) \cdot h(u+2, v+2); \\ & \quad x = 0, 1, ..., r - i; \end{aligned} \tag{17.39}$$

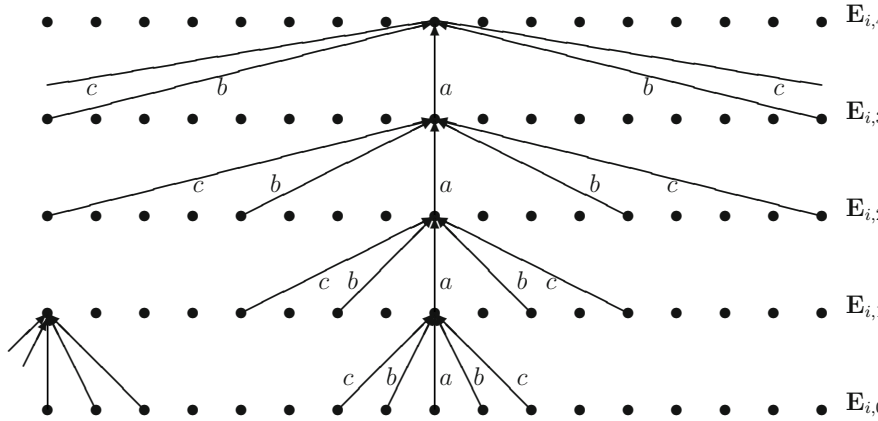


Bild 17.20: Berechnung der Energiehierarchiestufen (HDC-Verfahren). Am Rand liegende Bildpunkte müssen gesondert berechnet werden.

$$y = 0, 1, ..., r - i;$$
$$k = 0, 1, ..., r - (i + 1).$$

Wie bei der REDUCE-Operation kann man auch die HDC-Operation in Zeilen- und Spaltenoperationen auftrennen:

$$\tilde{e}_{i,k+1}(x, y) = \sum_{v=-2}^{+2} e_{i,k}(x, y + v \cdot 2^k) \cdot \hat{h}(v + 2); \tag{17.40}$$
$$x = 0, 1, ..., r - i;$$
$$y = 0, 1, ..., r - i;$$
$$e_{i,k+1}(x, y) = \sum_{u=-2}^{+2} \tilde{e}_{i,k+1}(x + u \cdot 2^k, y) \cdot \hat{h}(u + 2); \tag{17.41}$$
$$k = 0, 1, ..., r - (i + 1).$$

In Bild 17.20 ist dieses Verfahren für den jeweils mittleren Bildpunkt pro Zeile schematisch dargestellt. Man sieht hier auch, dass die Behandlung der Randpunkte etwas anders zu handhaben ist als bei der REDUCE-Operation: Beim Übergang von $\mathbf{E}_{i,0}$ zu $\mathbf{E}_{i,1}$ müssen zwei Bildpunkte am Rand extrapoliert werden, bei $\mathbf{E}_{i,1}$ zu $\mathbf{E}_{i,2}$ sind es vier und in der k-ten Iteration sind es 2^{k+1} Bildpunkte. Dadurch ergibt sich eine Begrenzung der maximalen

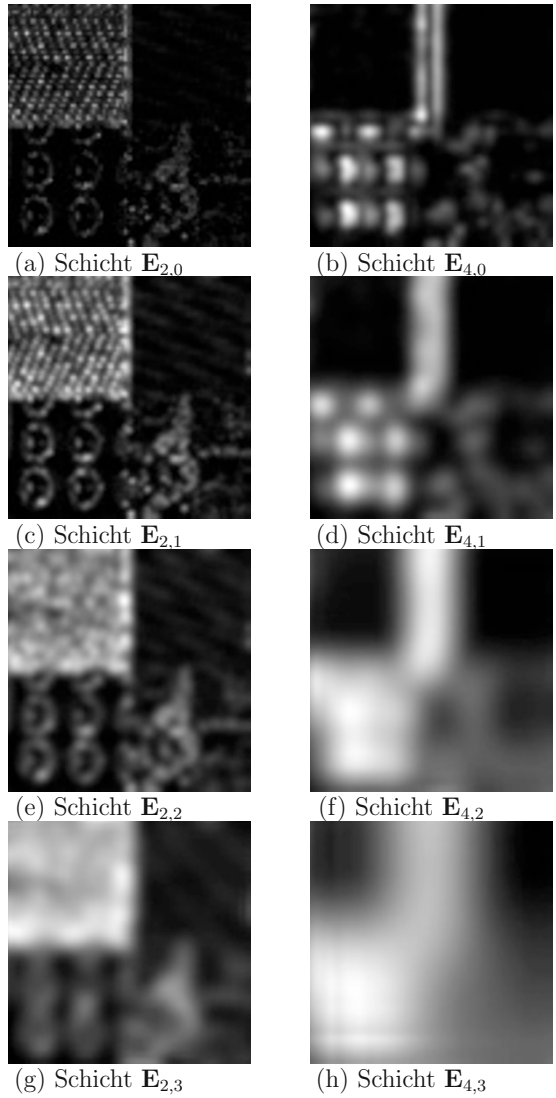
Anzahl von Energiehierarchiestufen, denn es müssen pro Zeilen- und Spaltenoperation von den fünf Bildpunkten, die zur Glättung verwendet werden, mindestens drei innerhalb des Bildes liegen.

Nun einige Anwendungsbeispiele zur Oberflächenuntersuchung mit dem HDC-Verfahren: Zunächst werden die Texturen von Bild 17.15 untersucht. In der Bildfolge 17.21 werden in der linken Spalte die Energiehierarchiestufen $\mathbf{E}_{2,0}$ bis $\mathbf{E}_{2,3}$ und in der rechten Spalte die Energiehierarchiestufen $\mathbf{E}_{4,0}$ bis $\mathbf{E}_{4,3}$ abgebildet. In der Laplace-Pyramide ist für die Textur 1 (Stoffgewebe) in der Schicht $\mathbf{L}_2$ die größte Signaldichte vorhanden. In den darunterliegenden Schichten der Energiehierarchie dehnen sich die signalstärksten Strukturen mehr und mehr aus, so dass die Textur 1 mit $\mathbf{E}_{2,3}$ am besten segmentiert werden kann.

Bei Textur 3 (Blisterpackung) ist die größte Energie in der Schicht $\mathbf{L}_4$ der Laplace-Pyramide vorhanden. In der Energierhierarchie ist deshalb diese Textur am besten mit $\mathbf{E}_{4,2}$ oder $\mathbf{E}_{4,3}$ zu segmentieren.

Ein weiteres Beispiel kommt aus dem Bereich der Vollständigkeitskontrolle. Bild 17.22 zeigt eine vollständig gefüllte Blisterpackung, daneben ist eine Packung mit einer fehlenden Tablette abgebildet. In der Schicht $\mathbf{E}_{4,1}$ ist diese „Texturstörung“ deutlich zu sehen. Nach einer Binarisierung dieser Schicht wurde damit die fehlende Tablette im Original markiert.

Abschließend zu den Gauß- und Laplace-Pyramiden sei noch bemerkt, dass in [Burt83] noch weitere interessante Anwendungen, wie z.B. das Multiresolution Spline, die Korrelation oder grafische Anwendungen, auch in Verbindung mit der fraktalen Geometrie (Kapitel 18) auf der Basis dieser Technik untersucht wurden.



(a) Schicht $\mathbf{E}_{2,0}$ (b) Schicht $\mathbf{E}_{4,0}$

(c) Schicht $\mathbf{E}_{2,1}$ (d) Schicht $\mathbf{E}_{4,1}$

(e) Schicht $\mathbf{E}_{2,2}$ (f) Schicht $\mathbf{E}_{4,2}$

(g) Schicht $\mathbf{E}_{2,3}$ (h) Schicht $\mathbf{E}_{4,3}$

Bild 17.21: Energiehierarchiestufen nach dem HDC-Verfahren. Linke Spalte (Bilder (a), (c), (e), (g)): Stufen $\mathbf{E}_{2,0}$, $\mathbf{E}_{2,1}$, $\mathbf{E}_{2,2}$ und $\mathbf{E}_{2,3}$. In der Schicht $\mathbf{E}_{2,0}$ treten die Bildstrukturen des Stoffgewebes mit der stärksten Signalenergie deutlich hervor. Durch das HDC-Verfahren dehnen sich diese Signale in den weiteren Schicht aus. Die Schicht $\mathbf{E}_{2,3}$ ist damit zur Segmentierung der Textur 1 geeignet. In der rechten Spalte ist derselbe Sachverhalt für die Textur 3 (Blisterpackung) dargestellt.

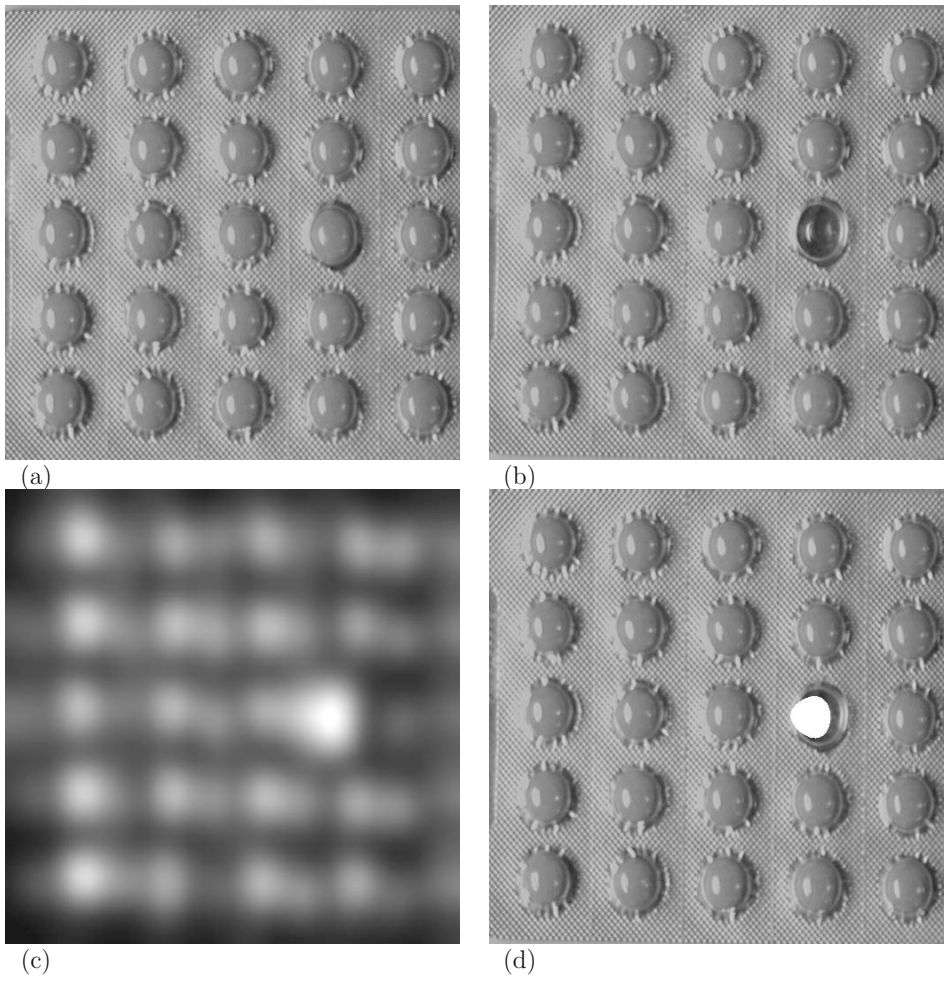


Bild 17.22: Vollständigkeitskontrolle mit dem HDC-Verfahren. (a) Vollständige Blisterpackung. (b) Blisterpackung mit einer fehlenden Tablette. (c) Die Schicht $\mathbf{E}_{4,1}$ der Energiehierarchie. (d) Die binarisierte Schicht $\mathbf{E}_{4,1}$ wurde in das Original eingeblendet und dadurch die fehlende Tablette markiert.

Kapitel 18

Scale Space Filtering

18.1 Anwendungen

Vor dem Hintergrund der fraktalen Geometrie wird in diesem Kapitel das *scale space filtering* erläutert. Dabei wird eine Oberfläche in unterschiedlichen „Vergrößerungen“ betrachtet und untersucht, ob dabei sich wiederholende Strukturen auftreten. Als praktische Anwendung hat diese Technik gute Ergebnisse bei der Qualitätskontrolle von Oberflächen, z.B. von Stoffen, geliefert.

18.2 Grundlagen: Fraktale Geometrie

Dieser Abschnitt enthält eine kurze Übersicht über die fraktale Geometrie. Es werden wichtige Begriffe, wie z.B. die gebrochene (fraktale) Dimension vorgestellt und anhand von Beispielen erläutert. Zur Vertiefung dieser Thematik sei auf die umfangreiche Literatur verwiesen (z.B. [Ekel92], [Mand87]).

Erste Ansätze in Richtung fraktaler Geometrie wurden bereits im vorletzten Jahrhundert beschrieben, wenn auch aus einem ganz anderen Blickwinkel: Der Mathematiker Karl Weierstrass stellte 1872 eine Funktion vor, die überall stetig, aber nirgends differenzierbar ist. Der Zusammenhang zwischen Stetigkeit und Differenzierbarkeit fällt in den Bereich der klassischen Infinitesimalrechnung. Lange Zeit ging man davon aus, dass stetige Funktionen bis auf „einige“ Ausnahmen auch differenzierbar sind. Weierstrass widerlegte diese Ansicht mit folgender Funktion: Zunächst konstruierte er eine Funktion, deren Graph wie ein Sägeblatt aussieht. Es ist offensichtlich, dass diese Funktion an den Spitzen der Sägezähne zwar stetig, aber nicht differenzierbar ist. An allen anderen Stellen ist sie differenzierbar. Nun definierte er eine neue Funktion, die durch den Grenzübergang entsteht, wenn man die Sägezähne unendlich nahe nebeneinander liegen lässt. Die so definierte Funktion ist dann zwar überall stetig aber an keiner Stelle mehr differenzierbar.

Sehr zögernd akzeptierten die Mathematiker die Existenz derartiger Funktionen. Sie wurden allgemein zunächst nicht sehr ernst genommen, da „Monster-Funktionen“ dieser Art sicher nichts mit dem mathematischen Alltag zu tun haben.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_18

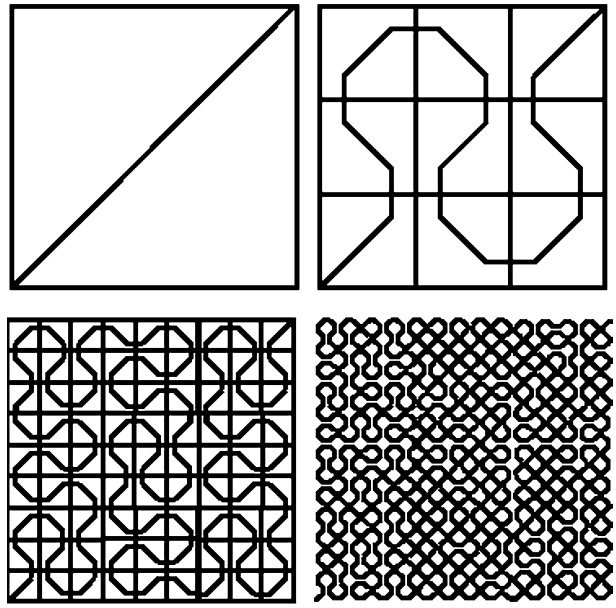


Bild 18.1: Die Peano-Kurve als Beispiel einer „raumfüllenden“ Kurve. Initiator ist die Diagonale eines Quadrats. Der Generator ist daneben abgebildet. Setzt man die Generierungsschritte bis ins Unendliche fort, so werden alle Punkte des umgebenden Quadrats erfasst.

Für weitere Diskussion sorgte 1890 Giuseppe Peano, als er eine „raumfüllende“ Kurve beschrieb (Bild 18.1). Um die Entstehung derartiger Kurven zu beschreiben, geht man von einem *Initiator* aus. Im Fall der Peano-Kurve ist der Initiator die Diagonale eines Quadrats. Mit dem *Generator* wird beschrieben, wie die Seiten iterativ verändert werden. Der Generator der Peano-Kurve ist in Bild 18.1 neben dem Initiator dargestellt. Setzt man die Generierungsschritte bis ins Unendliche fort, so werden alle Punkte des (zweidimensionalen) Quadrats erfasst. Damit kam der Begriff der Dimension ins Wanken: Wie kann eine eindimensionale Kurve alle Punkte einer zweidimensionalen Ebene erfassen?

An dieser Stelle sei folgende Betrachtung aus [Mand87] eingefügt. Der Begriff der Dimension ist im (mathematischen) Alltag einleuchtend: Punktmengen haben die Dimension null, Kurven die Dimension eins, Flächen die Dimension zwei, Körper die Dimension drei, usw. Ab der Dimension vier hat der Mensch zwar Schwierigkeiten bei der Vorstellung dieser Gebilde, aber der Mathematik macht es keine Probleme, mit n-dimensionalen Räumen zu arbeiten.

Man stelle sich nun einen Wollknäuel vor: Sieht man ihn aus weiter Entfernung, so ist

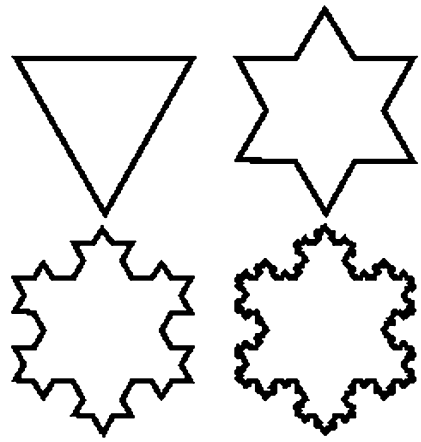


Bild 18.2: Die Koch'sche Insel (Schneeflockenkurve). Hier kann man sich fragen: Wie lang ist der Rand dieser Figur?

lediglich ein Punkt der Dimension null zu erkennen. Etwas näher betrachtet erscheint der Wollknäuel als Scheibe mit der Dimension zwei. Bei noch näherer Betrachtung erkennt man, dass es sich um ein dreidimensionales Gebilde handelt. Aber ist der Wollknäuel wirklich dreidimensional? Der aufgewickelte Faden ist ja eine Kurve, die eindimensional ist. Oder ist der Faden ein Zylinder, der dreidimensional ist? Bei mikroskopischer Betrachtung zerfällt der Faden in nahzu unendlich viele Härchen: Also doch eindimensional?

Der übliche Begriff der Dimension passt scheinbar nicht auf natürliche Problemstellungen. Man könnte entgegnen, dass die Mathematik als abstrahierte Wissenschaft nicht dazu gedacht ist, Problemstellungen wie einen Wollknäuel zu beschreiben. Jedoch ist letztlich die Mathematik ein Hilfsmittel zur Beschreibung unserer realen Umwelt, daher muss sie auch in der Lage sein, reale Sachverhalte so gut wie möglich zu beschreiben.

Weitere Gedankenexperimente wie bei der Peano-Kurve erfolgten: Helge von Koch stellte 1904 seine „Schneeflockenkurve“ (Koch'sche Kurve) vor (Bild 18.2). 1915 konstruierte Sierpinski seine „Pfeilspitze“ (Sierpinski-Dreieck , Bild 18.3).

Die Peano-Kurve, die Koch'sche Kurve und das Sierpinski-Dreieck sind Beispiele für *selbstähnliche Strukturen*, d.h. dass bei richtiger Skalierung jeder noch so kleine, geeignet gewählte Teil wieder das Ganze bilden kann.

Mit diesen Kurven wird ein weiterer Begriff der Geometrie in Frage gestellt, nämlich die Länge des Randes einer Figur. Wenn man das Bildungsgesetz z.B. der Koch'schen Kurve vor Augen hat, müsste sich doch ein unendlich langer Rand ergeben, da bei jedem Durchgang ein Teilstück um den Faktor 4/3 länger wird.

Benoit Mandelbrot griff um 1975 Ideen von Hausdorff und Julia auf, die zum Teil schon

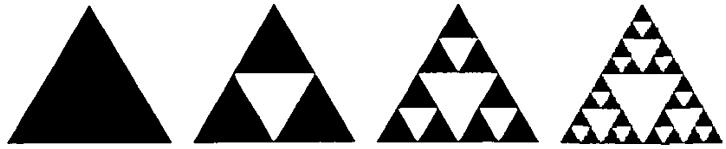


Bild 18.3: Das Sierpinski-Dreieck.

um 1900 publiziert waren. Er prägte den Begriff der *fraktalen Geometrie*. Ein wesentliches Merkmal der fraktalen Geometrie ist die *Selbstähnlichkeit*. Damit ist gemeint, dass sich die prinzipielle Struktur eines Fraktals immer wieder wiederholt, je „näher" man sie betrachtet, oder etwas mathematischer ausgedrückt, je kleiner der Maßstab ist, den man für die Betrachtung wählt. Dieser Sachverhalt wird auch als *Skaleninvarianz* bezeichnet. Schöne Beispiele für einfache, selbstähnliche Fraktale sind die bereits oben dargestellten Kurven.

Mandelbrot erkannte die Selbstähnlichkeit als ein „mächtiges Mittel zum Hervorbringen von Gestalten" und gab viele Beispiele, die mehr oder weniger skaleninvariant sind: Aktienkurse, Küstenlinien, Pflanzenblätter, Wolken, bestimmte komplexe Zahlen, usw.

Bekannt ist auch das Beispiel der Längenvermessung der Küstenlinien von Großbritannien, das 1961 zuerst von L.F. Richardson vorgestellt wurde. Wenn man die Länge der Küstenlinie vermessen will, so könnte man z.B. einen Meterstab als Skaleneinheit ($s = 1m$) nehmen und die Messung durchführen. Für die Länge wird sich dann ein bestimmter Wert ergeben. Möglicherweise wird sich bei der Durchführung der Messung die Frage erheben, ob denn die zugrunde gelegte Skala $s = 1m$ genügend genau ist, und man könnte die Messung mit einer Skala $s = 0.1m$ wiederholen. Es wird sich ein größerer Wert für die Länge ergeben, da man jetzt feinere Details der Küstenlinie erfasst hat. In einem nächsten Schritt könnte man $s = 0.01m$ verwenden. Die Länge der Kurve sollte sich als Grenzwert $L(s)$ der Längen der Polygonzüge ergeben. Kurven, deren Länge auf diese Weise ermittelt werden kann, heißen *rektifizierbar*. [1] Wenn man von einer Struktur sagt, sie zeige *Skalenverhalten*, so meint man damit, dass sie bei allen Skalen denselben Grad an Irregularität aufweist. Eine derartige Struktur bezeichnet man auch als *skaleninvariante Struktur*. Der Parameter s heißt *Skalenparameter*.

[1] Peitgen wollte in seinem Buch [Peit86] über fraktale Bilder ein ähnliches, „eingedeutschtes" Beispiel bringen: Wie lang ist die Grenze der Bundesrepublik Deutschland? Damit kam er aber in Schwierigkeiten: Die Grenze ist nach gesetzlicher Definition ein endlicher Polygonzug!

Macht man bei der Küstenlinie den Skalenwert s immer kleiner, so wird man immer mehr Details erfassen: Felsen, Steine, Staub, Moleküle... Diesem Gedankenexperiment folgend erkennt man, dass die wahre Länge der Küste Großbritanniens, wie auch die der Koch'schen Kurve, unendlich ist.

Aus den Experimenten, die Richardson durchführte, leitete er folgende Gesetzmäßigkeit ab:

$$L(s) \sim s^{1-D}. \tag{18.1}$$

Wenn man in ein Koordinatensystem auf der Abszisse $\log s$ und auf der Ordinate $\log L(s)$ aufträgt, so erhält man für die Küste Großbritanniens näherungsweise eine Gerade (Bild 18.4). Mandelbrot erkannte, dass die Steigung dieser Geraden eine Schätzung für $1 - D$ ist und dass D als eine *fraktale Dimension* betrachtet werden kann.

Fraktale Strukturen besitzen also eine fraktale Dimension. Nach Mandelbrot ist die fraktale Dimension die Hausdorff-Besicovitch-Dimension [Haus19]. Sie ist in der Praxis schwer zu bestimmen. Aus diesem Grund verwendet man oft einfachere Definitionen für fraktale Dimensionen, die jedoch, bei richtiger Wahl der Skalenparameter, der Hausdorff-Besicovitch-Dimension nahe kommen.

Dazu einige Beispiele: Teilt man eine Strecke in drei gleiche Teile, so ist die ganze Strecke dreimal so lang wie jede Teilstrecke. Die Anzahl a der Teilstrukturen ist also $a = 3^1 = 3$. Bei einem Quadrat, bei dem jede Seite dreigeteilt wird, erhält man $a = 3^2 = 9$ Teilstrukturen (Teilquadrate) und beim Würfel $a = 3^3 = 27$ Teilstrukturen (Teilwürfel). Man erkennt ein Potenzgesetz:

$$a = s^D, \tag{18.2}$$

wobei a die Anzahl der sich ergebenden Teilstrukturen, s die Anzahl der Unterteilungen der Strecken und D die Dimension der betrachteten Struktur ist. Löst man (18.2) nach D auf, so erhält man:

$$D = \frac{\log a}{\log s}. \tag{18.3}$$

Diese Betrachtungsweise kann man jetzt z.B. auf die Koch'sche Kurve anwenden: Der Generator besteht aus vier Teilstücken. Der Initiator ist dreimal so lang wie ein Teilstück. Somit ergibt sich: $D = \frac{\log 4}{\log 3} = 1.262$. Durch sinngemäße Betrachtungen erhält man für die Peano-Kurve $D = 2$ und für das Sierpinski-Dreieck $D = 1.585$.

Die so berechnete Größe D ist aber nicht die fraktale Dimension, da sie ja nur für streng selbstähnliche Strukturen - bei richtiger Wahl der Skalierung - zutrifft. In [Mand87] wird sie als *Ähnlichkeitsdimension* bezeichnet.

Die Beziehung (18.3) ist aber eine Motivation für die ziemlich willkürlich anmutende Definition der fraktalen Dimension, die im Folgenden anhand eines Beispiels erläutert wird. Dabei wird dargestellt, wie die fraktale Dimension einer Struktur definiert ist und wie sie ermittelt werden kann. Als Beispiel dient die Koch'sche-Kurve, die man sich auf

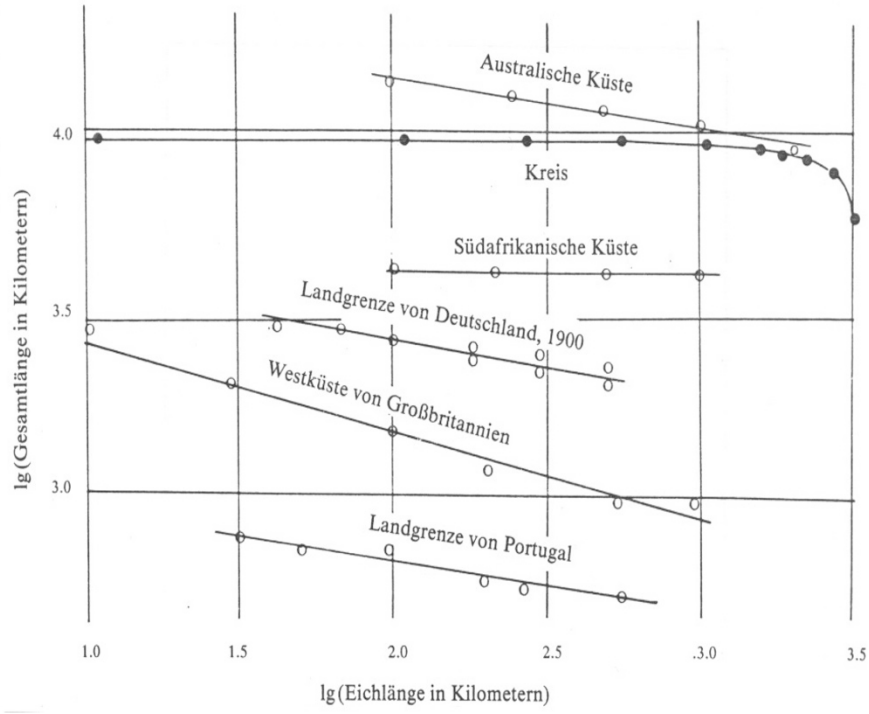


Bild 18.4: Zuwachsrate der Längen verschiedener Grenzlinien. Beim Auftragen von $\log L(s)$ über $\log s$ ergeben sich bei fraktalen Kurven Geraden. Die Untersuchung des Kreises ergibt eine Kurve, die sich schnell bei einem bestimmten Wert stabilisiert (aus [Mand87]).

einem quadratisch gerasterten Papier gezeichnet vorstellen möge. Man zählt nun, wieviele Rasterquadrate von der Kurve geschnitten werden. Dann wird das Raster verfeinert, etwa durch Halbierung der Rasterquadratseiten, und es wird wieder gezählt. Dieser Vorgang wird einige Male bis zu einer noch sinnvollen Rasterflächengröße wiederholt. Es lässt sich zeigen, dass man als Beziehung ein Potenzgesetz ansetzen kann:

$$N = \frac{C}{\epsilon^D}, \tag{18.4}$$

wobei N die Überdeckungszahl, ϵ die Maschenweite der Rasterflächen, C eine Konstante und D die fraktale Dimension ist. Durch Logarithmieren erhält man:

$$\log N = D \log \frac{1}{\epsilon} + \log C. \tag{18.5}$$

Trägt man die Logarithmen von N und $\frac{1}{\epsilon}$ in ein Koordinatensystem ein, so erhält man eine Gerade, deren Steigung die fraktale Dimension D ist.

In der Praxis wird man nicht exakt eine Gerade erhalten, da sich z.B. beim Auszählen der Rasterquadrate Fehler einschleichen. Man wird daher einige „Messungen“ mit verschiedenen Skalen durchführen, die Punkte $(\log N, \log \frac{1}{\epsilon})$ in das Koordinatensystem eintragen und mit Hilfe der linearen Regression die Steigung der Regressionsgeraden als fraktale Dimension ermitteln.

Diese Vorgehensweise kann auch bei natürlichen Strukturen, wie z.B. Wolken, Küstenlinien, Bäumen oder Flussystemen, sinngemäß angewendet werden. Als Beispiel wird angegeben, dass der Raum der Arterien des Menschen die fraktale Dimension $D = 2.7$ besitzt.

In [Mand87] werden ausführlich die verschiedensten Fraktalarten diskutiert und in ansprechenden Grafiken dargestellt. Außerdem werden die Bildungsgesetze mit Zufälligkeiten kombiniert, so dass natürlich wirkende Strukturen entstehen. Dies ist auch eine der Motivationen für die Untersuchung fraktaler Gebilde: Man möchte dadurch die in unserer natürlichen Umgebung auftretenden Fraktale besser verstehen und sie möglicherweise, zumindest innerhalb gewisser Schranken, durch bekannte, künstliche Fraktale annähern.

Damit schließt die Darstellung der Grundlagen der fraktalen Geometrie. Die fraktale Betrachtungsweise hat bereits verschiedene Anwendungen in der digitalen Bildverarbeitung und Mustererkennung gefunden. Ein interessanter Aspekt ist die Verwendung im Zusammenhang mit der Bilddatenkompression. Zu dieser Thematik, die hier nicht näher behandelt wird, sei auf [Fish92] verwiesen. Im folgenden Abschnitt werden Anwendungen der fraktalen Betrachtungsweise zur Texturerkennung untersucht.

18.3 Implementierung des Scale Space Filtering

Die in der Natur auftretenden Formen lassen sich oft, in gewissen Grenzen, der Klasse der fraktalen Strukturen zuordnen. Da diese Objekte auch in der digitalen Bildverarbeitung und Mustererkennung untersucht werden, liegt es nahe, die Erkenntnisse aus der fraktalen Geometrie auch in diesen Gebieten anzuwenden. Ursprünglich wurde die Idee der

Längenvermessung von Kurven, wie sie in Abschnitt 18.2 dargestellt wurde, sinngemäß auf digitalisierte Bilder übertragen. Das Problem ist dabei jedoch, dass natürliche Kurven nur in einem eingeschränkten Skalenbereich Skalenverhalten zeigen und die Ermittlung dieses Skalenbereichs wichtig und nicht trivial ist.

Von Witkin [Witk83] wird das *scale space filtering* vorgeschlagen, bei dem eine gegebene Struktur durch sukzessive Faltungen geglättet wird. Der Faltungskern ist von einem stetig veränderbaren Skalierungsparameter abhängig, der bei den verschiedenen Faltungen variiert wird. Nach der Glättung werden die Merkmale der Folge der geglätteten Strukturen in Abhängigkeit des Skalierungsparameters untersucht. Witkin hat in der zitierten Arbeit gezeigt, dass als Glättungsfunktion, auf Grund der notwendigen Voraussetzungen, nur die Gauß-Funktion in Frage kommt. In (18.6) ist die Faltung eines Signals $s(x)$ mit der Gauß-Funktion $g(x, \sigma)$ dargestellt. Das Ergebnis ist eine geglättete Kurve $c(x, \sigma)$:

$$c(x,\sigma) = \int_{-\infty}^{+\infty} s(u) \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-u)^2}{2\sigma^2}} du. \tag{18.6}$$

Die in Abschnitt 18.2 dargestellte Vorgehensweise von Richardson zur Ermittlung der Länge von Grenzlinien lässt sich auf das *scale space filtering* übertragen: Die Länge einer Kurve wird in Abhängigkeit des Skalierungsparameters σ der Gauß'schen Glättungskurve untersucht. Die experimentellen Ergebnisse lassen auch hier ein Potenzgesetz vermuten

$$L(\sigma) \sim \sigma^p, \tag{18.7}$$

wobei p ein „charakteristischer Skalenexponent“ ist, der linear mit einer fraktalen Dimension zusammenhängt.

Dazu ein Beispiel (Bild 18.5): Die Koch'sche Kurve wird mit unterschiedlichen σ-Werten geglättet. Die Logarithmen der Längen der geglätteten Kurven werden über die Logarithmen der unterschiedlichen Skalierungsparameter σ aufgetragen. In dieser Darstellung ergibt sich eine Gerade, deren Steigung der Parameter p ist.

Diese Vorgehensweise wird zur Erkennung von unterschiedlichen Texturen übertragen. Die grundlegende Idee ist, Umgebungen von Bildpunken (Texturfenster) mit zweidimensionalen Gauß-Funktionen mit unterschiedlichen σ-Werten zu glätten. Hier wird aber nicht wie oben die Länge, sondern die Oberfläche der Grauwertfunktion im Texturfenster verwendet. Es zeigt sich, dass auch hier wieder ein Potenzgesetz mit einem charakteristischen Skalenparameter gilt. Dieser Skalenparameter wird als Merkmalswert für die Textur dem zentralen Bildpunkt des Texturfensters zugewiesen. Diese Berechnungen werden für alle Bildpunkte durchgeführt. Es zeigt sich, dass gleiche Texturen auch gleiche Skalenparameter besitzen. Das so erzeugte Merkmalsbild kann in einem letzten Schritt in Bereiche gleicher Textur segmentiert werden.

Der Algorithmus dazu ist im Folgenden dargestellt, die einzelnen Teilprobleme werden in weiteren Abschnitten untersucht.

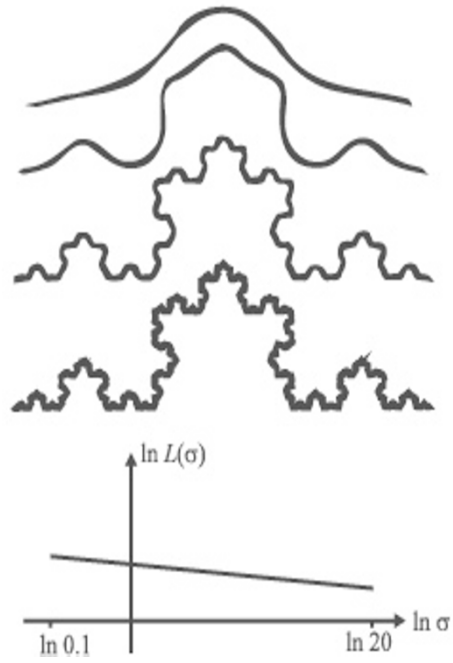


Bild 18.5: Analyse der Länge L einer Koch'schen Kurve in Abhängigkeit der Streuung σ einer Gauß'schen Glättungskurve.

A18.1: Scale Space Filtering.

Voraussetzungen und Bemerkungen:

◇ Es wird vorausgesetzt, dass das zu bearbeitende Bild als Grauwertbild $\mathbf{S}_e = (s_e(x, y))$ vorliegt. Weitere Vorverarbeitungsschritte sind nicht notwendig.

Algorithmus:

(a) Ermittlung der Größe der Grundtextur. Damit ist die Größe der Umgebung gemeint, die notwendig ist, um die Textureigenschaften von speziellen Texturen zu erfassen (Grundtexturfläche).

(b) Ermittlung der Gauß-Filterkerne für die Glättung.

(c) Berechnung der Oberflächen $O(\sigma)$ der Grauwertfunktion. Es gilt ein Potenzgesetz: $O(\sigma) \sim \sigma^p$.

Der Parameter p ist ein Skalenparameter, der für die jeweilige Textur charakteristisch ist.

(d) Berechnung der Skalenparameter p durch lineare Regression.

(e) Segmentierung.

Ende des Algorithmus

18.3.1 Ermittlung der Größe der Grundtextur

Unter der *Grundtexturfläche* versteht man die Umgebung eines Bildpunktes (x, y), die gerade hinreichend groß ist, um die charakteristischen Eigenschaften einer bestimmten Textur zu enthalten (Abschnitt 15.2). Diese Umgebung ist meistens nicht bekannt und wird auch keine regelmäßige Figur sein.

In der Praxis werden dazu rechteckige oder quadratische Umgebungen (Texturfenster) des Punktes (x, y) verwendet. Diese Umgebungen sollten jedoch die Grundtexturfläche so gut wie möglich annähern. Bei einer zu kleinen Umgebung werden nicht alle Textureigenschaften erfaßt und eine zu große Umgebung führt zu erhöhten Rechenzeiten.

Wenn der zu untersuchende Bildausschnitt verschiedene Texturen enthält, müsste man eigentlich mit unterschiedlich großen Umgebungen arbeiten, jeweils angepasst an die gerade untersuchte Textur. Dies ist in der Praxis aber nicht möglich. Wenn die Grundtexturflächen der verschiedenen Texturen sehr unterschiedlich groß sind, wird man in den Übergangsbereichen der verschiedenen Texturen breite Bereiche mit Mischtexturen erhalten, die bei der weiteren Verarbeitung, z.B. bei einer Segmentierung, störend sein können. Man könnte diese Bereiche nach der Segmentierung durch Dilatationen oder Erosionen (Kapitel 6) den benachbarten Klassen zuordnen.

In speziellen Anwendungsfällen wird man die sinnvoll zu verwendende Umgebung zur Erfassung der Grundtexturfläche am besten durch „Augenschein“ und durch Probieren ermitteln.

18.3.2 Ermittlung der Gauß-Filterkerne

Die Glättung des Eingabebildes wird im Ortsbereich mit Filterkernen der Größe $m \times m$ durchgeführt, die aus einer diskretisierten, zweidimensionalen Gauß-Funktion mit unterschiedlichen σ-Werten abgeleitet sind. Die Dichte einer zweidimensionalen Normalverteilung mit den Erwartungswerten $\xi_x = \xi_y = 0$ und den Standardabweichungen $\sigma_x = \sigma_y = \sigma$ ist gegeben durch:

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{18.8}$$

Diese Funktion muss in einem $m \times m$–Bereich um den Nullpunkt mit insgesamt m^2 Werten diskretisiert werden. Die einfachste Möglichkeit ist, an m^2 diskreten Stützstellen die Funktionswerte von $g(x, y, \sigma)$ zu ermitteln. Beispiele für Filterkerne der Größe 7×7 und $\sigma = 0.5$, $\sigma = 1.0$ und $\sigma = 2.0$ sind:

$$\sigma = 0.5: \quad \begin{pmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.011 & 0.084 & 0.011 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.084 & 0.619 & 0.084 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.011 & 0.084 & 0.011 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}, \tag{18.9}$$

$$\sigma = 1.0: \quad \begin{pmatrix} 0.000 & 0.000 & 0.001 & 0.002 & 0.001 & 0.000 & 0.000 \\ 0.000 & 0.003 & 0.013 & 0.022 & 0.013 & 0.003 & 0.000 \\ 0.001 & 0.013 & 0.059 & 0.097 & 0.059 & 0.013 & 0.001 \\ 0.002 & 0.022 & 0.097 & 0.159 & 0.097 & 0.022 & 0.002 \\ 0.001 & 0.013 & 0.059 & 0.097 & 0.059 & 0.013 & 0.001 \\ 0.000 & 0.003 & 0.013 & 0.022 & 0.013 & 0.003 & 0.000 \\ 0.000 & 0.000 & 0.001 & 0.002 & 0.001 & 0.000 & 0.000 \end{pmatrix}, \tag{18.10}$$

und

$$\sigma = 2.0: \quad \begin{pmatrix} 0.005 & 0.009 & 0.013 & 0.015 & 0.013 & 0.009 & 0.005 \\ 0.009 & 0.017 & 0.025 & 0.028 & 0.025 & 0.017 & 0.009 \\ 0.013 & 0.025 & 0.036 & 0.041 & 0.036 & 0.025 & 0.013 \\ 0.015 & 0.028 & 0.041 & 0.047 & 0.041 & 0.028 & 0.015 \\ 0.013 & 0.025 & 0.036 & 0.041 & 0.036 & 0.025 & 0.013 \\ 0.009 & 0.017 & 0.025 & 0.028 & 0.025 & 0.017 & 0.009 \\ 0.005 & 0.009 & 0.013 & 0.015 & 0.013 & 0.009 & 0.005 \end{pmatrix}. \tag{18.11}$$

Eine bessere Diskretisierung erhält man, wenn man die Umgebungen der m^2 Stützstellen berücksichtigt und die diskreten Werte durch eine numerische Integration bestimmt. Beispiele für Filterkerne der Größe 11×11 und $\sigma = 0.3$, $\sigma = 0.6$ und $\sigma = 0.9$ sind:

$$\sigma = 0.3: \quad \frac{1}{2^8} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 4 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 14 & 18 & 14 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 28 & 53 & 28 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 14 & 18 & 14 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 4 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{18.12}$$

$$\sigma = 0.6: \quad \frac{1}{2^8} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 3 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 & 6 & 7 & 6 & 3 & 1 & 0 & 0 \\ 0 & 1 & 2 & 6 & 10 & 12 & 10 & 6 & 2 & 1 & 0 \\ 0 & 1 & 3 & 7 & 12 & 15 & 12 & 7 & 3 & 1 & 0 \\ 0 & 1 & 2 & 6 & 10 & 12 & 10 & 6 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 6 & 7 & 6 & 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 3 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{18.13}$$

und

$$\sigma = 0.9: \quad \frac{1}{2^8} \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 2 & 2 & 3 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 2 & 3 & 3 & 3 & 2 & 2 & 1 & 0 \\ 1 & 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 & 1 \\ 1 & 2 & 3 & 5 & 6 & 6 & 6 & 5 & 3 & 2 & 1 \\ 1 & 2 & 3 & 5 & 6 & 7 & 6 & 5 & 3 & 2 & 1 \\ 1 & 2 & 3 & 5 & 6 & 6 & 6 & 5 & 3 & 2 & 1 \\ 1 & 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 & 1 \\ 0 & 1 & 2 & 2 & 3 & 3 & 3 & 2 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 2 & 2 & 3 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}. \tag{18.14}$$

Welche Filtergröße verwendet wird, hängt von der Größe der $m \times m$–Umgebung zur Erfassung der Grundtexturfläche ab (Abschnitt 15.2). Natürlich müssen die Filterkerne

nicht jedesmal neu berechnet werden, sondern können in einem vom *scale space filtering* unabhängigen Arbeitsgang einmal für verschiedene Umgebungen und mit verschiedenen σ-Werten berechnet und in einer Datenbank abspeichert werden.

Wenn ein Bild mit der Technik des *scale space filtering* verarbeitet werden soll, wird nun im ersten Schritt eine Folge von Gauß-gefilterten Bildern $\mathbf{C}(\sigma) = (c(x, y; \sigma))$ mit unterschiedlichen σ-Werten erzeugt.

Eine andere Möglichkeit, um zu einer Folge von tiefpassgefilterten Bildern zu kommen, ist der Einsatz der Gauß-Pyramiden (Kapitel 17). Wenn die einzelnen Schichten der Gauß-Pyramiden wieder auf die Originalgröße expandiert werden, so sind sie ebenfalls für die weitere Verarbeitung im Sinn des *scale space filtering* geeignet.

18.3.3 Berechnung der Oberflächen der Grauwertfunktion

Den nächsten Schritt bildet die Berechnung der Oberflächen der $m \times m$–Umgebungen. Die Oberfläche einer Struktur kann wie folgt berechnet werden:

$$\mathbf{O}(\sigma) = \int \sqrt{1 + \mathbf{C}_x(\sigma)^2 + \mathbf{C}_y(\sigma)^2} dxdy, \tag{18.15}$$

wobei $\mathbf{C}_x(\sigma)$ und $\mathbf{C}_y(\sigma)$ die partiellen Ableitungen der geglätteten Strukturen sind. In der praktischen Umsetzung sind dazu folgende Schritte notwendig:

- Berechnung der Ableitungsbilder $\mathbf{C}_x(\sigma)$ und $\mathbf{C}_y(\sigma)$ durch Differenzenbildung in Zeilen- und Spaltenrichtung.
- Berechnung der „Wurzelbilder“ $\mathbf{W}(\sigma) = (w(x, y; \sigma))$ gemäß

  $$w(x, y; \sigma) = \sqrt{1 + c_x(x, y; \sigma)^2 + c_y(x, y; \sigma)^2}.$$

- Berechnung der Oberfläche einer $m \times m$–Umgebung durch Aufsummierung der Werte $w(x, y; \sigma)$ in der $m \times m$–Umgebung des Bildpunktes (x, y). Dieser Wert wird dem Bildpunkt in der Position (x, y) des Ergebnisbildes $\mathbf{O}(\sigma) = (o(x, y; \sigma))$ zugewiesen.

18.3.4 Berechnung des Skalenparameters

Nach Berechnung der Oberflächenbilder von Abschnitt 18.3.3 liegt eine Folge von Bildern $\mathbf{O}(\sigma)$ vor, die durch die Wahl unterschiedlicher σ-Werte bei der Gauß-Filterung zustande kamen und deren Grauwerte ein Maß für die Oberfläche der Grauwertfunktion von jeweils $m \times m$ Bildpunkten sind.

Für jeden Bildpunkt (x, y) wird jetzt der Skalenparameter p berechnet, indem zu der Folge von Punktepaaren $(\log(o(x, y; \sigma)), \log(\sigma)), \sigma = ...$ die Regressionsgerade (Ausgleichsgerade) berechnet wird. Der Skalenparameter p ist dann die Steigung dieser Geraden. Im Ergebnisbild $\mathbf{S}_a$ hat jeder Bildpunkt den Wert des Skalenparameters p seiner $m \times m$–Umgebung.

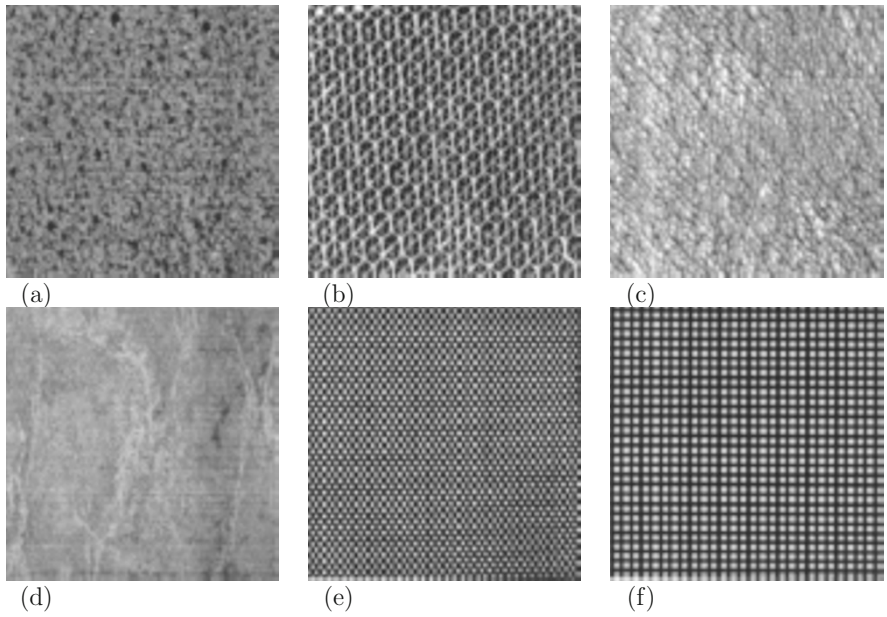


Bild 18.6: Sechs Texturen, deren Skalenverhalten untersucht wurde: (a) Schaumstoff, (b) Fahrradrückstrahler, (c) Frotteestoff, (d) Marmortapete, (e) Geschenkpapier, (f) synthetische Textur.

Wenn dieses Bild in die Grauwertmenge $G = \{0, ..., 255\}$ abgebildet wird, so werden Bereiche mit gleicher Textur gleiche Grauwerte besitzen. Im einfachsten Fall können jetzt die Bereiche mit gleicher Textur durch eine Schwellwertbildung getrennt werden. Natürlich kann das so erhaltene Texturmerkmal auch im Rahmen aufwendigerer Segmentierungsverfahren verwendet werden.

18.3.5 Beispiele und Ergebnisse

Die Texturen in Bild 18.6 wurden mit σ-Werten von 0.1, 0.2,...,0.9 und 0.95 und Umgebungen mit 5×5, 7×7, ... , 15×15 Pixel verarbeitet. In Bild 18.7 ist das Skalenverhalten dieser Texturen dargestellt.

Im markierten Bereich zeigen diese Texturen mit Ausnahme der Texturen (e) und (f) Skalenverhalten, d.h. sie gehorchen dem Potenzgesetz $O(\sigma) \sim \sigma^p$ und sind dadurch im

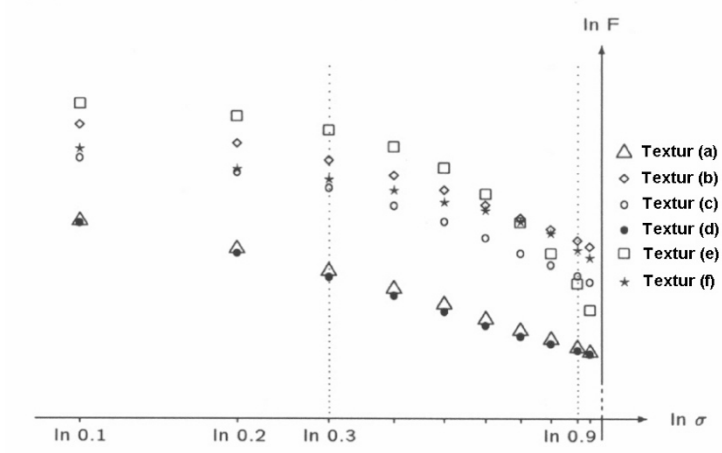


Bild 18.7: Skalenverhalten der sechs Texturen. Es zeigt sich, dass mit Ausnahme der Texturen (e) und (f) alle Texturen Skalenverhalten in einem markierten Bereich zeigen.

doppelt logarithmischen Papier durch eine Gerade repräsentiert. Die Texturen (e) und (f) sind offensichtlich keine fraktalen Texturen. Sie gehorchen deshalb auch nicht dem Potenzgesetz, was die dazu gehörigen Kurven bestätigen.

Ein Einflussparameter beim *scale space filtering* ist die Größe der $m \times m$–Umgebung, die zur Glättung und zur Berechnung der Oberflächen verwendet wird. Für die Texturen (a) und (b) sind die Ergebnisse der Filterung mit unterschiedlichen Umgebungen in Bild 18.8 gezeigt.

Bei Textur (a) ergeben sich Regressionsgeraden mit etwa gleicher Steigung. Bei Textur (b) ist das erst ab einer 13×13–Umgebung der Fall. Dies ist möglicherweise ein Hinweis, dass die Umgebung zur Erfassung der Grundtexturfläche erst ab dieser Größe ausreicht.

Die Technik des *scale space filtering* kann gut in der Qualitätskontrolle von Oberflächen eingesetzt werden: Wenn eine texturierte Oberfläche, z.B. ein Gewebe, Fehlerstellen (Löcher, Verschiebungen im Fadenlauf, ...) aufweist, so sind das Störungen in der Regelmäßigkeit der Textur. An diesen Stellen zeigt die Oberfläche ein anderes Skalenverhalten, was bedeutet, dass sich ein anderer Wert für den Skalenparameter p ergibt. Bild 18.9 zeigt Beispiele dazu. Die Bilder der oberen Reihe (18.9-a bis 18.9-c) zeigen ein Gewebe mit einer länglich ausgeprägten Fehlerstelle. Im mittleren Bild sind die Skalenparameter, abgebildet in die Grauwertmenge, dargestellt. Rechts ist das Ergebnis einer Binarisierung, in der die Fehlerstelle deutlich markiert ist. Die Teilbilder 18.9-d und 18.9-g zeigen eine Textur („Geschenkpapier“) mit einer kleinen Störung und darunter das segmentierte Ergebnis. Die Teilbilder 18.9-e und 18.9-f sind Aufnahmen desselben Gewebes, das nur mit unterschiedlichem Abstand und mit einem unterschiedlichen Drehwinkel aufgenommen wurde.

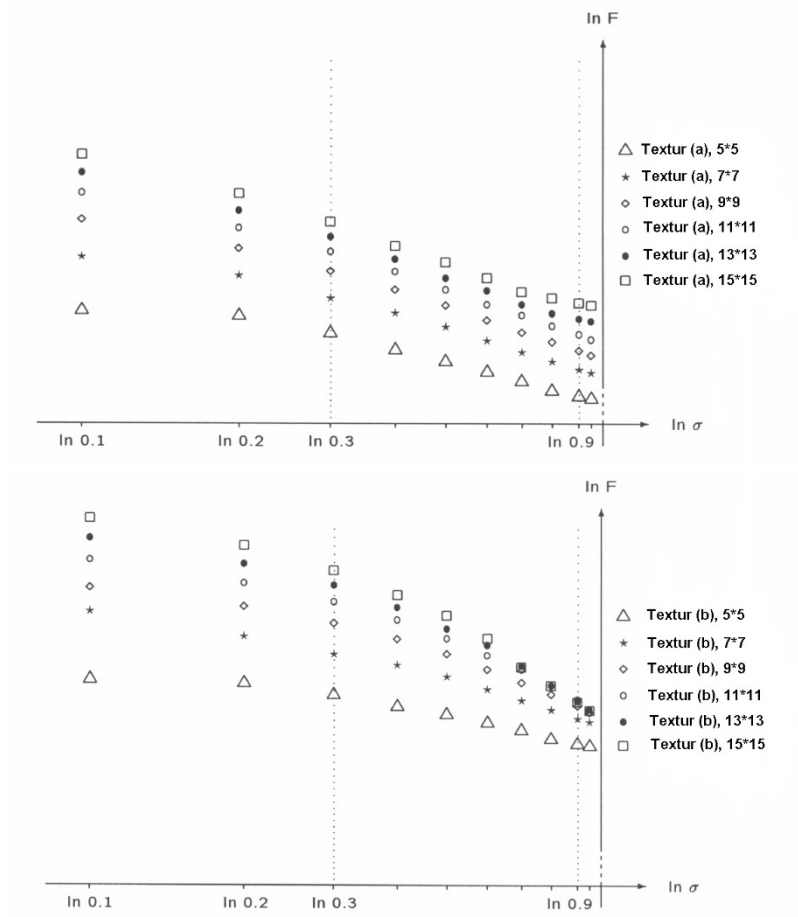


Bild 18.8: Skalenverhalten der Texturen (a) und (b) bei verschiedenen $m \times m$–Umgebungen. Bei Textur (a) ergeben sich Regressionsgeraden mit etwa gleicher Steigung. Bei Textur (b) ist das erst ab einer 13×13–Umgebung der Fall: Ein Hinweis, dass die Umgebung zur Erfassung der Grundtexturfläche erst ab dieser Größe ausreicht.

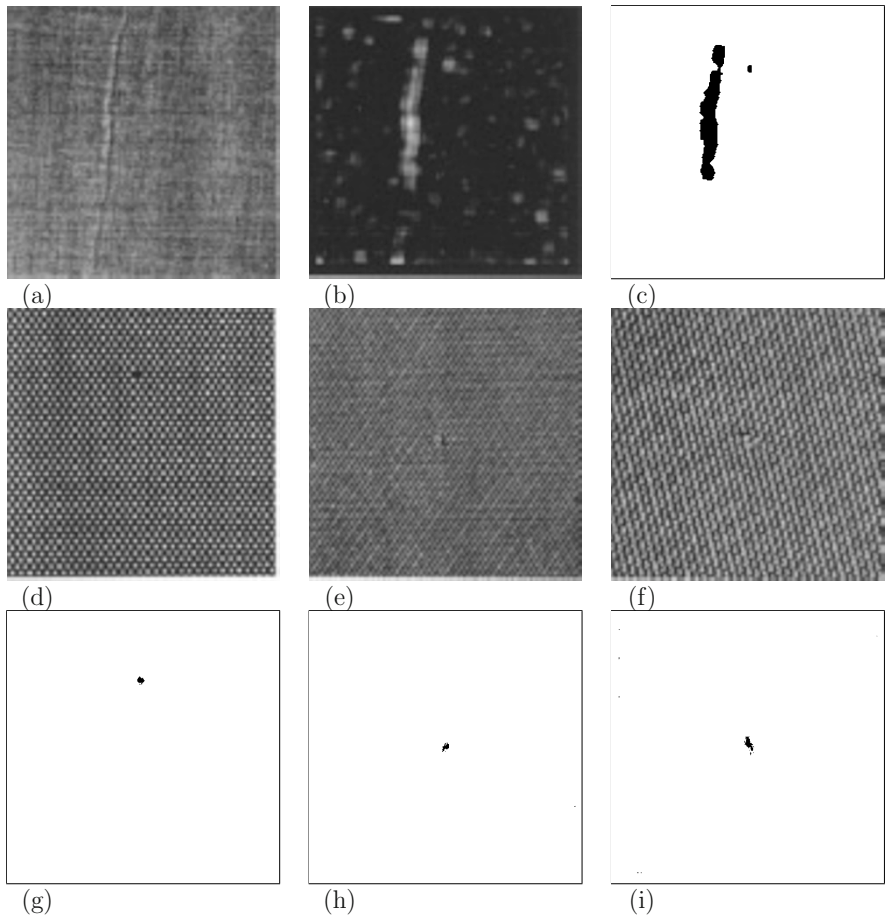


Bild 18.9: Beispiele zur Qualitätskontrolle von Oberflächen mit *scale space filtering*. (a) Gewebe mit einem länglich ausgeprägten Fehler. (b) Skalenparameter, abgebildet in die Grauwertmenge. (c) Binarisierung der Skalenparameter. Die Fehlerstelle ist deutlich markiert. (d) Textur „Geschenkpapier“. (e) Textur „Stoff“. (f) Textur „Stoff“ aus einem anderen Abstand und einem anderen Drehwinkel aufgenommen. (g) Segmentiertes Bild zu (d). (h) Segmentiertes Bild zu (e). (i) Segmentiertes Bild zu (f). Es zeigt sich, dass das *scale space filtering* größen- und rotationsinvariant ist.

Darunter sind jeweils die Ergebnisse abgebildet. Diese beiden Beispiele zeigen, dass das Verfahren des *scale space filtering* größen- und rotationsinvariant ist.

Als Merkmal kann man im Ergebnisbild alternativ (oder zusätzlich) zum Skalenparameter auch die Verschiebung der Regressionsgeraden parallel zur Ordinate verwenden. In einigen der gezeigten Beispiele wurde dieses Merkmal verwendet.

Nun noch einige Bemerkungen zum Berechnungsaufwand. Wenn man die einzelnen notwendigen Schritte betrachtet, erkennt man, dass der Aufwand zur Berechnung der Ergebnisbilder mit dem Skalenparameter p als Merkmal beträchtlich ist. In der Praxis wird man jedoch den zu untersuchenden Bereich in Vorverarbeitungsschritten eingrenzen können, so dass sich die Größe deutlich reduziert. Außerdem kann man viele der notwendigen Operationen mit spezieller Faltungshardware erledigen, mit der sich die Rechenzeiten spürbar reduzieren.

Kapitel 19

Baumstrukturen

19.1 Anwendungen

Neben den Gauß- und Laplac-Pyramiden (Kapitel 17) und der fraktalen Betrachtung beim *space scale filtering* (Kapitel 18) ist die Umwandlung eines Bildes in eine Baumstruktur eine weitere Variante der Betrachtung von Bilder in unterschiedlicher Feinheit. Es wird versucht, zusammenhängende Bildausschnitte der Größe $2 \cdot 2, 4 \cdot 4, 8 \cdot 8, \ldots$ mit homogenen Grauwerten in der Baumstruktur durch nur einen Knoten zu repräsentieren. Das bedeutet, dass Bilder mit großen zusammenhängenden, homogenen Bildbereichen sehr effektiv codiert werden können. Dieser Gesichtspunkt der *quad trees* wird vor allem in der Bilddatenreduktion und Bilddatenkompression verwendet.

Eine weitere Möglichkeit besteht in der Segmentierung von Bildern in homogene und inhomogene Bildbereiche. Auch diese Technik wird bei der Bilddatenreduktion und Bilddatenkompression verwendet, wenn homogene Bildbereiche feiner quantisiert werden sollen als inhomogene Bildbereiche.

Man kann Bilder, die als *quad trees* codiert sind, auch klassifizieren, wobei hier nicht einzelne Bildpunkte, sondern gesamte Bildbereiche, die durch einen Knoten des Baumes repräsentiert werden, einer Klasse zugewiesen werden.

Schließlich werden *quad trees* und ihre 3-dimensionale Verallgemeinerung *oct trees* auch in der 3D-Computergrafik verwendet (Band I Abschnitt 16.6.2).

19.2 Grundlegende Vorgehensweise beim Aufbau von Baumstrukturen

Der grundlegende Algorithmus wird anhand eines Binärbildes $\mathbf{S}_e = (s_e(x, y))$ mit $G = \{0, 1\}$ erläutert. Es wird zunächst angenommen, dass das Binärbild quadratisch mit einer Seitenlänge von $L = R = 2^k$ ist. Bild 19.1-a zeigt diesen Sachverhalt für ein Binärbild mit $k = 3$, also $8 \cdot 8 = 64$ Bildpunkten. Bei der Konstruktion des Baumes, der letztlich das Binärbild $\mathbf{S}_e$ repräsentieren soll, wird den $2^k \cdot 2^k$ Bildpunkten die Wurzel des Baumes

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_19

zugeordnet. Die Wurzel ist ein Knoten der Stufe k. In Bild 19.1-a ist die Wurzel ein Knoten der Stufe 3.

Im nächsten Schritt werden die $2^k \cdot 2^k$ Bildpunkte des Bildes untersucht: Falls alle Bildpunkte den Grauwert 0 (schwarz) oder alle den Grauwert 1 (weiß) besitzen, ist das gesamte Bild homogen schwarz oder weiß. In diesem Fall besteht der Baum nur aus der Wurzel, in der vermerkt ist, ob alle Bildpunkte den Grauwert 0 oder 1 haben. Falls einige Bildpunkte den Grauwert 0 und einige den Grauwert 1 besitzen, was in der Regel der Fall ist, wird das Bild in seine vier Quadranten unterteilt. Diese vier Quadranten werden, beginnend mit dem nordwestlichen, wie folgt bezeichnet (Bild 19.1-a):

$$Q_0^{(k-1)}, Q_1^{(k-1)}, Q_2^{(k-1)}, Q_3^{(k-1)}.$$

Jeder der vier Quadranten, der $2^{(k-1)\cdot(k-1)}$ Bildpunkte umfasst, wird in der Baumstruktur durch einen Knoten der Stufe $l = k - 1$ repräsentiert. Die Wurzel des Baumes wird durch vier Kanten mit jedem ihrer „Söhne“ (Nachfolger) verbunden. Nun wird für jeden der vier Quadranten sinngemäß dieselbe Untersuchung wie oben durchgeführt: Nur im Fall, dass in einem Quadranten die Grauwerte 0 und 1 auftreten, wird dieser wiederum in seine vier Quadranten unterteilt, denen im Baum Knoten der Stufe $l = k-2$ zugeordnet werden. Die Unterteilung in Quadranten endet spätestens nach k Schritten, wenn die Stufe der einzelnen Bildpunkte erreicht ist. Einem Bildpunkt ist in der Baumstruktur ein Knoten der Stufe $l = 0$ zugeordnet.

Knoten, die keine Nachfolger besitzen, werden als *Blätter des Baumes* oder *homogene Knoten* bezeichnet, da sie Bildausschnitte repräsentieren, die homogen schwarz oder weiß sind.

Die Adressierung der Knoten des Baumes erfolgt mit k-stelligen Zahlen zur Basis 4:

$$a_{k-1}, a_{k-2}, ..., a_1, a_0.$$

Die Ziffern $a_i \in (0, 1, 2, 3)$ stehen für die Position des zugehörigen Quadranten im Rahmen des ihn umgebenden Quadranten: 0-Nordwest, 1-Nordost, 2-Südwest und 3-Südost. Der Index i kennzeichnet die Stufe des Knotens. Durch die Stufennummer und die Knotenadresse ist jeder Knoten des Baumes eindeutig mit dem Bildausschnitt verbunden, den er repräsentiert. Die Wurzel in Bild 19.1-c hat die Stufennummer $l = 3$ und die Knotenadresse 000. Der nordöstliche Quadrant Q_1 hat die Stufennummer $l = 2$ und die Knotenadresse 100, und dem schraffiert markierten Bildpunkt entspricht ein Knoten der Stufe $l = 0$ mit der Knotenadresse 212.

Als Nächstes ist zu untersuchen, welche Informationen bei der Implementierung der Baumstruktur in den einzelnen Knoten abgespeichert werden müssen. Zunächst muss jeder Knoten seine Stufennummer und seine Knotenadresse enthalten, damit von der Baumstruktur aus der direkte Bezug zum Bild sichergestellt ist. Nur dadurch ist z.B. das Durchlaufen eines Pfades durch den Baum möglich. Weiter muss der Typ des Knotens festgehalten werden. Der Typ gibt an, ob es sich um einen homogenen weißen oder schwarzen Knoten handelt oder ob der Knoten inhomogen ist und Nachfolger besitzt. Die Kanten zwischen

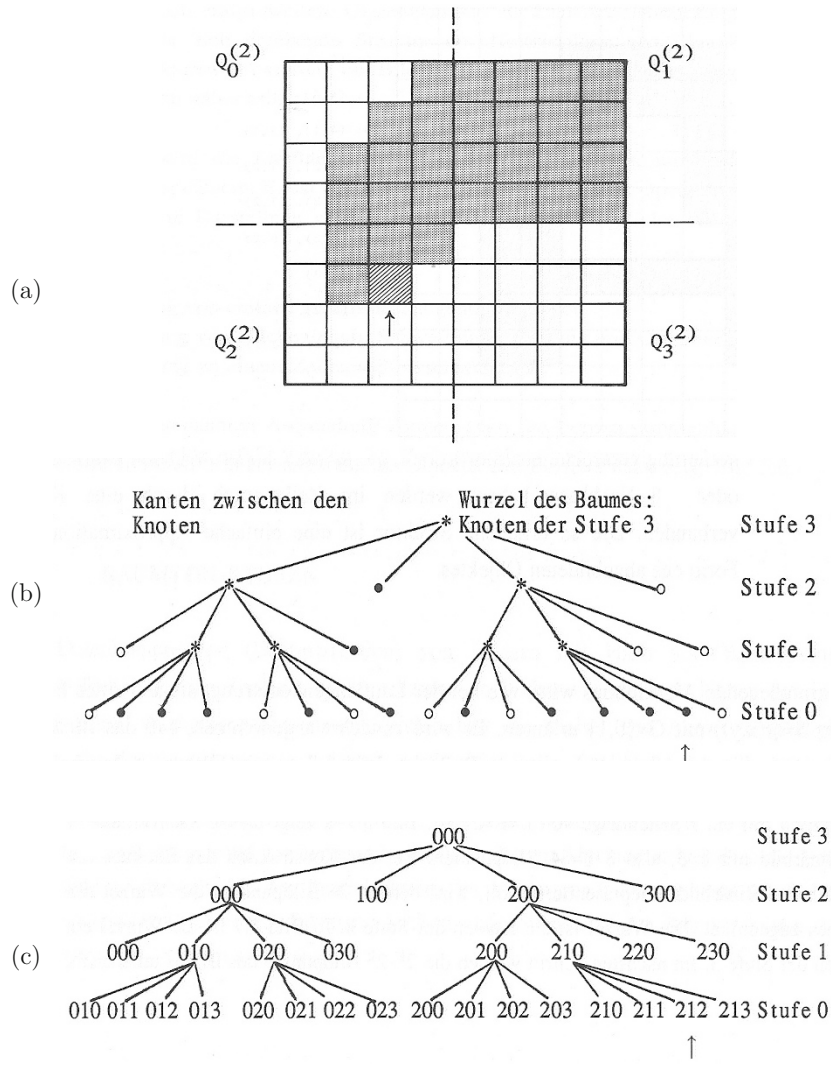


Bild 19.1: Beispiele zur Darstellung eines Bildes als Baumstruktur. (a) Originalbild mit $2^3 \cdot 2^3$ Bildpunkten. (b) Baumstruktur. (c) Adressierung der Knoten.

2	0	2	1
1	2	2	0
1	0	0	0
1	0	1	0
2	2	1	1
1	0	1	0
0	0	0	1

Tabelle 19.1: Baumstruktur zum Binärbild in Bild 19.1-a. Auf die Abspeicherung der Stufennummer, der Knotenadresse und der Adressverweise wurde in dieser Implementierungsvariante verzichtet. Die Rekonstruktion des Originals ist durch die spezielle, rekursive Speicherungstechnik möglich.

den Knoten werden durch Adressverweise realisiert. Jeder Knoten, mit Ausnahme der Blätter, hat vier Adressverweise auf seine vier Nachfolger („Söhne“) und, mit Ausnahme der Wurzel, einen Adressverweis auf seinen Vorgänger („Vater“). Zusätzlich können in jedem Knoten noch weitere Informationen abgelegt werden, die jedoch abhängig vom jeweiligen Anwendungsfall sind.

Neben der hier beschriebenen Implementierung der Baumstruktur sind noch andere Varianten möglich. Wird in einer aktuellen Anwendung dem durch die Baumstruktur erzielten Kompressionsfaktor mehr Bedeutung beigemessen als der flexiblen Auswertungs- und Weiterverarbeitungsmöglichkeit, so kann z.B. auf die Stufennummer, die Knotenadresse und die Adressverweise der Kanten verzichtet werden. Allerdings muss dann auf Grund einer speziellen, rekursiven Speicherungstechnik diese Information bei der Rekonstruktion des Originals wieder abgeleitet werden können. Dazu als Beispiel der Sachverhalt von Bild 19.1. Zur Darstellung des Typs eines Knotens (schwarz, weiß oder schwarz/weiß) benötigt man 2 Bit. Man könnte also festlegen: 0 schwarzer Knoten, 1 weißer Knoten und 2 schwarz/weiß gemischter Knoten, der weiter unterteilt ist. Die vier Knoten der Stufe 2 können dann durch das Codewort 2021 beschrieben werden. Jetzt wird zu jeder 2 in diesem Codewort, die ja ein Hinweis darauf ist, dass der Knoten weiter unterteilt ist, der zugehörige Teilbaum abgespeichert. Dieses rekursive Verfahren endet für den jeweiligen Teilbaum, wenn in einem Codewort nur mehr Nullen und Einsen enthalten sind. Für den Baum von Bild 19.1-b ergibt sich dann die Struktur von Tabelle 19.1.

Als weitere Variante ist es möglich, von der Unterteilung in vier Quadranten abzugehen und z.B. eine Zerlegung in $4 \cdot 4$ Teilquadrate vorzunehmen, was einer Aufteilung in 16 Knoten entspricht. Ein Codewort, das diese 16 Knoten in der oben beschriebenen Weise darstellt, benötigt 32 Bit. Implementierungen dieser Art lassen sich leicht auf byteorientierten Rechenanlagen durchführen. Es sei aber darauf hingewiesen, dass durch das Weglassen der Stufennummer, der Knotenadresse und der Adressverweise zwar ein optimaler Kompressionsfaktor erreicht wurde, jedoch die Auswertungsmöglichkeiten der Baumstruktur eingeschränkt wurden. In den folgenden Betrachtungen wird immer vorausgesetzt, dass in

den Knoten die volle Information verfügbar ist.

Eine Möglichkeit der Weiterverarbeitung von Binärbildern, die als Baumstrukturen codiert sind, ist die logische Verknüpfung von zwei Bäumen. Dazu müssen zunächst die Boole'schen Operatoren NOT, AND und OR für Bäume als Verknüpfungsobjekte implementiert werden. Dies ist direkt in der Baumstruktur möglich, es ist also nicht notwendig, das als Baum gespeicherte Bild zuerst in die Rasterdarstellung zurückzuwandeln. Die logische Verknüpfung von Bäumen ist auch in der komprimierten Form gemäß Bild 19.1 möglich. Es seien $\mathbf{S}_1$ und $\mathbf{S}_2$ zwei Binärbilder in ihrer Darstellung als Baumstruktur. Dann wird z.B. durch die logische Operation NOT $\mathbf{S}_1$ der Baum erzeugt, der dem Negativbild von $\mathbf{S}_1$ entspricht. Durch $\mathbf{S}_1$ OR $\mathbf{S}_2$ wird der Baum erzeugt, der sowohl den Bildinhalt von $\mathbf{S}_1$ als auch von $\mathbf{S}_2$ enthält, also die Vereinigung der beiden Originale $\mathbf{S}_1$ und $\mathbf{S}_2$. Schließlich erzeugt $\mathbf{S}_1$ AND $\mathbf{S}_2$ einen Baum, in dem nur diejenigen Bildbereiche den Grauwert 1 haben, die sowohl in $\mathbf{S}_1$ als auch in $\mathbf{S}_2$ den Grauwert 1 haben, was dem Durchschnitt der beiden Originale entspricht.

Dazu ein praktisches Beispiel: $\mathbf{S}_1$ sei eine Strichzeichnung, z.B. eine technische Dokumentation, eine Explosionszeichnung oder ein Kartenausschnitt. Nun soll aus diesem als Baumstruktur gespeicherten Bild ein rechteckiges Fenster ausgeblendet werden. $\mathbf{S}_2$ ist dann die Baumstruktur zu einem Rechteckausschnitt, der innerhalb des auszublendenden Bereiches den Grauwert 1 und außerhalb den Grauwert 0 hat. $\mathbf{S}_1$ AND $\mathbf{S}_2$ liefert den Baum, der nur die Bildinformation von $\mathbf{S}_1$ innerhalb des mit $\mathbf{S}_1$ definierten Fensters hat. Anwendungen dieser Art sind bei der digitalen Verarbeitung von Strichzeichnungen sinnvoll, die auf Grund ihres Bildinhaltes nicht mit grafischen, vektoriellen Systemen verarbeitet werden können. Da diese Bilder in der Regel mit einem sehr feinen Raster digitalisiert werden müssen, fallen große Datenmengen an. Zur Kompression können hier die Baumstrukturen eingesetzt werden. Es besteht dann die Möglichkeit einer interaktiven Weiterverarbeitung, z.B. einer Modifikation der Zeichnung, wie oben geschildert, direkt in der komprimierten Datenstruktur.

Bei der Beschreibung des grundlegenden Algorithmus wurde angenommen, dass die Binärbildvorlagen quadratisch sind und ihre Seitenlänge eine Potenz von zwei ist. Dies kann bei vielen praktischen Anwendungen nicht vorausgesetzt werden. Um trotzdem einen Baum aufbauen zu können, muss das i. allg. rechteckige Bild mit einem definierten Hintergrundgrauwert wie in Bild 19.2 dargestellt, aufgefüllt werden, um zu einem quadratischen Bild mit einer 2-er Potenz als Seitenlänge zu kommen.

Abschließend zu diesem Abschnitt noch einige Bemerkungen, die jedoch im nächsten Abschnitt noch ausführlicher behandelt werden. Beim Aufbau des Baumes zu einem Binärbild wurde als Kriterium, ob ein Quadrant unterteilt wird oder nicht, die Homogenität „ganz weiß“, „ganz schwarz“ oder alternativ dazu „schwarz/weiß“ verwendet. Soll der Baumalgorithmus auf Grautonbilder verallgemeinert werden, so muss ein anderes Homogenitätskriterium verwendet werden, z.B. ob alle Grauwerte des untersuchten Ausschnittes innerhalb eines bestimmten Grauwertintervalls liegen. Ist dies nicht der Fall, wird der Ausschnitt in seine vier Quadranten unterteilt. Repräsentierend für das Grauwertintervall wird in einem Feld „TYP“ jedes Knotens der Mittelwert des Intervalls eingetragen. Es ist offensichtlich, dass ein so codiertes Grauwertbild nicht mehr eindeutig reproduziert werden

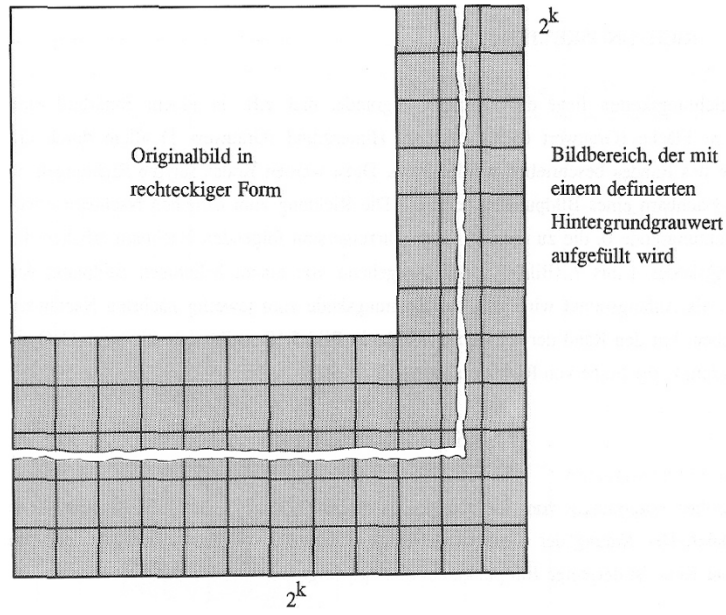


Bild 19.2: Rändelung eines rechteckigen Originalbildes mit einem definierten Hintergrundgrauwert, um zu einem quadratischen Bild mit einer 2-er Potenz als Seitenlänge zu kommen, das in eine Baumstruktur gewandelt werden kann.

kann. Ein Beispiel einer bildlichen Darstellung eines als Baum codierten Grauwertbildes wird im nächsten Abschnitt gegeben.

19.3 Regionenorientierte Bildsegmentierung mit *quad trees*

In Abschnitt 19.2 wurde auf die Darstellung von Bildern, besonders von Binärbildern, als Baumstrukturen eingegangen. Die Motivation des Einsatzes der Baumstrukturen war dort die Möglichkeit der Datenkompression. Es wurde aber bereits darauf hingewiesen, dass diese Technik auch gut zur Bildsegmentierung geeignet ist.

Beim Aufbau eines Baumes muss ein Homogenitätskriterium H angegeben werden, wann die Unterteilung in die vier Quadranten erfolgen soll. Im Fall von Binär- oder Zweipegelbildern wurde in Abschnitt 19.2 für H verwendet, ob alle Bildpunkte des betrachteten Quadranten weiß, schwarz oder schwarz und weiß sind. Dieses Kriterium kann folgendermaßen formuliert werden:

$$H_1 : min = max, \tag{19.1}$$

wobei min und max der minimale und der maximale Grauwert des Bildausschnittes sind.

Bei Grauwertbildern $\mathbf{S} = (s(x, y))$, $G = \{0, 1, ..., 255\}$ wird die Aufteilung in die vier Quadranten nicht durchgeführt, wenn die Grauwerte des Bildausschnittes in bestimmten Grenzen liegen. Damit lautet das Homogenitätskriterium z.B.

$$H_2 : \max\{|mean - max|, |mean - min|\} < c \tag{19.2}$$

oder

$$H_3 : max - min < c, \tag{19.3}$$

wobei $mean$, max und min der Mittelwert, der maximale und der minimale Grauwert des Bildausschnittes sind. c ist ein Schwellwert, mit dem festgelegt werden kann, wie breit das Grauwertintervall ist, in dem die Grauwerte zu einem Knoten zusammengefasst werden.

Die Bildfolge 19.3 zeigt ein Beispiel eines als *quad tree* codierten Grauwertbildes. Es hat ein Originalgröße von $(256 = 2^8) \cdot (256 = 2^8)$ Bildpunkten. Somit hat die Baumstruktur die Stufen 0 bis 8: Die Wurzel ist ein Knoten der Stufe 8, die einzelnen Bildpunkte sind Knoten der Stufe 0. Von links oben nach rechts unten sind die Stufen 8 bis 0 des Bildes als Grauwertbilder dargestellt, wobei die Flächen, die ein Knoten im Originalbild repräsentiert, durch den mittleren Grauwert dargestellt sind.

Bei mehrkanaligen Bildern $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1, ..., N - 1$ wird nicht aufgeteilt, wenn H für alle Kanäle zutrifft, also:

$$H_4 : \max\{|mean_i - max_i|, |maen_i - min_i|\} < c, i = 0, 1, ..., N - 1. \tag{19.4}$$

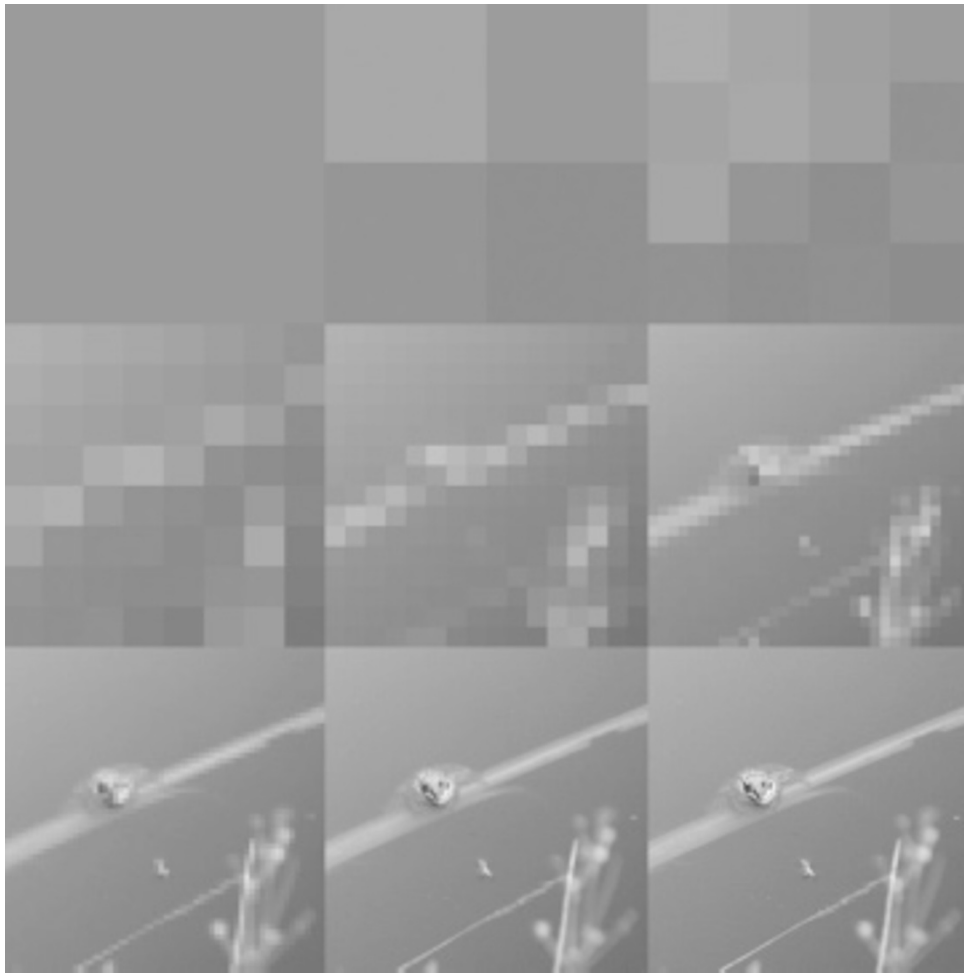


Bild 19.3: Als *quad tree* codiertes Grauwertbild der Größe $(256 = 2^8) \cdot (256 = 2^8)$ Bildpunkte. Die Baumstruktur hat die Stufen 0 bis 8: Die Wurzel ist ein Knoten der Stufe 8, die einzelnen Bildpunkte sind Knoten der Stufe 0. Von links oben nach rechts unten sind die Stufen 8 (Wurzel) bis 0 (einzelne Bildpunkte) des Bildes als Grauwertbilder dargestellt, wobei die Flächen die ein Knoten im Originalbild repräsentiert, durch den mittleren Grauwert dargestellt sind.

Dabei sind $mean_i, max_i$ und min_i wie oben die entsprechenden Werte für jeden der N Kanäle.

Die Wahl des Schwellwertes c beeinflusst den Baumaufbau wesentlich. Bei kleinem c ergibt sich eine große Knotenzahl, wodurch der Speicherbedarf und die Rechenzeit für nachfolgende Auswertungen steigt. Bei zu großem Schwellwert c geht Bildinformation verloren, da Bildinhalte mit geringen Grauwertunterschieden in homogenen Flächen aufgehen. Hier muss somit ein Kompromiss zwischen Speicher- und Rechenzeitaufwand und der Qualität des als Baum kodierten Bildes gefunden werden. Bei praktischen Anwendungen wird der Schwellwert c am besten empirisch ermittelt. So kann z.B. die Untersuchung der Grauwerte in strukturierten Bildbereichen, die von der Baumstruktur noch aufgelöst werden sollen, einen Hinweis auf den Schwellwert c geben.

Der Einfluss des Schwellwertes c wird mit den Bildern 19.4-a bis 19.4-d dokumentiert. Die Bilder wurden mit den Schwellwerten $c = 1, c = 4, c = 16$ und $c = 64$ und H_3 als Homogenitätskriterium codiert. Anschließend wurden sie in ein normales Rasterbild zurückgewandelt, um so die Auswirkung der unterschiedlichen Schwellwerte auf die Bildqualität darstellen zu können. Bei Bild 19.4-a ($c = 1$) wird unterteilt, wenn in dem jeweiligen Quadranten auch nur ein Bildpunkt ist, dessen Grauwert anders ist als die Grauwerte der restlichen Bildpunkte. Das bedeutet, dass die Baumstruktur das Grauwertbild exakt repräsentiert. Bei der Rückrasterung entsteht wieder das Original. Bei einem Schwellwert von $c = 4$ wird unterteilt, wenn die Abweichung in einem Quadranten größer oder gleich 4 ist (Bild 19.4-b). Hier ist im zurückgerasterten Bild noch kein wahrnehmbarer Unterschied zum Original festzustellen. Bei $c = 16$ (Bild 19.4-c) sieht man im ziemlich homogenen Hintergrund bereits eine Verschlechterung. Sehr deutlich wird es bei $c = 64$ (Bild 19.4-d). Allerdings werden die sehr fein strukturierten und kontrastreichen Bildbereiche (der Kopf des Frosches) auch hier noch bis auf Pixelebene aufgelöst. Dieser Sachverhalt ist in den drei Bildern Bild 19.5-a bis Bild 19.5-c dargestellt. Bild 19.5-c ist das Differenzbild zwischen dem Original (19.5-a) und der zurückgerasterten Baumstruktur mit $c = 64$. Zu den Differenzen wurde der Grauwert 128 addiert und das Ergebnis im Kontrast etwa angehoben, um die Unterschiede besser sichtbar zu machen. Man sieht hier deutlich, dass der stark strukturierte Kopf des Frosches noch fein aufgelöst wird.

Für jeden Knoten müssen Informationen gespeichert werden, die die Lage des Knotens im Baum beschreiben (Verweise zum Vater und zu den vier Söhnen), eine Zuordnung des durch den Knoten repräsentierten Ausschnittes im Originalbild gestatten (Knotenadresse und Stufe des Knotens) und eine möglichst gute Rekonstruktion der Grauwerte des Bildausschnittes erlauben. In Tabelle 19.2 sind die Knoteninformationen bei einer Anwendung der Baumstrukturen zur Bildsegmentierung von digitalisierten Luftbildern mit der Zielsetzung der Erstellung von thematischen Karten zusammengestellt ([Habe83], [Habe84]). Die digitalisierten Luftbilder lagen in diesem Fall als dreikanalige Bilder mit Rot-, Grün- und Blauauszug vor.

Der Algorithmus zum Aufbau der Baumstruktur wurde in Abschnitt 19.2 beginnend mit Wurzel des Baumes, also *top-down*, beschrieben. Diese Vorgehensweise hat den Nachteil, dass entweder große Bildausschnitte im Hauptspeicher des Datenverarbeitungssystems gehalten werden müssen oder viele Zugriffe auf den Hintergrundspeicher erfolgen. Die Baum-

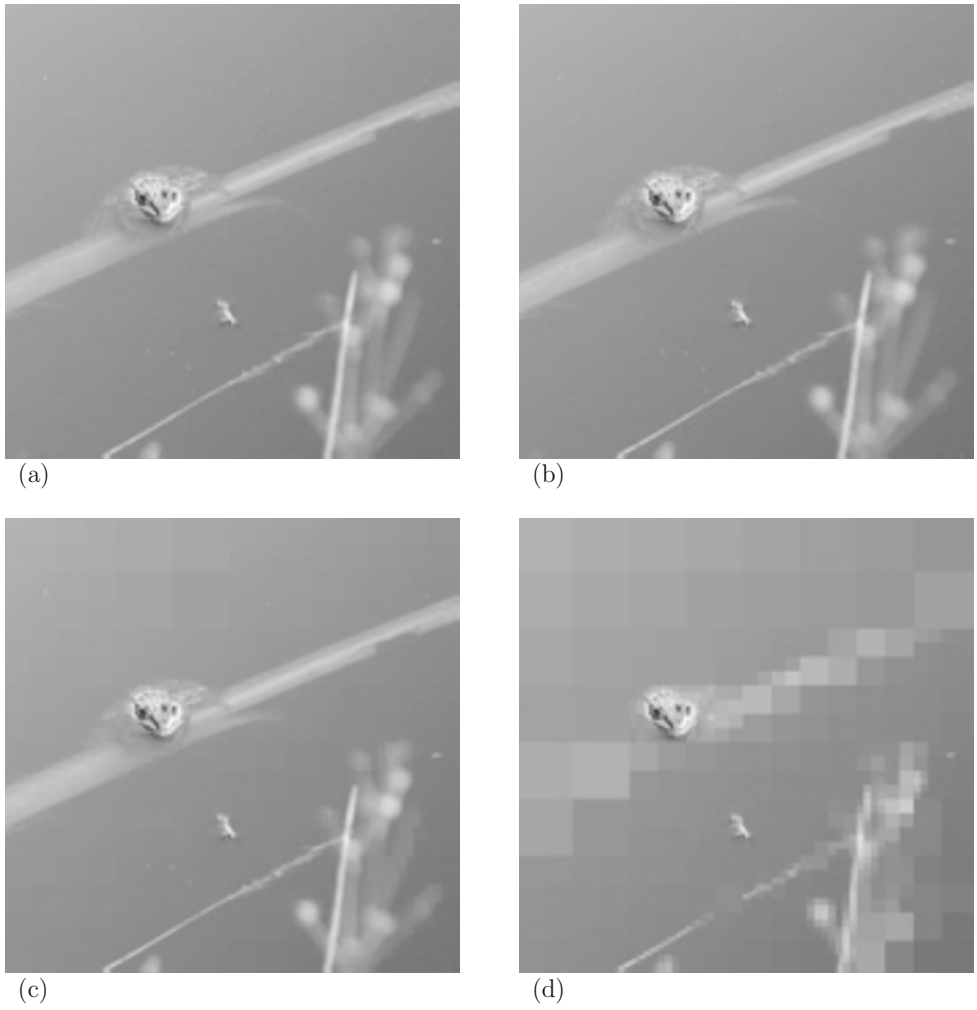
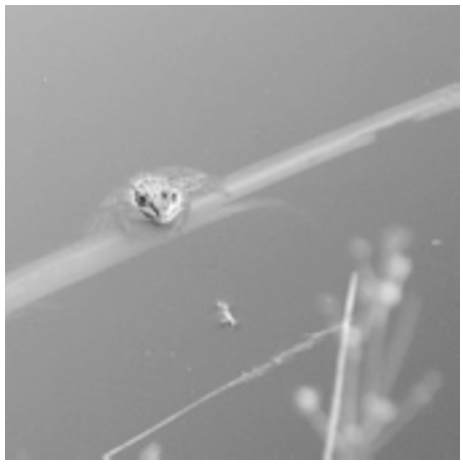
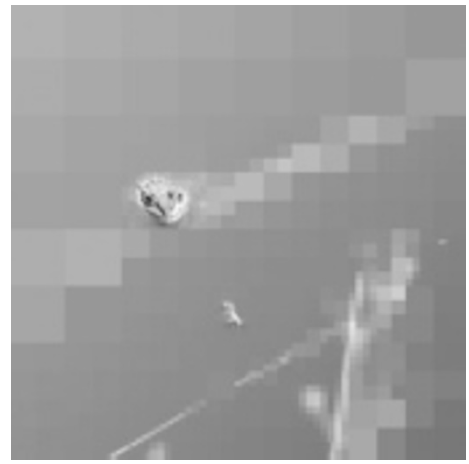


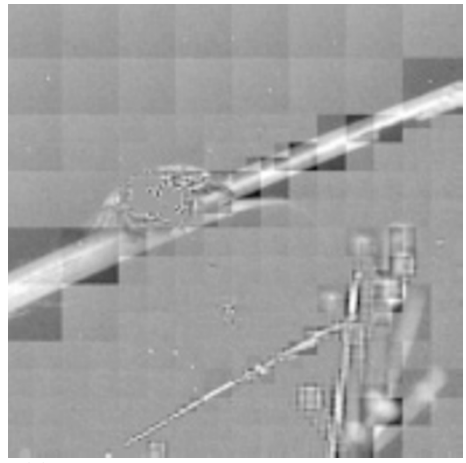
Bild 19.4: Der Einfluss des Schwellwertes c auf die Bildqualität bei der Rückrasterung. Je größer der Schwellwert, desto größere Grauwertintervalle werden zu einem Knoten zusammengefasst.



(a)



(b)



(c)

Bild 19.5: Differenzbild zwischen dem Original (a) und der mit einem Schwellwerte $c = 64$ erzeugten und zurückgerasterten Baumstruktur. Man sieht hier deutlich, dass der stark strukturierte Kopf des Frosches noch fein aufgelöst wird.

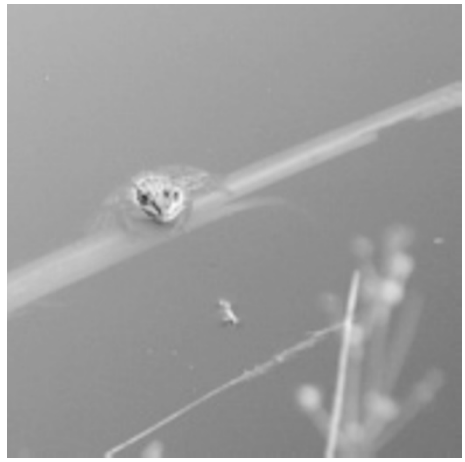
Zelle	Byte	Bezeichnung	Bedeutung
1	1	STUFE	Stufe l des Knotens
2	4	ADRESSE	Knotenadresse
3	4	VATER	Verweis auf den Vater
4	4	SOHN 0	Verweis auf den NW-Nachfolger
5	4	SOHN 1	Verweis auf den NO-Nachfolger
6	4	SOHN 2	Verweis auf den SW-Nachfolger
7	4	SOHN 3	Verweis auf den SO-Nachfolger
8	1	MITTEL	Mittelwert der $2^l \cdot 2^l$ Bildpunkte aus allen Kanälen
9	1	MINIMUM	Minimum der $2^l \cdot 2^l$ Bildpunkte aus allen Kanälen
10	1	MAXIMUM	Maximum der $2^l \cdot 2^l$ Bildpunkte aus allen Kanälen
11	1	MITTEL-GRÜN	Mittelwert der $2^l \cdot 2^l$ Bildpunkte im Grünkanal
12	1	MITTEL-ROT	Mittelwert der $2^l \cdot 2^l$ Bildpunkte im Rotkanal
13	1	MITTEL-BLAU	Mittelwert der $2^l \cdot 2^l$ Bildpunkte im Blaukanal
14	2	NACHFOLGER	Anzahl der Nachfolgeknoten des zugehörigen Teilbaumes
15	2	KNOTENCODE	(Zwischen-)Ergebniscode der Bildsegmentierung

Tabelle 19.2: Beispiel zur Struktur eines Baumknotens

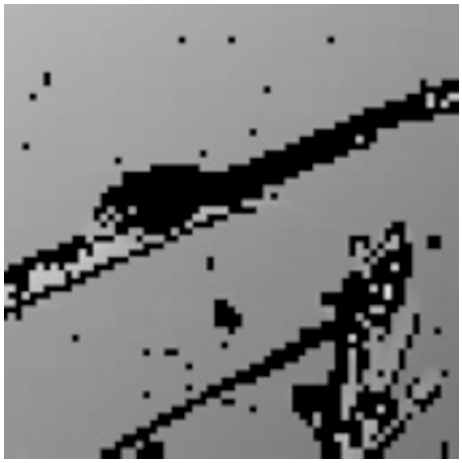
struktur kann auch während des sequentiellen Einlesens der Originalbilddaten von einem Hintergrundspeicher (oder einem Scanner) durchgeführt werden, wobei immer nur maximal zwei Bildzeilen im Hintergrundspeicher benötigt werden. Zu jeweils zwei Bildzeilen werden Teilbäume erzeugt, deren Wurzeln die Stufe 1 haben. Vier benachbarte Teilbäume einer Stufe $l, 1 \leq l \leq k-1$, werden dann zu einem Teilbaum der Stufe $l+1$ zusammengefasst, wobei durchaus der neue Teilbaum nur aus der Wurzel bestehen kann, wenn seine vier Söhne das Homogenitätskriterium erfüllen. Das Verfahren endet, wenn die letzten vier Knoten der Stufe $k-1$ zur Wurzel des Baumes zusammengefasst werden.

Um eine weitere Verarbeitung von Bildern, die als Baumstrukturen gespeichert sind, zu ermöglichen, müssen eine Reihe von Basisalgorithmen für Baumstrukturen implementiert sein. Ein Beispiel dazu ist ein Algorithmus, der ein als Baum gespeichertes Bild wieder in die normale Rasterbilddarstellung zurückwandelt. Ein derartiges Programm kann so implementiert werden, dass Ausgabebilder erzeugt werden, in denen homogene und inhomogene Bildbereiche getrennt dargestellt werden können, was ein erster Ansatz zur automatischen Aufteilung eines Bildes in homogene, also unstrukturierte, und inhomogene, also texturierte, Bildbereiche ist. In Bild 19.6-b und Bild 19.6-c wird dieser Sachverhalt am Beispiel eines Testbildes dargestellt. Bild 19.6-b zeigt alle homogenen Knoten bis zur minimalen Stufe l=3, während Bild 19.6-c alle inhomogenen Knoten bis zur maximalen Stufe 2 zeigt. Beim Aufbau des Baumes wurde hier das Homogenitätskriterium H_3 mit einem Schwellwert $c = 8$ verwendet.

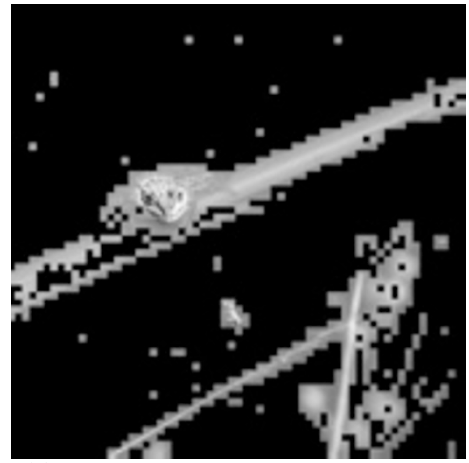
Bei vielen Anwendungen ist es notwendig, zur Inspektion von benachbarten Bildberei-



(a)



(b)



(c)

Bild 19.6: Rückrasterung des als Baumstruktur gespeicherten Testbildes. (a) Original (b) zeigt die homogenen Knoten bis zur minimalen Stufe $l = 3$ und (c) die inhomogenen Knoten bis zur maximalen Stufe $l = 2$.

chen in der Baumstruktur Pfade zu durchlaufen. Hierzu werden Nachbarschaftsalgorithmen benötigt, die es erlauben, ausgehend von einem beliebigen Knoten die Nachbarknoten in den acht Hauptrichtungen zu ermitteln. Das Auffinden eines Nachbarknotens basiert auf dem Auffinden eines gemeinsamen Vorgängers. Wird der Pfad zum gemeinsamen Vorgänger anschließend gespiegelt durchlaufen, so führt dieser Pfad zum Nachbarknoten [Same82]. Da die Stufe des Nachbarknotens höher, gleich oder niedriger sein kann, müssen hier Fallunterscheidungen berücksichtigt werden. Zunächst wird der Nachbar mit derselben oder größeren Stufe in einer der vier Richtungen Nord, West, Süd oder Ost ermittelt. Ist der Nachbarknoten in der gewünschten Richtung nicht homogen, so werden im zweiten Schritt die direkt anliegenden Nachbarn gesucht. Soll ein diagonal liegender Nachbar, z.B. der nordöstliche Nachbar eines Knotens, ermittelt werden, so wird dies zweistufig durchgeführt: Zuerst wird der östliche Nachbar und dann von diesem der nördliche bestimmt. Wegen der schon erwähnten Möglichkeit der unterschiedlichen Knotenstufen sind auch hier viele Fallunterscheidungen zu beachten.

Die Baumstrukturen legen es nahe, bei der Segmentierung Strategien einzusetzen, die auf die jeweilige Objektklasse abgestimmt sind. Wie schon oben erwähnt, ist es einfach, mit Hilfe der Baumstrukturen homogene von inhomogenen Bildbereichen abzugrenzen. Auf die homogenen Bildbereiche (z.B. Wasserflächen in einem Luftbild) werden dann spezielle Objektklassifikatoren angewendet. Ein derartiger Objektklassifikator wird im ersten Schritt alle homogenen Knoten des Baumes anhand bildpunktbezogener Merkmale klassifizieren und feststellen, ob es sich dabei um die gesuchte Klasse handelt. Dabei ist anzumerken, dass für größere homogene Flächen, d.h. für Ausschnitte, die durch Knoten mit einer höheren Stufennummer repräsentiert werden, nur eine Klassifizierungsentscheidung notwendig ist, was wesentlich zur Einsparung von Rechenzeit beiträgt.

Um im nächsten Schritt größere homogene Flächen zu extrahieren, werden alle Knoten, die im ersten Schritt als zur gesuchten Klasse gehörig erkannt wurden und eine bestimmte Minimalgröße erfüllen, darauf hin untersucht, ob sie ganz von Knoten derselben Art umgeben sind (8-Nachbarschaft). Diejenigen Knoten, für die diese Bedingungen erfüllt sind, werden gesondert markiert, da sie keiner weiteren Verarbeitung mehr unterzogen werden müssen. Zu den verbleibenden Knoten der gesuchten Klasse wird eine Randsuche durchgeführt, bei der schalenartig Knoten bis zur Stufe 0 der bereits erkannten Fläche zugewiesen werden.

Als Nächstes ein Beispiel für einen Objektklassifikator, der flächig ausgedehnte, inhomogene Bildbereiche segmentiert (z.B. Waldflächen in einem Luftbild). Ausgehend von dem Grundgedanken, dass die gesuchte Klasse großflächig ausgeprägt ist, wird zunächst ein Baumstrukturmaß ausgewertet, das bereits beim Baumaufbau berechnet werden kann. Dieses Maß beschreibt die Anzahl der Nachfolgeknoten eines bestimmten Knotens. Hat z.B. die Wurzel eines (Teil-)Baumes nur wenige Nachfolger, so repräsentiert sie ein Bild, das weitgehend aus homogenen Bereichen aufgebaut ist. Folgt auf die Wurzel des (Teil-) Baumes dagegen die maximal mögliche Anzahl von Nachfolgeknoten, die sich gemäß

$$N = \sum_{l=0}^{k} 4^l \tag{19.5}$$

berechnet, so beschreibt sie einen Bildausschnitt, der eine maximale Strukturiertheit aufweist. Inhomogene Knoten, die texturierte Gebiete repräsentieren, zeichnen sich also durch eine große Anzahl von Nachfolgern aus. Legt man nun für Knoten einer bestimmten Stufe eine Schwelle für die Anzahl der Nachfolgeknoten fest, so kann der Baum in Knoten, die texturierte und untexturierte Bildbereiche beschreiben, eingeteilt werden. Die so erkannten inhomogenen Knoten der festgelegten Stufe werden nun einer weiteren Texturanalyse in Richtung der gesuchten Klasse unterzogen, z.B. mit Hilfe der in Abschnitt 3.10 erläuterten Co-occurrence-Matrizen.

Inhomogene Knoten können auch von linienhaft ausgeprägten Objekten (z.B. Straßen) erzeugt worden sein. Zur Segmentierung derartiger Objekte ist eine Linienverfolgung (Kapitel 7) in der Baumstruktur möglich.

Kapitel 20

Segmentierung und numerische Klassifikation

20.1 Grundlegende Problemstellung

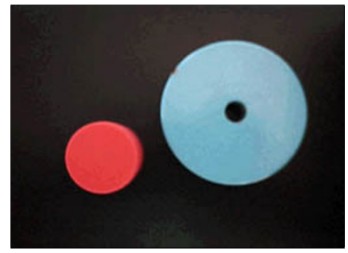
Die grundlegende Problemstellung bei der Segmentierung sei anhand eines Beispiels erläutert. Es wird eine zweikanalige Szene $\mathbf{S}_e = (s_e(x, y, n))$ angenommen, bei der $n = 0$ einen Rotauszug und $n = 1$ einen Blauauszug darstellt. In diesem Bild seien drei verschiedene Objekte abgebildet, die sich in den beiden Farbauszügen deutlich unterscheiden (etwa zwei verschiedenfarbige Oberflächen von Bauteilen vor einem dunklen Hintergrund). Es wird nun das zweidimensionale Histogramm (Abschnitt 3.10) dieses Bildes ermittelt und grafisch dargestellt. Auf den Koordinatenachsen werden dazu die Merkmale „Rotanteil“ (Kanal 0) und „Blauanteil“ (Kanal 1) aufgetragen. Gemäß der gewählten Definition sind die diskreten Werte für die Farbanteile aus der Grauwertmenge $G = \{0, 1, ..., 255\}$. Das bedeutet, dass im zweidimensionalen Histogramm alle Wertekombinationen $(i, k), i, k \in G$, mit bestimmten (relativen) Häufigkeiten auftreten können. Man sagt: Der *Merkmalsraum* ist zweidimensional. In der grafischen Darstellung des Histogramms von $\mathbf{S}_e$ werden sich zu den einzelnen Objekten im Merkmalsraum *Punktwolken* (*cluster*) ausbilden, da Bildpunkte von $\mathbf{S}_e$, die zum selben Objekt gehören, ähnliche Farbkombinationen aufweisen (Bild 20.1).

Aus diesem Beispiel ergeben sich drei wichtige Fragestellungen, die kennzeichnend für die numerische Klassifikation sind:

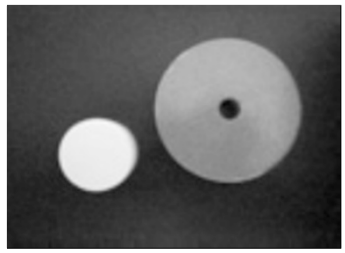
- Wie können die Punktwolken der abgebildeten Objekte ermittelt werden?
- Wie können die Punktwolken mit mathematischen Modellen beschrieben werden?
- Wie kann ein Bildpunkt, dessen Zugehörigkeit zu einem der abgebildeten Objekte nicht bekannt ist, mit einem Klassifizierungsverfahren einer der Klassen zugeordnet werden?

In den folgenden Abschnitten werden diese Fragestellungen näher untersucht. Zunächst aber einige Begriffsklärungen.

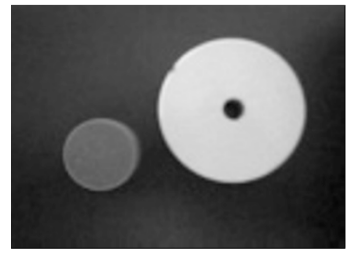
A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_20



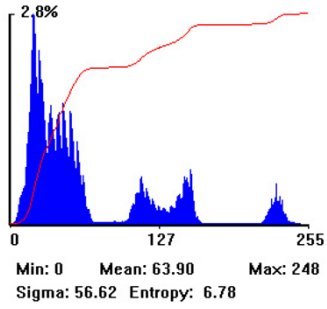
(a)



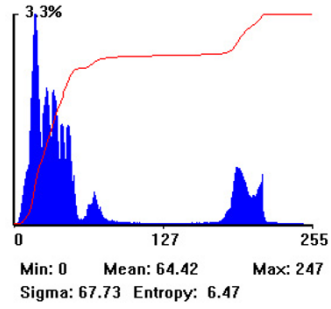
(b)



(c)

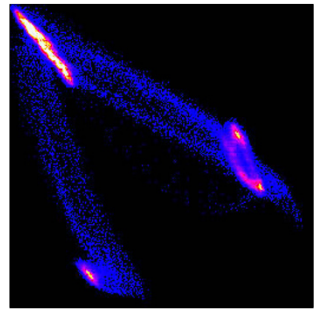


(d)



(e)

Objekt 3:
Bildhintergrund.



→
Kanal 1: Blauanteil.

Objekt 2:
Blaue Scheibe.
Deutlich ist das Zerfallen in zwei Teilcluster zu sehen. Dies drückt sich auch in den Farbkanälen und den beiden eindimensionalen Histogrammen aus.

Objekt 1:
Rote Scheibe.

(f) ↓
Kanal 0: Rotanteil.

Bild 20.1: Beispiel eines zweidimensionalen Merkmalsraums. (a) Farbbild mit drei Objekten: Rote Scheibe, blaue Scheibe und Hintergrund. (b) Rotanteil (Kanal 0). (c) Blauanteil (Kanal 1). (d) Histogramm des Rotanteils. (e) Histogramm des Blauanteils. (f) Zweidimensionales Histogramm mit drei Clustern.

Da ein Klassifizierungs- oder Mustererkennungssystem in der Regel für einen ganz bestimmten Anwendungsbereich oder *Problemkreis* geschaffen wird, ist es auch nur mit bestimmten Objekten konfrontiert. Manchmal lassen sich die möglichen Objekte aufzählen, z.B. alle Zeichen einer genormten Schrift. In anderen Fällen hat man nur eine intuitive Vorstellung: Wiesen, Wälder, Gewässer, Siedlungen usw. sind z.B. Objekte, die in einer Luftaufnahme abgebildet sein können. Man spricht dann auch von *Objektklassen* O. So ist z.B. ein Bauteil, das in einer Aufnahme abgebildet ist, die von einem sichtgesteuerten Roboter gemacht wird, ein Objekt der Objektklasse „Bauteil vom Typ XYZ". Durch die Verwendung der spezifischen Sensoren des Problemkreises werden bestimmte *Merkmale* der Objekte oder der Objektklassen erfasst. Von optischen Sensoren, die bildliche Informationen liefern, werden von den Objekten Merkmale erfasst, wie der Grauton, die Farbe, die multispektrale Signatur, die Oberflächenstruktur (Textur), die Form oder das zeitliche Verhalten. Der Duft eines Objektes wird von einem optischen Sensor z.B. nicht erfasst.

Als *Aufzeichnung* wird die Abbildung des Beobachtungsgebietes in die digitalisierte Form der Szene $\mathbf{S}_e = (s_e(x, y, n, t))$ bezeichnet. Bei einem dynamischen Vorgang wird die resultierende Szene eine Bildfolge sein. Im Fall einer statischen Graubildaufnahme genügt das Modell $\mathbf{S}_e = (s_e(x, y))$. Nach der Aufzeichnung wird eine Objektklasse $O_i, i = 0, 1, ..., t-1$ durch eine *Musterklasse* (ein *Muster*) $K_i, i = 0, 1, ..., t-1$ repräsentiert. Das Muster kann man sich aus allen möglichen Merkmalen zusammengesetzt vorstellen, die man aus den aufgezeichneten Originaldaten ableiten kann. Bei praktischen Anwendungen ist es allerdings nicht möglich, alle nur denkbaren Merkmale zu berechnen.

Zur Segmentierung der abgebildeten Objekte werden nun aus den aufgezeichneten Originaldaten weitere *beschreibende Merkmale* extrahiert. Besteht zu zwei Objekten nicht die Möglichkeit, aus den aufgezeichneten Daten Merkmale zu berechnen, anhand derer die Segmentierung durchgeführt werden kann, so ist die Trennung dieser Objekte nicht möglich. In diesem Fall muss die Aufzeichnung des Beobachtungsgebietes mit anderen Sensoren (z.B. mit anderen Spektralausschnitten) durchgeführt werden.

Die Merkmale, anhand derer ein Muster beschrieben wird, können recht unterschiedlich sein. Angenommen, die beiden Bauteile des einführenden Beispiels von Bild 20.1 seien durch folgende Merkmale zu unterscheiden: unterschiedliche Farbanteile im Rot- und im Blauauszug, verschiedene Oberflächenstruktur (z.B. glatt und mattiert) und verschiedene Form (z.B. kreisförmig und rechteckig). Die Farbunterschiede, die sich bereits in den aufgezeichneten Originaldaten ausdrücken, sind rein *bildpunktbezogen*. Sie würden hier unter Umständen bereits zur Segmentierung ausreichen.

Man könnte aber auch noch die unterschiedliche Oberflächenstruktur als beschreibendes Merkmal verwenden. Dazu muss zu einer bestimmten Umgebung jedes Bildpunktes eine Analyse der Oberflächenstruktur durchgeführt werden, die dann für diesen Bildpunkt und seine Umgebung eine oder mehrere Maßzahlen zur Oberflächenbeschreibung liefert. Beschreibende Merkmale dieser Art sind also *umgebungsabhängig*.

Auch die *Form* kann als beschreibendes Merkmal verwendet werden. Hier wird dann die flächige oder räumliche Anordnung der Bildpunkte eines abgebildeten Objektes verwendet. Allerdings werden kompliziertere Merkmale dieser Art erst in einer weiteren Klassifizierungs- oder Interpretationsstufe eingesetzt werden können, da ja zunächst Bildpunkte gefunden

werden müssen, die zu den Objekten gehören, und erst dann ihre Anordunung untersucht werden kann. Für die Darstellung der numerischen Klassifikation sei zunächst auf Merkmale dieser Art verzichtet, obwohl zu bemerken ist, dass die Segmentierung auch hier letztlich wieder auf ein Klassifizierungsproblem hinausläuft.

In der weiteren Betrachtung wird eine N-kanalige Szene angenommen, deren Kanäle die verschiedenen beschreibenden Merkmale (originale oder auch abgeleitete) repräsentieren. Die N Kanäle spannen einen *N-dimensionalen Merkmalsraum* auf.

Ein Bildpunkt wird daher im Weiteren auch als ein *N-dimensionaler Merkmalsvektor* $\mathbf{s}(x, y)$ bezeichnet. Wenn keine bildliche Darstellung der einzelnen Merkmalskanäle gewünscht ist, wird man hier die Forderung nach diskreten Grauwerten fallen lassen. Die Maßzahlen für die einzelnen beschreibenden Merkmale sind dann rational, reell oder komplex.

Da bei den folgenden Betrachtungen häufig nicht die Lage der Bildpunkte $\mathbf{s}(x, y)$ im Ortsbereich der (x, y)-Koordinaten, sondern nur die Lage der Bildpunkte im Merkmalsraum von Bedeutung ist, wird eine Szene $\mathbf{S}$ oft auch als eine Menge S von *Merkmalsvektoren*

$$S = \{\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{M-1}\}, \tag{20.1}$$

geschrieben, wobei M die Anzahl der Merkmalsvektoren ist. Treten in S identische Merkmalsvektoren auf, so sollen sie alle dasselbe Element der Menge bedeuten. Hier kann eventuell aber auch die Häufigkeit identischer Elemente mit berücksichtigt werden.

Zusammenfassend kann man also sagen: Eine Objektklasse wird vermöge der Aufzeichnung durch eine bestimmte Musterklasse repräsentiert, die anhand der gewählten beschreibenden Merkmale charakterisiert wird. Die Merkmale erzeugen im Merkmalsraum eine für die jeweilige Musterklasse *charakteristische Population*, die sich, bei geschickter Wahl der Merkmale, als zusammenhängende *Punktwolke* (*cluster*) ausprägt. Welche Punktwolke einem bestimmten Objekt zugeordnet ist und welche Bereiche des N-dimensionalen Merkmalsraumes das Muster einnimmt, ist nicht bekannt. Die wesentliche Aufgabe einer Segmentierung ist es somit, den N-dimensionalen Merkmalsraum durch geeignete Trennungsgrenzen in Bereiche mit ähnlichen Merkmalsvektoren aufzuteilen.

Um Aufschluß über die Population der Objekte im Merkmalsraum zu erhalten, sind unterschiedliche Vorgehensweisen möglich, die als

- *fest dimensionierte überwachte*,
- *fest dimensionierte unüberwachte*,
- *überwacht lernende* oder
- *unüberwacht lernende*

Klassifizierungsstrategien bezeichnet werden.

20.2 Fest dimensionierte überwachte Klassifizierungsstrategien

Bei vielen Problemstellungen besteht die Möglichkeit, die Population der Muster im Merkmalsraum (abkürzend dafür wird im Folgenden oft auch nur von „Muster“ gesprochen) durch eine Stichprobe mit bekannten Objekten zu ermitteln. Soll z.B. im Rahmen der Auswertung von Blutbildern die Zählung der unterschiedlichen Zelltypen automatisch durchgeführt werden, so besteht die Möglichkeit, die Merkmale anhand ausgesuchter Testbeispiele zu ermitteln. Werden Bilddaten aus der Fernerkundung verarbeitet, so werden die einzelnen Objektklassen anhand von Bildausschnitten (*Trainingsgebieten*) festgelegt.

Die Stichproben sollten dabei natürlich die Eigenschaften des zugehörigen Musters möglichst vollständig wiedergeben. Allerdings kann durch die Stichprobe die tatsächliche Lage der Muster im Merkmalsraum nur näherungsweise ermittelt werden. Die durch die Stichprobe erfassten Eigenschaften der Objekte werden im Merkmalsraum ebenfalls durch Punktwolken repräsentiert, die im Folgenden als *Realisationen* $k_i, i = 0, 1, 2...$ der Musterklassen bezeichnet werden.

Die Stichprobe ergibt zu jeder Klasse K_i eine *Realisation* k_i. Man kann sich das wie eine statistische Zufallsvariable X vorstellen, die bei der Durchführung eines Zufallsversuchs einen Wert x als Realisation annimmt. Wird eine andere Stichprobe verwendet, so ergibt sich auch eine andere Realisation k_i. Die k_i sollten die tatsächlichen K_i im Merkmalsraum so gut wie möglich annähern.

Es stellt sich die Frage, wie eine Realisation mathematisch beschrieben werden kann. Zunächst ist sie eine Menge von Merkmalsvektoren. Da zu jedem Merkmalsvektor noch wichtig ist, wie häufig er in der Realisation der Musterklasse auftritt, wird folgendes Modell gewählt:

$$k_i = \left\{ \left(\mathbf{g}_0^{(i)}, h_0^{(i)}\right), \left(\mathbf{g}_1^{(i)}, h_1^{(i)}\right), ..., \left(\mathbf{g}_{u_i-1}^{(i)}, h_{u_i-1}^{(i)}\right) \right\}, \tag{20.2}$$

wobei

k_i Realisation der Musterklasse,

$\mathbf{g}_j^{(i)}$ N-dimensionaler Merkmalsvektor,

$h_j^{(i)}$ Häufigkeit des Merkmalsvektors in der Realisation der Musterklasse,

u_i Anzahl der verschiedenen Merkmalsvektoren in der Realisation der Musterklasse.

Der Idealfall wäre nun, wenn die durch die Stichprobe gewonnenen Realisationen der Musterklassen mit der tatsächlichen Population deckungsgleich wären. Das wird in der Praxis aber nicht zu erreichen sein. Vielmehr werden sie nur einen gemeinsamen Durchschnitt haben. Bild 20.2 soll diesen Sachverhalt erläutern.

Werden durch die Stichprobe Eigenschaften erfasst, die die zugehörige Musterklasse nicht aufweist, so werden von den anschließenden Klassifizierungsverfahren Merkmalsvek-

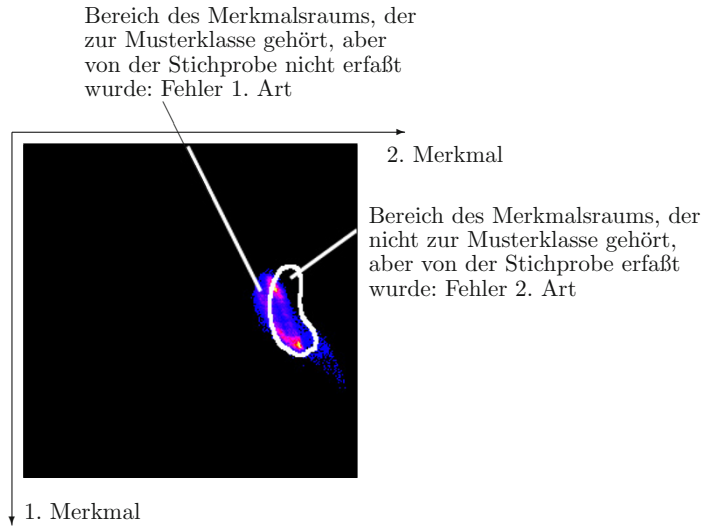


Bild 20.2: Fehler 1. und 2. Art bei der Approximation eines Musters mit Hilfe einer Stichprobe. In Farbe: Punktwolke der Musterklasse, deren Ausprägung in der Praxis aber in der Regel nicht bekannt ist. Der weiße Rand stellt eine Realisation einer Musterklasse nach Maßgabe eine Stichprobe dar.

toren, die eigentlich nicht zum Objekt gehören, fälschlich diesem zugeordnet (Fehler 2. Art). Der Fehler 1. Art tritt auf, wenn die Stichprobe nicht alle Eigenschaften der Klasse beinhaltet, so daß Bildpunkte, die eigentlich dazugehören würden, nicht oder falsch klassifiziert werden.

Die Realisationen der Musterklassen werden jetzt als Näherung der tatsächlichen Population der Musterklassen verwendet. Ein Merkmalsvektor, dessen Zugehörigkeit zu einem bestimmten Objekt nicht bekannt ist, wird demjenigen zugeordnet, zu dessen Realisation der Musterklasse er „am besten passt“. Was bei den einzelnen Klassifikatoren unter „am besten passt“ zu verstehen ist, wird bei der Darstellung der jeweiligen Verfahren in den folgenden Abschnitten erläutert. Diese Verfahren lassen sich, grob gesprochen, in zwei Bereiche aufteilen.

Bei der *statistischen Vorgehensweise* werden die Musterklassen im Merkmalsraum mit Hilfe von *Verteilungsfunktionen* oder *Verteilungsdichten* erfasst. Die Zuweisung eines unbekannten Bildpunktes erfolgt hier nach Gesichtspunkten der maximalen Wahrscheinlichkeit.

Bei der *geometrischen Vorgehensweise* werden *Trennungsfunktionen* zwischen den ein-

zelnen Musterklassen berechnet. Die Zuweisung der Bildpunkte erfolgt hier nach Maßgabe dieser Trennungsfunktionen.

Ein Klassifizierungssystem, das auf der Basis von bekannten Stichproben für die Realisationen der Musterklassen ermittelt, heißt ein fest *dimensioniertes überwachtes Klassifizierungssystem*. Zusammenstellend ist dabei Folgendes gegeben:

- $\mathbf{S} = (s(x, y, n))$ eine N-kanalige Szene, die einen N-dimensionalen Merkmalsraum aufspannt.
- t Objektklassen, denen bestimmte Musterklassen zugeordnet sind.
- t Realisationen $k_0, k_1, ..., k_{t-1}$, die das Ergebnis der Auswertung der Stichprobe sind.
- Zielsetzung ist die Zuweisung eines Merkmalsvektors $\mathbf{g} = \mathbf{s}(x, y)$, über dessen Zugehörigkeit zu einer der Objektklassen nichts bekannt ist, zu einer der Musterklassen. Algorithmen, die diese Zuordnung durchführen, heißen *Klassifikatoren*.

20.3 Fest dimensionierte unüberwachte Klassifizierungsstrategien

Bei einer fest dimensionierten überwachten Vorgehensweise werden die Realisationen der Musterklassen anhand von Stichproben ermittelt. Das heißt aber, dass damit die Anzahl der Klassen vorweg bekannt sein muss. Manchmal weiß man nicht, wieviele verschiedene Objekte abgebildet sind und wieviele Cluster sich demnach im Merkmalsraum ausprägen. Mit einer *unüberwachten Klassifizierungsstrategie* wird dann versucht, die Gruppierungstendenzen der Objekte im Merkmalsraum, also die Realisationen der Musterklassen, ohne bekannte Stichprobe zu ermitteln.

Hierzu werden Verfahren angewendet, die, ausgehend von einem beliebigen Merkmalsvektor als Zentrum einer ersten Punktwolke, versuchen, die Merkmalsvektoren dieser Musterklasse mit Hilfe geeigneter Kriterien zuzuordnen. Ist für einen Merkmalsvektor das Zuordnungskriterium nicht erfüllt, so wird eine neue Punktwolke mit einem neuen Zentrum eröffnet. Die Verfahren laufen (oft iterativ) so lange, bis alle Merkmalsvektoren einer Musterklasse zugeordnet sind. Abschließend werden Cluster, die auf Grund geeigneter Bewertungskriterien als „zu groß“ oder „zu klein“ erkannt wurden, aufgeteilt oder zusammengelegt.

Eine derartige *Clusteranalyse* liefert schließlich das Ergebnis, dass die Merkmalsvektoren dazu tendieren, t Realisationen $k_0, k_1, ..., k_{t-1}$ von Musterklassen mit den Zentren $\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{t-1}$ zu bilden. Da alle Bildpunkte einer der Musterklassen zugewiesen werden, kann das Eingabebild $\mathbf{S}_e$ als klassifiziert angesehen werden. Es besteht aber auch die Möglichkeit, ein derartiges Clusterverfahren einzusetzen, um sich einen ersten Überblick über die Aufteilung des Merkmalsraumes zu verschaffen und mit diesen Vorkenntnissen z.B. einen fest dimensionierten Klassifikator einzusetzen.

Diese Vorgehensweise wird am Beispiel des *unüberwachten Minimum-Distance-Cluster-Algorithmus* erläutert. Es sei

$$S_e = \left\{\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{M-1}\right\} \tag{20.3}$$

die Menge der Bildpunkte (Merkmalsvektoren) einer N-kanaligen Szene $\mathbf{S}_e$. Da die Lage der Bildpunkte im Ortsbereich hier nicht von Interesse ist, kann, wie schon in Abschnitt 20.1 erläutert, auf die Ortskoordinaten (x, y) bei der Darstellung des Verfahrens verzichtet werden. Der Wert $c > 0$ sei ein vorgegebener Schwellwert für die Distanz zweier Merkmalsvektoren im Merkmalsraum. Zu Beginn wird ein beliebiger Merkmalsvektor als Zentrum eines ersten Clusters (der ersten Musterklasse) definiert. Ohne Beschränkung der Allgemeinheit wird

$$\mathbf{z}_0 = \mathbf{g}_0 \tag{20.4}$$

angenommen. Für die folgenden Bildpunkte $\mathbf{g}_i, i = 1, 2, ...$ wird jetzt die Distanz

$$d_i^{(0)} = d(\mathbf{z}_0, \mathbf{g}_i) \tag{20.5}$$

berechnet. Als Distanzfunktion kann etwa die Euklidische Distanz im Merkmalsraum verwendet werden. Falls nun bis zu einem bestimmten Wert i gilt:

$$d_i^{(0)} \leq c, \quad i = 1, 2, ..., i_1 - 1, \tag{20.6}$$

so werden die dazugehörigen Bildpunkte der Musterklasse K_0 zugeordnet. Für den Bildpunkt $\mathbf{g}_{i_1}$ sei die Bedingung (20.6) nicht mehr erfüllt. Es wird dann

$$\mathbf{z}_1 = \mathbf{g}_{i_1} \tag{20.7}$$

als Zentrum eines zweiten Clusters festgelegt. Für die folgenden Bildpunkte

$$\mathbf{g}_{i_1}, \mathbf{g}_{i_1+1}, \cdots$$

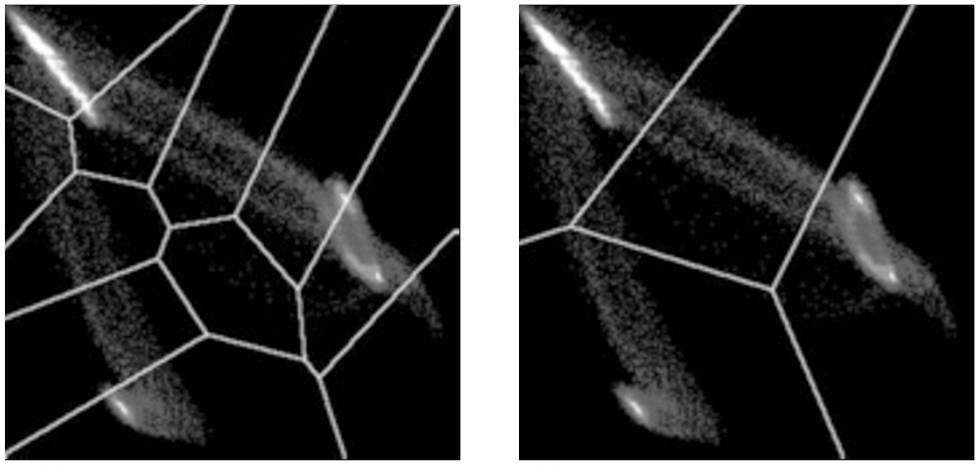
werden jetzt die Distanzen

$$d_k^{(0)} \text{ und } d_k^{(1)}, \quad k = i_1, i_1 + 1, ... \tag{20.8}$$

berechnet. Die Bildpunkte werden den entsprechenden Musterklassen zugeordnet, falls die Schwellwertüberprüfung dies zulässt. Gilt für einen Bildpunkt mit dem Index j

$$d_j^{(0)} \leq c \text{ und } d_j^{(1)} \leq c, \tag{20.9}$$

so wird er demjenigen Cluster zugewiesen, zu dem er die kleinere Distanz hat. Für einen Index i wird der Fall eintreten, dass keine der beiden Distanzen kleiner oder gleich c ist. In diesem Fall wird ein weiteres Clusterzentrum festgelegt. Aus dieser Darstellung ist leicht



(a) (b)

Bild 20.3: Die Wahl des Schwellwertes c beeinflusst das Ergebnis des Minimum-Distance-Clustering-Algorithmus wesentlich. (a) Bei kleinerer Wahl von c (hier: c=50) wird der Merkmalsraum feiner aufgeteilt als bei (b) größerer Wahl von c (hier: c=100).

zu ersehen, wie der Minimum-Distance-Cluster-Algorithmus abläuft, bis alle Merkmalsvektoren verarbeitet sind.

Die praktische Verwendung dieses Verfahrens ist allerdings nicht unproblematisch. Der verwendete Schwellwert c beeinflusst die Güte des Ergebnisses wesentlich, wie Bild 20.3 verdeutlicht.

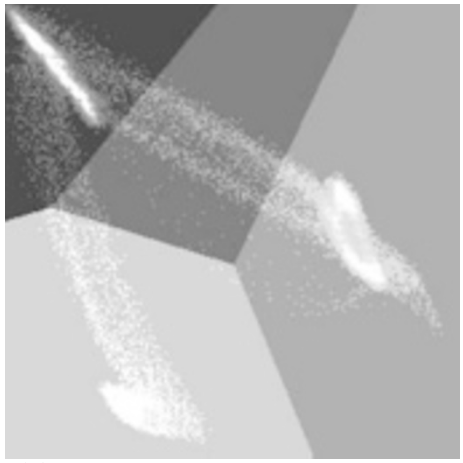
Zum anderen hängt das Ergebnis von der Verarbeitungsreihenfolge der Bildpunkte ab, da z.B. die Möglichkeit besteht, dass Bildpunkte, die der ersten Musterklasse zugewiesen werden, nach Kreation des zweiten Clusters diesem zugeordnet würden.

Als Vorteile sind zu nennen, dass das Verfahren einfach zu implementieren ist und meistens wenig Rechenzeit benötigt, da es nicht iterativ arbeitet.

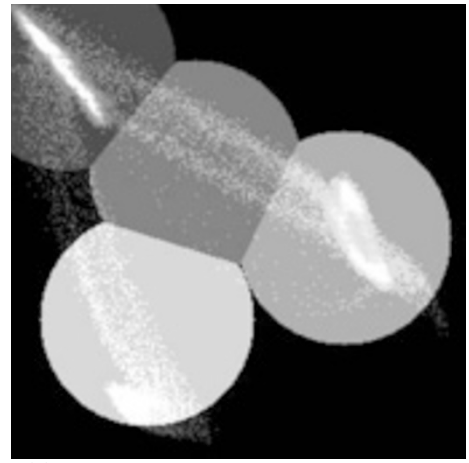
Bild 20.4-a zeigt ein Klassifizierungsergebnis zu diesem Verfahren. Bild 20.4-b stellt die Aufteilung des zweidimensionalen Merkmalsraums dar. Die Lage der zu apprximierenden Cluster wurde eingeblendet. Dieses Verfahren kann auch noch erweitert werden, wenn eine *Zurückweisungsklasse* eingeführt wird. Dieser Klasse werden alle Merkmalsvektoren zugewiesen, die weiter als ein vorzugebender Schwellwert von den generierten Clusterzentren entfernt liegen (Bild 20.4-c).

Es gibt eine Reihe von Clusterverfahren, die gegenüber dem dargestellten wesentliche Vorteile haben. Ein Beispiel eines Clusterverfahrens auf der Basis der *fuzzy logic* wird in Kapitel 22 gegeben.

(a)



(b)



(c)

Bild 20.4: Ergebnis des Clusterverfahrens angewendet auf den Rot- und den Blaukanal des Testbildes 20.1-a. (a) Ergebnis der Klassifizierung. Die drei Klassen „rote Scheibe“, „blaue Schweibe“ und „dunkler Hintergrund“ konnten gut klassifiziert werden. Der Merkmalsraum wurde jedoch in vier Bereiche aufgeteilt. In den vierten Bereich wurden z.B. die Ränder um die beiden Ringe klassifiziert. (b) Aufteilung des Merkmalsraums durch das Clusterverfahren bei einem Schwellwert $c = 90$. Die zu approximierenden Cluster wurden eingeblendet. (c) Hier wurde zusätzlich eine Zurückweisungsklasse eingefügt, der alle Merkmalsvektoren zugewiesen werden, die zu weit von den generierten Clusterzentren entfernt liegen (Schwellwert 60).

20.4 Überwachtes und unüberwachtes Lernen

Bei vielen Problemstellungen sind die Eigenschaften der Objekte, die mit Hilfe der numerischen Klassifikation segmentiert werden sollen, gewissen Änderungen unterworfen.

Ein Beispiel: Im Rahmen einer automatischen Qualitätskontrolle muss geprüft werden, ob verschiedenfarbige Flachbandkabel in einer bestimmten Reihenfolge an einen Bauteil angeschlossen sind. Dazu werden die Kabel mit einer Videokamera oder mit einem CCD-Sensor aufgezeichnet und anhand ihrer Farbgebung segmentiert. Nach der Segmentierung kann geprüft werden, ob die richtige Reihenfolge vorliegt. Die Merkmale zur Segmentierung sind hier Farbkombinationen der einzelnen Flachbandkabel, die mit Hilfe einer anfänglichen Stichprobe ermittelt werden können. Es ist nun durchaus denkbar, dass die Farbtöne geringfügigen Nuanzierungen, bedingt durch den Herstellungsprozess, unterworfen sind. Bei einem festdimensionierten Klassifizierungssystem würden diese Trends nicht mit berücksichtigt.

Ein *überwacht lernendes Klassifizierungssystem* versucht diesem Sachverhalt wie folgt gerecht zu werden (Bild 20.5): Anhand einer anfänglichen Stichprobe werden die Merkmale der einzelnen Klassen „gelernt“. Das Ergebnis besteht wie oben aus den t Realisationen der Musterklassen. Man bezeichnet diese Phase auch als die *Dimensionierung des Klassifizierungssystems*. Nach der Klassifizierung von Merkmalsvektoren werden jetzt aber ihre Merkmale rückgekoppelt und so die zugehörigen Realisationen der Musterklassen dem Trend angepasst.

Unüberwachtes Lernen liegt vor, wenn die Dimensionierung nicht anhand einer bekannten Stichprobe, sondern durch einen unüberwachten Clusteralgorithmus durchgeführt wird.

Damit soll die Untersuchung der ersten Fragestellung, wie die Populationen der Musterklassen werden können, beendet sein. In den folgenden Abschnitten werden nun einige Klassifikatoren erläutert und dabei die zweite und die dritte Fragestellung behandelt.

20.5 Der Minimum-Distance-Klassifikator

Die Voraussetzungen für die numerische Klassifikation werden einführend zu diesem Klassifikator nochmals zusammengestellt:

- $\mathbf{S}_e = (s_e(x, y, n))$ N-kanalige Szene, die einen N-dimensionalen Merkmalsraum aufspannt.
- t Objekte (Objektklassen), die in $\mathbf{S}_e$ abgebildet sind und denen vermöge der gewählten Aufzeichnung die Muster $K_i, i = 0, 1, ..., t-1$ zugeordnet sind.
- t Realisationen k_i der Musterklassen K_i. Sie sind das Ergebnis einer festdimensionierten Vorgehensweise (Stichproben) oder eines unüberwachten Cluster-Verfahrens.

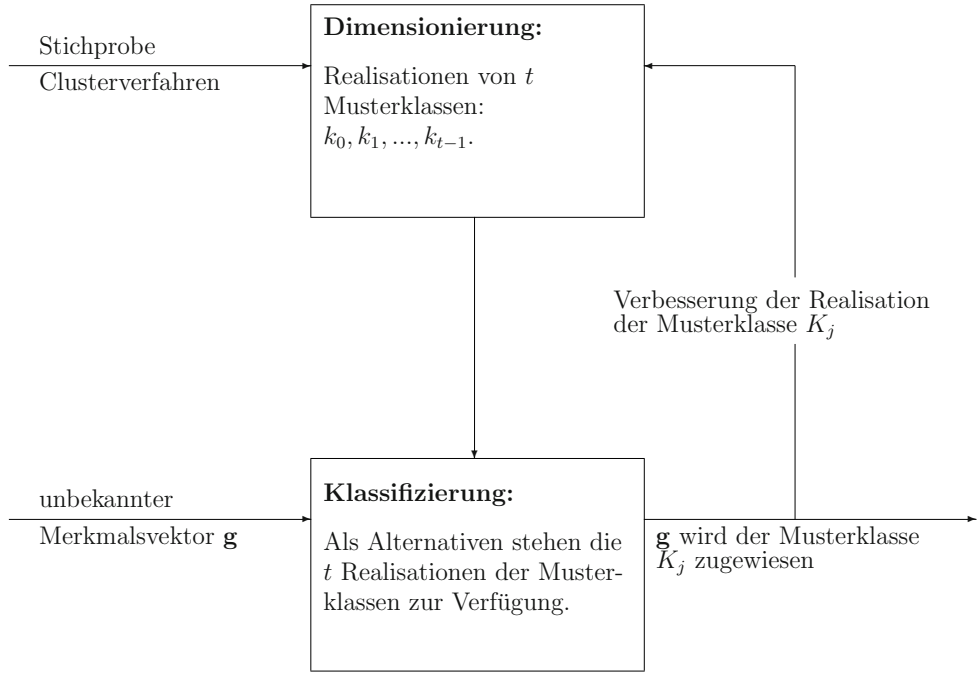


Bild 20.5: Überwacht oder unüberwacht lernendes Klassifizierungssystem. Die Musterklassen werden durch eine Rückkopplung der Merkmale von schon klassifizierten Bildpunkten dem Trend angepasst.

Die Aufgabe besteht nun darin, einen Merkmalsvektor (Bildpunkt) $\mathbf{g} = \mathbf{s}(x, y)$, über dessen Zugehörigkeit zu einer der Musterklassen nichts bekannt ist, mit Hilfe der Realisationen einer Klasse zuzuordnen.

Es sind somit noch die Fragen zu untersuchen, wie die Realisationen der Musterklassen im N-dimensionalen Merkmalsraum mathematisch beschrieben werden können und mit welchen Entscheidungskriterien ein unbekannter Merkmalsvektor klassifiziert werden kann.

Bei einem Klassifikator, der nach dem *Minimum-Distance-Verfahren* arbeitet, ist der Grundgedanke der Klassifizierung, dass Merkmalsvektoren, die im Merkmalsraum nahe beisammen liegen, wahrscheinlich zur selben Musterklasse gehören.

Als Abstandsmaß zweier Punkte $\mathbf{g}_1$ und $\mathbf{g}_2$ im Merkmalsraum wird z.B. die Euklidische Distanz verwendet:

$$d(\mathbf{g}_1, \mathbf{g}_2) = \sqrt{(\mathbf{g}_1 - \mathbf{g}_2)^T(\mathbf{g}_1 - \mathbf{g}_2)} = \sqrt{(g_{1,0} - g_{2,0})^2 + ... + (g_{1,N-1} - g_{2,N-1})^2} \quad (20.10)$$

Aufbauend auf diesem Distanzmaß wird ein Merkmalsvektor $\mathbf{g}$ derjenigen Klasse zugeordnet, zu deren Realisation er den geringsten Abstand hat.

Um den Abstand eines Merkmalsvektors zu einer Realisation berechnen zu können, wird zu jeder Musterklasse K_i ein Zentrum $\mathbf{z}_i, i = 0, 1, ..., t-1$ festgelegt. Als Zentrum können z.B. verwendet werden:

- derjenige Bildpunkt, der in der Realisation der Musterklasse am häufigsten auftritt,
- der Mittelwertvektor der Realisation der Musterklasse oder
- ein Zentrum, das durch einen unüberwachten Clusteralgorithmus ermittelt wurde.

Damit ergibt sich für den Minumum-Distance-Klassifikator folgender Algorithmus:

- Berechnung der t Zentren $\mathbf{z}_i$ zu den Realisationen k_i der Musterklassen.
- Berechnung der Abstände eines zu klassifizierenden Bildpunktes $\mathbf{g}$ zu den Zentren $\mathbf{z}_i$:

$$d_i = d(\mathbf{z}_i, \mathbf{g}), \quad i = 0, 1, ..., t-1. \quad (20.11)$$

- Zuweisung von $\mathbf{g}$ zur Musterklasse K_j, falls gilt: $d_j < d_i$ für alle $i \neq j$.

Im Falle eines zweidimensionalen Merkmalsraumes sind die Trennungsfunktion zweier Musterklassen bei diesem Klassifikator die Mittelsenkrechte zwischen den Zentrumsvektoren. Bei einem mehrdimensionalen Merkmalsraum sind die Trennungsfunktionen sinngemäß (Hyper-)Ebenen.

Die Berechnung des Abstandes eines Bildpunktes von einer Musterklasse nach (20.11) kann noch vereinfacht werden. Statt d_i wird d_i^2 verwendet und wie folgt umgeformt:

$$d_i^2 = (\mathbf{z}_i - \mathbf{g})^T(\mathbf{z}_i - \mathbf{g}) = \mathbf{z}_i^T\mathbf{z}_i - 2\mathbf{z}_i^T\mathbf{g}^T + \mathbf{g}\mathbf{g} = \mathbf{g}^T\mathbf{g} - 2\Big(\mathbf{g}^T\mathbf{z}_i - \frac{1}{2}\mathbf{z}_i^T\mathbf{z}_i\Big). \quad (20.12)$$

Der Term $\mathbf{g}^T\mathbf{g}$ ist für jeden Bildpunkt konstant und muss deshalb für die Berechnung der Entscheidung, welcher Musterklasse $\mathbf{g}$ zugewiesen wird, nicht berechnet werden. Der Term $1/2\mathbf{z}_i^T\mathbf{z}_i$ ist für jede Realisation k_i konstant und kann daher vorab berechnet werden. Der vereinfachte Algorithmus ist damit:

- Bestimmung der Zentren $\mathbf{z}_i$ der Realisationen k_i der Musterklassen K_i.
- Berechnung der *Klassenkonstanten* $c_i = -1/2\mathbf{z}_i^T\mathbf{z}_i$.
- Berechnung der Größen $d_i^{'} = \mathbf{g}^T\mathbf{z}_i + c_i$ für alle Realisationen k_i und für den zu klassifizierenden Merkmalsvektor $\mathbf{g}$.
- Klassifizierung von $\mathbf{g}$ zur Klasse K_j, falls gilt: $d_j^{'} > d_i^{'}$, für alle $i \neq j$.

Nun einige Beispiele. Die Bildfolgen 20.6-a bis 20.6-d und 20.7-a bis 20.7-d zeigen Klassifizierungsergebnisse und die durch die Trainingsgebiete und den gewählten Minimum-Distance-Klassifikator erzeugte Aufteilung des Merkmalsraums, der hier zweidimensional gewählt wurde.

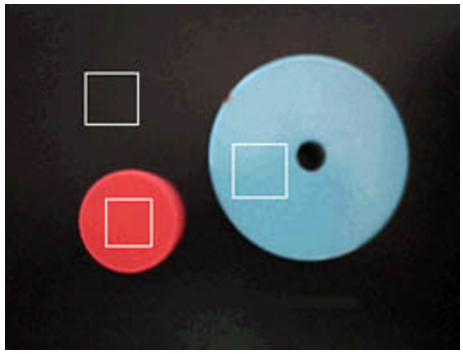
Dieses grundlegende Verfahren lässt eine Reihe von Modifikationen zu. Wie bereits oben erwähnt, können statt der Euklidischen Distanz auch andere Distanzmaße verwendet werden.

Zum anderen treten manchmal Problemstellungen auf, bei denen die Muster der einzelnen Objekte in mehrere Cluster zerfallen. In diesem Fall müssen zu den Musterklassen dementsprechend auch mehrere Zentrumsvektoren berechnet werden. Dieser Sachverhalt kann auch von einem Minimum-Distance-Klassifikator berücksichtigt werden. Man bezeichnet ihn dann als *Nächster-Nachbar-Klassifikator*.

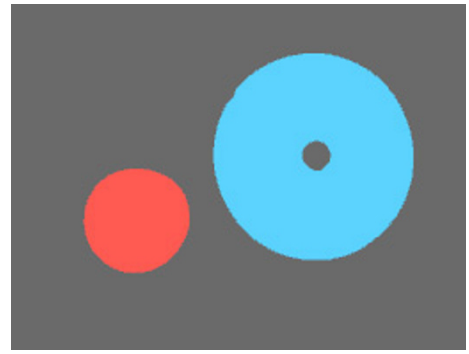
Schließlich ist es in den meisten Anwendungsfällen sinnvoll, eine *Zurückweisungsklasse* einzuführen, in die alle Bildpunkte klassifiziert werden, deren Abstand von der Musterklasse, an der sie am nächsten liegen, größer als ein Schwellwert (*Zurückweisungsradius*) r ist. Unter Verwendung des originalen Abstandes von (20.11) lautet dann die Zuordnungsvorschrift:

- Der Merkmalsvektor $\mathbf{g}$ wird der Musterklasse K_j zugewiesen, falls $d_j < d_i$ für alle $i \neq j$ und falls $d_j < r$ ist. Andernfalls wird er einer Zurückweisungsklasse zugewiesen.

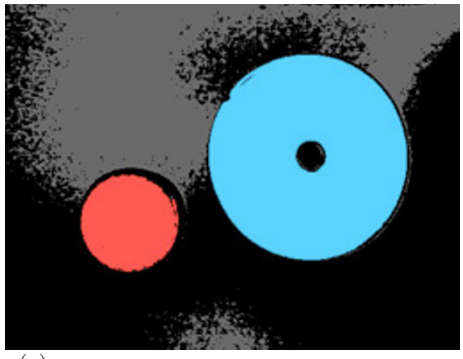
Durch diesen Minimum-Distance-Klassifikator mit *festem Zurückweisungsradius* werden die Musterklassen im N-dimensionalen Merkmalsraum durch N-dimensionale Kugeln angenähert, im Zweidimensionalen also durch Kreise.



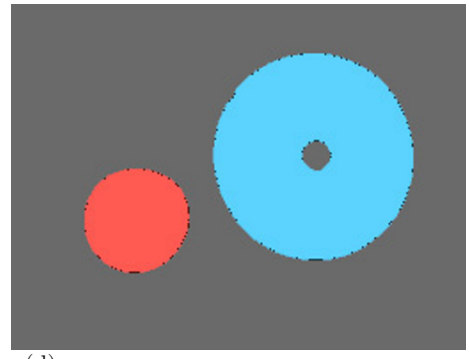
(a)



(b)



(c)



(d)

Bild 20.6: (a) Original mit drei Trainingsgebieten zu den drei Klassen „Hintergrund“, „rote Bildpunkte“ und „blaue Bildpunkte“. Um den Minimum-Distance-Klassifikator zweidimensional darstellen zu können, wurden zur Klassifizierung nur der rote und der blaue Kanal verwendet. (b) Ergebnis der Klassifizierung. Die Bildpunkte des klassifizierten Bildes (logisches Bild: die Grauwerte sind Codes für die Zugehörigkeit zu einer Klasse) wurden in der jeweiligen Farbe ihrer Klasse dargestellt. Man sieht, dass die Bildpunkte (Merkmalsvektoren) richtig klassifiziert wurden. Korrespondierend zu diesem Ergebnis ist die Aufteilung des Merkmalsraums gemäß Bild 20.7-b. (c) Hier wurde ein an die Clustergröße angepasster Zurückweisungsradius verwendet. Da das gewählte Trainingsgebiet die Klasse „Hintergrund“ nicht ausreichend erfasst (Vergleich: Bild 20.7-a und Bild 20.7-c) werden die Hintergrundbildpunkte zu einem großen Teil als „nicht klassifizierbar“ der Zurückweisungsklasse zugewiesen. Die Aufteilung des Merkmalsraums wird hier durch Bild 20.7-c dargestellt. (d) Hier wurde der Zurückweisungsradius vergrößert. Nur mehr in den Übergangsbereichen sind einige Bildpunkte nicht klassifizierbar. Die Aufteilung des Merkmalsraums entspricht Bild 20.7-d.

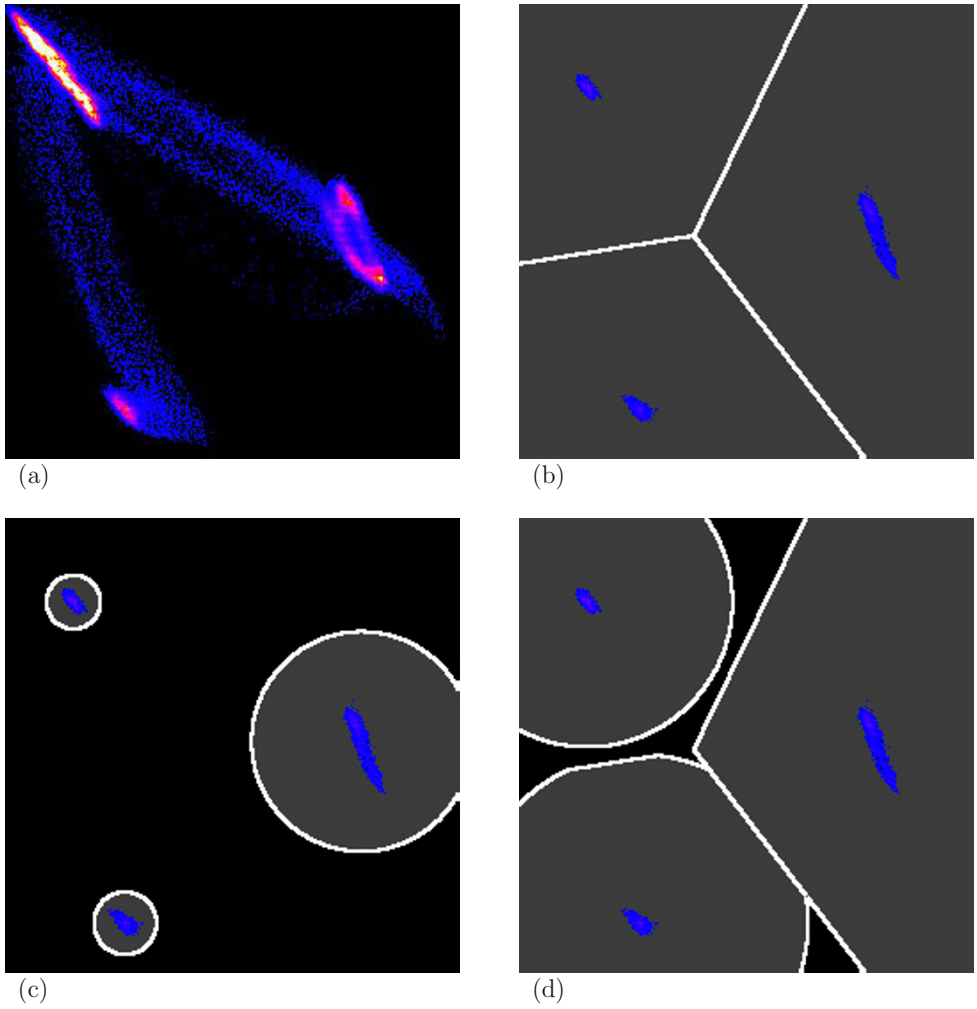


Bild 20.7: (a) Zweidimensionales Histogramm (zweidimensionaler Merkmalsraum) des Bildes 20.6-a unter Verwendung des roten und des blauen Kanals. (b) Durch die in Bild 20.6-a eingezeichneten Trainingsgebiete (Stichproben) werden im Merkmalsraum die dargestellten Realisationen der Klassen erzeugt. Man sieht durch einen Vergleich mit Bild a, dass die Klasse „Hintergrund“ nicht ausreichend erfasst wurde. Die weißen Linien zeigen die Aufteilung des Merkmalsraums durch den Minimum-Distance-Klassifikator ohne Zurückweisung. Die Trennungsgrenzen sind hier die Mittelsenkrechten zwischen den Clusterzentren. (c) Verwendung eines an die Clusterzentren angepassten Zurückweisungsradius. Alle Merkmalsvektoren, die außerhalb der Zurückweisungsbereiche liegen, werden als „nicht klassifizierbar“ ausgewiesen. (d) Vergrößerung des Zurückweisungsradius.

Eine weitere Möglichkeit besteht darin, für jede Musterklasse einen eigenen, an die Ausdehnung von k_i angepassten Zurückweisungsradius r_i zu verwenden, der z.B. aus der Streuung der Realisation der Musterklasse in den einzelnen Kanälen abgeleitet werden kann. Die Klassifizierungsvorschrift lautet dann:

- Der Merkmalsvektor $\mathbf{g}$ wird K_j zugewiesen, falls $d_j < d_i$ für alle $i \neq j$ und falls $d_j < r_j$ ist. Andernfalls wird er einer Zurückweisungsklasse zugewiesen.

Der Minimum-Distance-Klassifikator wurde bis jetzt als fest dimensionierter Klassifikator beschrieben, da nach der anfänglichen Stichprobe, aus der die Realisation der Musterklassen abgeleitet wurden, diese nicht durch eine Rückkopplung von bereits klassifizierten Bildpunkten verändert wurden. Um einen überwacht lernenden Klassifikator zu erhalten, ist bei der Zuweisung eines neuen Bildpunktes zu einer Musterklasse nur jeweils der Zentrumsvektor und evtl. der Zurückweisungsradius anzupassen. Wird als Zentrum der Mittelwertvektor verwendet, so wird er folgendermaßen modifiziert:

$k_j = \left\{\left(\mathbf{g}_0^{(j)}, h_0^{(j)}\right), ..., \left(\mathbf{g}_{u_j-1}^{(j)}, h_{u_j-1}^{(j)}\right)\right\}$
Realisation der Musterklasse vor der Zuweisung des Merkmalsvektors $\mathbf{g}$.

$$anz_j = \sum_{\mu=0}^{u_j-1} h_\mu^{(j)}$$
Anzahl der Bildpunkte, die k_j bis jetzt zugewiesen wurden.

$$\mathbf{z}_j = \frac{1}{anz_j} \sum_{\mu=0}^{u_j-1} h_\mu^{(j)} \mathbf{g}_\mu^{(j)} \text{ Mittelwertvektor von } k_j. \tag{20.13}$$

Anstatt $\mathbf{z}_j$ wird nach der Zuweisung von $\mathbf{g}$ zu K_j

$$\frac{1}{anz_j - 1}\left(anz_j \cdot \mathbf{z}_j + \mathbf{g}\right)$$

Auch der an die Musterklasse angepasste Zurückweisungsradius r_j kann bei Hinzunahme des Bildpunktes $\mathbf{g}$ zu K_j verbessert werden, wenn analog zu (20.13) die Streuung von k_j modifiziert wird.

Der Algorithmus lautet:

A20.1: Minimum-Distance-Klassifikator.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x,y,n))$ ein N-kanaliges Bild, dessen Kanäle einen N-dimensionalen Merkmalsraum aufspannen. Als Menge von Merkmalsvektoren geschrieben: $S = \{\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{M-1}\}$.

◇ $\mathbf{S}_a = (s_a(x,y))$ ein einkanaliges Grauwertbild mit der Grauwertmenge $G = \{0, 1, ..., 255\}$ (Ausgabebild). Die Grauwerte sind Codes für die Zugehörigkeit zu einer Klasse.

Algorithmus:

(a) Im Training werden aus den repräsentativen Stichproben, aus denen die Realisationen k_i der Klassen K_i abgeleitet wurden, die Zentrumsvektoren $\mathbf{z}_i$ und die Vektoren der Streuungen $\mathbf{q}_i$ berechnet.

(b) Für alle Klassen K_i werden berechnet:

(ba) Die Klassenkonstanten:
$c_i = -\frac{1}{2}\mathbf{z}_i^T\mathbf{z}_i$.

(bb) Die Zurückweisungsradien (hier angepasst an die Klasse):
$r_i = c \cdot \max_n \{\sqrt{q_{i,n}}\}, n = 0, 1, ..., N-1$. Falls die einzelnen Merkmalskanäle sehr unterschiedliche Streuungen besitzen, kann eine Normierung der Merkmale vor der Dimensionierung des Systems sinnvoll sein.

(c) Für alle Merkmalsvektoren $\mathbf{g}$ des Bildes:

(ca) Für alle Klassen $K_i, i = 0, 1, ..., t-1$:

Berechnung der Größen
$d_i' = \mathbf{g}^T\mathbf{z}_i + c_i$.

(cb) Zuweisung von $\mathbf{g}$ zur Klasse K_j, falls gilt:
$d_j' > d_i', \quad i = 0, 1, ..., t-1, \quad i \neq j$ und
$d_j' > \frac{1}{2}(\mathbf{g}^T\mathbf{g} - r_i^2)$.

Ende des Algorithmus

Nach dieser Beschreibung eines einfachen geometrischen Klassifikators mit seinen verschiedenen Varianten wird im folgenden Kapitel ein statistischer Klassifikator untersucht.

20.6 Maximum-Likelihood-Klassifikator

Dem Maximum-Likelihood-Klassifikator liegt ein statistischer Ansatz zugrunde. Die Musterklassen werden durch N-dimensionale Verteilungs- oder Dichtefunktionen beschrieben. Dazu müssen einige Wahrscheinlichkeiten definiert werden:

$p(K_i)$	apriori-Wahrscheinlichkeit der Musterklasse K_i.	(20.14)
$f(\mathbf{g})$	Verteilungsdichte der Merkmalsvektoren $\mathbf{g}$.	(20.15)
$f(\mathbf{g}\|K_i)$	Verteilungsdichte der Musterklasse K_i.	(20.16)
R_i	Bereich des Merkmalsraumes, in welchem die Merkmalsvektoren $\mathbf{g}$ zur Musterklasse K_i klassifiziert werden.	(20.17)
$\mathbf{L} = (l_{ij})$	Verlustmatrix: Es tritt der Verlust l_{ij} ein, wenn $\mathbf{g}$ aus K_i ist, jedoch zu K_j klassifiziert wird.	(20.18)

Damit berechnet sich die Wahrscheinlichkeit, dass $\mathbf{g}$ zu K_i klassifiziert wird, wenn $\mathbf{g}$ auch tatsächlich zu K_i gehört, wie folgt:

$$p(\mathbf{g} \text{ im Bereich } R_i|K_i) = \int_{R_i} f(\mathbf{g}|K_i)\mathrm{d}\mathbf{g}. \tag{20.19}$$

Die Wahrscheinlichkeit, dass $\mathbf{g}$ fälschlicherweise zu K_j klassifiziert wird, wenn es tatsächlich zu K_i gehört, ist:

$$p(\mathbf{g} \text{ im Bereich } R_j|K_i) = \int_{R_j} f(\mathbf{g}|K_i)\mathrm{d}\mathbf{g}. \tag{20.20}$$

Die Wahrscheinlichkeit, dass nun ein $\mathbf{g}$ aus K_i auftritt und zu K_i klassifiziert wird, ist dann

$$p(K_i) \cdot p(\mathbf{g} \text{ im Bereich } R_i|K_i) \tag{20.21}$$

und die Wahrscheinlichkeit, dass ein $\mathbf{g}$ aus K_i auftritt und falsch zu K_j klassifiziert wird ist

$$p(K_i) \cdot p(\mathbf{g} \text{ im Bereich } R_j|K_i). \tag{20.22}$$

Der zu erwartende Verlust bei der Klassifizierung von $\mathbf{g}$ zu einer der Klassen ist dann gegeben durch:

$$\sum_{i=0}^{t-1} p(K_i) \sum_{\mu=0;\mu\neq i}^{t-1} l_{i\mu} \cdot p(\mathbf{g} \text{ im Bereich } R_\mu|K_i). \tag{20.23}$$

Die Bereiche $R_i, \quad i = 0, 1, ..., t-1$, in die der Merkmalsraum aufgeteilt wird, sind so zu wählen, dass der zu erwartende Verlust minimiert wird. Dies ist der Fall, wenn $\mathbf{g}$ in R_j liegt, sofern gilt ([Ande69], [Niem74]):

$$\sum_{i=0;i\neq j}^{t-1} p(K_i) \cdot l_{ij} \cdot f(\mathbf{g}|K_i) < \sum_{i=0;i\neq \mu}^{t-1} p(K_i) \cdot l_{i\mu} \cdot f(\mathbf{g}|K_i) \qquad \mu = 0, ..., t-1; \quad \mu \neq j. \tag{20.24}$$

Ein Klassifikator, der nach dieser Regel entscheidet, heißt *Bayes'scher Klassifikator*. (20.24) ist die allgemeine Form eines Bayes'schen Klassifikators, der optimal ist, wenn alle statistischen Annahmen zutreffen und die Verlustmatrix bekannt ist. Um diese allgemeine Form für die Praxis verwenden zu können, werden eine Reihe von Vereinfachungen gemacht.

Die apriori-Wahrscheinlichkeiten $p(K_i)$ der Musterklassen sind in der Regel nicht bekannt. Eine grobe Näherung erhält man aus einer visuellen Beurteilung der zu klassifizierenden Szene, wenn man die Häufigkeiten der einzelnen Objekte abschätzt. Weiß man z.B. von einem medizinischen Präparat, dass etwa 70% des Bildausschnittes zum Hintergrund gehören, 10% von Zellen des Typs 1 und 20% von Zellen des Typs 2 eingenommen werden, so könnten die zugehörigen apriori-Wahrscheinlichkeiten mit 7/10, 1/10 und 2/10 angegeben werden.

Wenn der Klassifikation ein unüberwachter Cluster-Algorithmus vorausgegangen ist, so kann man die apriori-Wahrscheinlichkeiten aus der Häufigkeit der einzelnen Musterklassen ableiten.

In allen Fällen, in denen sich keine Information über die $p(K_i)$ gewinnen lässt, werden sie als gleichwahrscheinlich angenommen, d.h.:

$$p(K_i) = \frac{1}{t}. \tag{20.25}$$

Für die Verlustmatrix $\mathbf{L}$ wird vereinfachend angenommen, dass bei richtiger Klassifizierung kein Verlust ($l_{ij} = 0$) und bei falscher Klassifizierung immer derselbe Verlust eintritt ($l_{ij} = 1, i \neq j$). Damit reduziert sich (20.24) zu:

$$f(\mathbf{g}) - p(K_j) \cdot f(\mathbf{g}|K_j) < f(\mathbf{g}) - p(K_\mu) \cdot f(\mathbf{g}|K_\mu), \quad \mu = 0, ..., t-1, \mu \neq j. \tag{20.26}$$

Da sich für $f(\mathbf{g})$ hier immer derselbe Wert ergibt, wird es weggelassen. Die Größe, die zur Klassifizierungsentscheidung verwendet wird, ist dann:

$$d_i^{'}(\mathbf{g}) = p(K_i) \cdot f(\mathbf{g}|K_i) \tag{20.27}$$

und $\mathbf{g}$ wird zu K_j klassifiziert, falls gilt:

$$d_j^{'}(\mathbf{g}) > d_\mu^{'}(\mathbf{g}) \text{ für alle } \mu \neq j. \tag{20.28}$$

Für die Verteilungsdichten $f(\mathbf{g}|K_i)$ wird angenommen, dass sie mit Hilfe von N-dimensionalen Gauß'schen Normalverteilungen angenähert werden können:

$$f(\mathbf{g}|K_i) = \frac{1}{(2\pi)^{\frac{N}{2}}} \cdot \frac{1}{\det(\mathbf{C}_i)^{\frac{1}{2}}} \cdot \exp\Big(-\frac{1}{2}(\mathbf{g}-\mathbf{z}_i)^T \cdot \mathbf{C}_i^{-1} \cdot (\mathbf{g}-\mathbf{z}_i)\Big). \tag{20.29}$$

Dabei ist $\mathbf{C}_i$ die Kovarianzmatrix der Musterklasse K_i und $\mathbf{z}_i$ der Mittelwertvektor. Wegen der Monotonie des Logarithmus ist es möglich, obige Formel zu logarithmieren, so dass sie schließlich folgende Form erhält:

$$d_i(\mathbf{g}) = \ln\Big(p(K_i)\Big) - \frac{1}{2}\ln\Big(\det(\mathbf{C}_i)\Big) - \frac{1}{2}(\mathbf{g}-\mathbf{z}_i)^T \cdot \mathbf{C}_i^{-1} \cdot (\mathbf{g}-\mathbf{z}_i)\Big). \tag{20.30}$$

Ein unbekannter Merkmalsvektor $\mathbf{g}$ wird von diesem *Maximum-Likelihood-Klassifikator* der Musterklasse K_j zugewiesen, falls gilt:

$$d_j(\mathbf{g}) > d_i(\mathbf{g}) \text{ für alle } i \neq j. \tag{20.31}$$

Ähnlich wie im Falle des Minimum-Distance-Klassifikators ist es sinnvoll, auch hier eine Zurückweisungklasse einzuführen. Die Größe

$$(\mathbf{g}-\mathbf{z}_i)^T \cdot \mathbf{C}_i^{-1} \cdot (\mathbf{g}-\mathbf{z}_i)\Big). \tag{20.32}$$

heißt *Mahalanobis-Abstand* des Bildpunktes $\mathbf{g}$ von der Musterklasse K_i. In die Zurückweisungsklasse werden alle Bildpunkte eingewiesen, deren Mahalanobis-Abstand größer als eine vorgegebene Schwelle ist.

Der geometrische Ort aller Bildpunkte, die von einer Musterklasse vermöge der Mahalanobis-Distanz denselben Abstand haben, ist ein N-dimensionales Ellipsoid, im Zweidimensionalen also eine Ellipse. So werden von diesem Maximum-Likelihood-Klassifikator mit Zurückweisung die Musterklassen auch wieder durch geometrische Gebilde approximiert. Die Bilder 20.8 zeigen die Aufteilung des Merkmalsraums durch den Maximum-Likelihood-Klassifikator mit verschiedenen Zurückweisungsradien. Als Grundlage wurde die Szene mit den Trainingsgebieten von Bild 20.6-a und und des Merkmalsraums von Bild 20.7-a verwendet.

Die Dimension des Merkmalsraumes, die sich in (20.30) u.a. in der Kovarianzmatrix $\mathbf{C}_i$ der Musterklasse K_i widerspiegelt, geht in die Berechnung quadratisch ein. Dies kann bei höheren Dimensionen zu beachtlichen Rechenzeiten führen. Es gibt verschiedene Ansätze diese Rechenzeiten zu reduzieren. Eine Möglichkeit besteht darin, den Maximum-Likelihood-Algorithmus statt in einer normalen Rechenanlage in eigens für diese Art der Klassifizierung konstruierter Spezialhardware ablaufen zu lassen. Der eigentliche Algorithmus wird dabei nicht geändert.

Bei einer anderen Möglichkeit wird zu jeder Musterklasse ein *Kern* berechnet. Ein Kern ist dabei ein N-dimensionales Ellipsoid, das so gewählt wird, dass, wenn ein Bildpunkt innerhalb des Kernes einer Musterklasse liegt, sofort entschieden werden kann, dass er zu dieser Musterklasse gehört.

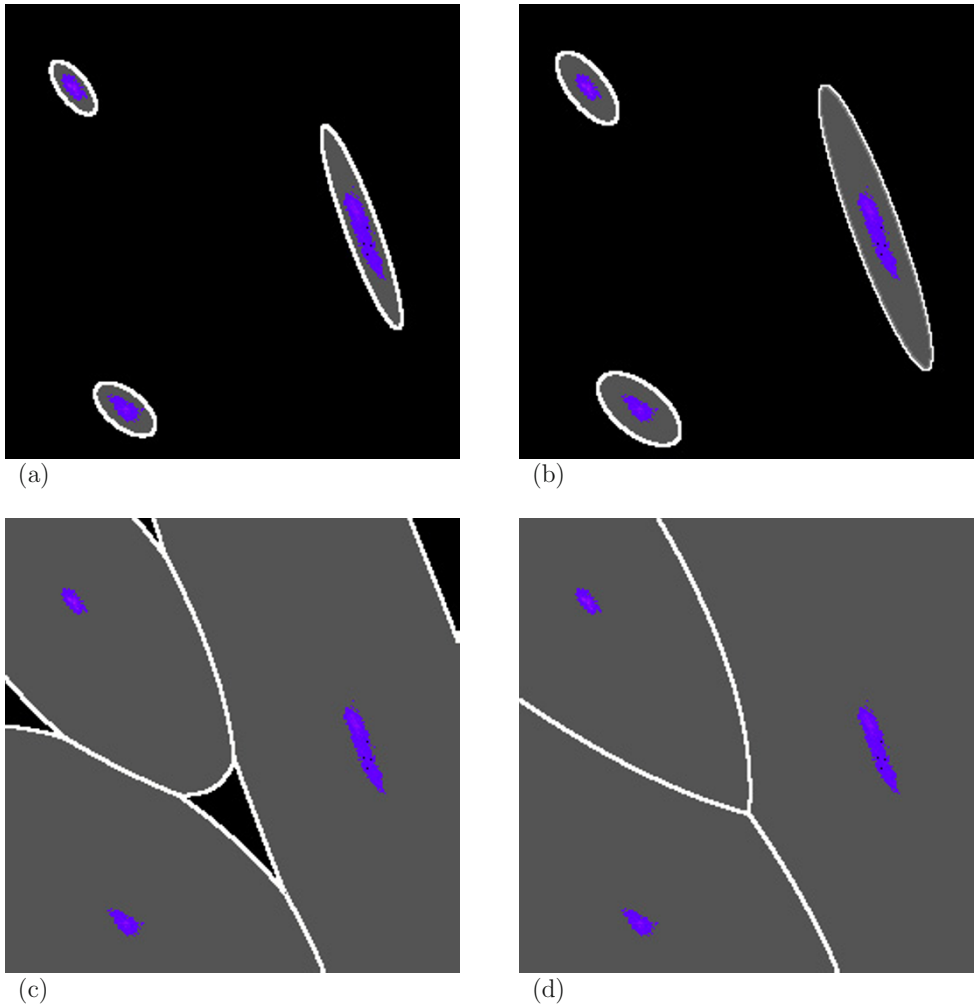


Bild 20.8: (a) - (c) Aufteilung des Merkmalsraums durch den Maximum-Likelihood-Klassifikator zum Klassifizierungstest von Abschnitt 20.5 mit unterschiedlichen Zurückweisungsradien. (d) Aufteilung ohne Zurückweisung.

Der Maximum-Likelihood-Klassifikator kann auch in einem überwacht lernenden Klassifizierungssystem verwendet werden. Dazu wird hier, wie in Abschnitt 20.5 beschrieben, bei der Zuweisung eines Bildpunktes $\mathbf{g}$ zur Musterklasse K_i der Mittelwertvektor $\mathbf{z}_i$ verbessert. Zusätzlich muss dann hier auch noch die Kovarianzmatrix $\mathbf{C}_i$ angepasst werden. Dies sei hier für eine Musterklasse und ein Kanalpaar (n_1, n_2) dargestellt. Der Index für die Musterklasse wird aus Gründen der Übersichtlichkeit weggelassen.

$\mathbf{z}_n^{(anz)}$ sei der Mittelwertvektor der Musterklasse im Kanal n, nach der Zuweisung von anz Merkmalsvektoren.

$\mathbf{z}_n^{(anz+1)}$ der verbesserte Mittelwertvektor nach der Zuweisung des $(anz+1)$-ten Merkmalsvektors $\mathbf{g}$.

Es gilt:

$$\mathbf{z}_n^{(anz+1)} = \frac{1}{anz+1}\Big(anz \cdot \mathbf{z}_n^{(anz)} + \mathbf{g}\Big). \tag{20.33}$$

$v_{n_1,n_2}^{(anz)}$ sei die Kovarianz der Musterklasse in den Kanälen n_1 und n_2 nach der Zuweisung von anz Merkmalsvektoren.

$v_{n_1,n_2}^{(anz+1)}$ sei die Kovarianz der Musterklasse in den Kanälen n_1 und n_2 nach der Zuweisung des (anz+1)-ten Merkmalsvektors $\mathbf{g}$.

Dann berechnet sich die verbesserte Kovarianz wie folgt:

$$v_{n_1,n_2}^{(anz+1)} = \frac{1}{anz+1}\Big(anz \cdot v_{n_1,n_2}^{(anz)} + g_{n_1} g_{n_2} + anz \cdot z_{n_1}^{(anz)} \cdot z_{n_2}^{(anz)}\Big) - z_{n_1}^{(anz)} \cdot z_{n_2}^{(anz)}. \tag{20.34}$$

Werden zu allen Paaren von Kanälen die Kovarianzen mit (20.34) neu berechnet, so wurde damit in der Kovarianzmatrix $\mathbf{C}_j$ die Zuweisung des Bildpunktes $\mathbf{g}$ zur Musterklasse K_j berücksichtigt.

20.7 Der Quader-Klassifikator

Die beiden bis jetzt dargestellten Klassifikatoren haben die Musterklassen durch N-dimensionale Kugeln (Minimum-Distance-Klassifikator, Abschnitt 20.5) oder durch N-dimensionale Ellipsoide (Maximum-Likelihood-Klassifikator, Abschnitt 20.6) approximiert. Es liegt nahe, die Klassifizierung auch noch mit anderen, mathematisch einfach zu handhabenden Gebilden, wie z.B. N-dimensionalen, achsenparallelen Quadern zu versuchen. Wie können diese Quader beschrieben werden? Es sei:

$$k_i = \left\{(\mathbf{g}_0^{(i)}, h_0^{(i)}), (\mathbf{g}_1^{(i)}, h_1^{(i)}), ..., (\mathbf{g}_{u_i-1}^{(i)}, h_{u_i-1}^{(i)})\right\} \tag{20.35}$$

die Realisation der i-ten Musterklasse.

Der Quader Q_i, der der Musterklasse K_i zugeordnet wird, ist ein N-Tupel von Zahlenpaaren, gemäß:

$$Q_i = \left\{(a_0^{(i)}, b_0^{(i)}), (a_1^{(i)}, b_1^{(i)}), ..., (a_{N-1}^{(i)}, b_{N-1}^{(i)})\right\}. \tag{20.36}$$

Dabei sind die Größen $a_n^{(i)}$ und $b_n^{(i)}$ die jeweils linke und rechte Begrenzung des Quaders im Kanal $n = 0, 1, ..., N-1$ (Bild 20.9). Sie können z.B. aus dem Mittelwertvektor $\mathbf{z}_i$ und dem Vektor der Streuungen q_i berechnet werden:

$$a_n^{(i)} = z_{i,n} - c \cdot \sqrt{q_{i,n}}; \quad b_n^{(i)} = z_{i,n} + c \cdot \sqrt{q_{i,n}}. \tag{20.37}$$

Auch hier ist wieder überwachtes Lernen möglich, wenn die Quaderbegrenzungen bei jeder Zuweisung eines neuen Bildpunktes angepasst werden.

Wird zunächst vorausgesetzt, dass die Quader paarweise disjunkt sind, also

$$Q_i \bigcap Q_j = \{\}, \text{für } i \neq j, \tag{20.38}$$

so kann folgende Zuordnungsvorschrift verwendet werden:

> Ein unbekannter Merkmalsvektor $\mathbf{g}$ wird der Musterklasse K_j zugewiesen, falls er im Quader Q_j dieser Musterklasse liegt, d.h. falls $\mathbf{g}$ im Intervall $[a_n^{(j)}, b_n^{(j)}], n = 0, 1, ...N-1$ liegt. (20.39)

Dieser Quader-Klassifikator ist sehr einfach zu implementieren und rechenzeitsparend, da die Prüfung, ob ein Bildpunkt in einem Quader liegt, nur aus Vergleichsoperationen besteht. Er ist in solchen Fällen gut geeignet, in denen die verwendeten Merkmale unkorreliert sind, d.h. dass sich die Musterklassen nicht schmal und schräg-langgezogen ausprägen (Bild 20.10). So ist hier ein gewisser Querbezug zur Hauptkomponententransformation (Abschnitt 13.4) gegeben, die ja neue, unkorrelierte Merkmale berechnet.

Der Quader-Klassifikator wäre aber praktisch nicht verwendbar, wenn man die Forderung, dass sich die Quader der einzelnen Musterklassen nicht überdecken dürfen, aufrecht

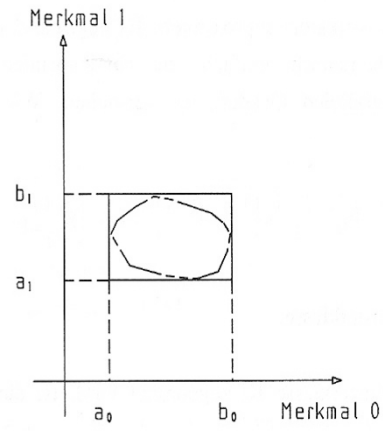


Bild 20.9: Beispiel eines „zweidimensionalen“ Quaders (Rechtecks) zur Approximation einer Musterklasse.

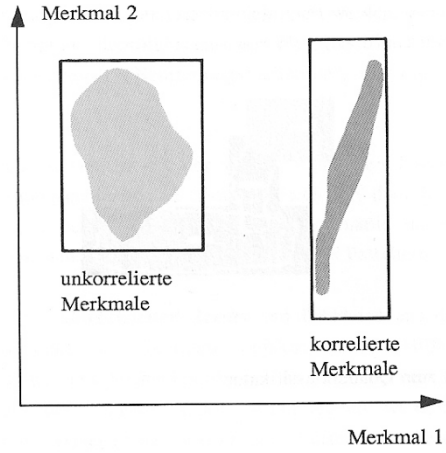


Bild 20.10: Die Approximation der Musterklassen bei der Quadermethode ist in der Regel besser, wenn die verwendeten Merkmale unkorreliert sind.

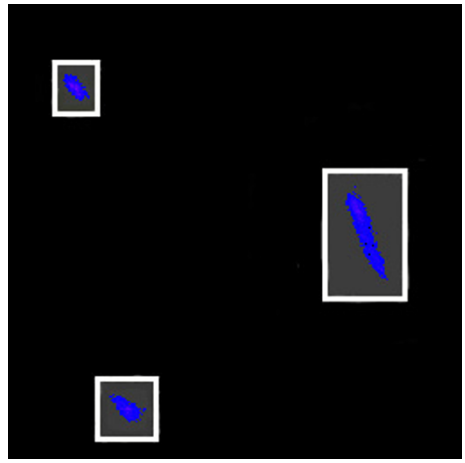


Bild 20.11: Beispiel zum Quader-Klassifikator (zum Testbeispiel in Bild 20.6-a und in Bild 20.7-a: Hier liegen die Quader deutlich voneinander getrennt.

halten würde. Was geschieht aber dann mit den Bildpunkten, die im Überdeckungsbereich von einem oder sogar von mehreren Quadern liegen?

Als eine Möglichkeit kann man diese Bildpunkte einer Sonderklasse „Überdeckungsbereich“ zuweisen. Diese Vorgehensweise stellt sich in der Praxis aber auch als unbefriedigend heraus, denn was soll mit den Bildpunkten bei der weiteren Verarbeitung geschehen?

Die beste Möglichkeit ist es, die Bildpunkte in Überdeckungsbereichen mit anderen, etwa den in Abschnitt 20.5 und 20.6 besprochenen Verfahren, zu klassifizieren. Praktische Untersuchungen haben gezeigt, dass in der Regel weniger als 1/3 der Bildpunkte in Überdeckungsbereichen liegen, so dass die Rechenzeitvorteile immer noch gegeben sind. Bild 20.11 ist das Ergebnis des Quader-Klassifikators, angewendet auf das in den Abschnitten 20.5 und 20.6 verwendete Beispiel.

Der Algorithmus zu diesem Klassifikator lautet somit:

A20.2: Quader-Klassifikator.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S}_e = (s_e(x, y, n))$ ein N-kanaliges Bild, dessen Kanäle einen N-dimensionalen Merkmalsraum aufspannen. Als Menge von Merkmalsvektoren geschrieben: $S = \{\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_{M-1}\}$.

◇ $\mathbf{S}_a = (s_a(x,y))$ ein einkanaliges Grauwertbild mit der Grauwertmenge $G = \{0,1,...,255\}$ (Ausgabebild). Die Grauwerte sind Codes für die Zugehörigkeit zu einer Klasse.

Algorithmus:

(a) Im Training werden aus der repräsentativen Stichprobe, aus denen die Realisationen k_i der Klassen K_i abgeleitet wurden, die Quadergrenzen $(a_n^{(i)}, b_n^{(i)})$ berechnet.

(b) Für alle Merkmalsvektoren $\mathbf{g}$ des Bildes:

(ba) Für alle Klassen $K_i, i = 0,1,...,t-1$: Überprüfen, ob $\mathbf{g}$ in Q_i liegt.

(bb) Falls $\mathbf{g}$ in nur einem Quader Q_j liegt: Zuordnung von $\mathbf{g}$ zur Klasse K_j.

(bc) Falls $\mathbf{g}$ in mehreren Quadern liegt:

(bca) Berechnung der Distanzen von $\mathbf{g}$ zu allen Zentrumsvektoren derjenigen Quader, in denen $\mathbf{g}$ liegt.

(bcb) Zuordnung von $\mathbf{g}$ zu der Klasse K_j, falls er zu dieser Klasse den kürzesten Abstand hat. Bei gleichen Abständen kann er zufällig einer Klasse zugewiesen werden.

(bd) Falls $\mathbf{g}$ in keinem Quader liegt, wird er der Zurückweisungsklasse zugeordnet.

Ende des Algorithmus

20.8 Beurteilung der Ergebnisse

Zur Beurteilung der Klassifizierungsergebnisse bieten sich mehrere Möglichkeiten an. Bei vielen praktischen Anwendungen ist es sinnvoll, vor der eigentlichen Klassifizierung eine apriori-Beurteilung durchzuführen. Dazu werden bei einem festdimensionierten Klassifikator zunächst, wie in den vorhergehenden Abschnitten beschrieben wurde, die benötigten Größen (Mittelwertvektor, Vektor der Streuungen, Kovarianzmatrix, usw.) berechnet. Bevor aber die eigentliche Szene klassifiziert wird, werden jetzt die einzelnen Stichproben klassifiziert. Im Idealfall sollte natürlich eine Stichprobe für ein bestimmtes Objekt auch zu 100% als zu diesem Objekt gehörig klassifiziert werden. Die Abweichungen von diesem Idealwert erlauben es, die zu erwartende Qualität zu beurteilen: Wenn nämlich die Klassifizierung der Stichproben bereits schlechte Ergebnisse liefert, so ist es unwahrscheinlich, dass die Klassifizierung der gesamten Szene besser ist. Diese Beurteilung kann auch automatisiert werden. Man kann etwa die Ergebnisse der Klassifizierung aller Stichproben zu einer Maßzahl zusammenziehen, die, wie die mathematische Wahrscheinlichkeit, Werte zwischen 0 und 1 annimmt und dann die zu erwartende Güte z.B. in Prozent angibt.

Diese Art der apriori-Beurteilung kann noch objektiviert werden, wenn man die Beurteilung nicht anhand der Stichprobe durchführt, aus der die Klassifizierungsparameter

abgeleitet wurden, sondern anhand einer „Beurteilungsstichprobe“, die eigens für diesen Zweck festgelegt wurde.

Eine weitere Möglichkeit besteht aus der visuellen und manuellen Interpretation von speziell ausgewählten Beurteilungsgebieten und dem Vergleich mit dem Klassifizierungsergebnis. Hierbei ist allerdings zu bedenken, dass die visuelle und manuelle Interpretation auch Fehler beinhaltet und unter Umständen „etwas Falsches mit etwas Falschem“ verglichen wird.

Handelt es sich bei den klassifizierten Szenen um Bilddaten aus der Fernerkundung, so bietet sich eine Möglichkeit an, die relativ objektiv ist. Mit Hilfe eines Zufallszahlengenerators werden Zeilen- und Spaltenkoordinaten im klassifizierten Bild erzeugt und deren Zuordnung zu einem der Objekte notiert. Anschließend werden die entsprechenden Positionen entweder in vergleichbaren Karten oder Luftbildern überprüft oder es wird im Rahmen einer Feldbegehung die tatsächliche Zugehörigkeit zu einer der Klassen bestimmt und mit dem Klassifizierungsergebnis verglichen. Diese Vorgehensweise ist zwar etwas zeitaufwändig, liefert aber ein Ergebnis, das auch nach dem Abschluss noch überprüfbar ist.

20.9 Ergänzungen

In den vorhergehenden Abschnitten wurden einige grundlegende Algorithmen zur Klassifizierung dargestellt. Bei praktischen Anwendungen wird man viele Varianten und Kombinationen mit anderen Lösungsansätzen finden.

So wurden bei den geschilderten Verfahren die Bildpunkte im N-dimensionalen Merkmalsraum parallel klassifiziert, was bedeutet, dass zur Entscheidung alle Merkmalskanäle gleichzeitig herangezogen werden. Bei einem sequentiellen Klassifikator werden nur diejenigen Komponenten $g_0, g_1, ..., g_{N-1}$ eines Bildpunktes $\mathbf{g}$ verwendet, die zu einer Klassifizierung mit ausreichender Sicherheit notwendig sind.

Bei den dargestellten Verfahren wurden überall implizit Trennungsfunktionen berechnet, nach deren Maßgabe die Bildpunkte zugewiesen wurden. Im Falle des Minimum-Distance-Klassifikators waren dies z.B. N-dimensionale Hyperebenen. Die explizite Berechnung der Trennungsfunktionen ist eine weitere Alternative.

Wenn die Dimension des Merkmalsraumes klein ist, etwa N=1, 2 oder 3, so kann die Klassifizierung auch über *table-look-up-Verfahren* durchgeführt werden. Dazu wird z.B. der zweidimensionale Merkmalsraum diskretisiert und durch eine Tabelle von $256 \cdot 256$ Positionen repräsentiert. Für jede Position der Tabelle wird dann angegeben, zu welcher Musterklasse sie gehört. Die dazu notwendigen Informationen können aus der Stichprobe gewonnen werden. Welcher Klassifikator dazu verwendet wird, spielt keine Rolle. Die eigentliche Klassifizierung eines Bildpunktes $\mathbf{g}$ besteht nur mehr darin, in der Tabelle nachzusehen, zu welcher Klasse er gehört. Auch überwachtes Lernen ist möglich, wenn die Tabelle nach jeder Zuweisung modifiziert wird. Dieses Verfahren, das sehr rechenzeitoptimal abläuft, findet aber bei höherdimensionalen Merkmalsräumen schnell seine Grenzen, da bei einem 4-dimensionalen Merkmalsraum die Tabelle bereits $256 \cdot 256 \cdot 256 \cdot 256$ Einträge umfassen müsste.

Kapitel 21

Klassifizierung mit neuronalen Netzen

21.1 Grundlagen: Künstliche neuronale Netze

In diesem Abschnitt wird eine kurze Übersicht zur Thematik *künstliche neuronale Netze* gegeben. Zu ausführlichen Darstellungen sei auf die vielfältige Literatur, z.B. [Koho88], [Wass89] oder [Krat90] verwiesen. Nach der Erläuterung des prinzipiellen Aufbaus neuronaler Netze werden einige Netztypen kurz vorgestellt. In Abschnitt 21.2 wird die Eignung der neuronalen Netze für die digitale Bildverarbeitung und Mustererkennung untersucht.

21.1.1 Prinzipieller Aufbau

Der prinzipielle Aufbau eines künstlichen Neurons ist in Bild 21.1 dargestellt.

Die Größen $x_i, i = 1, ..., m$ sind die Eingabewerte (die *Eingangsaktivierungen*) des Neurons. Sie werden häufig auch als Vektor $\mathbf{x} = (x_1, x_2, ..., x_m)^T$ geschrieben. Jedem Eingang ist ein *(Netz-)Gewicht* $w_i, i = 1, ..., m$ zugeordnet. Die *Netzaktivität* net wird wie folgt berechnet:

$$net = \sum_{i=1}^{m} x_i w_i. \tag{21.1}$$

Nachgeschaltet ist eine *Aktivierungsfunktion* $f(x)$, mit der die Ausgabe (*Ausgangsaktivierung*) out berechnet wird:

$$out = f(net). \tag{21.2}$$

Als Aktivierungsfunktion werden je nach Neuronentyp unterschiedliche Funktionen verwendet:

- $f(x)$ ist eine lineare Funktion (möglicherweise auch die Identität). Dieser Fall, der einem Klassifikator entspricht, welcher den Merkmalsraum mit Hyperebenen aufteilt,

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_21

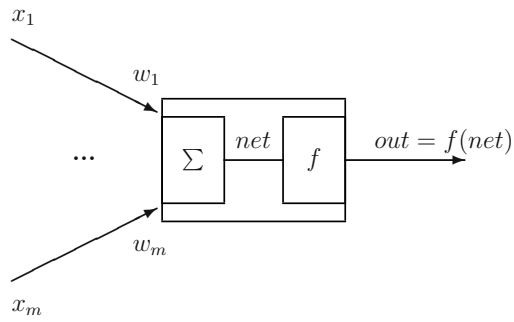


Bild 21.1: Grundlegende Struktur eines künstlichen Neurons.

wird als neuronales Netz in der Praxis selten verwendet, da die Einsatzmöglichkeiten eingeschränkt sind.

- $f(x)$ ist eine Schwellwert- oder Treppenfunktion. Hierbei handelt es sich um eine weit verbreitete Klasse von Aktivierungsfunktionen, da sie es erlauben, diskrete Ausgangsaktivierungen zu erzeugen.
- $f(x)$ ist die *Sigmoidfunktion*:

$$f(x) = \frac{1}{1 + e^{-\frac{x-a}{r}}}. \tag{21.3}$$

  Der Parameter a bestimmt die Lage des Durchgangs durch das Niveau 0.5 und der Parameter r die Steigung der Sigmoidfunktion. Die Sigmoidfunktion gehört zur Klasse der semilinearen Funktionen, die u.a. die Eigenschaft besitzen, überall differenzierbar zu sein.
- Manche Netze verwenden auch stochastische Aktivierungen (z.B. die Boltzmann-Maschine).

21.1.2 Adaline und Madaline

Zwei Spezialisierungen des allgemeinen Konzepts sind das Adaline (Adaptive Linear Neuron, Bild 21.2) und das Madaline (Multiple Adaline). Netze dieser Typen wurden in der Anfangszeit der Forschungen auf dem Gebiet der neuronalen Netze untersucht. Wegen ihrer eingeschränkten Fähigkeiten werden sie heute in der Praxis selten verwendet. Es lassen sich jedoch die grundlegenden Fragestellungen gut anhand dieser Netze erläutern.

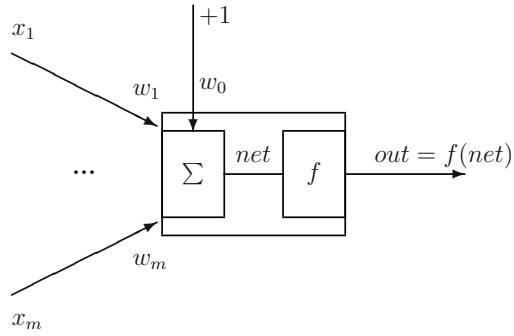


Bild 21.2: Struktur eines Adaline (Adaptive Linear Neuron).

Für die Größen $x_i, i = 1, ..., m$, also die Eingangsaktivierungen, gilt: $x_i \in \{-1, +1\}$. Das Gewicht w_0 übernimmt die Funktion eines Schwellwertes. Es ist die Aktivierung eines fiktiven Eingabeelements, das als Eingabeaktivierung immer mit $+1$ besetzt ist. Die Netzaktivität *net* und die Ausgabe *out* werden wie folgt berechnet:

$$net = \sum_{i=1}^{m} x_i w_i + w_0, \tag{21.4}$$

$$out = \begin{cases} +1, & \text{falls } net > 0, \text{ also } \sum\limits_{i=1}^{m} x_i w_i > -w_0 \\ -1, & \text{sonst.} \end{cases} \tag{21.5}$$

Es wird hier also eine Schwellwertfunktion als Aktivierungsfunktion verwendet.

Im Folgenden wird die Arbeitsweise eines Adaline am Beispiel des logischen AND gezeigt. Die Wahrheitswerte TRUE und FALSE werden durch $+1$ und -1 dargestellt. Als Gewichte werden verwendet: $w_1 = 0.1$ und $w_2 = 0.1$. Der Schwellwert ist $w_0 = -0.1$. Damit sehen die Werte wie folgt aus:

Eingabe x_1/x_2	*net*	*out*	Soll
$-1/-1$	-0.3	-1	-1
$-1/+1$	-0.1	-1	-1
$+1/-1$	-0.1	-1	-1
$+1/+1$	$+0.1$	$+1$	$+1$

Andere logische Operatoren lassen sich ähnlich realisieren. Schwierigkeiten gibt es jedoch bei XOR und bei NXOR. Eine Betrachtung des zweidimensionalen Raumes der Ein-

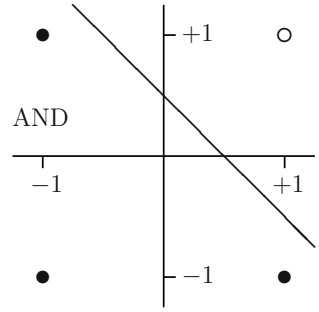


Bild 21.3: Zweidimensionaler Raum der Eingabevektoren für die zweistelligen logischen Operatoren der Boole'schen Algebra. Die Wahrheitswerte TRUE und FALSE werden durch $+1$ und -1 dargestellt.

gabevektoren für die zweistelligen logischen Operatoren der Boole'schen Algebra zeigt, warum das so ist (Bild 21.3).

Die Formel (21.4) sieht ausgeschrieben wie folgt aus:

$$x_1 w_1 + x_2 w_2 + w_0 = net. \tag{21.6}$$

Ein Adaline teilt den Raum der Eingabevektoren durch eine Gerade in zwei Teilmengen M^+ und M^-, so dass alle Eingabevektoren mit $out = +1$ in M^+ und alle Eingabevektoren mit $out = -1$ in M^- liegen. Im Fall des **XOR** ist das nicht möglich: Es müssten nämlich die Punkte $(x_1, x_2) = (-1, -1)$ und $(x_1, x_2) = (+1, +1)$ auf der einen Seite der Gerade und die Punkte $(x_1, x_2) = (+1, -1)$ und $(x_1, x_2) = (-1, +1)$ auf der anderen Seite liegen. Ein Adaline in dieser Form ist also nur bei Problemstellungen einsetzbar, bei denen die Eingabevektoren $\mathbf{x}$ linear separierbar sind. Bei hochdimensionalen Eingabevektoren $\mathbf{x}$ ist es nicht mehr trivial möglich zu entscheiden, ob der Merkmalsraum linear separierbar ist.

Ein anderes Problem ist die Wahl der Gewichte. Bei allgemeinen m-dimensionalen Eingabevektoren ist die Wahl der Gewichte $w_i, i = 0, 1, ..., m$ nicht durch geometrische Überlegungen möglich. Vielmehr müssen die Gewichte von anfänglichen Startwerten aus an die jeweilige Problemstellung adaptiert werden. Einen beispielhaften Algorithmus dazu beschreibt A21.1:

A21.1: Automatische Adaption der Gewichte eines Adaline.

Voraussetzungen und Bemerkungen:

◇ Die Gewichte werden mit $w_i, i = 0, 1, ..., m$ bezeichnet. w_0 ist der Schwellwert.

Algorithmus:

(a) Die Gewichte $w_i, i = 0, 1..., m$ werden mit zufälligen Werten besetzt.

(b) Nacheinander werden zufällige, gleichverteilte Eingabevektoren $\mathbf{x} = (x_1, ..., x_m)^T$ angelegt.

(c) Es wird eine Soll-Ausgabe out_{soll} ermittelt:

$$out_{soll} = \begin{cases} +1, & \text{falls } \mathbf{x} \in M^+ \\ -1, & \text{falls } \mathbf{x} \in M^- \end{cases}$$

(d) Die Abweichung von net vom Zielwert out_{soll} wird berechnet:

$\delta_{ges} = out_{soll} - net.$

(e) Um den ermittelten Fehler auszugleichen, werden alle Gewichte um einen Betrag δ verändert:

$\delta = \frac{\delta_{ges}}{m+1}.$

$w_i \Leftarrow w_i + \eta x_i \delta; \qquad i = 1, ..., m;$

$w_0 \Leftarrow w_0 + \eta\delta.$

Der Faktor η wird als *Lernfaktor* bezeichnet. Er bestimmt, in welchem Maß der ermittelte Fehler die Gewichtskorrektur beeinflusst.

Ende des Algorithmus

In dieser Form hat ein Adaline den entscheidenden Nachteil, dass der m-dimensionale Raum der Eingabevektoren nur linear separiert werden kann. Einfache Problemstellungen, wie z.B. die Realisierung eines logischen XOR, sind damit nicht möglich. Aus diesem Grund wurden Erweiterungen des Konzepts durchgeführt. Eine davon führt zum *Madaline* (Multiple Adaline), das in Bild 21.4 abgebildet ist.

Die Eingabeaktivierungen werden auf alle Adalines gelegt. Diese arbeiten autonom nach dem oben beschriebenen Verfahren. Jedes Adaline liefert eine Ausgabeaktivierung out_i, die durch ein *Verdichtungsverfahren* im Madaline zusammengefasst werden. Beispiele für Verdichtungsverfahren sind:

- Das *Mehrheitsverfahren*.

$$out_{mada} = \begin{cases} +1, & \text{falls } \sum_{i=1}^{n} out_i > 0 \\ -1, & \text{sonst} \end{cases} \tag{21.7}$$

Als Endaktivierung wird die mehrheitlich erzielte Aktivierung der Adalines verwendet.

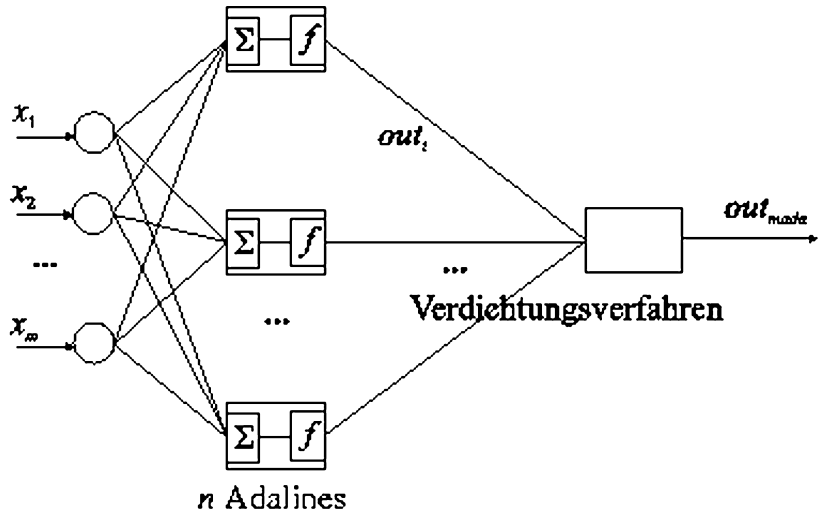


Bild 21.4: Architektur eines Madaline.

- Das *Einstimmigkeitsverfahren*.

$$out_{mada} = \min\{out_i, i = 1, ..., n\}. \tag{21.8}$$

  Hier wird als Gesamtergebnis nur dann +1 ausgegeben, falls alle Adalines den Wert +1 liefern.

- Das *Singulärverfahren*.

$$out_{mada} = \max\{out_i, i = 1, ..., n\}. \tag{21.9}$$

  Hier wird als Gesamtergebnis schon +1 ausgegeben, falls nur ein Adaline den Wert +1 liefert.

Damit können auch nicht linear separierbare Probleme behandelt werden. Ein Problem bei der Anwendung eines Madalines ist, dass man bei einer gegebenen Anwendung nicht weiß, welches Verdichtungsverfahren man verwenden soll. Im Fall des NXOR (Not-XOR, auch Äquivalenz genannt) ist das Einstimmigkeitsverfahren geeignet (Bild 21.5).

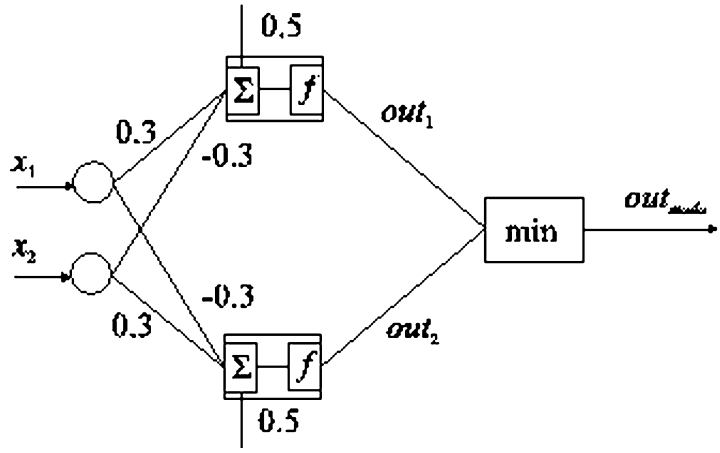


Bild 21.5: Realisierung des NXOR mit einem Madaline.

21.1.3 Das Perceptron

Das Perceptron ist ein künstliches neuronales Netzwerk, das ebenfalls in der Anfangszeit der Forschung auf dem Gebiet der neuronalen Netze ausführlich untersucht wurde. Es besitzt im Prinzip dieselbe Struktur wie das Adaline. Netze, die durch die Parallelschaltung mehrerer Elemente entstehen, werden auch als Perceptron bezeichnet. Der Eingabevektor $\mathbf{x} = (x_1, ..., x_m)^T$ wird auf die *Eingabeschicht* gegeben. Für die Eingabewerte gilt: $x_i \in \{0, 1\}$. Das i-te Neuron der Eingabeschicht ist mit dem j-ten Neuron der *verdeckten Schicht* (d.h. einer Schicht aus Neuronen zwischen Ein- und Ausgabeschicht) mit dem Gewicht w_{ji} verbunden. Die Netzaktivitäten net_j bilden mit der Aktivierungsfunktion f die Ausgangsaktivierungen out_j, wobei gilt:

$$out_j = \begin{cases} 1, & \text{falls } net_j \geq 0 \\ 0, & \text{sonst} \end{cases} \tag{21.10}$$

Die Beziehung zur Berechnung der Größen net_i lautet somit wie folgt:

$$(net_1, net_2, \cdots, net_n) = (x_1, \cdots, x_m) \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{n1} \\ w_{12} & w_{22} & \cdots & w_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ w_{1m} & w_{2m} & \cdots & w_{nm} \end{pmatrix} = \mathbf{x}^T \mathbf{W}. \tag{21.11}$$

Es wäre denkbar, statt nur einer Summationsschicht mehrere hintereinander zu schalten. Auf Grund der linearen Beziehung zwischen dem Eingabevektor und den Netzaktivie-

rungen lässt sich aber leicht zeigen, dass derartige mehrschichtige Netze mit einschichtigen äquivalent sind:

$$\Big(\Big(\cdots(\mathbf{x}^T\mathbf{W_1})\mathbf{W_2}\Big)\cdots\Big)\mathbf{W_k} = \mathbf{x}^T\mathbf{W}. \tag{21.12}$$

Dies gilt nicht, wenn mehrschichtige Netze mit kompletten Perceptronbausteinen und nichtlinearen Aktivierungsfunktionen verwendet werden, was weiter unten noch ausführlicher untersucht wird.

Wie das Training abläuft, ist in Algorithmus A21.2 beschrieben.

A21.2: Adaption der Gewichte bei einem Perceptron-Netz.

Voraussetzungen und Bemerkungen:

◇ Die Gewichte werden mit $w_{ji}, i = 1, ..., m; j = 1, .., n$ bezeichnet.

Algorithmus:

(a) Die Gewichte w_{ji} werden mit zufälligen Werten besetzt (z.B. mit 0 initialisiert).

(b) Eingabevektoren $\mathbf{x}$ werden zufällig ausgewählt. Zu jedem Eingabevektor sei die Soll-Ausgabe $\mathbf{out}_{soll}$ bekannt.

(c) Wenn für die Ausgabekomponente j ein korrektes Ergebnis erzielt wird, so wird keine Gewichtskorrektur durchgeführt.

(d) Falls das Ergebnis falsch war, wird ein Fehlermaß δ_j berechnet:

$\delta_j = out_{soll,j} - out_j$.

Die δ_j gehören zur Menge $\{-1, +1\}$, da die out_j entweder 0 oder 1 sind und die Gewichtskorrektur nur bei einem falschen Ergebnis durchgeführt wird.

(e) Die Gewichte w_{ij} werden jetzt nach folgender Regel korrigiert:

$w_{ij} \Leftarrow w_{ij} + \eta \cdot x_i \cdot \delta_j$.

η ist ein Lernfaktor. Da die x_i entweder 0 oder 1 sind, werden nur solche Gewichte korrigiert, die von einer aktiven Eingabe angesteuert wurden.

Ende des Algorithmus

21.1.4 Backpropagation

Es liegt die Vermutung nahe, dass unter Umständen Netze mit mehreren Schichten doch leistungsfähiger sind. Das ist aber nur der Fall, wenn die beteiligten Neuronen nichtlineare Aktivierungsfunktionen besitzen. Ein weiteres Problem ist das Training: Bei nur einer Schicht können mit Hilfe eines geeigneten Fehlermaßes die Gewichte, wie oben dargestellt, angepasst werden, da ja von der Ausgabeschicht her ein Sollwert vorliegt. Gibt es aber dazwischen verdeckte Schichten, so liegen für die dazu gehörigen Ausgaben keine Sollwerte vor. Bei vorwärts- und rückwärtsvermittelnden Netzen wird dieses Problem mit Hilfe des *Backpropagation*-Algorithmus gelöst.

Bei Netzen dieser Art wird ein Neuron zugrunde gelegt, wie es oben schon beschrieben wurde. Die Eingabeaktivierungen x_i werden hier meistens aus dem Intervall $[-1, +1]$ oder $[0, +1]$ gewählt. Die Größen *net* und *out* berechnen sich wie gewohnt. Als Aktivierungsfunktion wird die Sigmoidfunktion verwendet:

$$f(net) = \frac{1}{1 + e^{-net}}. \tag{21.13}$$

Neben anderen günstigen Eigenschaften der Sigmoidfunktion (z.B. Differenzierbarkeit, Verhalten im Unendlichen, usw.) gilt für ihre Ableitung:

$$f'(net) = f(net)(1 - f(net)). \tag{21.14}$$

Diese Eigenschaft wird später, bei der Adaption der Netzgewichte, verwendet (Gradientenabstiegsverfahren). Die Einführung eines Bias (zusätzliche Eingabe, die immer +1 ist und ein eigenes Gewicht w_0 besitzt) ist möglich, wird in dieser Darstellung aber weggelassen. Mit diesem Neuron als Grundelement werden nun mehrschichtige Netze aufgebaut (Bild 21.6).

In der Eingabeschicht geschieht keine Verarbeitung, sondern es werden nur die Eingabewerte x_i auf die Eingabewerte a_i der (ersten) verdeckten Schicht verteilt. Die letzte Schicht ist die Ausgabeschicht. Ihre Ausgabewerte sind zugleich die Ausgabewerte out_j des gesamten Netzes.

Das Training wird mit zufällig ausgewählten Eingabevektoren $\mathbf{x}$ und den dazu gehörigen Sollvektoren $\mathbf{out}_{soll}$ durchgeführt. Es läuft wie in Algorithmus A21.3 dargestellt ab:

A21.3: Backpropagation.

Algorithmus:

(a) Die Gewichte der Neuronen werden mit zufälligen Werten besetzt.

(b) Es wird zufällig ein Trainingspaar $(\mathbf{x}, \mathbf{out}_{soll})$ ausgewählt.

(c) Mit Hilfe der *Vorwärtsvermittlung* wird ein Ausgabevektor **out** berechnet.

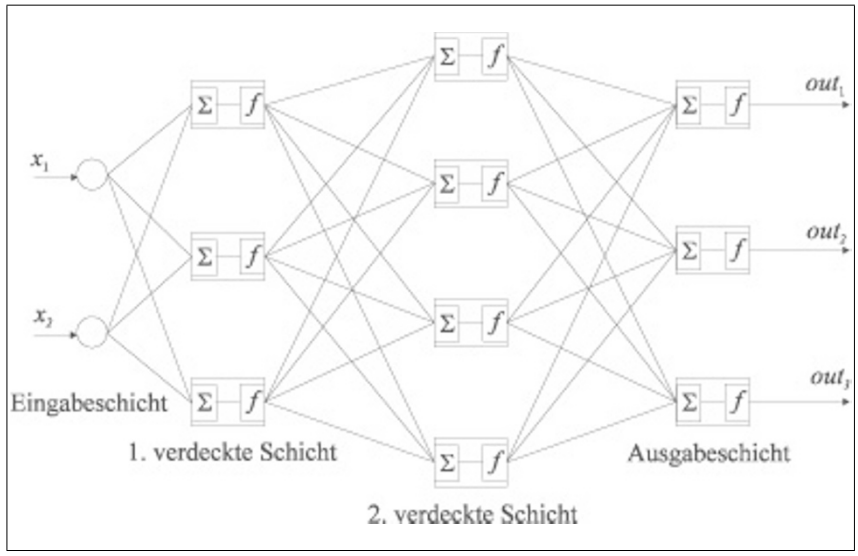


Bild 21.6: Mehrschichtiges Backpropagation-Netz.

(d) Es wird ein Fehlermaß zwischen **out** und $\mathbf{out}_{soll}$ berechnet.

(e) Die Gewichte werden so modifiziert, dass das Fehlermaß minimiert wird.

(f) Die Schritte (b)-(e) werden solange wiederholt, bis der Fehler akzeptabel klein ist.

Ende des Algorithmus

Die Positionen (b) und (c) von Algorithmus A21.3 werden als *Vorwärtsschritt* bezeichnet und die Positionen (d) und (e) als *Rückwärtsschritt*.

Zur genaueren Beschreibung werden folgende Annahmen gemacht (für ein Neuron sind die Bezeichnungen in Bild 21.7 zusammengestellt):

Das Netz habe die Schichten $k = 0, 1, ..., l$, wobei $l > 1$ sei. Mit $k = 0$ wird die Eingabeschicht bezeichnet. Hier wird keine Verarbeitung durchgeführt, sondern es werden die Elemente des Eingabevektors $\mathbf{x} = (x_1, x_2, ..., x_m)^T$ auf die Neuronen der ersten verdeckten Schicht ($k = 1$) verteilt.

Die Schicht mit dem Index $k = l$ ist die Ausgabeschicht. Die Schichten mit den Indizes $1 \leq k \leq l - 1$ sind die verdeckten Schichten. Jede Schicht besitzt n_k Neuronen. Für $k = 0$ gilt $n_0 = m$ und für $k = l$ gilt $n_l = n$. Die Eingabeschicht hat also m Neuronen und die Ausgabeschicht n Neuronen. Ein Trainingspaar $(\mathbf{x}, \mathbf{out}_{soll})$ besteht aus einem Eingabevektor und dem dazugehörigen Ausgabesollvektor.

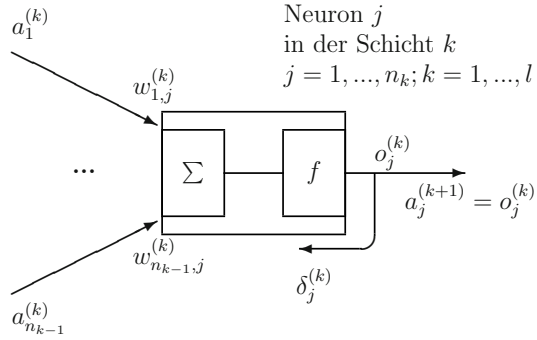


Bild 21.7: Bezeichnungen eines Neurons im Rahmen des Backpropagation-Algorithmus.

Die Eingangsaktivierungen eines Neurons j in der Schicht $k = 1, 2, ..., l$ seien $a_1^{(k)}, ..., a_{n_{k-1}}^{(k)}$. Die Ausgabe dieses Neurons wird mit $o_j^{(k)}$ bezeichnet. Die Eingaben der Neuronen der Schicht k sind, mit Ausnahme der ersten Schicht, die Ausgaben der Schicht $k-1$:

$$k = 1: \qquad a_i^{(k)} = x_i, i = 1, ..., n_0 = m. \tag{21.15}$$

$$k > 1: \qquad a_i^{(k)} = o_i^{(k-1)}, i = 1, ..., n_{k-1}; \tag{21.16}$$

Im Vorwärtsschritt wird zu jedem Neuron $j = 1, ..., n_k$ der Schicht $k = 1, 2, ...l$ die Ausgabe $o_j^{(k)}$ berechnet:

$$net_j^{(k)} = \sum_{i=1}^{n_{k-1}} a_i^{(k)} w_{ij}^{(k)}, \qquad o_j^{(k)} = f_j(net_j^{(k)}). \tag{21.17}$$

Dabei ist $w_{ij}^{(k)}$ das Gewicht der i-ten Eingangsaktivierung ($i = 1, ..., n_{k-1}$) des Neurons j, $j = 1, ..., n_k$ in der Schicht $k = 1, ...l$. Hier wurde berücksichtigt, dass die einzelnen Neuronen unterschiedliche Aktivierungsfunktionen haben können. Die Ausgabe einer Schicht k sind die Eingaben der nächsten Schicht $k+1$ ($k = 1, .., l-1$):

$$a_j^{(k+1)} = o_j^{(k)}, j = 1, ..., n_k. \tag{21.18}$$

Die Ausgabewerte des gesamten Netzes sind die Ausgaben der letzten Schicht $k = l$:

$$out_j = o_j^{(l)}, j = 1, ..., n_l = n. \tag{21.19}$$

Nun der Rückwärtsschritt: Zunächst wird in der Ausgabeschicht mit Hilfe der Sollausgabe ein Korrekturwert $\delta_j^{(l)}$ für jedes Neuron berechnet. Dieses Maß wird nach der *Gradientenabstiegsmethode* ermittelt. Hier gehen die speziellen Eigenschaften der Ableitung der Sigmoidfunktion mit ein:

$$\delta_j^{(l)} = out_j(1 - out_j)(out_{soll,j} - out_j). \tag{21.20}$$

Für die verdeckten Schichten $k = l-1, ..., 1$ wird das Korrekturmaß wie folgt berechnet:

$$\delta_j^{(k)} = o_j^{(k)}(1 - o_j^{(k)}) \sum_{i=1}^{n_{k+1}} w_{ji}^{(k+1)} \delta_i^{(k+1)}. \tag{21.21}$$

Nun wird die Korrektur der Gewichte $w_{ij}^{(k)}$ vom Iterationsschritt t zum Iterationsschritt $t+1$ durchgeführt:

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + \eta \delta_j^{(k)} a_i^{(k)}, j = 1, ..., n_k, i = 1..., n_{k-1}. \tag{21.22}$$

Der Parameter η ist, wie schon oben, die Lernrate. Die Iteration wird beendet, wenn die Streuung der Abweichungen der Ausgabevektoren von den Ausgabesollvektoren kleiner als eine vorgegebene Schranke ist:

$$\frac{1}{2} \sum_{j=1}^{n_l} (out_{soll,j} - out_j)^2 < \epsilon. \tag{21.23}$$

Mit diesen Ausführungen soll die Darstellung der neuronalen Netze beendet werden. Es gibt noch viele weitere Netztypen, z.B. solche, die die Eingangsvektoren ohne Sollausgabe zu Punktwolken zusammenfassen und somit als unüberwachte Klassifikatoren verwendet werden können. Andere Netze arbeiten wie ein Signalverstärker: Ein Eingabevektor wird gelernt. Die Ausgabe ist derselbe Eingabevektor. Wenn nun der Eingabevektor leicht „verrauscht“ ist, wird er durch diesen Netztyp wieder rekonstruiert (z.B. [Koho88]).

21.2 Neuronale Netze als Klassifikatoren

Nach der obigen Darstellung ist leicht zu sehen, wie neuronale Netze in der digitalen Bildverarbeitung und Mustererkennung eingesetzt werden können, nämlich überall dort, wo Merkmalsvektoren klassifiziert werden. Dabei kann es sich um Merkmalsvektoren handeln, wie sie in den vorhergehenden Abschnitten verwendet wurden, es können aber auch solche sein, die z.B. die Form eines Segments oder Objekts beschreiben. Es ist auch denkbar, die Bildpunkte ganzer Bilder oder von Bildausschnitten als Eingabevektoren eines neuronalen Netzes zu verwenden. Beispiele hierzu sind Bildausschnitte, die Objekte wie Buchstaben oder Symbole enthalten. Auch Bildausschnitte, die primitive Objekte enthalten (Grundbausteine für komplexere Formen, wie z.B. Linienstücke, Ecken, Verzweigungen, usw.), können als Eingabevektoren dienen.

In den folgenden Abschnitten werden einige einfache Beispiele dargestellt. Sie zeigen auch die Problematik bei der Verwendung neuronaler Netze auf. Ein Beispiel zur rotations- und größeninvarianten Segmenterkennung mit neuronalen Netzen wird in Kapitel 26 gegeben.

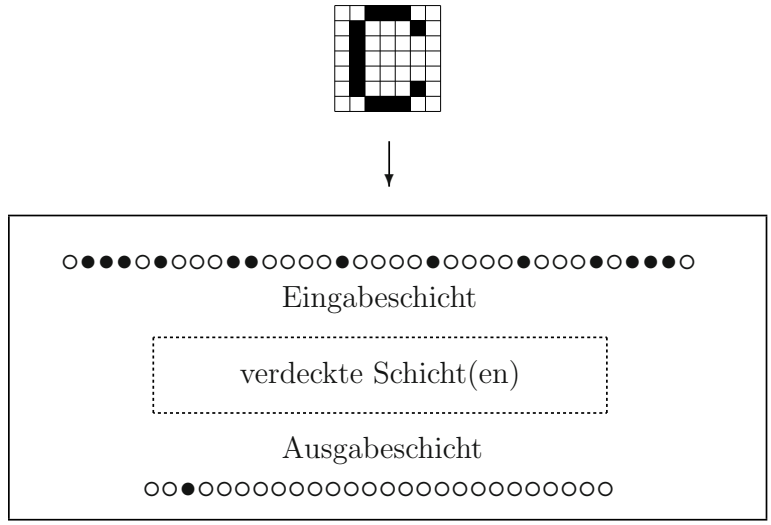


Bild 21.8: Klassifikation von Bildern (Bildausschnitten), die $7 \cdot 5$ Bildpunkte große Großbuchstaben enthalten. Die Bildpunkte werden hier direkt als Eingabe für das neuronale Netz verwendet. Da die linke und die rechte Randspalte bei allen Buchstaben nicht besetzt ist, werden diese Bildpunkte in der Eingabeschicht nicht verwendet. Die 26 Ausgabeneuronen geben die Positionen der Buchstaben im Alphabet an.

21.2.1 Verarbeitung von Binärbildern

Bei der Anwendung in diesem Abschnitt werden binäre Bilddaten vorausgesetzt. Es könnte sich dabei z.B. um Bilder (Bildausschnitte) handeln, die Großbuchstaben enthalten. Bild 21.8 zeigt den Buchstaben „C“ mit einer Buchstabengröße von $7 \cdot 5$ Bildpunkten. Die linken und rechten Randspalten sind bei allen Buchstaben weiß. Sie werden deshalb nicht in der Eingabeschicht mitverwendet.

Die 35 Bildpunkte wurden in verschiedenen Testläufen ohne weitere Verarbeitung auf die 35 Eingangsneuronen eines Backpropagation-Netzes gegeben. Die Ausgabeschicht bestand aus 26 Neuronen. Bei einem bestimmten Großbuchstaben als Eingabe wurde als Sollausgabe die Position des Buchstaben im Alphabet (10...0 $\hat{=}$ A, 010...0 $\hat{=}$ B, ...) trainiert.

In der folgenden Tabelle ist zusammengefasst, wie die verdeckten Schichten bei unterschiedlichen Tests gewählt wurden:

verd.Schichten	Neuronen	Training	Systemfehler	0 Fehler	1 Fehler	2 Fehler
1	15	267	0.025	0	2	8
1	15	811	0.01	0	2	8
1	20	474	0.025	0	0	1
1	20	845	0.01	0	0	1
1	25	673	0.01	0	0	5
2	15 15	283	0.01	0	3	5
3	11 13 11	882	0.01	0	10	15

Der Test in der letzten Zeile der Tabelle wurde mit einer anderen Anzahl von Ausgabeneuronen durchgeführt. Auf diesen Test wird weiter unten näher eingegangen. Alle anderen Tests wurden, wie oben beschrieben, mit 35 Eingabe- und 26 Ausgabeneuronen durchgeführt.

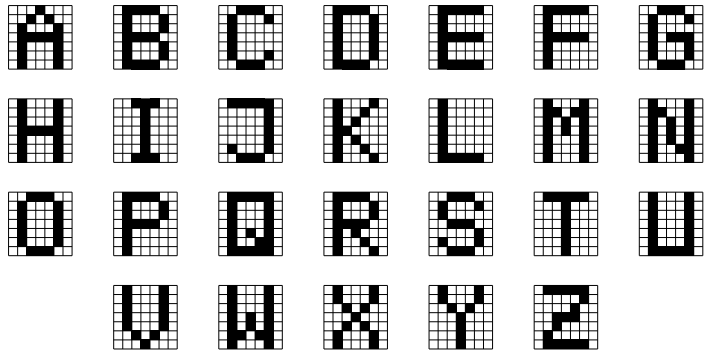
In der Spalte „Neuronen“ ist die jeweilige Anzahl der Neuronen der verdeckten Schicht(en) bei den verschiedenen Tests angegeben.

Die Spalte „Training“ gibt an, wie viele Trainingsdurchläufe mit der Trainingsmenge durchgeführt wurden, um ein zufriedenstellendes Ergebnis zu erreichen. Die Trainingsdaten bestanden aus den Buchstaben „A“ bis „Z“ und den entsprechenden Sollausgaben „10...0“ bis „0...01“. In der nächsten Spalte „Systemfehler“ ist der erzielte Systemfehler angegeben. Darunter versteht man die Streuung der Abweichungen der Soll-Werte von den trainierten Ist-Werten.

Die letzten drei Spalten der Tabelle enthalten die Ergebnisse von Erkennungsläufen. In der Spalte „0 Fehler“ ist das Ergebnis der Klassifizierung der Buchstaben „A“ bis „Z“ zusammengefasst, wobei die Buchstaben exakt so in das Netz eingegeben wurden, wie sie trainiert wurden (siehe Bild 21.9 oben). In der Ausgabeschicht wurde das Neuron mit dem maximalen Wert als „gesetzt“ und alle anderen als „nicht gesetzt“ interpretiert. In allen Fällen konnten die trainierten Buchstaben richtig erkannt werden. Das ist aber nicht weiter verwunderlich, da man ja in der Regel so lange trainiert, bis alle Eingabevektoren richtig erkannt werden. Wenn das nicht gelingt, so ist das ein Hinweis, dass die Struktur des Netzes ungeeignet ist.

In den Spalten „1 Fehler“ und „2 Fehler“ wurden Buchstaben verwendet, die jeweils einen (Bild 21.10 oben) bzw. zwei zufällig gestörte Pixel enthielten. Die Spalten geben an, wieviele Buchstaben falsch erkannt wurden. Man sieht hier deutliche Unterschiede zwischen den einzelnen Testläufen.

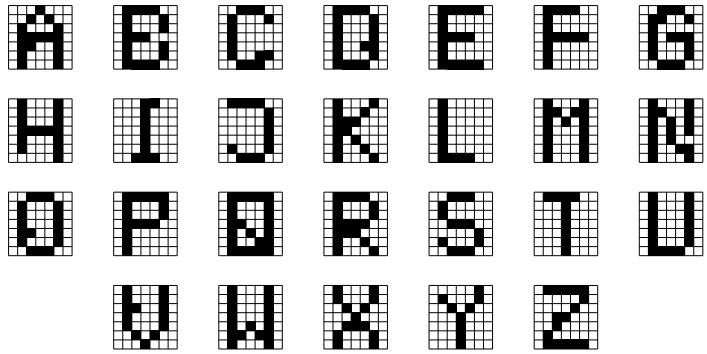
Die Ergebnisse der letzten Zeile der Tabelle beschreiben einen Test mit einer anderen Anzahl von Ausgabeneuronen und einem anderen Training. Die Eingabeschicht bestand wie vorher aus 35 Neuronen. Es wurde ein Netz mit drei verdeckten Schichten mit jeweils 11, 13 und 11 Neuronen verwendet. Als Ausgabe wurden nur fünf Neuronen verwendet, die die Position des jeweiligen Buchstaben im Alphabet als Dualzahl angeben (00001 $\hat{=}$ A, 00010 $\hat{=}$ B, 00011 $\hat{=}$ C,...). Die Ausgabe ist hier codiert, im Gegensatz zu den anderen Tests, wo für die jeweilige Position des Buchstaben im Alphabet ein Bit gesetzt wurde. Damit wird dem Netz mehr zugemutet, da es die Ausgaberepräsentation codieren muss.



Werte der Ausgabeneuronen:

```
0.020837  0.028464  0.017265  0.021185  0.982142 -> 00001 ->  1  A
0.020670  0.020221  0.014934  0.989956  0.019147 -> 00010 ->  2  B
0.027935  0.023546  0.024885  0.986592  0.999935 -> 00011 ->  3  C
0.000572  0.014370  0.999676  0.018239  0.018549 -> 00100 ->  4  D
0.002510  0.015419  0.999610  0.020248  0.981162 -> 00101 ->  5  E
0.022271  0.003551  0.998852  0.974118  0.005454 -> 00110 ->  6  F
0.001091  0.008356  0.990704  0.989253  0.999025 -> 00111 ->  7  G
0.018397  0.999732  0.023999  0.021194  0.019648 -> 01000 ->  8  H
0.015209  0.995435  0.015903  0.011473  0.997451 -> 01001 ->  9  I
0.017719  0.995189  0.000752  0.976232  0.001950 -> 01010 -> 10  J
0.015654  0.988395  0.004384  0.981284  0.999403 -> 01011 -> 11  K
0.000068  0.982192  0.999956  0.018638  0.018289 -> 01100 -> 12  L
0.005537  0.985710  0.998434  0.016890  0.984309 -> 01101 -> 13  M
0.015791  0.982911  0.999132  0.978255  0.000060 -> 01110 -> 14  N
0.000073  0.972075  0.979134  0.979410  0.984619 -> 01111 -> 15  O
0.991416  0.024989  0.024072  0.032943  0.000741 -> 10000 -> 16  P
0.986835  0.017315  0.013341  0.013476  0.991117 -> 10001 -> 17  Q
0.993490  0.013013  0.000066  0.976666  0.027819 -> 10010 -> 18  R
0.984228  0.013388  0.000033  0.999302  0.986272 -> 10011 -> 19  S
0.976467  0.018553  0.968682  0.026200  0.023099 -> 10100 -> 20  T
0.975068  0.003692  0.983546  0.022007  0.977489 -> 10101 -> 21  U
0.976925  0.000857  0.982606  0.972932  0.014265 -> 10110 -> 22  V
0.987626  0.000696  0.977273  0.978641  0.989696 -> 10111 -> 23  W
0.997931  0.990342  0.009320  0.001847  0.009105 -> 11000 -> 24  X
0.975234  0.976796  0.000583  0.016451  0.971005 -> 11001 -> 25  Y
0.969912  0.973222  0.000023  0.992793  0.014657 -> 11010 -> 26  Z
```

Bild 21.9: Ergebnisse der Großbuchstabenerkennung. Unter den Großbuchstaben sind die Werte der fünf Ausgabeneuronen aufgelistet. Mit einem Schwellwert von 0.5 können sie als Dualzahlen interpretiert werden, die die Nummer des Buchstaben im Alphabet angeben. Es ist zu sehen, dass hier ohne Schwierigkeiten die 26 Großbuchstaben trainiert und wiedererkannt werden konnten.



Werte der Ausgabeneuronen:

```
0.022017  0.026538  0.011898  0.039725  0.976535 -> 00001 ->  1  A
0.001897  0.061533  0.974516  0.902196  0.006236 -> 00110 ->  6  B -> F
0.045946  0.024845  0.011010  0.979124  0.999941 -> 00011 ->  3  C
0.003161  0.016500  0.999421  0.094493  0.004494 -> 00100 ->  4  D
0.654648  0.000696  0.755598  0.865997  0.992804 -> 10111 -> 23  E -> W
0.000370  0.013200  0.999785  0.369456  0.027534 -> 00100 ->  4  F -> D
0.000028  0.780247  0.983394  0.997842  0.973749 -> 01111 -> 15  G -> O
0.010467  0.999752  0.013863  0.049535  0.030799 -> 01000 ->  8  H
0.030367  0.996544  0.009684  0.005725  0.998188 -> 01001 ->  9  I
0.000111  0.985478  0.495491  0.615509  0.004989 -> 01010 -> 10  J
0.038400  0.996255  0.001789  0.832148  0.998877 -> 01011 -> 11  K
0.000060  0.981270  0.999959  0.069995  0.020088 -> 01100 -> 12  L
0.005423  0.986693  0.977855  0.034662  0.994876 -> 01101 -> 13  M
0.106422  0.871062  0.998939  0.986739  0.000044 -> 01110 -> 14  N
0.000059  0.880481  0.999446  0.959640  0.934473 -> 01111 -> 15  O
0.997912  0.017865  0.003772  0.075668  0.004611 -> 10000 -> 16  P
0.999167  0.010610  0.153588  0.023471  0.099921 -> 10000 -> 16  Q -> P
0.995054  0.018077  0.001841  0.336459  0.003432 -> 10000 -> 16  R -> P
0.004964  0.019297  0.001310  0.973724  0.990766 -> 00011 ->  3  S -> C
0.644564  0.043035  0.988861  0.030728  0.003821 -> 10100 -> 20  T
0.976208  0.008697  0.987211  0.162659  0.970364 -> 10101 -> 21  U
0.997931  0.007157  0.968808  0.136341  0.310576 -> 10100 -> 20  V -> T
0.996570  0.001039  0.541286  0.994480  0.986531 -> 10111 -> 23  W
0.979371  0.998029  0.001152  0.019924  0.651716 -> 11001 -> 25  X -> Y
0.681093  0.990846  0.000309  0.290778  0.994714 -> 11001 -> 25  Y
0.012139  0.999637  0.000950  0.523317  0.044018 -> 01010 -> 10  Z -> J
```

Bild 21.10: Ergebnisse der Großbuchstabenerkennung mit Buchstaben, die jeweils durch ein falsches Pixel gestört sind. Es ergeben sich beträchtliche Fehlklassifizierungen.

Es wird folglich eine größere Netzkomplexität notwendig.

Nach 882 Trainingszyklen wurde der Systemfehler auf einen Wert von 0.01 reduziert. Bild 21.9 zeigt die Ergebnisse zu dieser Problemstellung. Unter den Großbuchstaben sind die Werte der fünf Ausgabeneuronen aufgelistet. Mit einem Schwellwert von 0.5 können sie als Dualzahlen interpretiert werden, die die Nummer des Buchstaben im Alphabet angeben. Es ist zu sehen, dass hier ohne Schwierigkeiten die 26 Großbuchstaben trainiert und wiedererkannt werden konnten.

In einem zweiten Erkennungsdurchlauf wurden alle Buchstaben durch ein zufälliges fehlerhaftes Pixel gestört. Das Ergebnis dazu zeigt Bild 21.10. Hier treten erhebliche Fehlklassifizierungen auf. Für den menschlichen Betrachter ist z.B. das gestörte „E“ ohne weiteres zu erkennen. Das neuronale Netz klassifiziert es aber als den Buchstaben „W“. Im dritten Lauf wurden die Buchstaben durch zwei zufällige Pixel gestört, was zur Folge hat, dass mehr als die Hälfte aller Buchstaben falsch erkannt wurde (siehe Bild 21.10 unten).

Damit werden Fragen deutlich, denen man bei der Verwendung neuronaler Netze begegnet:

- Wieviele Ein- und Ausgabeneuronen sind zu verwenden?
- Wieviele verdeckte Schichten mit wie vielen Neuronen sollte man wählen?
- Welche Netztopologie soll überhaupt gewählt werden?
- Wie sollen Parameter wie z.B. die Lernrate η oder die Steigung der Sigmoidfunktion gewählt werden?
- Wie lange soll trainiert werden?
- Auf welchen Wert soll der Systemfehler reduziert werden?

Diese Fragen lassen sich nicht generell beantworten. Bei vielen praktischen Anwendungen ist man auf reines Probieren (oder „eine gute Nase“) angewiesen. Trotzdem seien einige Hinweise gegeben, die man beachten sollte:

- Die Zahl der Eingabeneuronen ist in den Anwendungen des vorliegenden Bereichs oft durch die jeweilige Problemstellung vorgegeben. Durch eine geeignete Vorverarbeitung der Eingabedaten kann das neuronale Netz von der Notwendigkeit des Aufbaus interner Repräsentationen befreit werden. Eine Codierung der Eingabedaten kann die Anzahl der Eingabeneuronen reduzieren. Sie kann sich aber auch nachteilig auswirken, wenn das Netz die Repräsentation decodieren muss. Die Entfernung von Redundanzen in den Eingabedaten, z.B. mit der Hauptkomponententransformation, kann ebenfalls die Anzahl der notwendigen Eingabeneuronen reduzieren.
- Die Ausgabe sollte man so einfach wie möglich gestalten, was aber nicht immer bedeuten muss, dass das Netz mit weniger Ausgabeneuronen die besseren Ergebnisse liefert. Die Beispiele der obigen Tests bestätigen dies: Das eindeutig bessere Ergebnis

liefern die Tests, bei denen die Position des Buchstaben im Alphabet als Bitposition innerhalb der 26 Ausgabeneuronen codiert wurde. Bei der Codierung als Dualzahl benötigt man zwar weniger Ausgabeneuronen, aber dem Netz wird mehr Codierungsaufwand zugemutet. Es ist in den einzelnen Anwendungsfällen zu prüfen, ob es nicht besser ist, eine weitgehend uncodierte Ausgabe zu erzeugen und die gewünschten Ergebnisse in der Nachverarbeitung zu ermitteln.

- Je größer die Anzahl der zu lernenden Musterzuordnungen und je komplexer die Ein-/Ausgabebeziehungen sind, desto mehr verdeckte Neuronen werden notwendig. Die Anzahl der verdeckten Schichten und die Anzahl der Neuronen pro Schicht sollte aber trotzdem, problemangepasst, so gering wie möglich sein.

  Wird ein Netz zu groß dimensioniert, so schwindet der *Generalisierungseffekt*, d.h. die Fähigkeit, die Beziehungen zwischen nicht gelernten Daten korrekt wiederzugeben.

  Im obigen Beispiel liegt das Optimum anscheinend in der Nähe von 20 Neuronen bei einer verdeckten Schicht. Werden mehr oder weniger Neuronen verwendet, so schwindet der Generalisierungseffekt, d.h. es werden bei gestörten Eingabemustern mehr Fehler gemacht.

- Bei komplexen Beziehungen kann die Topologie des Netzes aufwändiger gestaltet werden als bei einem Standard-Backpropagationnetz. Hier sind Strukturen denkbar, bei denen z.B. die Ausgabeschicht zusätzlich direkt mit der Eingabeschicht verbunden ist oder bei denen zwei Systeme von verdeckten Schichten nebeneinander verwendet werden. Dadurch werden aber mehr Neuronen und Verbindungen notwendig, was zu einer schlechteren Generalisierung führen kann.

- Die Lernrate und die Steigung der Sigmoidfunktion beeinflussen die Konvergenz des Netzes in der Trainingsphase. Ein hoher Wert η kann zum Alternieren des Systemfehlers beitragen. Ein zu kleiner Wert kann eine sehr langsame Konvergenz zur Folge haben. In den vorliegenden Beispielen wurde $\eta = 0.7$ gewählt.

  Die Sigmoidfunktion ist für die Nicht-Linearität des Netzes verantwortlich. Wird die Steigung reduziert, so kann im Grenzfall ein nahezu lineares System erreicht werden. Das andere Extremum ist eine Schwellwertfunktion.

  Nichtlineare Systeme werden auch in der *Chaostheorie* untersucht. Möglicherweise sind Erkenntnisse aus der Chaostheorie auf neuronale Netze anwendbar, z.B. dass in einem nichtlinearen System geringfügige Änderungen an den Eingabedaten kolossale Änderungen der Ausgabedaten bewirken können. Um ein gutes Generalisierungverhalten zu erzielen, müsste man folglich möglichst „wenig nichtlineare“ Netze verwenden. Dann ist aber das Konvergenzverhalten in der Regel schlechter.

- Wenn zu lange trainiert wird, sinkt das Generalisierungsverhalten. Bei verrauschten Eingabedaten kann es dann z.B. passieren, dass das Rauschen mit trainiert wird. Um ein generalisierendes Netz zu erhalten, trainiert man häufig gerade so lange, bis alle Eingabedaten richtig erkannt werden.

- Der erzielte Systemfehler kann als Maß für die Güte verwendet werden. Trainiert man aber solange, bis der Systemfehler sehr klein wird, so geht das wieder auf Kosten des Generalisierungsverhaltens. In den obigen Beispielen hat sich gezeigt, dass etwa ab einem Systemfehler, der kleiner ist als 0.05, alle unverfälschten Eingabemuster richtig klassifiziert wurden.

21.2.2 Verarbeitung von mehrkanaligen Bildern

In diesem Abschnitt werden die Bildpunkte $\mathbf{s}(x, y)$ eines mehrkanaligen Bildes $\mathbf{S}$ direkt als Eingabevektoren eines neuronalen Netzes verwendet. Die einzelnen Kanäle des Bildes können dabei das Ergebnis einer längeren Folge von Verarbeitungsschritten zur Merkmalsgewinnung (Kapitel 11) sein.

Als Beispiel wird ein dreikanaliges RGB-Farbbild $\mathbf{S}_e = (s_e(x, y, n)), n = 0, 1, 2$ gewählt. Das Netz wird somit zur Farberkennung trainiert. Die Bildpunkte $\mathbf{s}_e(x, y) = (r, g, b)^T$ sind dreidimensionale Vektoren, die den Rot-, Grün- und Blauauszug repräsentieren. In der Trainingsphase werden ausgewählte Farb- (Merkmals-)Vektoren angeboten und dazu angegeben, um welchen Farbton es sich handelt. Die Eingabeschicht besteht hier somit aus drei Neuronen. Die Ausgabeschicht hängt von der Anzahl der zu lernenden Farbtöne ab: Für jede Klasse (Farbe) wird ein Ausgabeneuron verwendet.

In der Trainingsphase zu einem ersten Beispiel wurden die Farbvektoren zu den Farben Rot, Gelb, Grün, Zyan, Blau, Magenta, Weiß und Schwarz trainiert. In folgender Tabelle sind die Eingabevektoren (Spalte „Tr.Vekt."), die Sollausgabe für das Training (Spalte „Soll"), sowie die Eingabedaten für einen Bewertungstest (Spalte „Kl.Vekt.") zusammengefasst. Ein Vergleich mit der grafischen Darstellung des RGB-Farbraumes in Abschnitt 3.2.4 (Bild 3.5) zeigt, dass hier die acht Eckpunkte des Farbwürfels trainiert wurden.

Farbe	Tr.Vekt	Soll	Kl.Vekt.
Rot	1 0 0	1 0 0 0 0 0 0 0	0.80 0.20 0.10 und 0.80 0.20 0.20
Gelb	1 1 0	0 1 0 0 0 0 0 0	0.90 0.85 0.10 und 0.85 0.85 0.20
Grün	0 1 0	0 0 1 0 0 0 0 0	0.20 0.85 0.10 und 0.15 0.85 0.25
Zyan	0 1 1	0 0 0 1 0 0 0 0	0.15 0.85 0.75 und 0.25 0.90 0.75
Blau	0 0 1	0 0 0 0 1 0 0 0	0.10 0.20 0.95 und 0.20 0.35 0.90
Magenta	1 0 1	0 0 0 0 0 1 0 0	0.85 0.15 0.90 und 0.80 0.10 0.85
Schwarz	0 0 0	0 0 0 0 0 0 1 0	0.15 0.05 0.25 und 0.20 0.15 0.20
Weiß	1 1 1	0 0 0 0 0 0 0 1	0.75 0.85 0.85 und 0.80 0.80 0.85

In der nächsten Tabelle sind die Ergebnisse unterschiedlicher Testläufe zusammengestellt. Die Anzahl der Eingabeneuronen war immer drei und die Anzahl der Ausgabeneuronen immer acht. Spalten, die mit „-" gekennzeichnet sind, beziehen sich auf Tests, bei denen der Systemfehler nicht gegen den geforderten Wert von 0.05 konvergierte, sondern in einem lokalen Minimum hängenblieb. Bei den Ausgabeneuronen wurde das Neuron mit dem maximalen Wert als „gesetzt" interpretiert.

verd.Schichten	Neuronen	Training	Systemfehler	Tr.Vekt.	Kl.Vekt.
1	3	134	0.05	0	1
1	3	1790	0.01	0	1
1	5	79	0.05	0	0
1	5	990	0.01	0	0
1	7	-	0.05	-	-
1	11	-	0.05	-	-
2	3 3	184	0.05	0	0
2	3 3	1800	0.01	0	0
2	5 5	104	0.05	0	0
2	5 5	644	0.01	0	0
2	7 7	97	0.05	0	0
2	7 7	706	0.01	0	0
2	11 11	-	0.05	-	-
3	3 5 3	-	0.05	-	-
3	5 7 5	157	0.05	0	0
3	5 7 5	914	0.01	0	0
3	10 20 10	198	0.05	0	0
3	10 20 10	423	0.01	0	0

Um das Generalisierungsverhalten etwas genauer zu untersuchen, wurden in einem zweiten Beispiel aus dem RGB-Farbraum äquidistante Punkte ausgewählt: In der RG-Ebene (Blauanteil = 0.0) die $11 \cdot 11$ Punkte mit einem Rotanteil von 0.0, 0.1 bis 1.0 und und einem Grünanteil von 0.0, 0.1 bis 1.0, in der darüberliegenden Ebene (Blauanteil 0.1) sinngemäß wieder $11 \cdot 11$ Punkte, bis zur oberen Ebene mit einem Blauanteil von 1.0. Diese $11 \cdot 11 \cdot 11$ Punkte wurden klassifiziert. In den resultierenden Farbebenen wurden die Ergebnisvektoren in den trainierten Grundfarben (Rot, Gelb, ..., Weiß) dargestellt. Man sieht so, wie durch die verschiedenen Tests mit unterschiedlichen Netzen der Farbraum aufgeteilt wurde.

Die Ergebnisse sind in den Bildern 21.11, 21.12 und 21.13 zusammengestellt. Die Teilbilder zeigen von links oben nach rechts unten die einzelnen Ebenen des diskretisierten Farbraums. Links oben ist die unterste Ebene mit dem Blauanteil 0.0 (RG-Ebene). Daneben ist die Ebene mit einem Blauanteil von 0.1, usw. Die Ebene im Teilbild rechts unten hat einen Blauanteil von 1.0. Die weiteren Erläuterungen zu den Ergebnissen stehen in den Bildunterschriften.

Im letzten Beispiel zur Klassifizierung mit neuronalen Netzen wurden aus einem gescannten RGB-Farbbild mit 24 Bit pro Bildpunkt, also 8 Bit pro Farbauszug (Bild 21.14-a), zu verschiedenen Farben Trainingsgebiete ausgewählt und einem neuronalen Netz trainiert. Das Merkmal für die Segmentierung der (Farb-)Klassen ist somit der Farbton, d.h. ähnliche Farben werden in einer Klasse zusammengefasst. Es wurden fünf Trainingsgebiete definiert. Die Bildausschnitte wurden aus folgenden Bereichen gewählt: Weiß: Ei, Gelb: Zitrone, Hellblau: Schmetterling, Rot: Apfel, Grün: Hintergrund der Margerite. In der Tabelle sind die dazugehörigen Koordinaten angegeben:

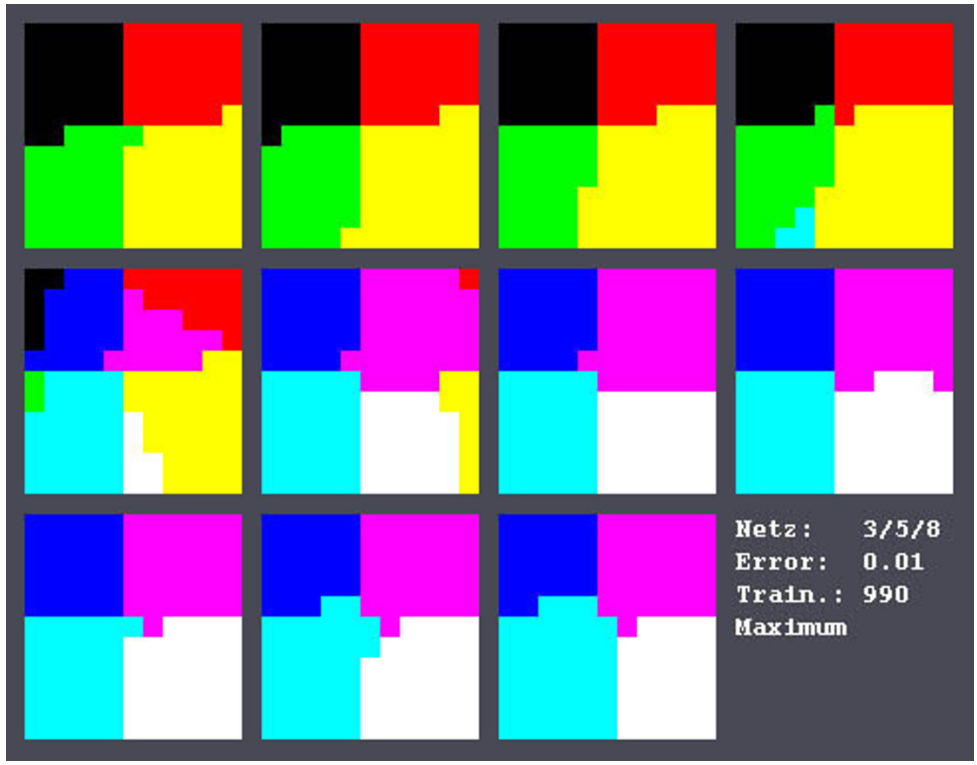


Bild 21.11: Klassifizierung des diskretisierten Farbraums. Hier wurde ein neuronales Netz mit drei Eingabeneuronen, einer verdeckten Schicht mit fünf Neuronen und einer Ausgabeschicht mit acht Neuronen verwendet. Man sieht deutlich, dass in der untersten Ebene die Farben Schwarz, Rot, Gelb und Grün fast symmetrisch verteilt sind. In der vierten Ebene tritt Zyan und in der fünften Ebene treten Blau, Magenta und Weiß zum ersten Mal auf. In den weiteren Ebenen mit zunehmendem Blauanteil ist zu sehen, wie sich Blau, Magenta, Zyan und Weiß durchsetzen.

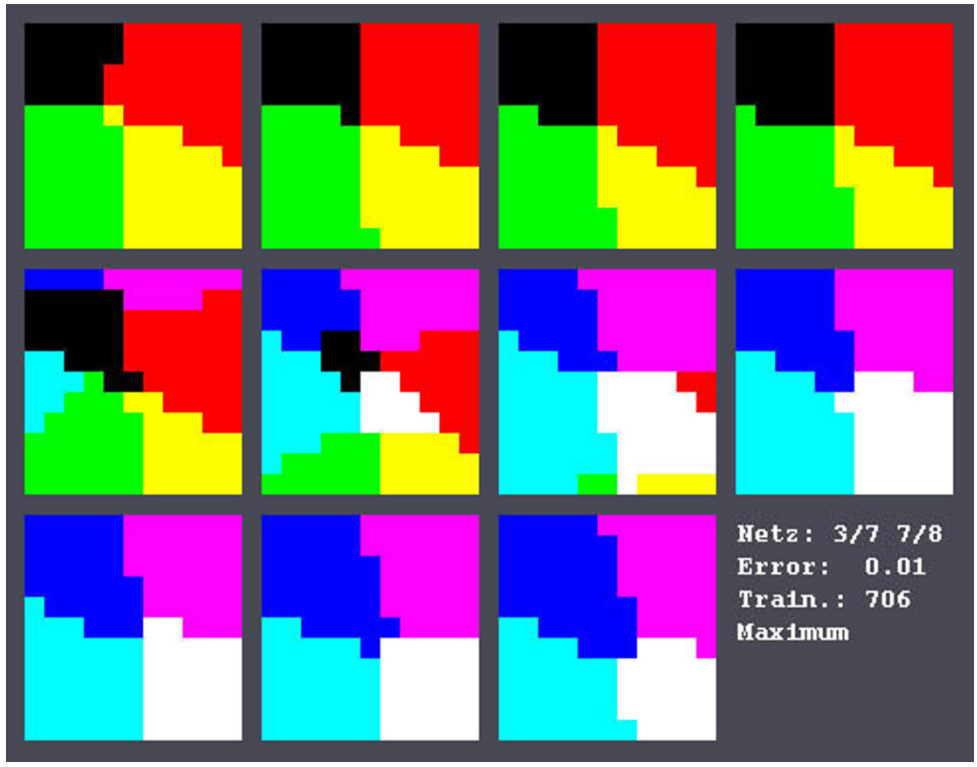


Bild 21.12: Klassifizierung des diskretisierten Farbraums. Hier wurde ein neuronales Netz mit drei Eingabeneuronen, zwei verdeckten Schichten mit jeweils sieben Neuronen und einer Ausgabeschicht mit acht Neuronen verwendet. In der untersten und obersten Ebene ist die Aufteilung auf die Farben Schwarz/Rot/Gelb/Grün und Blau/Magenta/Weiß/Zyan nicht mehr so symmetrisch wie im vorhergehenden Bild. Auch die Übergänge in den dazwischen liegenden Ebenen sind komplexer geworden.

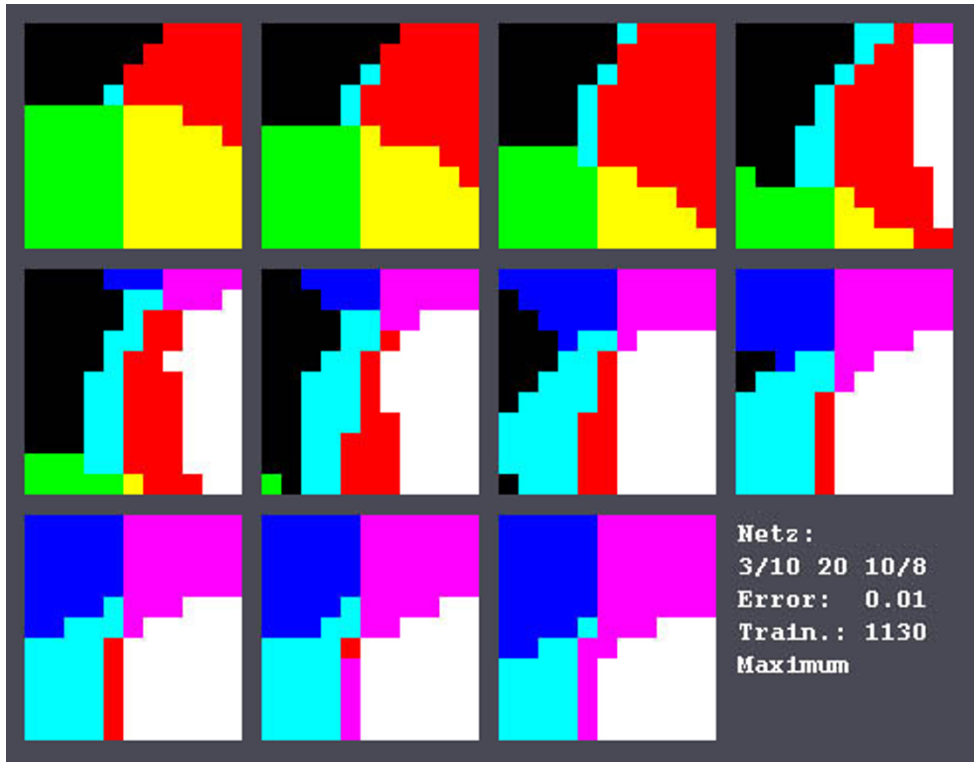


Bild 21.13: Klassifizierung des diskretisierten Farbraums. Hier wurde ein neuronales Netz mit drei Eingabeneuronen, drei verdeckten Schichten mit 10/20/10 Neuronen und einer Ausgabeschicht mit acht Neuronen verwendet. Hier zieht sich z.B. die Farbe Zyan bis zur untersten Ebene durch. Die Farbe Rot tendiert ab der vierten Ebene zum mittleren Bereich des Farbraums. Auch die Übergänge zwischen den anderen Farben entsprechen nicht der intuitiven Vorstellung, wie die Aufteilung sein sollte. Das liegt daran, dass ein Netz mit zu vielen Neuronen verwendet wurde.

Klasse	Anfangszeile/spalte	Endzeile/spalte	Anzahl der Vektoren
Weiss	254/387	301/419	1584
Gelb	89/244	127/279	1404
Hellblau	122/370	142/392	483
Rot	258/230	295/278	1862
Grün	293/ 48	328/ 65	648

Nach Maßgabe der dreidimensionalen Eingabevektoren und der fünf Ausgabeklassen wurde ein Netz mit drei Neuronen in der Eingabeschicht und fünf Neuronen in der Ausgabeschicht verwendet. Das Netz hatte eine verdeckte Schicht mit sieben Neuronen.

Zum Training des Netzes wurden aus den Trainingsgebieten jeweils 20 Farbvektoren pro Klasse, also insgesamt 100 Farbvektoren ausgewählt. Es wurde so lange trainiert, bis alle 100 Eingabevektoren richtig erkannt wurden.

Zur apriori-Beurteilung der Güte der Klassifizierung wurden die Trainingsgebiete, also die jeweils 20 ausgewählten und die restlichen, nicht für das Training des Netzes verwendeten Farbvektoren klassifiziert. Das Ergebnis dieses Schritts ist in folgender Tabelle zusammengefasst:

Klasse	Weiss	Gelb	Hellblau	Rot	Grün
Weiss	100.0%	0.0%	0.0%	0.0%	0.0%
Gelb	0.0%	100.0%	0.0%	0.0%	0.0%
Hellblau	0.0%	0.0%	100.0%	0.0%	0.0%
Rot	0.0%	0.1%	0.0%	99.9%	0.0%
Grün	0.0%	0.0%	0.0%	0.0%	100.0%

Diese Tabelle besagt z.B., dass bei der Klassifizierung der Farbvektoren der Klasse „Rot“ 0.1% fälschlicherweise zur Klasse „Gelb“ und die restlichen 99.9% richtig zur Klasse „Rot“ klassifiziert wurden. Dieses Ergebnis zeigt einen guten Generalisierungseffekt, da ja die Farbvektoren eines Trainingsgebietes in den einzelnen Farbkanälen durchaus streuten.

Das Ergebnis zeigt Bild 21.14-b. Die Klassen wurden durch den mittleren Farbton ihrer Trainingsgebiete dargestellt. Da keine Zurückweisungsklasse eingeführt worden war, wurden alle Farbvektoren einer der fünf Farbklassen zugeordnet. Das bedeutet z.B., dass der dunkle Hintergrund des roten Apfels zur Klasse „Grün“ (Eichenblatt) klassifiziert wurde.

Abschließend sei nochmals darauf hingewiesen, dass weitere Anwendungen mit neuronalen Netzen in Kapitel 14 zu finden sind.

(a)

(b)

Bild 21.14: (a) RGB-Farbbild als Testbeispiel für die Farbklassifizierung mit einem neuronalen Netz. (b) Ergebnis der Klassifizierung des Farbbildes. Die Farbklassen sind durch die Mittelwerte der Trainingsgebiete codiert. Es wurde keine Zurückweisungsklasse definiert.

Kapitel 22

Segmentierung mit Fuzzy Logic

22.1 Anwendungen

In diesem Kapitel wird ein unüberwachter Klassifikator vorgestellt, der eine Menge von Merkmalsvektoren einer vordefinierten Anzahl von Klassen zuordnet. Die Lage der *cluster*-Zentren im Merkmalsraum wird, ausgehend von einer initialen Verteilung, in mehreren Iterationsschritten so verändert, dass ein Minimalisierungskriterium erfüllt wird. Als Ergebnis wird zu jedem Merkmalsvektor angegeben, wie groß seine Zugehörigkeit zu den verschiedenen Klassen ist. Im Rahmen eines Systems, das mit der Modellierung von vagem Wissen operiert, kann diese Zugehörigkeitsaussage mit weiteren auf andere Weise gefundenen Aussagen unscharf verknüpft werden.

22.2 Grundlagen: Fuzzy Logic

22.2.1 Einführende Beispiele

In diesem Abschnitt werden die wesentlichen Grundlagen der *fuzzy logic* kurz zusammengefasst. Für den mathematisch interessierten Leser sei auf weitere einführende und vertiefende Literatur, etwa [Zimm92], [Boeh93] oder [Kosk92] verwiesen.

Was heißt „*fuzzy*“? Ein Englischlexikon gibt die folgenden Bedeutungen an: fusselig, faserig, kraus, verschwommen, trüb. Im Sinne der *fuzzy logic* wäre das Wort „*fuzzy*“ eher in Richtung „ungenau“, „unscharf“ oder „unpräzise“ zu interpretieren. Aus diesem Grund lautet die deutsche Übersetzung auch „Unscharfe Logik“ oder „Unscharfe Mengenlehre“. Aber auch die Bezeichnungen „Fuzzy Logik“ oder *fuzzy logic* haben sich im deutschen Sprachgebrauch eingebürgert.

Im alltäglichen Leben werden häufig Begriffe verwendet, die sich nicht exakt abgrenzen lassen. Beispiele dazu sind:

- Das Wetter ist schön.
- Es ist warm.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_22

- Ein Auto kostet viel und hat einen hohen Benzinverbrauch.
- Wenn ein Steak zu lange gebraten wird, ist es zäh.

Das sind alles vage Formulierungen, bei denen man sich nicht auf einen konkreten „Wert“ festlegt. Außerdem kann sich je nach Standpunkt des Verwenders dieser Formulierungen eine unterschiedliche Interpretation verbergen: Der Benzinverbrauch eines Autos beispielsweise ist eine sehr subjektive Angelegenheit. Bei einem Mittelklassewagen würde man derzeit von einem angemessenen Verbrauch bei etwa 8 *Litern pro 100 km* sprechen. Für einen Wagen der gehobenen Klasse wäre das ein eher niederer Verbrauch. Dazu kommt noch eine weitere Unschärfe: In einigen Jahren könnte man bei einem Mittelklassewagen durchaus 5-6 *Liter pro 100 km* als angemessen ansehen. Man hat hier also zwei unscharfe Formulierungen, nämlich „angemessener Verbrauch“ und „derzeit“.

Diese sprachliche Ungenauigkeit ist jedoch kein Nachteil, sondern ermöglicht es uns, in Situationen, in denen nur unvollständige oder sogar teilweise widersprüchliche Informationen vorliegen, eine Entscheidung zu fällen.

Die *fuzzy logic* ist eine exakte mathematische Theorie, mit der versucht wird, nicht exakten Begriffen gerecht zu werden. Dabei stellt sie andere Logikkalküle nicht in Frage, sondern bildet eine Verallgemeinerung, aus der sich z.B. die klassische Aussagenlogik als Sonderfall ableiten lässt.

An dieser Stelle sei auch noch ein Hinweis auf die Wahrscheinlichkeitstheorie eingefügt: Das Ereignis A=„Würfeln einer Sechs“ tritt mit einer mathematischen Wahrscheinlichkeit $p(A) = \frac{1}{6}$ ein. Wenn die Voraussetzung richtig ist, dass es sich um einen symmetrischen, idealisierten Würfel handelt, bei dem alle sechs möglichen Seiten gleichwahrscheinlich oben liegen, ist die obige Aussage exakt richtig. In der Praxis ist das aber nicht erfüllbar, da es keinen idealen Würfel gibt. Man könnte nun in vielen Versuchen näherungsweise die Wahrscheinlichkeiten ermitteln. Ein Ergebnis könnte dabei sein, dass z.B. die Zahl sechs mit einer Wahrscheinlichkeit von $\frac{13}{60}$ erscheint. Aber dies ist wiederum nur eine Näherung. Eine unscharfe Aussage wäre, dass bei einem Würfel die Zahl sechs „ziemlich sicher“ mit der Wahrscheinlichkeit $\frac{13}{60}$ oben liegt. Hier ist „ziemlich sicher“ eine unscharfe Aussage, die mit bestimmten Werten zu belegen ist, die angeben, was unter „ziemlich sicher“ zu verstehen ist.

Unscharfen Begriffen der Art „ziemlich sicher“, „eher klein“, „nahe bei eins“, „normal“, „hohes Fieber“, usw. müssen somit Bereiche der jeweiligen Skala zugeordnet werden, und es muss angegeben werden, wie stark die Zugehörigkeit der möglichen Werte zu dem unscharfen Begriff ist. Diese Zugehörigkeit nimmt Werte zwischen 0 und 1 an, wobei festgelegt wird, dass 0 als „sicher nicht dazu gehörig“ und 1 als „sicher dazu gehörig“ verstanden wird.

Dazu ein Beispiel: Der „angemessene Bezinverbrauch“ für Mittelklassewagen könnte etwa so beschrieben werden:

Literverbrauch pro 100 km	<6.5	6.5	7.0	7.5	8.0	8.5	9.0	9.5	>9.5
Zugehörigkeit	0.0	0.2	0.5	0.9	1.0	0.9	0.5	0.2	0.0

Die Tabelle ist so zu lesen, dass z.B. ein konkreter Verbrauch von 7.5 *Litern pro 100 km* mit einer Zugehörigkeit von 0.9 als angemessen für Mittelklassewagen anzusehen ist, während ein Verbrauch, der größer als 9.5 *Liter pro 100 km* ist, sicher nicht als angemessen bezeichnet werden kann. Natürlich müssen die Werte nicht immer, wie in diesem Beispiel, diskret sein. Sie können auch kontinuierlich und die Zugehörigkeiten funktional angegeben werden.

Nun noch ein paar Worte zur Geschichte der *fuzzy logic*. Die ersten Veröffentlichungen erschienen 1965 von Lofti Asker Zadeh. Er gehörte dem Department of Electrical Engineering and Electronics Research Laboratory der Berkeley University an. Die Weiterentwicklung erfolgte auch in Europa und Japan. Heute ist die *fuzzy logic* ein Teilbereich der Mathematik mit einer großen Zahl von Publikationen. Praktische Anwendungen gibt es in der Regelungstechnik, bei Expertensystemen, in der digitalen Bildverarbeitung und Mustererkennung und, wenn der Reklame zu trauen ist, bei Waschmaschinen und Videokameras.

Nach dieser Einführung wird im folgenden die *fuzzy logic* etwas genauer definiert. Die Darstellung und Notation orientiert sich an [Boeh93].

22.2.2 Definitionen und Erläuterungen

Es sei G eine Grundmenge und μ_A eine Funktion von G in das Intervall $[0, 1]$ der reellen Zahlenachse:

$$\mu_A : G \to [0, 1]. \tag{22.1}$$

Die Menge

$$A = \{(x, \mu_A(x)) | x \in G\} \tag{22.2}$$

heißt *fuzzy-Menge* (*unscharfe Menge*, *fuzzy set*) über G.

Die Funktion μ_A heißt *Mitgliedsgrad-* oder *Zugehörigkeitsfunktion*. Die Angabe der Mitgliedsgradwerte von $\mu_A(x)$ ist nicht Aufgabe der *fuzzy logic*, sie werden vielmehr nach Erfahrung, sprachlicher Gewohnheit usw. festgelegt. Die Werte selbst können wieder unscharf sein, was zu *fuzzy*-Mengen höherer Ordnung führt. Dies ist auch schon in Beispielen von Abschnitt 22.2.1 angeklungen.

Eine klassische Menge M auf einer Grundmenge G kann als Sonderfall einer *fuzzy*-Menge abgeleitet werden: Die Mitgliedsgradwerte für alle $x \in M$ sind 1, während die Mitgliedsgradwerte für $x \notin M$ gleich 0 sind. Die Funktion μ_A ist dann gleich mit der charakteristischen Funktion μ_M der Menge M.

Fuzzy-Mengen dieser Art werden häufig auch als *linguistische Variable* bezeichnet. Ein Beispiel soll eine Möglichkeit der Notation verdeutlichen: Die *fuzzy*-Menge (linguistische Variable) HOHES FIEBER besitzt die Grundmenge $G = \{37, 38, 39, 40, 41, 42\}$ (Maßeinheit: *Grad Celsius*). Die *fuzzy*-Menge HOHES FIEBER ist:

$$\text{HOHES FIEBER} = \{(39, 0.5), (40, 0.8), (41, 1.0), (42, 1.0)\} \tag{22.3}$$

Die Zugehörigkeitsfunktion $\mu_{\text{HOHES FIEBER}}$ lautet tabellarisch:

x	37	38	39	40	41	42
$\mu_{\text{HOHES FIEBER}}(x)$	0.0	0.0	0.5	0.8	1.0	1.0

Es sei darauf hingewiesen, dass bei *fuzzy*-Mengen in der Notation von (22.3) Elemente, deren Mitgliedsgradwert 0 ist, weggelassen werden.

In diesem Beispiel wurden zwei übliche Notationen für endliche *fuzzy*-Mengen benützt: Die Aufzählung aller Paare $(x, \mu_A(x))$ mit $x \in G$ und Darstellung in Vektor- oder Tabellenform.

Eine andere Notation für eine endliche *fuzzy*-Menge A über der Grundmenge $G = \{x_1, x_2, ..., x_n\}$ ist:

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + ... + \frac{\mu_A(x_n)}{x_n} = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i} = \sum_{i=1}^{n} \mu_A(x_i) \Big/ x_i. \tag{22.4}$$

Es sei bemerkt, dass die Bruchstriche, Pluszeichen und Summenzeichen hier nicht die übliche Bedeutung wie in der Mathematik haben, sondern zur formalen Darstellung dienen.

Für unendliche *fuzzy*-Mengen wird, angelehnt an obige Notation, das Integralzeichen verwendet. Dazu ein weiteres Beispiel aus [Boeh93]:

Die Grundmenge G sei die Menge der reellen Zahlen. Die *fuzzy*-Menge A=NAHE NULL mit der Mitgliedsgradfunktion

$$\mu_{\text{NAHE NULL}}(x) = \frac{1}{1 + x^2} \tag{22.5}$$

könnte dann so geschrieben werden:

$$A = \text{NAHE NULL} = \int_{-\infty}^{+\infty} \frac{1}{1 + x^2} \Big/ x. \tag{22.6}$$

Häufig hat der Graph der Mitgliedsgradfunktion einen triangulären oder trapezförmigen Verlauf. Diese Mitgliedsgradfunktion und damit die *fuzzy*-Menge kann dann durch mehrere Terme definiert werden. Als Beispiel dient die Funktion von Bild 22.1.

Die Grundmenge ist das Intervall $G = [a, b]$ und es gelte $a \leq c_1 < m < c_2 \leq b$. Dann lässt sich die Mitgliedsgradfunktion wie folgt beschreiben:

$$\mu_A(x) = \begin{cases} \max\left(0, \frac{x - c_1}{m - c_1}\right) & \text{für} \quad x \in [a, m] \\ \max\left(0, \frac{c_2 - x}{c_2 - m}\right) & \text{für} \quad x \in [m, b] \end{cases} \tag{22.7}$$

Diese *fuzzy*-Menge könnte etwa als folgende linguistische Variable NORMALE TEMPERATUR interpretiert werden: „Der Sollwert ist m (*Grad Celsius*). Von a (z.B. $a = -273$

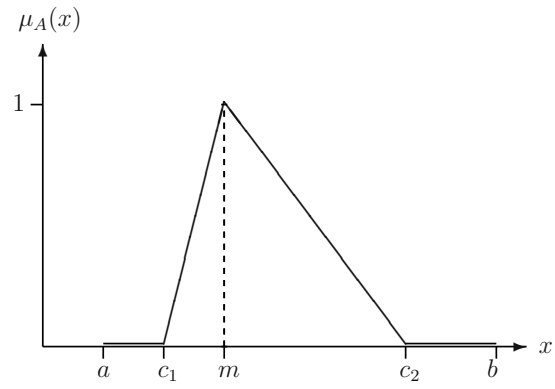


Bild 22.1: Beispiel für eine *fuzzy*-Menge mit triangulärer Mitgliedsgradfunktion.

Grad) bis c_1 ist die Zugehörigkeit 0, dann nimmt sie linear bis zum Wert m zu. Danach fällt die Zugehörigkeit linear bis zum Wert c_2 und ist von c_2 bis b wieder 0."

Oft müssen die Elemente einer unscharfen Menge bezüglich ihrer Zugehörigkeit bewertet werden: Es interessieren z.B. nur solche Elemente aus der Grundmenge G, deren Zugehörigkeit zur unscharfen Menge A größer ist als ein bestimmter Schwellwert α. Man spricht dann von der *α-Niveaumenge* (*α-level set*, *α-cut*, *α-Schnitt*):

$$A_\alpha = \{x \in G | \mu_A(x) \geq \alpha\} \tag{22.8}$$

Wird in (22.8) statt $\mu_A(x) \geq \alpha$ gefordert, dass gilt $\mu_A(x) > \alpha$, so spricht man von der *strengen α-Niveaumenge*. Die α-Niveaumengen sind klassische (scharfe) Mengen. Dazu ein Beispiel: Die linguistische Variable GS (geeigneter Schwellwert zur Binarisierung eines Bildes) sei:

$$\mathrm{GS} = \tfrac{0.1}{140} + \tfrac{0.4}{141} + \tfrac{0.7}{142} + \tfrac{0.9}{143} + \tfrac{1.0}{144} + \tfrac{0.9}{145} + \tfrac{0.7}{146} + \tfrac{0.4}{147} + \tfrac{0.1}{148}$$

Beispiele für α-Niveaumengen sind dann: $\mathrm{GS}_{0.4} = \{141, 142, 143, 144, 145, 146, 147\}$ oder $\mathrm{GS}_{1.0} = \{144\}$.

Zwei weitere Begriffe: Die *Höhe* und die *Normalisierung* einer *fuzzy*-Menge. Die Höhe ist

$$h(A) = \sup_{x \in G} \mu_A(x). \tag{22.9}$$

Dabei ist sup das Supremum (die kleinste obere Schranke) des Wertebereichs von μ_A. Falls es einen größten $\mu_A(x)$-Wert gibt (z.B. bei endlichem Wertevorrat), ist $\sup \mu_A(x) = \max \mu_A(x)$. Die *fuzzy*-Menge A heißt normalisiert, wenn gilt:

$$h(A) = 1. \tag{22.10}$$

Jetzt werden einige Beziehungen und Verknüpfungen von *fuzzy*-Mengen definiert:

Zunächst die *Gleichheit* von zwei *fuzzy*-Mengen und der Begriff der *Teilmenge*. Dazu werden zwei *fuzzy*-Mengen A und B über derselben Grundmenge G betrachtet:

$$A = B \Longleftrightarrow \mu_A(x) = \mu_B(x) \text{ für alle } x \in G. \tag{22.11}$$

und

$$A \subset B \Longleftrightarrow \mu_A(x) \leq \mu_B(x) \text{ für alle } x \in G. \tag{22.12}$$

Damit können auch Begriffe wie *Reflexivität* $(A = A)$, *Symmetrie* $(A = B \Longrightarrow B = A)$ und *Transitivität* $(A = B \text{ und } B = C \Longrightarrow A = C)$ für *fuzzy*-Mengen definiert werden.

Als Beispiel sei nochmals die *fuzzy*-Menge GS (geeigneter Schwellwert zur Binarisierung eines Bildes) von oben betrachtet. Wenn man eine *fuzzy*-Menge GGS „gut geeigneter Schwellwert ...“

$$\text{GGS} = \tfrac{0.9}{143} + \tfrac{1.0}{144} + \tfrac{0.9}{145}$$

festlegt, dann gilt GGS $\subset$ GS. Außerdem sind GS und GGS normalisiert, da $h(\text{GS}) = h(\text{GGS}) = 1$.

Sprachlich werden unscharfe Aussagen oft verstärkt (geeignet, gut geeignet, sehr gut geeignet) oder abgeschwächt (groß, nicht groß, eher klein). Diese Sprechweisen lassen sich auch auf *fuzzy*-Mengen übertragen. Dazu verwendet man Begriffe wie *Konzentration* und *Dilatation*. Da die Mitgliedsgradfunktion $\mu_A(x)$ nur Werte aus dem Intervall $[0, 1]$ annehmen darf, werden durch Potenzieren mit einem Exponenten $n > 0$ die Mitgliedsgradwerte verkleinert und durch Radizieren vergrößert.

Der *Konzentrations-Operator* kon_n und der *Dilatations-Operator* dil_n für eine *fuzzy*-Menge $A = \{(x, \mu_A(x))|x \in G\}$ sind somit wie folgt definiert:

$$\text{kon}_n A = \{(x, \mu_A(x)^n)|x \in G\} \tag{22.13}$$

und

$$\text{dil}_n A = \{(x, \sqrt[n]{\mu_A(x)})|x \in G\}. \tag{22.14}$$

Es gilt offenbar für $x \in G$ und $n > 1$:

$$\text{kon}_n A \subset A \subset \text{dil}_n A \tag{22.15}$$

Wie die Verstärkung und die Abschwächung von linguistischen Variablen durchzuführen ist, ist nicht die Aufgabe der *fuzzy logic*. Vielmehr muss eine passende Modellierung in jedem konkreten Anwendungsfall festgelegt werden.

Weitere wichtige Mengenoperationen sind der *Durchschnitt*, die *Vereinigung* und das *Komplement*. Auch hier werden wieder Definitionen verwendet, die mit der klassischen Mengenlehre verträglich sind. Betrachtet werden zwei *fuzzy*-Mengen A und B auf der Grundmenge G. Dann ist der Durchschnitt:

$$A \cap B : \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)), \tag{22.16}$$

die Vereinigung

$$A \cup B : \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)), \tag{22.17}$$

und das Komplement

$$\bar{A} : \mu_{\bar{A}}(x) = 1 - \mu_A(x). \tag{22.18}$$

Anhand der Definitionen ist leicht zu sehen, wie die Verträglichkeit mit der scharfen Mengenlehre gewährleistet ist. In der Literatur werden auch noch andere Definitionen für diese Operatoren diskutiert.

Die Wirkungsweise dieser Mengenperatoren ist nicht immer mit dem sprachlichen Verständnis im Einklang. Dazu ein Beispiel: Bei der Auswahl und Implementierung von Bildverarbeitungsoperatoren ist es wichtig, mit welchem Aufwand sie zu realisieren und wie ihre Ergebnisse zu beurteilen sind. Auf einer Menge zur Verfügung stehender Bildverarbeitungsoperatoren $G = \{O_1, O_2, ..., O_n\}$ zu einer bestimmten Problemstellung könnte man somit die linguistischen Variablen ANGEMESSEN in der Realisierung und GUT in der Qualität definieren. Für einen bestimmten Bildverarbeitungsoperator O_i gelte nun:

$$\mu_{\text{ANGEMESSEN}}(O_i) = 0.2 \text{ und } \mu_{\text{GUT}}(O_i) = 0.9,$$

was bedeutet, dass er zwar ziemlich gute Ergebnisse liefert, aber nicht einfach zu realisieren ist. In der *fuzzy*-Menge ANGEMESSEN $\wedge$ GUT, die man aus dem Durchschnitt der beiden *fuzzy*-Mengen bildet, erhält der Bildverarbeitungsoperator O_i den Wert

$$\mu_{\text{ANGEMESSEN} \cap \text{GUT}}(O_i) = \min(\mu_{\text{ANGEMESSEN}}, \mu_{\text{GUT}}) = \min(0.2, 0.9) = 0.2,$$

d.h. er wird insgesamt ziemlich schlecht bewertet. Legt man fest, dass ein Bildverarbeitungsoperator angemessen ODER gut sein soll, so erhält man in diesem Beispiel einen Wert von 0.9, was intuitiv einer zu guten Bewertung entspricht.

In der Praxis wird man häufig Kompromisse schließen müssen. Es kann sein, dass man einen hohen Implementierungsaufwand in Kauf nimmt, um gute Ergebnisse zu erhalten. Für die Mengenoperatoren auf *fuzzy*-Mengen heißt das, dass man Entscheidungen treffen möchte, die „nicht ganz UND“ sind, also Werte liefert, die zwischen min und max liegen.

Mengenoperatoren dieser Art werden *kompensatorische* oder *kombinatorische* Operatoren genannt. Auch hierzu wird ein Beispiel gegeben und darauf hingewiesen, dass in der Literatur eine Vielzahl von verschiedenen Definitionen untersucht wird.

Der *min-max-Kompensations-Operator* $K_\gamma(x, y)$ wird folgendermaßen definiert:

$$K_\gamma(x, y) = \min(x, y)^{1-\gamma} \cdot \max(x, y)^\gamma. \tag{22.19}$$

Der Parameter $\gamma \in [0, 1]$ heißt *Kompensationsgrad* und die Argumente $x, y \in [0, 1]$ stehen für die Mitgliedsgrade. Für $\gamma = 0$ ist $K_0(x, y) = \min(x, y)$ und für $\gamma = 1$ ist $K_1(x, y) = \max(x, y)$.

Für das obige Beispiel sind die Zugehörigkeitswerte des Operators O_i zur *fuzzy*-Menge ANGEMESSEN $\wedge$ GUT für verschiedene Werte des Kompensationsgrads γ in einer Tabelle zusammengestellt:

γ	$K_\gamma(\mu_{\text{ANGEMESSEN}}(O_i), \mu_{\text{GUT}}(O_i))$
0.0	0.200
0.1	0.232
0.2	0.270
0.3	0.314
0.4	0.365
0.5	0.424
0.6	0.493
0.7	0.573
0.8	0.666
0.9	0.774
1.0	0.900

In [Boeh93] oder [Zimm92] werden weitere interessante Aspekte der *fuzzy*-Mengen untersucht, etwa *fuzzy*-Relationen, *fuzzy*-Zahlen oder die *fuzzy*-Aussagenlogik.

Nun noch einige Bemerkungen zum „unscharfen Schließen“. Darunter versteht man das Herleiten von nichtpräzisen Folgerungen aus nichtpräzisen Voraussetzungen. Eine umfassende Darstellung dieser Problematik würde den Rahmen dieser kurzen Zusammenfassung bei weitem sprengen, so dass hier nur versucht wird darzustellen, welche Probleme auftreten. Zu Fragen, wie sie mathematisch modelliert werden können, sei auf die schon mehrfach zitierte Literatur verwiesen.

Nach [Boeh93], wo diesen Fragestellungen ein Kapitel am Ende gewidmet ist, müssen an eine Theorie zum unscharfen Schließen einige Forderungen gestellt werden:

- Die in *fuzzy*-Aussagen (z.B. „Das Auto fährt schnell“, „Gallo ist intelligenter als Hugo“, „Die Temperatur ist zu hoch“) enthaltene Unschärfe muss zahlenmäßig darstellbar gemacht werden.

Dazu wird die Möglichkeitsverteilung (Possibilitätsverteilung) einer *fuzzy*-Aussage durch eine *fuzzy*-Menge charakterisiert (eingeschränkt, restringiert). Die Possibilität wird durch die Mitgliedsgradfunktion bestimmt.

- Wenn Schlussfolgerungen abgeleitet werden, so muss der Vorgang der Ableitung einem mathematischen Formalismus gehorchen und berechenbar sein. Typische Schlussfolgerungen sind: „Wenn X und Y dann Z“ oder „Wenn X oder Y dann Z“. Dabei sind X, Y und Z *fuzzy*-Aussagen, die durch *fuzzy*-Mengen A, B und C dargestellt werden. Es muss somit ein mathematischer Weg gefunden werden, wie aus den *fuzzy*-Mengen A und B die *fuzzy*-Menge C berechnet werden kann.

- Schlussfolgerungen, die auf diese Weise abgeleitet werden, müssen so gut wie möglich mit der Realität übereinstimmen, d.h. dass sie in einem bestimmten Anwendungsgebiet ein Ergebnis liefern, das mit „gesundem Menschenverstand“ auch erzielt worden wäre.

Um diesen Forderungen gerecht zu werden, wird in [Boeh93] zunächst der in dieser Zusammenstellung etwas vage verwendete Begriff der *fuzzy*-Variablen oder linguistischen Variablen exakt definiert. Dazu wird eine Possibilitätsverteilung definiert, die auf den *fuzzy*-Mengen aufbaut. Dann werden das Projektionsprinzip und die zylindrische Erweiterung definiert, mit denen Possibilitätsverteilungen eingeschränkt bzw. erweitert werden können.

Die Partikularisation beschäftigt sich mit der Frage, wie man eine Possibilitätsverteilung zu ändern hat, wenn man als neues Wissen erfährt, dass sich die Possibilitätsverteilung einer Subvariablen geändert hat.

Regeln der maximalen und minimalen Restriktion beschäftigen sich mit Ableitungen der Form: Wenn X und Y *fuzzy*-Aussagen sind, wie können dann die *fuzzy*-Aussagen berechnet werden, die durch beide Aussagen beeinflusst werden? Ein Beispiel: X und Y werden durch die *fuzzy*-Mengen A und B dargestellt. Wie soll dazu die *fuzzy*-Menge berechnet werden, die aus dem kartesischen Produkt $A \times B$ besteht?

Aufbauend darauf werden „Wenn-dann-Inferenzregeln“ (Wenn X dann Y) und der „generalisierte Modus ponens“ (X ist ...; Wenn X dann Y; Was ist Y?) modelliert.

Beim Aufbau einer *fuzzy-logic*-Applikation ist somit praktisch wie folgt vorzugehen:

- Im ersten Schritt muss festgelegt werden, welcher Sachverhalt fuzzifiziert werden soll. Bei einer Bildverarbeitungsanwendung könnte man sich z.B. entscheiden, die Schwellwertbestimmung einer Binarisierung zu fuzzifizieren.

- Als Nächstes ist festzulegen, wie die Zugehörigkeitsfunktionen definiert werden. Hier spielt oft die persönliche Erfahrung oder Bewertung der Sachverhalte eine wichtige Rolle. Bei Anwendungen in der digitalen Bildverarbeitung und Mustererkennung können die Werte der Mitgliedsgradfunktionen häufig aus den zu verarbeitenden Daten abgeleitet werden. Bei einer dynamischen Schwellwertbestimmung könnte man z.B. den Schwellwert für einen Bildausschnitt durch eine *fuzzy*-Menge darstellen, deren Mitgliedsgradwerte aus einer bestimmten Bildumgebung berechnet werden.

- Des Weiteren ist festzulegen, wie die Verknüpfungen der *fuzzy*-Mengen realisiert werden sollen. Hier muss man z.B. definieren, welche Operatoren man für UND und ODER verwendet. Auch die Wahl bestimmter kompensatorischer Operatoren fällt in diesen Bereich.
- Der nächste Schritt ist das Aufstellen von problemrelevanten Regeln. Hier muss also das Wissen der jeweiligen Problemstellung modelliert und für die Anwendung in ein geeignetes Regelsystem gepackt werden.
- Der letzte Schritt ist die Defuzzifizierung: Hier muss festgelegt werden, wie die Abbildung der abgeleiteten *fuzzy*-Ergebnisse auf reale Werte geschehen soll, die dann z.B. zur Steuerung einer Anlage verwendet werden können.

22.3 Fuzzy Klassifikator

In diesem Abschnitt wird ein Klassifikator beschrieben, der die Aufteilung des N-dimensionalen Merkmalsraums mit einem *fuzzy*-Ansatz durchführt. Die Grundlagen der numerischen Klassifikation sind in den Kapiteln 11 und 20 zusammengestellt. Im Folgenden sind nochmals kurz die relevanten Größen mit ihren Bezeichnungen zusammengefasst:

t	Anzahl der *cluster*,
N	Dimension des Merkmalsraums,
M	Anzahl der Merkmalsvektoren,
$\mathbf{g}_i \in R^N$	i-ter Merkmalsvektor, i=0,...,M-1
$\mathbf{z}_j \in R^N$	j-tes *cluster*-Zentrum, j=0,...,t-1.

Zusätzlich werden noch eingeführt:

$w_{ij} = \mu_j(\mathbf{g}_i)$	Mitgliedsgradwert des Merkmalsvektors $\mathbf{g}_i$ zum *cluster* j,
m	ein Parameter, der den *fuzzy*-Grad steuert.

Gesucht sind t *cluster*-Zentren $\mathbf{z}_j$, die so im Merkmalsraum verteilt liegen, dass der Ausdruck

$$\sum_{i=0}^{M-1} \sum_{j=0}^{t-1} w_{ij}^m d_{ij} \tag{22.20}$$

minimal wird. Eine Randbedingung ist dabei

$$\sum_{j=0}^{t-1} w_{ij} = 1 \tag{22.21}$$

mit $w_{ij} > 0$, für $i = 0, ..., M-1$ und $j = 0, ..., t-1$. Die Größe d_{ij} ist der euklidische Abstand des Merkmalsvektors $\mathbf{g}_i$ zum *cluster*-Zentrum $\mathbf{z}_j$ im Merkmalsraum:

$$d_{ij} = \sqrt{(\mathbf{g}_i - \mathbf{z}_j)^T (\mathbf{g}_i - \mathbf{z}_j)}. \tag{22.22}$$

Die Optimierung läuft hier über die Mitgliedsgradwerte und die Lage der *cluster*-Zentren im $N-$dimensionalen Merkmalsraum, d.h. dass bei jedem Iterationsschritt die Mitgliedsgradwerte und die *cluster*-Zentren neu berechnet werden, bis ein Abbruchkriterium (siehe unten) erfüllt ist.

Wo liegt hier der *fuzzy*-Ansatz? Bei einem herkömmlichen („scharfen“) Klassifikator wird als Ergebnis jeder Merkmalsvektor $\mathbf{g}_i$ genau einem *cluster*-Zentrum $\mathbf{z}_j$ und damit genau einer Klasse zugeordnet. Der unscharfe Klassifikator ordnet einen Merkmalsvektor allen *cluster*-Zentren zu, jedoch mit unterschiedlichen Mitgliedsgradwerten. Mit der Nomenklatur und der Notation der *fuzzy logic* heißt das, dass jedem Merkmalsvektor $\mathbf{g}_i$ eine unscharfe Menge MV_i zugeordnet wird:

$$MV_i = \frac{w_{i0}}{\vec{z_0}} + \frac{w_{i1}}{\vec{z_1}} + ... + \frac{w_{it-1}}{\vec{z_{t-1}}}, i = 0, ..., M-1. \tag{22.23}$$

Bei praktischen Anwendungen kann die Zugehörigkeit zu den *cluster*-Zentren noch durch weitere Restriktionen eingeschränkt werden. Dazu drei Beispiele:

- Jeder Merkmalsvektor $\mathbf{g}_i$ kann nur maximal c_1 *cluster*-Zentren mit einem Mitgliedsgradwert größer Null zugeordnet werden.
- Falls für die Distanz eines Merkmalsvektors $\mathbf{g}_i$ zu einem *cluster*-Zentrum $\mathbf{z}_j$ gilt $d_{ij} > c_2$, wird $w_{ij} = 0$ gesetzt.
- Der Mitgliedsgradwert eines Merkmalsvektors $\mathbf{g}_i$ zum *cluster*-Zentrum $\mathbf{z}_j$ wird zu $w_{ij} = 0$ gesetzt, falls $w_{ij} < c_3$ gilt.

Der folgende Algorithmus beschreibt den *fuzzy*-Klassifikator. Die zusätzliche Restriktion wird in diesem Algorithmus nach der dritten Variante implementiert.

A22.1: Fuzzy-Klassifikator.

Voraussetzungen und Bemerkungen:

◇ Die Voraussetzungen und Bezeichnungen sind in der Einführung zu diesem Abschnitt zusammengestellt.

Algorithmus:

(a) Festlegen von t anfänglichen *cluster*-Zentren $\mathbf{z}_j$, $j = 0, ..., t-1$. Diese *cluster*-Zentren könnten z.B. das Ergebnis eines Minimum-Distance-Cluster-Verfahrens sein (Abschnitt 20.3). Sie können aber auch „aus der Erfahrung“ oder gleichverteilt im Merkmalsraum festgelegt werden.

(b) Iteration:

(ba) Berechnung der Distanzen d_{ij}^2 für $0 \leq i \leq M-1$ und $0 \leq j \leq t-1$.

$d_{ij}^2 = (\mathbf{g}_i - \mathbf{z}_j)^T(\mathbf{g}_i - \mathbf{z}_j)$.

(bb) Berechnung der nicht normierten Mitgliedsgradwerte v_{ij} für $0 \leq i \leq M-1$ und $0 \leq j \leq t-1$:

(bba) Falls $d_{ij}^2 = 0$: $v_{ij} = 1$ und $v_{ik} = 0$, $0 \leq k \leq t-1, k \neq j$.

(bbb) Falls $d_{ij}^2 \neq 0$:

$$v_{ij} = \frac{1}{\sum\limits_{k=0}^{t-1} \left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}}, 0 \leq k \leq t-1, k \neq j.$$

(bc) Berechnung der neuen *cluster*-Zentren:

$$\mathbf{z}_j = \sum_{i=0}^{M-1} v_{ij}^m \mathbf{g}_i \Big/ \sum_{i=0}^{M-1} v_{ij}^m, 0 \leq j \leq t-1.$$

(bd) Berechnung des Abbruchkriteriums. Hier sind zwei verschiedene Varianten möglich:

$$R_\gamma = \Big[\sum_{j=0}^{t-1} \sum_{n=0}^{N-1} |z_{jn}(s) - z_{jn}(s-1)|^\gamma \Big]^{\frac{1}{\gamma}} < \varepsilon,$$

oder

$$Q_\gamma = \Big[\sum_{i=0}^{M-1} \sum_{j=0}^{t-1} |v_{ij}(s) - v_{ij}(s-1)|^\gamma \Big]^{\frac{1}{\gamma}} < \varepsilon,$$

Dabei steht s für den s-ten Iterationsschritt. ε ist ein vorzugebender kleiner Wert. Mit γ können unterschiedliche Normen verwendet werden: $\gamma = 1$ ist die $l1$-Norm, $\gamma = 2$ die euklidische Norm und $\gamma = \infty$ die Tschebyscheff-Norm.

(c) Schwellwertbildung bei den Mitgliedsgradwerten: Falls $v_{ij} < c_3$: $v_{ij} = 0$, $0 \leq i \leq M-1$ und $0 \leq j \leq t-1$.

(d) Normalisierung der Mitgliedsgradwerte:

$$w_{ij} = \frac{v_{ij}}{\sum\limits_{k=0}^{t-1} v_{ik}}$$

für $0 \leq i \leq M-1$ und $0 \leq j \leq t-1$.

Ende des Algorithmus

Wie oben bereits dargestellt, liegen nach Ablauf dieses Algorithmus M *fuzzy*-Mengen vor, und zwar zu jedem Merkmalsvektor eine. Die Möglichkeiten der Weiterverarbeitung sind unterschiedlich. Wenn z.B. mit dem Klassifizierungsschritt die Verarbeitung beendet ist, kann eine Defuzzifizierung vorgenommen werden, indem man etwa einen Merkmalsvektor derjenigen Klasse zuordnet, zu der er den größten Mitgliedsgradwert hat.

Der *fuzzy*-Ansatz kommt aber dann erst richtig zum Tragen, wenn die so erhaltenen unscharfen Mengen im weiteren Verlauf der Verarbeitung mit anderen unscharfen Mengen verknüpft werden. So könnten in einem zum *fuzzy*-Klassifikator parallel verlaufenden Arbeitsschritt, der jedoch ganz andere Merkmale zur Klassifizierung verwendet, auch *fuzzy*-Mengen als Ergebnis auftreten. Die letztliche Zuordnung erfolgt dann auf Grund einer UND-Verknüpfung der beiden *fuzzy*-Mengen.

Nach der Darstellung soll das Verfahren anhand von zwei Beispielen getestet werden. Das erste Beispiel kann bei der Implementierung als Vergleichsbeispiel verwendet werden. Es wird ein eindimensionaler Merkmalsraum verwendet. Die Eingabedaten und die Ergebnisse sind in folgender Tabelle zusammengestellt:

Eingabe-Merkmalsvekt.	-5.00	-4.00	1.00	2.00	5.00	6.00
Anf. *cluster*-Zentren	-4.00	5.00				
Iterierte *cluster*-Zentren	-4.35	3.76				
Mitgliedsgradwerte						
w_{i0}	0.99455	0.99795	0.21002	0.07126	0.01729	0.04475
w_{i1}	0.00545	0.00205	0.78998	0.92874	0.98271	0.95525

Der *fuzzy*-Parameter m wurde hier zu $m = 2$ gesetzt. Als Abbruchkriterium wurde Q_γ mit $\gamma = 2$, also die euklidische Norm, verwendet. Es waren neun Iterationen notwendig, um eine Fehlerschranke von $\varepsilon < 0.001$ zu erreichen. Eine Schwellwertbildung der Mitgliedsgradwerte wurde nicht durchgeführt.

Als zweites Beispiel wurden dreidimensionale Farbvektoren klassifiziert. Insgesamt wurden 16 Merkmalsvektoren verwendet:

Nummer	Rotanteil	Grünanteil	Blauanteil
0	220	20	14
1	218	18	23
2	244	6	8
3	223	28	35
4	22	220	35
5	65	238	12
6	14	228	4
7	12	251	18
8	12	28	235
9	23	14	220
10	2	31	242
11	18	22	249
12	180	85	22
13	93	141	31
14	12	185	242
15	164	143	156

Bei genauerer Betrachtung dieser Farbvektoren erkennt man, dass die ersten zwölf Vektoren der Reihe nach jeweils viermal die Farbe Rot, Grün und Blau, mit leicht unterschiedlichen Farbanteilen repräsentieren. Bei den letzten vier Vektoren sind die Farbanteile weiter verfälscht, der letzte repräsentiert einen Grauton, wobei der Rotanteil etwas überwiegt.

Sinnvollerweise wurden drei *cluster*-Zentren vorgegeben, die anfänglich mit den reinen additiven Grundfarben vorbesetzt waren. Die iterierten Zentren zeigt folgende Tabelle:

Anfängliche *cluster*-Zentren	Rot	Grün	Blau
0	255.00	0.00	0.00
1	0.00	255.00	0.00
2	0.00	0.00	255.00
Iterierte *cluster*-Zentren	Rot	Grün	Blau
0	218.40	27.52	23.78
1	31.71	226.59	23.49
2	15.33	31.40	235.24

In der folgenden Tabelle sind die Mitgliedsgradwerte für verschiedene Werte des *fuzzy*-Parameters m zusammengestellt. Eine Schwellwertbildung der Mitgliedsgradwerte wurde nicht durchgeführt. Für $m = 2$ waren sechs Iterationen, für $m = 3$ sieben Iterationen und für $m = 10$ vierzehn Iterationen notwendig (Fehlerschranke $\varepsilon = 0.001$ mit Q_2-Abbruchkriterium). Es ist klar, dass sich bei unterschiedlichen m-Werten auch geringfügig veränderte *cluster*-Zentren ergeben, die hier jedoch nicht aufgeführt sind.

	$m = 2$			$m = 3$			$m = 10$		
Nr	w_{i0}	w_{i1}	w_{i2}	w_{i0}	w_{i1}	w_{i2}	w_{i0}	w_{i1}	w_{i2}
0	0.9912	0.0048	0.0040	0.9211	0.0409	0.0380	0.5243	0.2398	0.2359
1	0.9929	0.0038	0.0033	0.9375	0.0320	0.0305	0.5790	0.2116	0.2094
2	0.9609	0.0210	0.0181	0.8098	0.0977	0.0926	0.4536	0.2747	0.2717
3	0.9948	0.0028	0.0024	0.9208	0.0405	0.0387	0.4894	0.2565	0.2541
4	0.0046	0.9908	0.0046	0.0533	0.8933	0.0534	0.2469	0.5057	0.2474
5	0.0181	0.9691	0.0128	0.1125	0.7923	0.0952	0.2886	0.4326	0.2788
6	0.0127	0.9760	0.0113	0.0779	0.8484	0.0737	0.2697	0.4634	0.2670
7	0.0167	0.9672	0.0161	0.0866	0.8280	0.0853	0.2725	0.4553	0.2722
8	0.0023	0.0024	0.9953	0.0157	0.0159	0.9685	0.1380	0.1390	0.7230
9	0.0126	0.0117	0.9757	0.0750	0.0718	0.8532	0.2682	0.2665	0.4653
10	0.0042	0.0045	0.9913	0.0443	0.0461	0.9096	0.2469	0.2502	0.5029
11	0.0063	0.0062	0.9874	0.0504	0.0498	0.8998	0.2553	0.2556	0.4891
12	0.8712	0.0840	0.0448	0.6293	0.2123	0.1584	0.3928	0.3133	0.2939
13	0.2409	0.6487	0.1105	0.3034	0.4866	0.2101	0.3290	0.3664	0.3045
14	0.1166	0.2633	0.6201	0.2117	0.3215	0.4668	0.3036	0.3352	0.3612
15	0.3813	0.3040	0.3147	0.3564	0.3194	0.3242	0.3376	0.3312	0.3312

Die Werte zeigen, wie der *fuzzy*-Parameter m die Mitgliedsgradwerte beeinflusst: Je größer m wird, desto mehr nähern sich die Werte an. Die Tendenz der Zugehörigkeit zu den

einzelnen *cluster*-Zentren bleibt aber immer erhalten. Sogar beim letzten Merkmalsvektor, einem Grauwert, bei dem der Rotanteil geringfügig überwiegt, ist bei allen m-Werten die Zugehörigkeit zum ersten *cluster*-Zentrum am größten.

Nachdem zu den Farbvektoren die *fuzzy*-Mengen berechnet sind, kann jetzt z.B. eine scharfe Zuordnung zu der Klasse (zu dem *cluster*-Zentrum) mit dem höchsten Mitgliedsgradwert erfolgen. Allerdings hat sich dann der Aufwand der Verwendung des *fuzzy*-Klassifikators eigentlich nicht gelohnt. Denn die Vorteile des Ansatzes über die *fuzzy logic* kommen erst zum Tragen, wenn weiter mit den unscharfen Mengen gearbeitet wird und diese im Rahmen einer umfangreicheren Problemstellung mit anderen unscharfen Aussagen verknüpft werden.

Bei praktischen Anwendungen dieses *fuzzy-clustering* sollte man darauf achten, dass die Menge der Merkmalsvektoren nicht zu groß wird. Denn wenn man den obigen Algorithmus betrachtet, erkennt man, dass ein beachtlicher Rechenaufwand notwendig ist. Man könnte z.B. mit anderen, einfacheren Verfahren eine Vorauswahl der Merkmalsvektoren durchführen und erst mit einer reduzierten Datenmenge den *fuzzy-cluster*-Algorithmus durchlaufen.

Kapitel 23

Speicherung von Segmenten mit Run-Length-Coding

23.1 Anwendungen

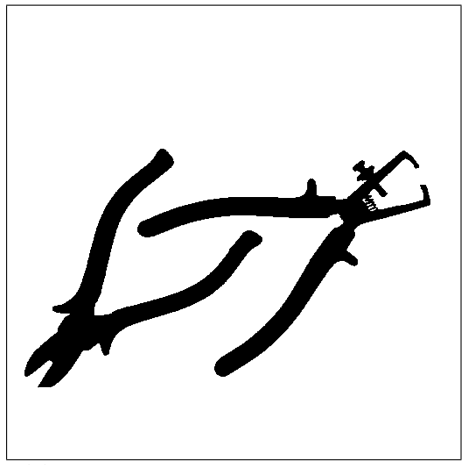
Immer mehr Anwendungen erfordern, Objekte in Echtzeit zu lokalisieren, zu erkennen oder zu verfolgen, wobei die zu verwendende Bildverarbeitungshardware nur über eingeschränkte Ressourcen verfügt. Derartige Anforderungen treten oft in autonomen Systemen im Automotive-Bereich oder in der Robotik auf. Bekanntestes Beispiel dürfte die Bildverarbeitung auf eingebetteten Systemen im Roboter-Fußball sein. Die von der Kamera gelieferte Datenmenge muss hierbei möglichst frühzeitig und effizient verdichtet werden, um eine Ressourcen schonende (CPU, Speicher) Weiterverarbeitung zu ermöglichen.

Lauflängencodierung (engl.*run length encoding, RLE*) bietet hierzu die Möglichkeit, die Pixeldaten zu komprimieren und auf diesem reduzierten Datenvolumen effiziente Segmentierungs- und Merkmalsextraktionsalgorithmen aufzusetzen. Bild 23.5 zeigt die typische Zusammenarbeit von Lauflängencodierung und Union-Find-Algorithmen in diesem Umfeld effizienter Echtzeitbildverarbeitung, wie sie im Folgenden dargestellt wird.

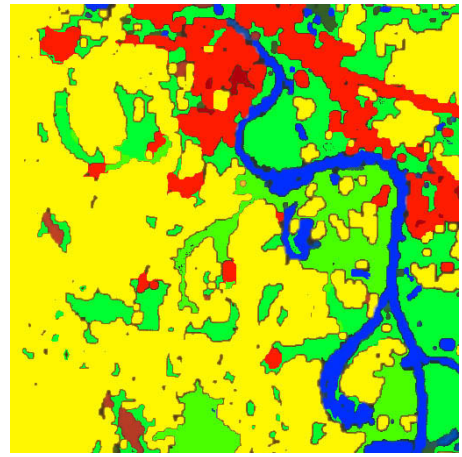
In den vorhergehenden Kapiteln wurde erläutert, wie Bildpunkte (Merkmalsvektoren) eines, eventuell mehrkanaligen, Bildes zu Segmenten zusammengefasst werden. Das Eingabebild beinhaltet dabei die Merkmale, anhand derer die Segmentierung durchgeführt wird. Im einfachsten Fall kann das ein Grauwertbild sein, aus dem durch eine Binarisierung ein Binärbild mit Hintergrund- und Vordergrundbildpunkten erzeugt wird. Bei aufwändigeren Problemstellungen werden aus den aufgezeichneten Originaldaten zunächst weitere Merkmalsdaten berechnet (z.B. Kapitel 11, 17, 18), die einen mehrdimensionalen Merkmalsraum aufspannen. Zur Segmentierung werden dann aufwändigere Verfahren, wie multivariate Klassifikatoren, neuronale Netze oder Fuzzy-Logik (Kapitel 20, 21, 22) verwendet.

Als Eingabebild für dieses Kapitel wird ein segmentiertes Bild vorausgesetzt. Die Segmente können dabei alle den gleichen Grauwert besitzen, wie etwa bei einem Binär- oder Zweipegelbild mit dem Hintergrundgrauwert 0 und dem Vordergrundgrauwert 1 (oder 255) (Bild 23.1-a). Ein anderes Beispiel ist ein klassifiziertes Bild. Die Segmente besitzen hier

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_23



(a)



(b)

Bild 23.1: (a) Beispiel für ein Binärbild. Die Hintergrundbildpunkte sind mit dem Grauwert 255 codiert. Die Vordergrundbildpunkte (Segmente) besitzen alle den Grauwert 0. (b) Beispiel für ein klassifiziertes Satellitenbild. Die Segmente besitzen unterschiedliche Grauwerte, die ein Code für die Zugehörigkeit zu einer Klasse sind. In diesem Beispiel wurden die Segmente mit einer Pseudofarbcodierung unterschiedlich eingefärbt. Die Farben der Segmente entsprechen hier folgenden Klassen: Gelb – landwirtschaftliche Nutzungsfläche, rot – Bebauung, blau – Gewässer, grün – Wald, schwarz – nicht klassifizierbar (farbig im Farbteil).

in der Regel unterschiedliche Grauwerte, die als Code für die Zugehörigkeit zu einer Klasse interpretiert werden können (Bild 23.1-b).

Eine Zielsetzung dieses Kapitels ist die Darstellung von Verfahren zur kompakten Speicherung von segmentierten Bildern. Dazu wird die *run-length*-Codierung (Lauflängencodierung) ausführlich dargestellt. In diesem Zusammenhang wird ein Verfahren benötigt, mit dem zusammengehörige Bildstrukturen, die lauflängencodiert sind, erkannt werden können. Auf der Basis von kompakt gespeicherten Segmenten bauen weitere Möglichkeiten der Beschreibung von Segmenten auf. In diesen Bereich fallen auch Verfahren, die Eigenschaften von Segmenten ermitteln. Beispiele dazu sind die Berechnung des Flächeninhalts, des Umfangs, des Schwerpunktes, der Orientierung, der Form, usw. Auf diese Thematik wird ab Kapitel 24 eingegangen.

Um einen Eindruck der Kompressionsrate dieses Verfahrens zu vermitteln, betrachten wir folgendes Beispiel: Das Binärbild einer Zange, Bild 23.1-a besteht aus $512 \cdot 512 = 262144$ Pixel. Durch Lauflängencodierung wird das Bild auf 806 Streifen reduziert. Das farbsegmentierte Bild, Abb. 23.1-b derselben Größe wird durch die Lauflängencodierung auf 6780 Streifen reduziert. Für die Laufzeitersparnis bedeutet dies einen Faktor 325 (a) bzw. 38 (b), der lediglich durch minimal komplexere Algorithmen zur Segmentierung bzw. Merkmals-

extraktion abgeschwächt wird, für die Speicherplatzeffizienz steht der Repräsentation eines Pixels (Binär oder farbsegmentiert: typisch 1 Byte/Pixel) durch ein Byte die Repräsentation eines Streifens durch weniger als 10 Byte gegenüber. Problematisch in der Theorie, aber von geringer praktischer Relevanz ist lediglich die Tatsache, dass keine Obergrenze für die Anzahl der Streifen existiert, die unter der Pixelanzahl des Originalbilds liegt.

Im Rahmen einer Verarbeitungsfolge kennzeichnet die Verwendung des *run-length*-Codes den Übergang von der pixelorientierten Verarbeitung zur listenorientierten Verarbeitung.

23.2 Run-Length-Codierung

23.2.1 Prinzipielle Problemstellung und Implementierung

Vorausgesetzt wird zunächst ein Binärbild mit den Grauwerten 0 (Hintergrundbildpunkte) und 1 (Vordergrundbildpunkte). Es werden nun die Längen der 0- und 1-Ketten angegeben. Der Anfang der Bildzeile

$$00000000111000000011111000000...$$

ergibt folgenden *run-length*-Code:

$$(8 \cdot 0), (3 \cdot 1), (7 \cdot 0), (5 \cdot 1), (5 \cdot 0),... \ .$$

Da die Hintergrundbildpunkte meistens nicht interessieren, werden nur die Positionen und die Längen der 1-Ketten codiert. Im obigen Beispiel ergibt sich dann folgender *run-length*-Code:

$$(8, 3), (18, 5),... \ .$$

Der Code sagt also aus, dass in der betrachteten Zeile x ab der Spaltenposition $y = 8$ eine 1-Kette (*Sehne*) der Länge 3 und ab der Position $y = 18$ eine Sehne der Länge 5 beginnt. Ob in einer Implementierung für eine Sehne die Anfangsposition y_a und die Länge l angegeben wird oder die Anfangsposition y_a und die Endposition y_e, macht keinen prinzipiellen Unterschied. Die Entscheidung kann möglicherweise von spezieller Hardware abgenommen werden, die den *run-length*-Code in der einen oder anderen Form liefert.

Ein Beispiel zeigt Bild 23.2. Es entstand aus Bild 23.1-a durch Verkleinerung auf $100 \cdot 100$ Pixel. Es wurden die schwarzen Vordergrundpixel codiert. Ein Ausschnitt aus einem lesbaren Ausdruck des *run-length*-Codes ist im Folgenden wiedergegeben. Die Bedeutung des Wertes „`Code`“ wird später erläutert:

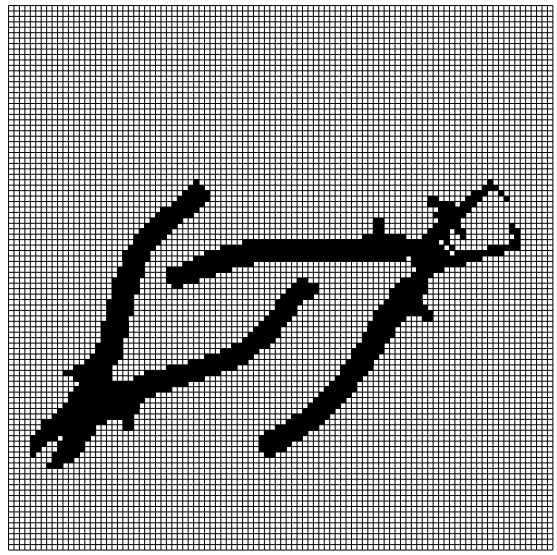


Bild 23.2: Beispiel zur *run-length*-Codierung. Das Testbild 23.1-a wurde auf eine Größe von $100 \cdot 100$ Pixel verkleinert. Dann wurden die scharzen Vordergrundbildpunkte codiert. Ein Auszug des zugehörigen *run-length*-Codes ist in den laufenden Text eingefügt.

```
Leerzeile: 1
Leerzeile: 2
...
Leerzeile: 32
In Zeile:  33 sind 2 Ketten
               Anfang:   34 Ende:   34 Code:   1
               Anfang:   88 Ende:   88 Code:   4
In Zeile:  34 sind 2 Ketten
               Anfang:   33 Ende:   35 Code:   1
               Anfang:   87 Ende:   89 Code:   4
In Zeile:  35 sind 3 Ketten
               Anfang:   32 Ende:   36 Code:   1
               Anfang:   85 Ende:   87 Code:   4
               Anfang:   90 Ende:   90 Code:   2
In Zeile:  36 sind 4 Ketten
               Anfang:   31 Ende:   36 Code:   1
               Anfang:   77 Ende:   78 Code:   4
               Anfang:   84 Ende:   86 Code:   4
               Anfang:   91 Ende:   91 Code:   3
In Zeile:  37 sind 3 Ketten
               Anfang:   30 Ende:   35 Code:   1
               Anfang:   77 Ende:   81 Code:   4
               Anfang:   83 Ende:   85 Code:   4
...
In Zeile:  83 sind 3 Ketten
               Anfang:    4 Ende:    4 Code:   1
               Anfang:    8 Ende:   12 Code:   1
               Anfang:   47 Ende:   48 Code:   4
In Zeile:  84 sind 1 Ketten
               Anfang:    8 Ende:   11 Code:   1
In Zeile:  85 sind 1 Ketten
               Anfang:    7 Ende:    9 Code:   1
Leerzeile: 86
Leerzeile: 87
...
Leerzeile: 100
```

Dieser Ausdruck zeigt auch, welche Informationen in der Datenstruktur eines *run-length*-codierten Bildes vorhanden sein müssen: Als erstes müssen die Bildgröße und die Zeilennummern verfügbar sein. Außerdem werden die Anzahl der Sehnen pro Zeile und die Anfangs- und Endangaben (oder Längenangaben) der Sehnen in der Zeile benötigt. Schließlich sollte noch ein Feld mitgeführt werden, in dem die Zugehörigkeit einer Sehne zu einem bestimmten Segment festgehalten wird. Auf diese Problematik wird im nächsten

Abschnitt eingegangen. Folgende Tabelle zeigt ein einfaches Beispiel einer Datenstruktur für den *run-length*-Code:

Globale Größen	
	Anzahl der Bildzeilen
	Anzahl der Bildspalten
run-length-Code	
	Zeilennummer
	Anzahl der Sehnen pro Zeile
	Sehnenanfang
	Sehnenende
	Code der Sehne
	...
	Zeilennummer
	Anzahl der Sehnen pro Zeile
	Sehnenanfang
	Sehnenende
	Code der Sehne
	...

23.2.2 Vereinzelung von Segmenten

Bei einem klassifizierten Bild sind die verschiedenen Klassen mit unterschiedlichen Grauwerten codiert. Wenn ein derartiges Bild lauflängencodiert wird, so können die Klassencodes sofort im Feld `Code der Sehne` eingetragen werden. Bei einem binarisierten Bild haben alle Segmente denselben (Vordergrund-)Grauwert. Wenn die zusammenhängenden Flächen zu einem Segment zusammengefasst werden sollen, so muss zu jeder Sehne des *run-length*-Codes im Feld `Code der Sehne` angegeben werden, zu welchem Segment sie gehört. Dieser Vorgang wird oft auch als *Vereinzelung der Segmente* bezeichnet.

Der *run-length*-Code wird dazu vom Anfang an abgearbeitet, was einer zeilenweisen Verarbeitung des Bildes von oben nach unten entspricht. Angenommen, es tritt die erste Sehne des korrespondierenden Binärbildes in Zeile x auf. Es ist zu prüfen, ob die Sehne mit Sehnen der Zeile $x + 1$ gemeinsame Nachbarn besitzen. Dabei wird entweder 4- oder 8-Nachbarschaft vorausgesetzt. Ist dies der Fall, so gehören diese Sehnen zum selben Segment. Es können aber auch zwei Sehnen einer Zeile x zum selben Segment gehören. Das ist der Fall, wenn das Segment nach oben oder unten geöffnet ist. Der Fall „nach unten geöffnet“ ist einfacher zu entscheiden, da ja bei der Verarbeitung von oben nach unten dann Überdeckungen von Sehnen vorliegen. Der Fall „nach oben geöffnet“ ist schwieriger: Hier kann erst in einem zweiten Durchlauf entschieden werden, welche Sehnen zusammen gehören.

In Bild 23.3 ist das Ergebnis der Vereinzelung von Bild 23.3-a abgebildet. Im verkleinerten Bild 23.2 treten, im Gegensatz zum Original, im rechten oberen Bereich der rechten Zange (Zeilen 35 und 36) zwei Bildpunkte auf, die schräg nach rechts/unten liegen. Da hier

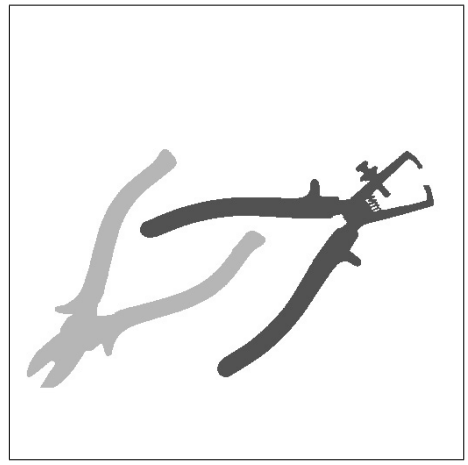


Bild 23.3: Ergebnis der Vereinzelung des Testbildes 23.1-a

4-Nachbarschaft verwendet wurde, wurden diese Bildpunkte als eigene Segmente codiert. Im Ausdruck des *run-length*-Codes sind aus diesem Grund in den Zeilen 35 und 36 zwei Segmente mit den Codes 2 und 3 aufgeführt. Die linke Zange hat den Code 1 und die rechte den Code 4.

Wenn die Segmentvereinzelung abgeschlossen ist, kann der *run-length*-Code bezogen auf die gefundenen Segmente effektiver gespeichert werden. Im Folgenden wird ein Vorschlag für eine *run-length*-Code-Struktur gegeben, die sich in der Praxis bewährt hat. Es wurden zusätzliche Informationen mit aufgenommen. Das erleichtert die spätere Weiterverarbeitung, geht aber zu Lasten der Datenkompression.

Der *run-length*-Code der Segmente wird bezüglich eines frei wählbaren Referenzpunktes (x_r, y_r) erzeugt. Dieser Referenzpunkt kann für alle in einem Bild auftretenden Segmente identisch sein, etwa die linke obere Bildecke mit den Koordinaten $(0, 0)$. Es besteht auch die Möglichkeit, für jedes Segment i einen eigenen Referenzpunkt (x_{r_i}, y_{r_i}) festzulegen, z.B. einen beliebigen Punkt innerhalb des Segments. Die Koordinatenwerte aller Referenzpunkte (x_{r_i}, y_{r_i}) werden dann bezüglich eines „globalen“ Referenzpunktes (x_r, y_r) (z.B. Position $(0, 0)$), angegeben. Als weitere globale Größen müssen noch Zeilen- und Spaltenanzahl des Bildes gespeichert werden. In folgender Tabelle ist die gesamte Datenstruktur wiedergegeben:

Globale Größen	
	Anzahl der Bildzeilen
	Anzahl der Bildspalten
	Referenzpunkt (x_r, y_r)
Lokale Größen des Segments i	
	Referenzpunkt (x_{r_i}, y_{r_i})
	Code des Segments
	Anzahl der Zeilen
	Zeilennummer bzgl. des Referenzpunktes
	Anzahl der Sehnen pro Zeile
	Sehnenanfang bzgl. des Referenzpunktes
	Sehnenende bzgl. des Referenzpunktes
	...
	Zeilennummer bzgl. des Referenzpunktes
	Anzahl der Sehnen pro Zeile
	Sehnenanfang bzgl. des Referenzpunktes
	Sehnenende bzgl. des Referenzpunktes
	...

In 23.4 ist ein einfaches Beispiel dazu gegeben. Das Bild ist $16 \cdot 16$ Pixel groß und enthält zwei Segmente. Als globaler Referenzpunkt (x_r, y_r) wird die linke obere Ecke $(0, 0)$ gewählt. Die folgenden Tabellen enthalten die Datenstruktur des dazugehörigen *run-length*-Codes.

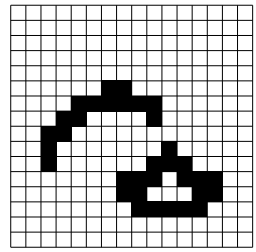


Bild 23.4: Beispiel zur *run-length*-Codierung. Das Bild ist $16 \cdot 16$ Pixel groß und enthält zwei Segmente. Als globaler Referenzpunkt (x_r, y_r) wird die linke obere Ecke $(0, 0)$ gewählt.

Globale Größen	
Anzahl der Bildzeilen	16
Anzahl der Bildspalten	16
Referenzpunkt	$(0, 0)$

Segment 1	
Referenzpunkt	$(8, 5)$
Code des Segments	1
Anzahl der Zeilen	6
Zeilennummer	-3
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	1
Sehnenende	2
Zeilennummer	-2
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	-1
Sehnenende	4
Zeilennummer	-1
Anzahl der Sehnen pro Zeile	2
Sehnenanfang	-2
Sehnenende	-1
Sehnenanfang	4
Sehnenende	4
Zeilennummer	0
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	-3
Sehnenende	-2
Zeilennummer	1
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	-3
Sehnenende	-3
Zeilennummer	2
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	-3
Sehnenende	-3

Segment 2	
Referenzpunkt	$(9, 7)$
Code des Segments	1
Anzahl der Zeilen	5
Zeilennummer	0
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	3
Sehnenende	3
Zeilennummer	1
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	2
Sehnenende	4
Zeilennummer	2
Anzahl der Sehnen pro Zeile	2
Sehnenanfang	0
Sehnenende	2
Sehnenanfang	4
Sehnenende	6
Zeilennummer	3
Anzahl der Sehnen pro Zeile	2
Sehnenanfang	0
Sehnenende	1
Sehnenanfang	5
Sehnenende	6
Zeilennummer	4
Anzahl der Sehnen pro Zeile	1
Sehnenanfang	1
Sehnenende	5

Wenn Segmente als *run-length*-Code gespeichert sind, ist es möglich, eine Reihe von segmentbeschreibenden Merkmalen zu berechnen, ohne zuvor in die ursprüngliche Rasterdarstellung zurückgehen zu müssen. Beispiele dazu werden in den folgenden Kapiteln gegeben.

Eine zur *run-length*-Codierung verwandte Technik ist die Codierung von Bildern als *quadtrees* oder *octrees* (Kapitel 19).

23.2.3 Effiziente Vereinzelung mit Union-Find-Algorithmen

Eine Methode zur Bildung von Zusammenhangskomponenten, die wegen ihrer Laufzeit- und Speicherplatzeffizienz eine immer größere Verbreitung findet, stellt die Klasse der Union-Find-Algorithmen dar. Bild 23.5 verdeutlicht das Zusammenspiel zwischen der Lauflängencodierung und dem Union-Find-Algorithmus: Aus den unzusammenhängenden Streifen der Lauflängencodierung werden Zusammenhangskomponenten gebildet, für die dann - wie im folgenden Kapitel beschrieben - Merkmale bestimmt werden können.

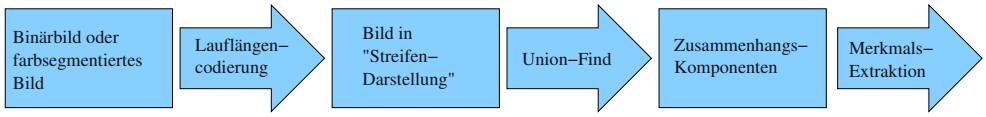


Bild 23.5: Zusammenspiel zwischen Lauflängencodierung und Union-Find-Algorithmus

Die unzusammenhängenden Streifen aus der Lauflängencodierung kann man sich nun als Graph denken, wobei die Streifen Knoten darstellen und zwischen zwei benachbarten, sich berührenden Streifen eine Kante besteht. Glücklicherweise stellt die Graphentheorie sehr einfache, effektive Algorithmen zur Verfügung, die Zusammenhangskomponenten sehr effizient – sowohl Speicherplatz als auch Rechenzeit betreffend – ermitteln können, die sogenannten Union-Find-Algorithmen. Der Name bezieht sich dabei auf die zwei Operationen, die für den Umgang mit Zusammenhangskomponenten erforderlich sind: `Union` für die Vereinigung zweier Zusammenhangskomponenten (beim dynamischen Erzeugen eines Graphen durch Einfügen einer Kante) sowie `Find` für die Abfrage, ob zwei Knoten des Graphen zur selben Zusammenhangskomponente gehören.

Als Datenstruktur zur Verwaltung der Zusammenhangskomponenten wird meist ein Wald von Bäumen verwendet, wobei die Bäume durch Rückwärtsverweise bis zur Wurzel realisiert werden (Bild 23.6). Die Wurzel eines Baumes ist dadurch gekennzeichnet, dass sie keinen weiterführenden Verweis enthält. Als Repräsentant für eine (durch einen Baum dargestellte) Zusammenhangskomponente wird die jeweilige Wurzel verwendet. Die Operation `Find` durchsucht iterativ die Kette der Vorgänger, bis ein Knoten ohne Vorgänger (der Repräsentant dieser Zusammenhangskomponente) gefunden wird:

```
Id Find ( Id x )          // findet den Repräsentanten der Zusammenhangs-
{                         // komponente, zu der x gehört
  Id rootX = x;           // Initialisierung, gleichzeitig Ergebnis
                          // bei einelementiger Komponente
  while hasParent(rootX) // iterativ rückwärts bis zur Wurzel laufen
    rootX = rootX.parent;
  return rootX;
}
```

Die Operation `Union` ermittelt die Repräsentanten der beiden Elemente, d.h. die Wurzeln der beiden Bäume. Sind die beiden Wurzeln nicht identisch, wird als Vorgänger der einen Wurzel die Wurzel der anderen Zusammenhangskomponente eingetragen:

```
Union ( Id x, Id y)          // vereinigt die Zusammenhangskomponenten,
{                            // denen x und y angehören
   Id rootX, rootY;
   rootX = Find(x);          // die beiden Wurzeln (Repräsentanten
   rootY = Find(y);          // der Komponenten) ermitteln...
   if ( rootX != rootY )
      rootX.parent = rootY; // ... und gemeinsamen Repräsentanten setzen
}
```

Die Grafiken in Bild 23.6 visualisieren die Verzeigerung in dieser Datenstruktur, die entsteht, wenn die lauflängencodierte Bildrepräsentation zu Zusammenhangskomponenten zusammengefügt wird, wobei jeweils nur die Streifen zweier aufeinanderfolgender Zeilen `prev` und `akt` betrachtet werden.

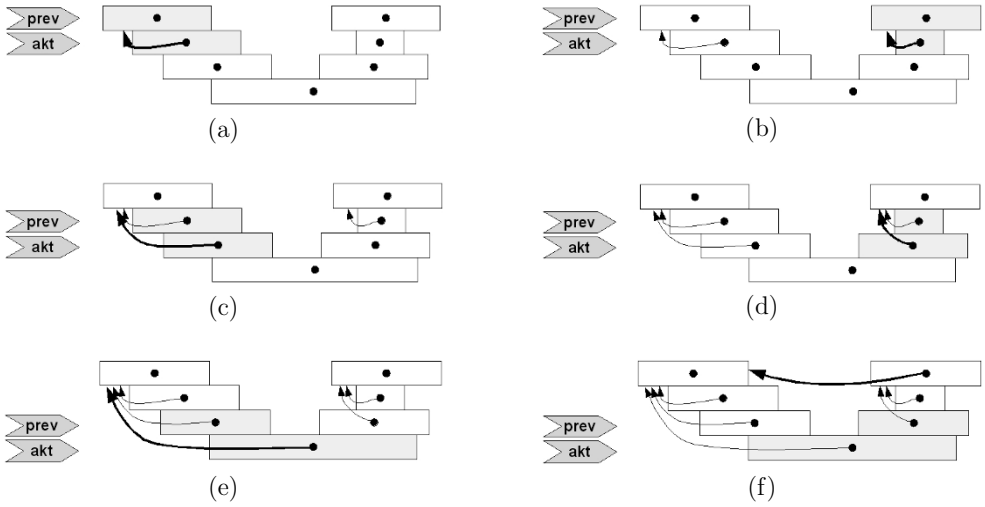


Bild 23.6: Visualisierung der Arbeitsweise des Union-Find-Algorithmus: Das Bild wird zeilenweise durchlaufen, wobei jeweils zwei benachbarte Zeilen (`akt` und `prev`) betrachtet werden. Berühren sich zwei Streifen, so werden diese (sowie ihre bereits bestehenden Zusammenhangskomponenten) zu einer Zusammenhangskomponente vereinigt.

Da bei obigem naiven Algorithmus entartete (z.B. links- oder rechtsgekämmte) Bäume entstehen können, werden in der Praxis meist Kombinationen der folgenden beiden Optimierungsverfahren verwendet:

Pfadkomprimierung (*path compression*): Wird die Wurzel eines Knotens x gesucht, kann dies bedeuten, dass ein mehr oder weniger langer Pfad über mehrere Vorgänger (s. gekämmter Baum) durchlaufen werden muss, bis die Wurzel gefunden wird. Ist die Wurzel gefunden, kann sie für den Knoten x als direkter Vorgänger gesetzt werden. Wird bei einem späteren Aufruf erneut die Wurzel von x gesucht, hat der Suchpfad die Länge 1. Der Zusatzaufwand für die Pfadkompression (z.B. das zusätzliche Statement `x.parent = rootX;` am Ende der Funktion `getRoot`) ist minimal. Manche Implementierungen machen dies für alle Knoten des durchlaufenen Pfades, was von Anwendungsfall zu Anwendungsfall aber auch einen Mehraufwand bedeuten kann.

Gewichtsausgleich (*weight balancing*): Man kann das Entstehen entarteter Bäume einfach dadurch verhindern, dass die Baumtiefe in der jeweiligen Baumwurzel mitcodiert wird und bei der Vereinigung zweier Bäume der jeweils flachere an die Wurzel des tieferen Baums gehängt wird, statt diesen Vorgang – wie in der naiven Darstellung oben – ohne Betrachtung der Baumtiefe durchzuführen.

Tarjan [Tarj75] hat nachgewiesen, dass durch diese Optimierungsverfahren die Baumtiefe für alle in der Realität denkbaren Kantenanzahlen nie größer als 4 wird, sodass die Zeit für die Operationen `union` und `find` nahezu konstant ist.

Zusammenfassend sei erwähnt,

- dass für die Repräsentation der Zusammenhangskomponenten lediglich ein Array mit den Indizes der Streifen (oder alternativ dazu je ein zusätzlicher Zeiger pro Streifen) erforderlich ist (Speicherplatzeffizienz),
- dass die Anzahl der für die Erzeugung der Zusammenhangskomponenten (`Union` ist nahezu konstant) erforderlichen Operationen nahezu linear in der Anzahl der Kanten (Nachbarschaftsbeziehungen zwischen Streifen) sind, sowie
- dass die Anzahl der für die Anfrage, ob zwei Streifen zur selben Zusammenhangskomponente gehören (`Find`), erforderlichen Operationen nahezu konstant ist.

Kapitel 24

Einfache segmentbeschreibende Parameter

24.1 Anwendungen

In diesem Kapitel werden einfache Verfahren angegeben, mit denen Segmente, die in einer Segmentierungsstufe erzeugt wurden, näher beschrieben werden. Dazu werden Parameter berechnet, die die Segmente charakterisieren. In dieser Verarbeitungsstufe findet in Verbindung mit der kompakten Speicherung von Segmenten (z.B. Kapitel 23) der Übergang von der pixelorientierten Verarbeitung zur listenorientierten Verarbeitung statt. Damit ist gemeint, dass die Segmente zusammen mit den sie beschreibenden Größen als Datenstrukturen gespeichert werden, die algorithmisch gut zu handhaben sind.

Häufig müssen die Segmente nach diesem Verarbeitungsschritt daraufhin untersucht werden, ob sie einem bestimmten vorgegebenen oder gesuchten Typ entsprechen. Man kann diese Problemstellung als eine Klassifizierung von Segmenten auffassen. Dabei wird man nach dem „Prinzip des minimalen Aufwands“ vorgehen: Wenn z.B. zwei Segmente allein durch ihren Flächeninhalt zu unterscheiden und damit entsprechenden Klassen zuzuordnen sind, so ist es nicht notwendig, weitere Parameter zu berechnen. Hier wird deshalb oft von einer sequentiellen Segmentklassifizierung gesprochen, bei der der Reihe nach bestimmte Segmentparameter abgeprüft werden. Anhand ihrer Werte werden die Segmente den entsprechenden Klassen zugeordnet.

Bei einer parallelen Segmentklassifizierung werden zu den einzelnen Segmenten n-dimensionale Merkmalsvektoren berechnet, die dann mit einem Klassifikator weiterverarbeitet werden. Dazu können z.B. alle Verfahren verwendet werden, die in den Kapiteln 6, 7 und 15 erläutert wurden.

Zur Thematik der Segmentbeschreibung werden neben der Berechnung von einfachen Eigenschaften und der auf der Basis dieser Eigenschaften durchgeführten Klassifizierung auch komplexere Verfahren untersucht. Eine wichtige Zielsetzung dieser Verfahren ist die translations-, rotations- und größeninvariante Segmenterkennung. Beispiele dazu werden in diesem Kapitel und in den Kapiteln 7, 25 und 26 gegeben.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_24

Am Ende einer derartigen Verarbeitung kann beispielsweise eine Aussage der folgenden Form stehen: „Das Segment ist die Abbildung eines Bauteils vom Typ X, das allerdings eine Fehlerstelle aufweist.“ Es kann sich dann eine geeignete Reaktion anschließen, z.B. das Markieren oder Aussteuern des Bauteils.

Im Folgenden wird vorausgesetzt, dass die Segmente vereinzelt, d.h. mit unterschiedlichen Grauwerten codiert, sind. Es besteht auch die Möglichkeit, dass sie *run-length*-codiert vorliegen und die Segmentvereinzelung gemäß Abschnitt 23.2 durchgeführt wurde.

24.2 Flächeninhalt

Der Flächeninhalt eines Segments ist einfach zu berechnen: Es müssen nur für jedes Segment die Pixel aufsummiert werden, die zum Segment gehören. Der Flächeninhalt wird in der Maßeinheit *Pixel* angegeben. Ist das Segment *run-length*-codiert (Kapitel 10), so werden alle Sehnenlängen addiert, die den Sehnencode des Segments haben. Der Algorithmus ist zusammen mit der Berechnung der Schwerpunkte der Segmente in Abschnitt 24.3 dargestellt.

24.3 Flächenschwerpunkt

Auch der Flächenschwerpunkt eines Segments ist einfach zu berechnen. Der Schwerpunkt (x_{cp}, y_{cp}) eines „Systems materieller Punkte“ $M_j(x_j, y_j)$ mit den Massen m_j berechnet sich nach der Formel:

$$x_{cp} = \frac{\sum\limits_j m_j \cdot x_j}{\sum\limits_j m_j}, \quad y_{cp} = \frac{\sum\limits_j m_j \cdot y_j}{\sum\limits_j m_j}. \tag{24.1}$$

Übertragen auf die vorliegende Problemstellung sind die Punkte (x_{ij}, y_{ij}) die Pixel, die zum Segment mit der Nummer i gehören. Die „Massen“ der Bildpunkte sind alle gleich eins, so dass im Nenner der Formeln (24.1) der Flächeninhalt des Segments i auftritt.

Die Segmentflächen und die Segmentschwerpunkte werden in Abschnitt 24.9 noch in einem anderen Zusammenhang, nämlich bei der Berechnung von Momenten, benötigt.

Im Folgenden ist der Algorithmus zur Berechnung der Segmentflächen und der Schwerpunkte angegeben, falls die Segmente *run-length*-codiert sind. Dieser Algorithmus kann auch als Vorlage für die in den weiteren Abschnitten nicht explizit angegebenen Algorithmen angesehen werden.

A24.1: Berechnung der Flächen und der Schwerpunkte.

Voraussetzungen und Bemerkungen:

◇ $\mathbf{S} = (s(x, y))$ segmentiertes Bild, das *run-length*-codiert ist. Der *run-length*-Code sei im Feld rl abgelegt. Die Struktur des *run-length*-Codes entspricht der Datenstruktur von Kapitel 10.

◇ Im Feld `Code der Sehne` ist die jeweilige Segmentnummer eingetragen.

◇ In $area[i]$ werden die Flächen der Segmente i gespeichert. $h1[i]$ und $h2[i]$ sind zwei Hilfsfelder.

◇ In $x_{cp}[i]$ und $y_{cp}[i]$ werden die Zeilen- und Spaltenkoordinaten des Schwerpunkts des Segments i gespeichert.

Algorithmus:

(a) Für alle Segmente i: $area[i] = 0; h1[i] = 0; h2[i] = 0;$

(b) akt_zeile = erste Zeile des RL-Codes;

(c) Solange akt_zeile nicht die letzte Zeile des RL-Codes ist:

(ca) Falls akt_zeile eine Leerzeile ist: akt_zeile = nächste Zeile;

(cb) Sonst:

(cba) Für alle Sehnen akt_sehne von akt_zeile:

$k = akt\_end - akt\_anf + 1;$

$area[akt\_code] = area[akt\_code] + k;$

$h1[akt\_code] = h1[akt\_code] + (akt\_anf + akt\_end) \cdot k;$

$h2[akt\_code] = h2[akt\_code] + akt\_zeile \cdot k;$

(d) Für alle Segmente i:

$x_{cp}[i] = h2[i]/area[i];$

$y_{cp}[i] = h1[i]/(2 \cdot area[i]);$

Ende des Algorithmus

Wenn die Koordinaten des Segmentschwerpunkts ermittelt sind, kann man als zusätzliches Merkmal verwenden, ob der Schwerpunkt innerhalb oder außerhalb des Segments liegt. Dieses Merkmal ist in der Praxis oft gut zur Unterscheidung von Segmenten geeignet. Ist von einem Segment bekannt, dass der Schwerpunkt immer innerhalb liegt, so können bei einer Klassifizierung alle Segmente, bei denen der Schwerpunkt außerhalb liegt, ignoriert werden.

24.4 Umfang

Bei vielen Anwendungen wird auch die Länge des Randes eines Segments benötigt. Soll für das Segment i dieser Wert ermittelt werden, so werden alle mit dem Wert i codierten Bildpunkte danach untersucht, ob die 4- (oder 8-) Nachbarn ebenfalls denselben Code besitzen. Bildpunkte, für die das nicht zutrifft, sind Randpunkte, die aufsummiert werden. Bei dieser Berechnung kann auch der Rand des Segments als Datenstruktur, z.B. in der Form eines *run-length*-Codes, codiert werden. Weitere Verfahren zur Randverarbeitung werden in Abschnitt 24.8 gegeben.

24.5 Kompaktheit

Aus dem Flächeninhalt und der Länge des Umfangs eines Segments lässt sich eine Maßzahl berechnen, die als *Kompaktheit* bezeichnet wird. Als Maßzahl verwendet man

$$V = \frac{Umfang^2}{Fläche}. \tag{24.2}$$

In der folgenden Tabelle ist für einige einfache geometrische Formen die Kompaktheit angegeben:

Form	Umfang	Fläche	Kompaktheit
Kreis	2π r	$r^2\pi$	$4\pi = 12.57$
Quadrat	$4a$	a^2	16
Rechteck	$2a + 2b$	ab	$8 + \frac{4a}{b} + \frac{4b}{a}$
gleichseitiges Dreieck	$3a$	$\frac{a^2}{4}\sqrt{3}$	20.78

Die kompakteste Form ist der Kreis. Man sieht, dass die Maßzahl für die Kompaktheit desto mehr vom Idealwert $V = 4\pi$ abweicht, je deutlicher sich die Form von einer Kreisform unterscheidet. Betrachtet man z.B. das Rechteck: Falls für die Seitenlängen $b = a$ gilt, ergibt sich ein Quadrat und damit der Wert $V = 16$. Bei $b = 2a$ erhält man $V = 18$ und bei $b = 4a$ ist $V = 25$, d.h. je länglicher eine Fläche ist, desto größer ist der Wert von V. Man kann somit dieses Maß verwenden, um längliche Segmente von nicht länglichen Segmenten zu unterscheiden. Eine weitere Methode dazu wird in Abschnitt 24.6 gegeben.

Praktische Anwendungen haben gezeigt, dass die Signifikanz dieses Maßes, vor allem bei Segmenten mit kleinem Flächeninhalt und in unterschiedlichen Drehlagen, nicht immer zufriedenstellend ist.

24.6 Orientierung

Zur Lagebestimmung von Segmenten wird häufig auch die *Orientierung* eines Segments benötigt. Im Folgenden wird ein einfaches Verfahren dargestellt, das bei deutlich länglichen Segmenten gut die beiden Hauptrichtungen ermittelt. Es beruht auf der Annäherung

eines Segments durch eine Ellipse, deren große und kleine Hauptachse die Orientierung des Segments bestimmen. Bei nicht länglichen Segmenten, z.B. bei nahezu kreisförmigen oder quadratischen Segmenten, liefert das Verfahren zwar auch Orientierungswerte, die aber meistens ohne Signifikanz sind.

Es wird ein bestimmtes Segment betrachtet, das aus der Menge der diskreten Punkte $\{(x, y)\}$ besteht. Alle Bildpunkte des Segments besitzen als Code den gleichen Grauwert g, für den ohne Beschränkung der Allgemeinheit $g = 1$ angenommen werden kann.

Zur Berechnung der Orientierung des Segments wird als Erstes die Kovarianzmatrix C des Segments berechnet:

$$C = \begin{pmatrix} v_{xx} & v_{xy} \\ v_{yx} & v_{yy} \end{pmatrix}, \tag{24.3}$$

wobei

$$v_{xx} = \frac{\sum\limits_x \sum\limits_y (x - x_{cp})^2}{A}, \quad v_{yy} = \frac{\sum\limits_x \sum\limits_y (y - y_{cp})^2}{A}, \tag{24.4}$$

$$v_{xy} = v_{yx} = \frac{\sum\limits_x \sum\limits_y (x - x_{cp})(y - y_{cp})}{A}.$$

Die Größe A in (24.4) ist die Fläche des Segments. x_{cp} und y_{cp} sind die Koordinaten des Schwerpunkts. Im nächsten Schritt werden die Eigenwerte λ_1 und λ_2 der Kovarianzmatrix C berechnet:

$$\lambda_1 = \frac{(v_{xx} + v_{yy}) + \sqrt{(v_{xx} - v_{yy})^2 + 4 v_{xy} v_{yx}}}{2}, \tag{24.5}$$

$$\lambda_2 = \frac{(v_{xx} + v_{yy}) - \sqrt{(v_{xx} - v_{yy})^2 + 4 v_{xy} v_{yx}}}{2}.$$

Die Orientierung des Segments lässt sich jetzt über die zu den Eigenwerten λ_1 und λ_2 gehörigen Eigenvektoren $\vec{ev}_1$ und $\vec{ev}_2$ berechnen. Die beiden Eigenvektoren werden auch als die beiden *Hauptrichtungen des Segments* bezeichnet. Die Orientierung wird mit Hilfe von λ_1, dem größeren der beiden Eigenwerte, berechnet. Die Berechnung über λ_2 würde einen dazu senkrechten Winkel ergeben.

Wegen der Beziehung

$$\begin{pmatrix} v_{xx} - \lambda_1 & v_{xy} \\ v_{yx} & v_{yy} - \lambda_1 \end{pmatrix} \cdot \begin{pmatrix} ev_{1x} \\ ev_{1y} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{24.6}$$

gilt zwischen den Komponenten des Eigenvektors folgende Gleichung:

$$ev_{1y} = \frac{\lambda_1 - v_{xx}}{v_{xy}} \cdot ev_{1x}. \tag{24.7}$$

Die Orientierung erhält man mit

$$\varphi = \arctan \frac{\lambda_1 - v_{xx}}{v_{xy}}. \tag{24.8}$$

Bei der numerischen Berechnung ist zu beachten, dass in der Nähe von $\varphi = \pm 90°$ der Nenner $v_{xy} = 0$ oder zumindest sehr klein sein kann. Diese Fälle sind gesondert zu behandeln, um Divisionen durch null zu vermeiden.

Ein weiteres segmentbeschreibendes Merkmal lässt sich aus den Eigenwerten ableiten: λ_1 und λ_2 geben die Ausdehnung des Segments entlang der 1. und 2. Hauptrichtung an. Der Quotient $\frac{\lambda_1}{\lambda_2}$ kann als Maß dafür verwendet werden, ob das Segment länglich ist oder nicht. Bei einem länglichen Segment ist λ_1 deutlich größer als λ_2, deswegen ist der Quotient deutlich größer als eins. Bei nicht länglichen Segmenten ist der Quotient ungefähr gleich eins.

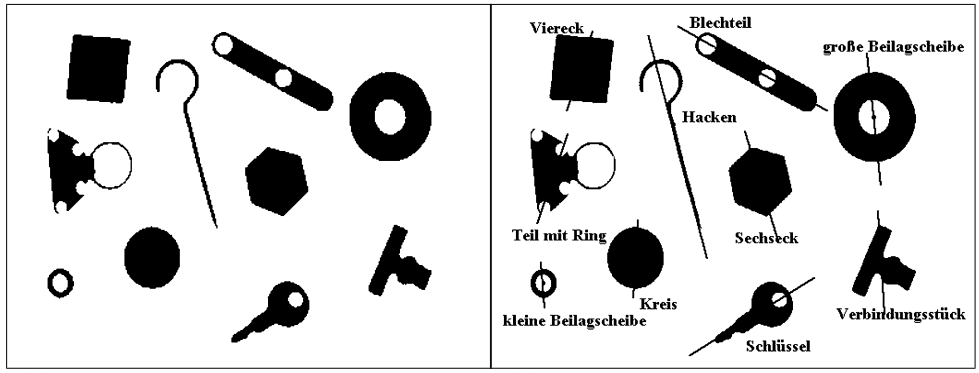
Bild 24.1 ist ein Beispiel für ein *run-length*-codiertes Binärbild mit verschiedenen Segmenten. Die Parameter sind in folgender Tabelle zusammengefasst:

Bezeichnung	x_{cp}	y_{cp}	Fläche	λ_1	λ_2	$\frac{\lambda_1}{\lambda_2}$	φ
Viereck	64	92	4500	419.4	334.8	1.25	168°
Blechteil	69	285	3252	1639.0	63.2	25.93	61°
große Beilagscheibe	117	409	5971	685.4	575.3	1.19	4°
Teil mit Ring	174	65	3162	425.9	404.4	1.05	159°
Hacken	125	192	1195	3299.5	109.1	30.24	14°
Sechseck	186	286	3751	321.1	281.6	1.14	10°
kleine Beilagscheibe	298	51	497	83.3	72.3	1.15	4°
Kreisscheibe	270	151	3281	283.8	240.6	1.18	178°
Schlüssel	329	287	2267	532.0	123.4	4.31	125°
Verbindungsstück	279	418	2397	325.6	266.6	1.22	2°

24.7 Fourier-Transformation der Randpunkte

Um ein Segment bezüglich der Lage und Orientierung zu normieren, kann man die Randpunkte des Segments Fourier-transformieren. Die Grundlagen zur Fourier-Transformation sind in Kapitel 8 zusammengestellt. Um die Randpunkte Fourier-transformieren zu können, werden ihre (x, y)-Koordinaten als komplexe Zahlen aufgefasst: Die x-Werte sind der Realteil und die y-Werte der Imaginärteil. Einem Randpunkt mit den Koordinaten (x, y) ist durch diese Vorschrift die komplexe Zahl $x + iy$ zugeordnet. Wenn man die Fourier-Transformation auf diese Punkte anwendet, erhält man die Fourier-Komponenten der Randpunkte. Da die Fourier-Transformation umkehrbar ist, gehen bei dieser Transformation keine Informationen verloren.

Falls der Rand des Segments eine geschlossene Kurve ist und die Randpunkte im Uhrzeigersinn ermittelt wurden, wird die erste Fourier-Komponente die betragsmäßig größte



(a) (b)

Bild 24.1: (a) Originalbild mit zehn Teilen. (b) *run-length*-codiertes Bild. Die Schwerpunkte und die Orientierungen sind eingezeichnet. Die weiteren Parameterwerte sind in der Tabelle zusammengestellt. Bei nicht länglichen Segmenten sind die berechneten Orientierungen meistens wenig bedeutsam. Die Drehwinkel in der zugehörigen Tabelle sind, ausgehend von der nach unten gehenden x-Achse, im mathematisch positiven Sinn zwischen 0° und 180° angegeben.

sein. Eine Größennormierung kann jetzt durch die Division aller Fourier-Komponenten durch den Betrag der ersten Komponente erhalten werden.

Einer Drehung der Randpunkte im Ortsbereich entspricht eine Multiplikation der Fourier-Komponenten mit dem Wert $e^{i\varphi}$, wobei φ der Drehwinkel ist. Diese Operation ändert somit den Betrag der Fourier-Komponenten nicht.

24.8 Chain-Codierung

Neben der Länge des Randes eines Segments (Abschnitt 24.4) ist oft auch der gesamte Umriss, die *Kontur*, von Interesse. Die *chain*-Codierung bietet eine Möglichkeit, zu einem Segment eine Datenstruktur zu erzeugen, mit der die Kontur kompakt gespeichert ist. Außerdem ist die Möglichkeit weiterer Verarbeitungsschritte gegeben.

Das Grundprinzip besteht darin, dass man ausgehend von einem Randpunkt des Segments als Startpunkt im Uhrzeigersinn entlang der Kontur wandert. Das Verfahren wird beendet, wenn man wieder am Startpunkt angelangt ist oder wenn kein neuer Randpunkt gefunden werden kann (z.B. wenn der Bildrand erreicht wird oder eine Linie endet).

Geht man vom Startpunkt aus, so kann der nächste Randpunkt nur in einer Richtung von $a \cdot 45°$ liegen, mit $a = 0, 1, ..., 7$. Diese Werte werden als *Richtungscodes* bezeichnet. Üblicherweise sind sie wie in Bild 24.2 dargestellt festgelegt.

Bild 24.3 zeigt ein einfaches Beispiel für ein Segment. Der *chain*-Code der äußeren

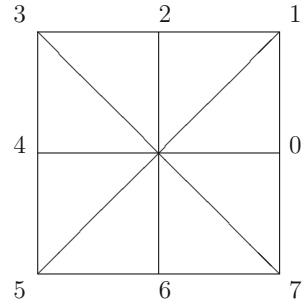


Bild 24.2: Richtungscodes der 8-Nachbarn.

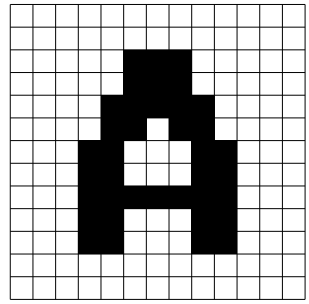


Bild 24.3: Beispiel für ein einfaches Segment.

Kontur dieses Segments sieht folgendermaßen aus:

040500200500676766664234456422221212

Diese Ziffernfolge muss noch etwas erläutert werden: Die Richtungscodekombination 04 wird als *Signalcode* verwendet, der anzeigt, dass danach „Sonderinformation“ kommt. Eine tatsächlich auftretende Richtungscodekombination 04 wird dann mit 0404 codiert. In diesem Beispiel folgt ein weiterer Code 05, der anzeigt, dass anschließend die Koordinaten $(x, y) = (002, 005)$ des Startpunkts folgen. Die Richtungscodes der Kontur beginnen dann mit 00676...

Das Segment von Bild 24.3 enthält aber noch eine Aussparung innerhalb des Segments. Es wäre somit durch obigen *chain*-Code nicht zufriedenstellend wiedergegeben. Dieses Problem kann man wie folgt lösen: Es wird der Rand der Aussparung codiert und die beiden

Startpunkte der beiden Konturen werden durch eine „virtuelle Kette“ verbunden (Signalcode 0421), die nur zur strukturellen Beschreibung des Segments dient. Damit könnte der *chain*-Code wie folgt aussehen:

040500200500676766664234456422221212	äußere Kontur
0421040500200576	virtuelle Kette
04050040067765443211	Kontur der Aussparung

Der *chain*-Code eignet sich nicht nur zur Codierung der Kontur eines Segments, sondern erlaubt auch die Berechnung von segmentbeschreibenden Parametern, z.B. die Länge der Kontur:

$$A = \Delta l(n_g + n_u\sqrt{2}). \tag{24.9}$$

Hier ist n_g die Anzahl der geraden und n_u die Anzahl der ungeraden Richtungscodes der Kette. Δl ist der Abstand der Gitterpunkte, die Länge der Kontur kann dann in der Einheit von Δl, z.B. in *mm* oder *cm*, angegeben werden.

Auch die vertikale und die horizontale Ausdehnung des Segments kann leicht aus der Kette ermittelt werden. Dazu ordnet man den Richtungscodes Richtungsvektoren zu, gemäß folgender Tabelle:

Richtungscode	x-Komponente	y-Komponente
0	0	1
1	-1	1
2	-1	0
3	-1	-1
4	0	-1
5	1	-1
6	1	0
7	1	1

Um die Ausdehnung in x- und y-Richtung zu berechnen, werden die zu den Richtungscodes gehörigen Komponenten der Richtungsvektoren getrennt für beide Richtungen akkumuliert und dabei das Minimum und das Maximum ermittelt.

24.9 Momente

In diesem Abschnitt wird eine Methode beschrieben, die es erlaubt, Segmente translations-, rotations- und größeninvariant zu beschreiben. Es wird ein Segment betrachtet, das aus der Menge der diskreten Punkte $\{(x, y)\}$ besteht. Hier muss das Segment nicht, wie in den vorhergehenden Abschnitten, einen homogenen Grauwert als Klassencode besitzen. Vielmehr wird angenommen, dass ein Segment die Grauwerte $s(x, y)$ besitzt. Ein zulässiger Sonderfall wäre, wenn alle $s(x, y)$ gleich sind (z.B. $s(x, y) = 1$).

Eine dazu passende Aufgabenstellung könnte etwa sein:

- Mit den Methoden der Klassifizierung wurde ein Bild segmentiert. Den verschiedenen Klassen wurden unterschiedliche Codes zugeordnet. Es kann natürlich vorkommen, dass verschiedene Segmente denselben Code besitzen, wenn ihre Punkte zur selben Klasse gehören.
- Die Segmente werden jetzt mit dem Original verknüpft, so dass für jedes Segment die Grauwerte (die Merkmalswerte) in einem bestimmten Kanal zur Verfügung stehen.
- Jetzt soll untersucht werden, ob zwei Segmente derselben Klasse ähnlich sind, also sich nur hinsichtlich der Position im Bild, der Skalierung (Größe) und der Drehlage unterscheiden.

Die *Momente der Ordnung* $p+q$ sind definiert durch:

$$m_{pq} = \sum_x \sum_y x^p y^q s(x,y) \tag{24.10}$$

In dieser Form hängen die Momente von der Position des Segments im Bild ab, so dass die oben angesprochene Translationsinvarianz nicht gegeben ist.

Aus diesem Grund definiert man die *zentrierten Momente der Ordnung* $p+q$:

$$\mu_{pq} = \sum_x \sum_y (x - x_{cp})^p (y - y_{cp})^q s(x,y). \tag{24.11}$$

Dabei sind x_{cp} und y_{cp} die Koordinaten des Schwerpunkts des Segments:

$$x_{cp} = \frac{\sum\limits_x \sum\limits_y x s(x,y)}{\sum\limits_x \sum\limits_y s(x,y)} = \frac{m_{10}}{m_{00}}, \tag{24.12}$$

$$y_{cp} = \frac{\sum\limits_x \sum\limits_y y s(x,y)}{\sum\limits_x \sum\limits_y s(x,y)} = \frac{m_{01}}{m_{00}}. \tag{24.13}$$

Falls das Segment homogen ist ($s(x,y) = 1$), tritt im Nenner von (24.12) und (24.13) die Fläche A des Segments auf. Nach der Koordinatentransformation gemäß

$$x' = x - x_{cp}, \qquad y' = y - y_{cp}, \tag{24.14}$$

liegt der Koordinatenursprung im Schwerpunkt, und es gilt: $\mu_{10} = \mu_{01} = 0$.

Nun zur Größennormierung: Die *normierten, zentrierten Momente der Ordnung* $p+q$ sind definiert durch:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}+1}} \tag{24.15}$$

Als Beispiel ist

$$\eta_{20} = \frac{\mu_{20}}{\mu_{00}^2} = \frac{\sum\limits_x \sum\limits_y (x - x_{cp})^2 s(x,y)}{(\sum\limits_x \sum\limits_y s(x,y))^2}, \tag{24.16}$$

und falls $s(x, y) = 1$ gilt, steht im Nenner das Quadrat A^2 der Fläche des Segments.

Die Momente η_{pq} sind invariant gegenüber Größenänderungen der Art $x' = ax, y' = ay$ des Segments, d.h. wenn in einem Bild an zwei unterschiedlichen Positionen zwei ähnliche Segmente auftreten, werden sich, abgesehen von Quantisierungsfehlern, gleiche Werte für η_{pq} ergeben.

Als Letztes wird die Rotationsinvarianz untersucht. Man erhält rotationsinvariante Momente, wenn das Koordinatensystem so gedreht wird, dass es mit den Hauptachsen des Segments übereinstimmt. Der Drehwinkel kann aus den Momenten berechnet werden:

$$\varphi = \frac{1}{2} \arctan \frac{2\eta_{11}}{\eta_{20} - \eta_{02}}. \tag{24.17}$$

Die Koordinatentransformation hat dann folgende Form:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{24.18}$$

Aufbauend auf den Momenten wurden in [Hu62] sieben invariante Merkmale definiert, die im Folgenden dargestellt sind. Sie werden aus den normierten, zentrierten Momenten der Ordnung $p + q$ berechnet. Eine Transformation gemäß (24.17) ist nicht notwendig:

$$\begin{aligned}
c_1 &= \eta_{20} + \eta_{02} \\
c_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
c_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
c_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
c_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
c_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad +4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
c_7 &= (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{24.19}$$

Da diese Merkmale sehr unterschiedliche Dynamikbereiche besitzen, wurden sie in den folgenden Bildbeispielen durch Logarithmierung normiert:

$$c_i \Leftarrow -lnc_i. \tag{24.20}$$

Im Folgenden werden einige Beispiele zu diesen sieben invarianten Parametern gegeben. Bild 24.4 zeigt Segmente, die sich durch die Position im Bild, die Größe und die Drehlage unterscheiden.

In der folgenden Tabelle sind für jedes Segment die sieben Parameter zusammengestellt. Man sieht, dass die Werte ungefähr gleich sind. Die Abweichungen sind durch die Fehler der diskreten Verarbeitung zu erklären. Zur weiteren Verarbeitung dieser Merkmalsvektoren können Klassifikatoren oder neuronale Netze verwendet werden. So könnte man z.B. ableiten, dass es sich hier immer um dasselbe Objekt in unterschiedlichen Lagen handelt.

Parameter	Original	andere Position	andere Größe	gedreht 5°	gedreht 123°
c_1	6.042	6.042	6.045	6.046	6.025
c_2	12.981	12.981	12.985	12.991	12.953
c_3	21.437	21.437	21.623	21.646	21.594
c_4	21.968	21.968	21.962	21.982	21.887
c_5	43.674	43.674	43.763	43.806	43.639
c_6	28.492	28.492	28.481	28.640	33.619
c_7	43.720	43.720	43.774	44.914	44.007

In Bild 24.5 sind sieben verschieden geformte Segmente abgebildet. Die folgende Tabelle enthält die Werte der c_i-Parameter. Zu einigen Parametern konnte der Logarithmus nicht berechnet werden, da der c_i-Wert nahezu null war. Diese Werte sind mit -1 markiert. Man sieht bei diesem Beispiel, dass sich durchaus unterschiedliche Werte ergeben und mit einer geeigneten Auswahl von Parametern die Segmente leicht unterschieden werden können.

Parameter	Niere	Kreis	Sichel	Blatt	Eck	Dreieck	Viereck
c_1	6.726	6.878	5.950	6.961	6.705	6.708	6.640
c_2	14.546	15.662	12.528	18.801	14.843	16.745	14.148
c_3	23.063	28.934	18.712	26.586	21.677	20.604	28.908
c_4	24.714	30.316	20.003	29.078	24.622	25.368	28.566
c_5	48.642	-1.000	39.854	-1.000	47.799	49.431	57.305
c_6	33.727	38.210	26.772	-1.000	32.165	34.886	38.063
c_7	48.984	-1.000	44.073	-1.000	49.056	48.581	57.142

Bild 24.6 behandelt eine weitere Problemstellung: Es sind von zwei verschiedenen Zangentypen jeweils vier Zangen in unterschiedlichen Drehlagen und Größen angeordnet. Die dazugehörige Tabelle zeigt, dass sich hier auf den ersten Blick keine deutlichen und verwendbaren Unterschiede ergeben. Sieht man sich jedoch die Parameter c_3 und c_4 genauer an, so ist zu erkennen, dass auch hier eine Klassifizierung in die Klassen „Typ 1“und „Typ 2“ möglich ist.

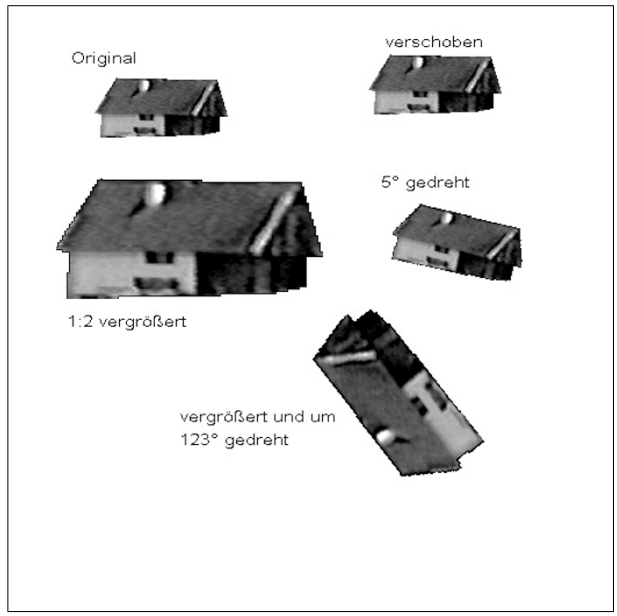


Bild 24.4: Beispiel für Segmente, die sich durch die Position im Bild, die Größe und die Drehlage unterscheiden. Die aus den Momenten abgeleiteten sieben Parameter sind ungefähr gleich (siehe Tabelle).

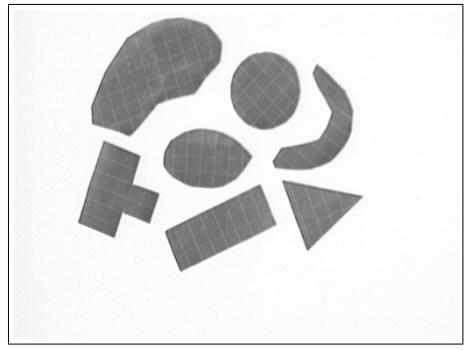


Bild 24.5: Beispiel für unterschiedlich geformte Segmente. Mit einer geeigneten Parameterauswahl können die Segmente leicht unterschieden werden (siehe Tabelle).

Bild 24.6: Das Bild enthält zwei Zangen in jeweils vier unterschiedlichen Drehlagen und Größen. Die Tabelle der c_i-Werte zeigt hier keine so deutlichen Unterschiede. Die Parameter c_3 und c_4 machen in diesem Beispiel aber eine Trennung der Klassen möglich.

Parameter	Typ 1				Typ 2			
c_1	4.654	4.736	4.703	4.761	4.798	4.706	4.747	4.869
c_2	11.175	11.285	11.263	11.339	11.103	10.913	11.025	11.255
c_3	14.870	15.147	15.036	15.255	15.613	15.372	15.456	15.867
c_4	17.070	17.203	17.134	17.279	18.600	17.993	18.330	18.515
c_5	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
c_6	25.105	23.755	24.261	-1.000	24.933	-1.000	26.213	24.661
c_7	-1.000	34.056	35.519	-1.000	-1.000	-1.000	36.899	-1.000

Die Momente wurden in der obigen Darstellung flächenhaft anhand aller Punkte eines Segments berechnet. Soll bei einer Anwendung nur der Rand des Segments verarbeitet werden, so können Momente auch nur für die Randpunkte berechnet werden. Dazu führt man folgende Schritte durch:

- Man berechnet den Schwerpunkt des Segments.
- Für jeden Punkt i der Kontur berechnet man den Abstand d_i zum Schwerpunkt.

- Man berechnet die relativen Häufigkeiten der Abstände d_i. Dazu müssen die Distanzen mit einer passenden Einheit (z.B. *Pixel*) diskretisiert werden. Angenommen, es gibt k verschiedene Distanzen, die mit a_1 bis a_k bezeichnet werden. $p(a_j)$ ist die relative Häufigkeit der Distanz a_j.

- Die zentrierten Momente $q - ter$ Ordnung sind dann:

$$\mu_p = \sum_{j=1}^{k} (a_j - m)^q p(a_j) \tag{24.21}$$

mit

$$m = \sum_{j=1}^{k} a_j p(a_j) \tag{24.22}$$

Die Größe m ist die mittlere Länge der Distanzen und μ_2 ein Schätzwert für die Streuung. Mit den so berechneten Momenten kann sinngemäß wie bei der flächenhaften Betrachtung vorgegangen werden.

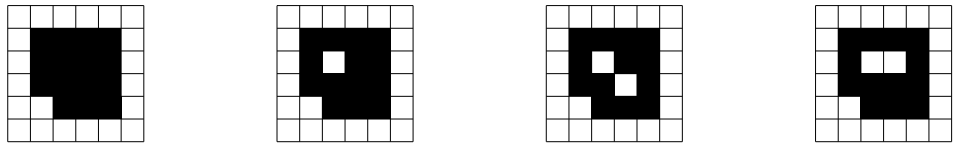
24.10 Euler'sche Charakteristik

Ein wichtiges Merkmal zur Beschreibung und Unterscheidung von Segmenten ist die Anzahl der „Löcher", die das Segment besitzt. Als „Loch" bezeichnet man eine Ansammlung von zusammenhängenden Bildpunkten (4-Nachbarschaft), die nicht zum Segment gehören, aber ganz von Bildpunkten des Segments umgeben werden. Zum Zählen der Löcher kann die Euler'sche Charakteristik verwendet werden.

In Abschnitt 7.9 wurde die Euler'sche Charakteristik $E = e - k + f$ in einem anderen Zusammenhang vorgestellt. In dieser Formel ist e die Anzahl der Ecken, k die Anzahl der Kanten und f die Anzahl der Flächen. Ecken oder Kanten, die zu mehreren Bildpunkten des Segments gehören, werden nur einmal gezählt. Für ein Polygonnetz gilt $E = 1$. Betrachtet man die Segmente in Bild 24.7, erhält man für das Segment in Bild 24.7-a $E = 1$, für Bild 24.7-b $E = 0$, für Bild 24.7-c $E = -1$ und für Bild 24.7-d $E = 0$. Die Anzahl L der Löcher ist dann

$$L = 1 - E. \tag{24.23}$$

Bei einer praktischen Anwendung kann man mit diesem Verfahren die Anzahl der Löcher der Segmente bestimmen. An einer unrichtigen Löcheranzahl könnte man z.B. fehlerhafte Teile erkennen. Eine andere Möglichkeit wäre, dass man nur die Segmente, die eine bestimmte Anzahl von Löchern haben, weiter verarbeitet und evtl. aufwändigere Verfahren darauf anwendet.



(a) (b) (c) (d)

Bild 24.7: Beispiele zur Euler'schen Charakteristik zum Zählen der Löcher eines Segments (schwarze Segmentbildpunkte): (a) e=24, k=38 und f=15, also E=1. (b) e=24, k=38 und f=14, also E=0. (c) e=24, k=38 und f=13, somit E=−1. (d) e=24, k=37 und f=13, somit$E$=0. Die Größe $L = 1 - E$ gibt die Anzahl der Löcher an.

24.11 Auswahl von Segmenten mit morphologischen Operationen

Wenn man bei einer aktuellen Problemstellung die Aufgabe hat, die Umgebung eines bestimmten Segments aus einem Bild auszublenden, so dass im Ergebnisbild nur mehr dieses Segment vorhanden ist, so kann man auch mit einfachen morphologischen Operationen arbeiten. Die Grundlagen der morphologischen Operationen sind in Kapitel 6 zusammengestellt.

Angenommen, es liegt ein Zweipegelbild mit verschiedenen Segmenten vor (Hintergrundgrauwert 0, Segmente mit Grauwert 255). Um ein bestimmtes Segment herauszugreifen, geht man wie folgt vor:

- Das Eingabebild wird in ein Eingabefeld eingelesen.
- Es wird ein beliebiger Punkt des Segments (z.B. der Schwerpunkt) als „Initialpunkt“ definiert.
- In einem Ausgabefeld, das die Größe des Eingabefeldes hat, wird nur dieser Punkt auf den Grauwert 255 gesetzt. Alle anderen Punkte haben den Grauwert 0.
- Nun wird das Ausgabefeld dilatiert. Der ursprüngliche Initialpunkt wird um einen ein Pixel breiten Rand vergrößert.
- Jetzt wird die logische UND-Operation mit dem Eingabefeld durchgeführt. Falls durch die Dilatation Punkte dazu gekommen sind, die nicht zum gewünschten Segment gehören, werden diese durch diese Operation wieder eliminiert.
- Jetzt wird wieder das Ausgabefeld dilatiert, dann die UND-Operation ausgeführt.
- Diese Operationen werden so lange wiederholt, bis sich die Fläche der Bildpunkte mit dem Grauwert 255 vor und nach der UND-Verknüpfung nicht mehr ändert.

- Im Ausgabefeld ist nur mehr das gewünschte Segment.

Dieses Verfahren, das auch *grassfire*-Operation genannt wird [Abma94], kann auch verwendet werden, um Störungen am Bildrand zu eliminieren. Als Initialpunkte kann man die Randpunkte des Bildes verwenden. Das Ergebnisbild wird vom Ausgangsbild abgezogen, so dass nur mehr die Segmente übrig bleiben, die den Bildrand nicht berühren.

24.12 Segmentbeschreibung mit Fuzzy Logic

In diesem Abschnitt soll der Versuch kurz dargestellt werden, Segmente mit Methoden der *fuzzy logic* zu klassifizieren. Die Grundlagen dazu wurden in Kapitel 22 erläutert. Die Vorgehensweise ist etwa wie folgt:

- Transformation der Segmente in eine geeignete Datenstruktur, etwa *run-length*-Code.
- Berechnung von segmentbeschreibenden Merkmalen: Fläche, Umfang, Schwerpunkt, Orientierung, Momente, usw.
- Normierung der Segmente bezüglich Translation (Normierung über die Schwerpunktkoordinaten), Rotation und Größe (Normierung über Momente, siehe Abschnitt 24.9).
- Berechnung eines Merkmalsvektors für jedes Segment mit geeigneten Merkmalen.
- *fuzzy clustering* gemäß Kapitel 22.
- Zuordnung des Segments zur Klasse (zum *cluster*) mit dem höchsten Mitgliedsgradwert oder weitere Verwendung der *fuzzy*-Mengen der einzelnen Segmente bei anschließenden Verarbeitungsschritten.

Praktische Erfahrungen haben gezeigt, dass dieses Verfahren gut funktioniert. Es hat sich vor allem als sehr vorteilhaft erwiesen, dass für eine gewisse Anzahl von Segmenttypen, die die verschiedenen Klassen festlegen, nur ein einziger Trainingslauf benötigt wurde.

Kapitel 25

Segmentbeschreibung mit dem Strahlenverfahren

25.1 Anwendungen

In den vorangehenden Kapiteln wurden viele Beispiele gegeben, wie segmentbeschreibende Merkmale und Datenstrukturen berechnet werden können. Einige dieser Merkmale waren unabhängig von der Lage des Segments (z.B. der Flächeninhalt), andere unabhängig von der Größe des Segments (z.B. die Orientierung). Bei Datenstrukturen, die ebenfalls zur Charakterisierung von Segmenten verwendet werden können, wie z.B. der *run-length*-Code oder der *chain*-Code, hängt es von der Implementierung ab, ob sie lageinvariant sind oder nicht. Auf jeden Fall sind beide Verfahren nicht orientierungsunabhängig.

Wenn Segmente in Klassen eingeteilt werden sollen (z.B. „Bauteil vom Typ X/Y“ oder „defektes/nicht defektes Bauteil“), so wird man genau diejenigen charakteristischen Merkmale verwenden, die bei einer gegebenen Problemstellung gerade noch zur Unterscheidung ausreichen. Wenn ein einfaches Merkmal, wie der „Flächeninhalt“, zur Unterscheidung genügt, so wäre es unsinnig und würde den Rechenaufwand nur unnötig erhöhen, wenn man weitere, aufwändigere Segmentmerkmale mit verwenden würde.

Bei manchen Anwendungen, bei denen die Segmente komplexe Formen besitzen und durch einfache Merkmale nicht zu unterscheiden sind, muss man etwas mehr Aufwand betreiben. Eine zusätzliche Forderung kann noch sein, dass man die Segmente, unabhängig von ihrer Größe, ihrer Position im Bildausschnitt und ihrer Drehlage, immer erkennt. Dazu wurden Verfahren zur rotations-, translations- und größeninvarianten Segmenterkennung entwickelt.

In diesem Kapitel und im folgenden Kapitel 26 werden zwei Verfahren zur dieser Problemstellung erläutert, anhand von Beispielen dargestellt und auf ihre Leistungsfähigkeit untersucht. Beide Verfahren tasten den Rand eines Segments ab und verwenden den so entstehenden Merkmalsvektor der Distanzen zum Rand als charakterisierendes Merkmal. Das erste Verfahren, das *Strahlenverfahren*, verwendet einen multivariaten Klassifikator. Das zweite Verfahren arbeitet mit einem neuronalen Backpropagation-Netz.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_25

25.2 Prinzipieller Ablauf des Strahlenverfahrens

Das Strahlenverfahren arbeitet nach dem Prinzip einer überwachten Klassifizierungsstrategie, die fest dimensioniert oder lernend implementiert werden kann (Kapitel 11 und 20). Im hier beschriebenen Fall wird die feste Dimensionierung gewählt. Der prinzipielle Ablauf der *Trainingsphase* des Strahlenverfahrens ist im folgenden Algorithmus zusammengestellt:

25.1: Trainingsphase des Strahlenverfahrens.

Algorithmus:

(a) Einzug des Bildes über einen Videosensor (oder für Testbeispiele aus einer Datei).

(b) Segmentierung des Bildes.

(c) Kompakte Speicherung der Segmente mit Hilfe spezieller Datenstrukturen (z.B. *run-length*-Code).

(d) Für alle zu trainierenden Segmente:

(da) Auswahl eines Segments für das Training.

(db) Aufbau der Trainingsdaten.

(dc) Eintragen der Trainingsdaten in die Trainingsdatei.

(e) Dimensionierung eines multivariaten Klassifikators.

(f) Abspeichern der Ergebnisse des Trainings in einer Datei.

Ende des Algorithmus

In der *Erkennungsphase* (*Klassifizierungsphase*, *Produktionsphase*, *recall*-Phase) werden Segmente verarbeitet, deren Zugehörigkeit zu einer der möglichen Klassen nicht bekannt ist. Mit Hilfe des trainierten Klassifikators kann eine Zuordnung zu einer Klasse erfolgen. Die Produktionsphase ist der Trainingsphase sehr ähnlich:

25.2: Klassifizierungsphase des Strahlenverfahrens.

Algorithmus:

(a) Einzug des Bildes über einen Videosensor.

(b) Segmentierung des Bildes.

(c) Kompakte Speicherung der Segmente mit Hilfe spezieller Datenstrukturen (z.B. *run-length*-Code).

(d) Einlesen der Dimensionierungsdaten des Klassifikators aus einer Datei.

(e) Für alle zu erkennenden Segmente:

(ea) Auswahl eines Segments.

(eb) Aufbau des Merkmalsvektors.

(ec) Klassifizierung des Merkmalsvektors.

Ende des Algorithmus

Der wichtigste Punkt in der Trainingsphase ist der Aufbau der Trainingsdaten. Dieser Problemkreis wird im folgenden Abschnitt behandelt.

25.3 Aufbau des Merkmalsvektors für ein Segment

Der Grundgedanke beim Strahlenverfahren ist die Tatsache, dass der Rand eines Segments zu seiner Charakterisierung verwendet werden kann. Es muss aber ein Verfahren verwendet werden, das der Forderung nach der Translations-, Rotations- und Größeninvarianz gerecht wird. Dazu werden, um ein Segment möglichst eindeutig zu beschreiben, Strahlen ermittelt, indem man die Distanzen von bestimmten Randpunkten zum Schwerpunkt berechnet und als Merkmale verwendet. Die Randpunkte eines Segments werden durch dessen Kontur festgelegt. In die Berechnung werden jedoch nicht alle Randpunkte einbezogen, sondern nur die Schnittpunkte der Kontur mit Strahlen, die in gleichen Winkelabständen vom Schwerpunkt ausgehen. Dazu werden zu einem zu trainierenden Segment

- der Schwerpunkt berechnet,
- die Orientierung ermittelt,
- die Randpunkte bestimmt und,
- ausgehend vom Schwerpunkt und nach Maßgabe der Orientierung, ein Merkmalsvektor mit Distanzen Schwerpunkt/Rand in vorgegebenen Winkelintervallen berechnet.

Die Berechnung des Segmentschwerpunkts und der Randpunkte ist einfach und wurde bereits in den Abschnitten 24.3 und 24.8 dargestellt.

Die Berechnung der Orientierung des Segments ist ein wesentliches Kernstück des Strahlenverfahrens, da ausgehend von der Orientierung der Längen der Strahlen Schwerpunkt/Rand berechnet werden.

Verfahren zur Orientierung wurden in den Abschnitten 24.6 und 24.9 erläutert: Es werden aus der Kovarianzmatrix des Segments die Eigenwerte berechnet und zur Ermittlung der Hauptrichtung verwendet. Praktische Tests haben gezeigt, dass dieses Verfahren nur verwendet werden kann, wenn die in Frage kommenden Segmente eine ausgeprägte

Hauptrichtung besitzen, also länglich sind. Bei Segmenten, bei denen das nicht der Fall ist, wird sich mehr oder weniger zufällig eine Hauptrichtung ergeben, die bei unterschiedlichen Drehlagen auch unterschiedlich sein kann. Dieses Verfahren kann dann nicht eingesetzt werden.

Eine andere Möglichkeit, die sich als verhältnismäßig stabil gezeigt hat, ist die Bestimmung der Orientierung durch die Ermittlung der kürzesten Distanz Rand/Schwerpunkt/Rand (Bild 25.1-a).

Bei diesem Verfahren werden viele Segmentformen berücksichtigt. Sicherlich finden sich Segmente, bei denen auch durch diese Methode keine eindeutige Drehlage ermittelt werden kann, d.h. wenn mehr als eine kürzeste Distanz vorkommt. Zu bemerken ist noch, dass das Verfahren mit der Berechnung der Orientierung über die Eigenwerte kompatibel ist: Bei einem ausgeprägt länglichen Segment stimmen die berechneten Orientierungen der beiden Verfahren überein.

Zur Bestimmung der Orientierung mit diesem Verfahren werden zu allen Randpunkten die Distanzen zum Schwerpunkt berechnet. Dann werden die Strahlen Schwerpunkt/Rand zu Strecken Rand/Schwerpunkt/Rand ergänzt, indem zu einem Strahl, wenn möglich, derjenige Strahl gesucht wird, der um 180° weiter liegt. Diese Ergänzung wird meistens nicht mit einem exakten Winkel von 180° möglich sein, so dass hier bei der Implementierung Toleranzen eingebaut werden müssen. Aus diesen Strecken wird dann die kürzeste gesucht und die zugehörige Richtung als Hauptrichtung verwendet.

Wenn ein Segment theoretisch mehrere gleiche Hauptrichtungen besitzt (Bild 25.1-b), so werden sich bei einer praktischen Analyse doch geringfügige Längenunterschiede ergeben, so dass sich ein Minimum ermitteln lässt. Das Problem ist dann allerdings, dass bei einer anderen Drehlage eine andere Hauptrichtung zufällig den minimalen Wert annimmt und somit die Berechung der Orientierung nicht mehr eindeutig ist. In diesem Fall kann man versuchen, das Segment mehrmals zu trainieren. Falls auch das nicht möglich ist, ist das Verfahren zur Bestimmung der Hauptrichtung nicht verwendbar.

Wenn der Schwerpunkt außerhalb des Segments liegt, muss die Bestimmung der Orientierung etwas anders durchgeführt werden. Man kann hier z.B. die kürzeste Distanz Rand/Rand in Richtung des Schwerpunkts verwenden (Bild 25.1-c).

Nachdem die Orientierung des Segments bestimmt ist, wird im nächsten Schritt ein Merkmalsvektor mit Distanzen Schwerpunkt/Rand berechnet. Dazu können die bei der Bestimmung der Orientierung berechneten Distanzen aller Randpunkte zum Schwerpunkt verwendet werden. Der Vollkreis wird in vorgegebene Winkelintervalle eingeteilt, wobei die Richtung 0° durch die berechnete Hauptrichtung des Segments (oder senkrecht dazu) festgelegt ist (Bild 25.2-a).

Jetzt werden für jede Strahlenrichtung die Schnittpunkte mit dem Rand des Segments gesucht. Das wird in der Praxis nicht exakt möglich sein, da ja die Randpunkte nicht genau auf den gewählten Strahlenrichtungen liegen müssen. In diesem Fall werden die Randpunkte gewählt, die der jeweiligen Richtung am nächsten liegen (Bild 25.2-b).

Es wird häufig der Fall sein, dass sich in bestimmten Richtungen mehrere Schnittpunkte mit dem Rand ergeben (Bild 25.2-c). Alle diese Schnittpunkte werden zunächst erfasst. Erst bei der Aufstellung des Merkmalsvektors wird entschieden, welche davon verwendet

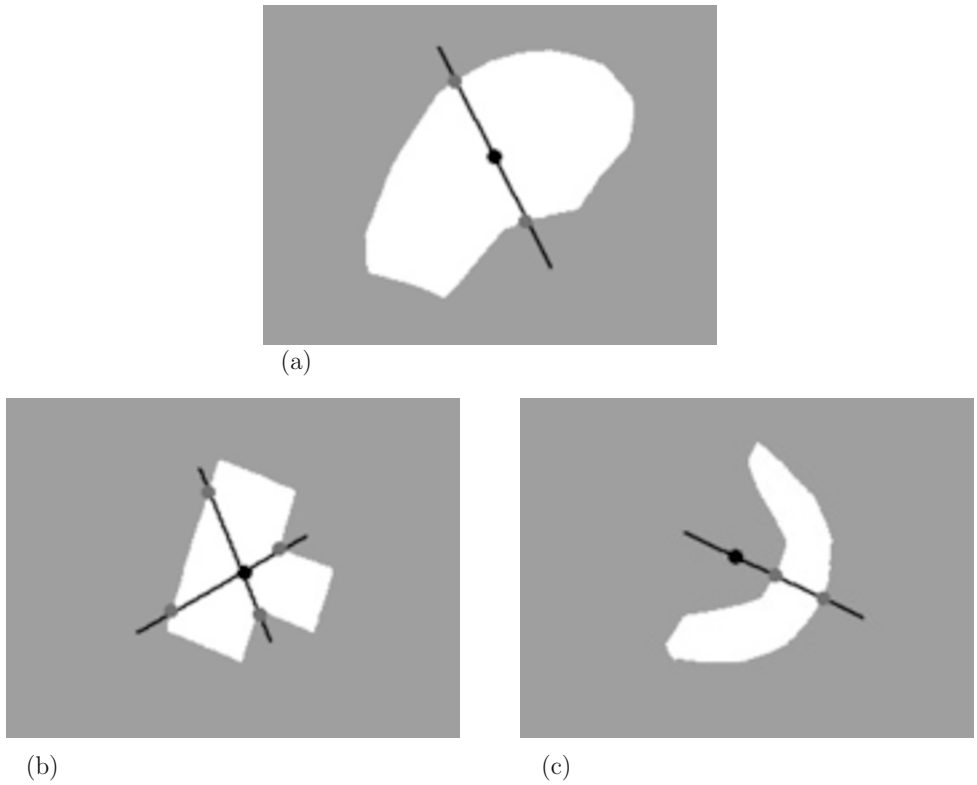


Bild 25.1: (a) Ermittlung der Orientierung eines Segments durch das Finden der kürzesten Distanz Rand/Schwerpunkt/Rand. (b) Manche Segmente besitzen mehrere gleichgroße Distanzen dieser Art. (c) Liegt der Schwerpunkt außerhalb des Segments, so kann die kürzeste Distanz Rand/Rand in Richtung Schwerpunkt verwendet werden.

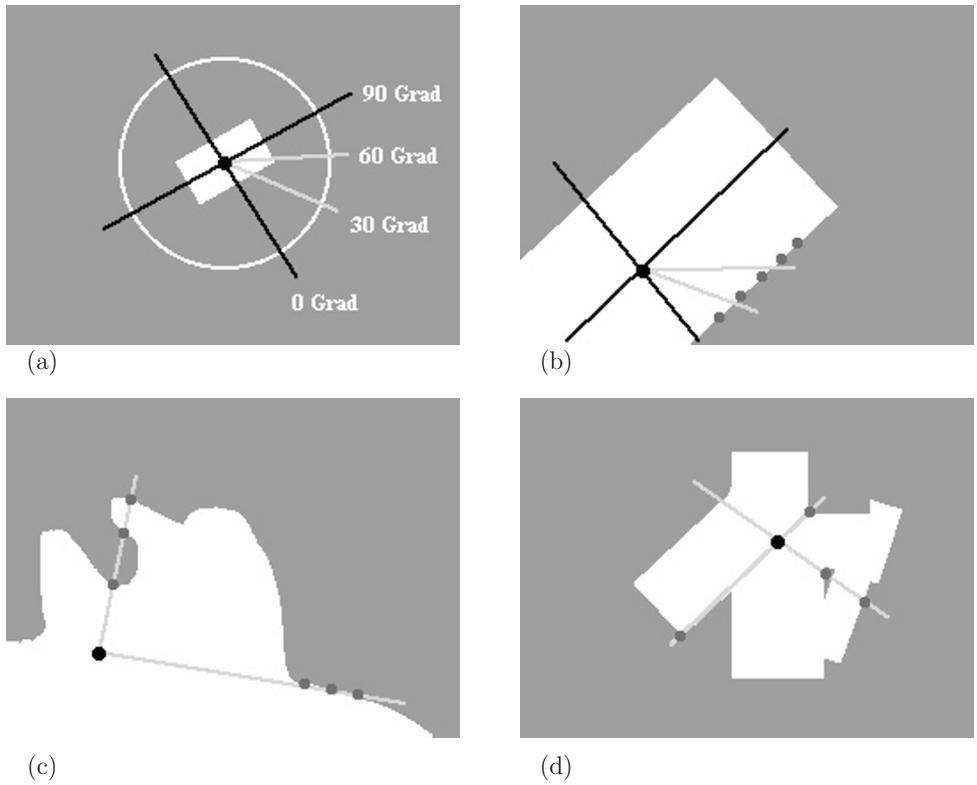


Bild 25.2: (a) Einteilung des Vollkreises in vorgegebene Winkelintervalle. Die Richtung 0° ist durch die berechnete Hauptrichtung des Segments (oder senkrecht dazu) festgelegt. (b) Die Randpunkte liegen nicht exakt auf den gewählten Strahlenrichtungen. Es werden die Randpunkte verwendet, die der jeweiligen Richtung am nächsten liegen. (c) Bei einigen Richtungen ergeben sich möglicherweise mehrere Schnittpunkte mit dem Rand des Segments. (d) Beim Aufbau des Merkmalsvektors, der das Segment charakterisieren soll, wird pro Richtungspaar die minimale und die maximale Distanz verwendet.

werden.

Nachdem zu den Richtungsstrahlen alle Distanzen zum Rand berechnet sind, kann ein Merkmalsvektor aufgebaut werden, der die Kontur des Segments charakterisieren soll. Hier muss man sich entscheiden, welche der Distanzen dazu verwendet werden sollen. Im vorliegenden Fall werden pro Richtungsstrahlenpaar (eine Richtung zusammen mit der um 180° weiter liegenden Richtung, z.B. 0° und 180°, 12° und 192°,...) jeweils die minimale und die maximale Distanz verwendet (Bild 25.2-d). Bei einem Winkelintervall von 12° werden somit 180/12=15 Richtungen verwendet. Der Merkmalsvektor ist dementsprechend 30-dimensional.

Mit dieser Vorgehensweise bei der Berechnung der Merkmalsvektoren wurde die erste Forderung, nämlich die Rotationsinvarianz der Merkmale, erfüllt, da ja die Distanzen nach Maßgabe der ermittelten Hauptrichtung berechnet wurden. Es muss an dieser Stelle nochmals darauf hingewiesen werden, dass das Verfahren voraussetzt, dass die Hauptrichtung mit genügender Genauigkeit bestimmt werden kann.

Die Größeninvarianz wird durch Normierung der Distanzen erreicht. Dazu folgende Überlegungen: Bei einem Quadrat Q_1 mit der Seitenlänge a ist die Fläche $A_1 = a^2$ und die Länge der Diagonalen $d_1 = a\sqrt{2}$. Bei einem Quadrat Q_2 mit der doppelten Seitenlänge $2a$ ergibt sich: $A_2 = 4a^2$ und $d_2 = 2a\sqrt{2}$. Wenn die Strecken d_1 und d_2 dadurch normiert werden, dass sie durch die Wurzel aus ihrem Flächeninhalt dividiert werden, ergibt sich in beiden Fällen der Wert $\sqrt{2}$, obwohl das zweite Quadrat eine viermal so große Fläche besitzt. Dieser Sachverhalt wird zur Normierung der Distanzen verwendet: Es wird jede Distanz durch die Wurzel aus dem Flächeninhalt des Segments dividiert.

Die folgende Tabelle zeigt die verwendeten normierten Distanzen für ein Segment, das eine ähnliche Lage und Form hat wie das Segment von Bild 25.2-b:

Richtung	maximale Distanz	minimale Distanz
0°/180°	38.41	34.94
12°/192°	39.88	35.50
24°/204°	43.26	37.74
36°/216°	49.66	40.81
48°/228°	58.23	48.91
60°/240°	72.49	61.17
72°/252°	93.94	86.62
84°/264°	89.85	88.33
96°/276°	89.82	88.36
108°/288°	94.15	90.02
120°/300°	95.61	79.61
132°/312°	76.28	56.43
144°/324°	54.72	45.81
156°/336°	44.43	40.04
168°/348°	38.83	36.15

Nachdem zu allen in Frage kommenden Segmenten die Merkmalsvektoren berechnet

Bild 25.3: Drei Verkehrszeichen als Testbeispiele für das Strahlenverfahren: „Viehtrieb, Tiere“, „Wildwechsel“, „Flugbetrieb“.

wurden, kann ein multivariater Klassifikator dimensioniert werden. Im vorliegenden Beispiel wird ein Minimum-Distance-Klassifikator verwendet (Abschnitt 20.5). Mit einem geeigneten Zurückweisungsradius kann zusätzlich eine Klasse „nicht klassifizierbar“ eingeführt werden, in die Segmente eingeteilt werden, die zu weit von den *cluster*-Zentren entfernt liegen.

25.4 Klassifizierungs- / Produktionsphase

Wie der Algorithmus von Abschnitt 25.2 zeigt, ist die Klassifizierungs- oder Produktionsphase sehr ähnlich der Trainingsphase. Für ein zu untersuchendes Segment wird in der Weise wie in Abschnitt 25.3 beschrieben ein Merkmalsvektor berechnet. Nach dem Training ist der Klassifikator dimensioniert. Somit kann ein unbekannter Merkmalsvektor einer Klasse (oder der Zurückweisungsklasse) zugewiesen werden.

25.5 Strahlenverfahren: Ein Beispiel

Zum Abschluss ein Beispiel, das dieses Verfahren illustrieren soll. Im Bereich „intelligente Fahrzeuge“ könnte man z.B. Systeme vorsehen, die Verkehrszeichen erkennen. Dabei ist zunächst die Aufgabe zu lösen, ein Verkehrszeichen aus seiner Bildumgebung zu isolieren. Im zweiten Schritt kann man versuchen, den Sinngehalt des Verkehrszeichens zu ermitteln. Für das Beispiel wurden die drei Verkehrszeichen „Viehtrieb, Tiere“, „Wildwechsel“ und „Flugbetrieb“ gewählt (Bild 25.3).

In Vorverarbeitungsschritten wurden die drei Symbole segmentiert. Im nächsten Schritt wurden zu jedem Symbol in zwölf verschiedenen Drehlagen (0°, 30°, ..., 330°, Bilder 25.4-a

bis 25.4-c) wie oben beschrieben die Merkmalsvektoren berechnet und damit ein Minimum-Distance-Klassifikator trainiert. Dabei wurden pro Segment 15 Richtungen gesetzt, so dass die Merkmalsvektoren 30-dimensional waren. Für das Segment „Kuh, Lage 4“ (links oben) sind in der folgenden Tabelle sämtliche Distanzen aufgeführt. Die im Merkmalsvektor verwendeten Distanzen sind mit einem (*) markiert:

Richtung	Schnitte
0°/180°	33.94* 20.95*
12°/192°	67.87* 37.43 20.95*
24°/204°	67.59* 60.83 42.10 21.80*
36°/216°	61.86* 55.16 50.31 24.47*
48°/228°	102.83* 97.95 92.16 86.27 80.46 59.35 38.61 34.80 28.36*
60°/240°	111.65* 88.96 71.19 60.50*
72°/252°	130.79* 95.71 92.26 87.70 83.99*
84°/264°	98.09* 93.90 82.79 69.31*
96°/276°	100.46* 85.50 61.99*
108°/288°	103.84* 97.51 59.12*
120°/300°	72.50* 63.15*
132°/312°	88.17* 84.55 66.80 56.21 47.15*
144°/324°	51.48* 44.75 42.43 38.61 25.74*
156°/336°	68.47* 64.62 59.39 38.48 22.01*
168°/348°	36.03* 21.80*

In der Produktionsphase wurde jedes Symbol in vier unterschiedlichen Drehlagen dem Klassifikator angeboten (Bild 25.4-d). Das Ergebnis der Klassifizierung ist in folgender Tabelle zusammengefasst:

Eingabesymbol	klassifiziert zu Klasse	Soll
Segment „Kuh“, Lage 1	1	1
Segment „Kuh“, Lage 2	1	1
Segment „Kuh“, Lage 3	1	1
Segment „Kuh“, Lage 4	1	1
Segment „Hirsch“, Lage 1	2	2
Segment „Hirsch“, Lage 2	2	2
Segment „Hirsch“, Lage 3	2	2
Segment „Hirsch“, Lage 4	2	2
Segment „Flugzeug“, Lage 1	3	3
Segment „Flugzeug“, Lage 2	3	3
Segment „Flugzeug“, Lage 3	3	3
Segment „Flugzeug“, Lage 4	3	3

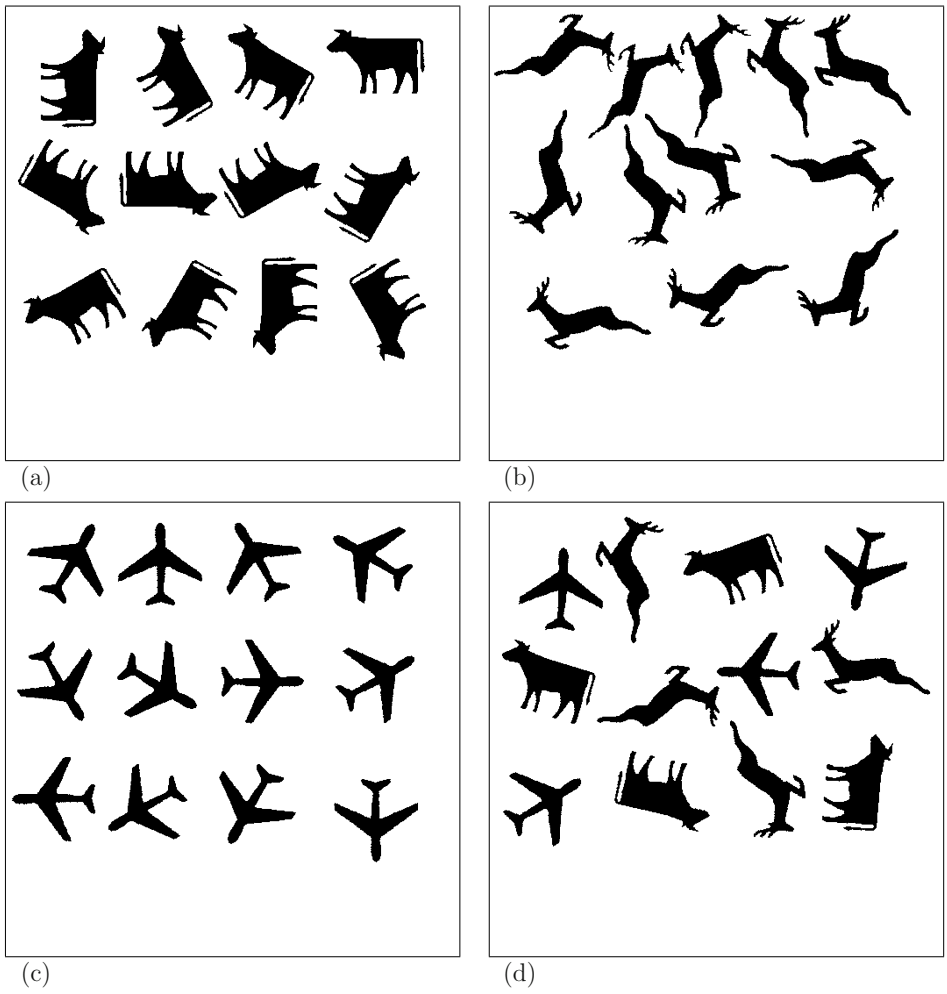


Bild 25.4: Jedes Symbol wurde in zwölf Drehlagen trainiert. (a) Kuh. (b) Hirsch. (c) Flugzeug. (d) Zur Klassifizierung wurde jedes Symbol viermal in unterschiedlichen Drehlagen, die nicht mit den trainierten Drehlagen identisch waren, verarbeitet. Es wurden alle Symbole richtig klassifiziert.

Die Ergebnisse zeigen, dass in diesem Fall alle Segmente richtig erkannt wurden. Weitere Tests mit unterschiedlich großen Symbolen ergaben ebenfalls gute Ergebnisse. Allerdings sind hier den erkennbaren Segmentgrößen nach unten und oben Grenzen gesetzt (etwa halber bzw. doppelter Flächeninhalt).

Kapitel 26

Neuronale Netze und Segmentbeschreibung

26.1 Anwendungen

Die Anwendungen sind hier dieselben wie im vorhergehenden Kapitel: Die translations-, rotations- und größeninvariante Erkennung von Segmenten. Es wird sich zeigen, dass hier das Training einfacher ist als im Fall des Strahlenverfahrens.

26.2 Prinzipieller Ablauf

Die theoretischen Grundlagen zu neuronalen Netzen sind in Kapitel 21 zusammengefasst. Im vorangehenden Kapitel 25 wurde auf die Thematik der translations-, rotations- und größeninvarianten Segmenterkennung ausführlich eingegangen. In diesem Abschnitt wird beschrieben, wie neuronale Netze in diesem Zusammenhang zur Erkennung von Segmenten eingesetzt werden können.

Der prinzipielle Ablauf ist ähnlich wie beim Strahlenverfahren in Kapitel 25. Auch hier kann eine Einteilung in die Trainingsphase und die Produktionsphase (*recall*-Phase) vorgenommen werden. Die Trainingsphase läuft wie folgt ab:

26.1: Training der Segmenterkennung mit einem neuronalen Netz.

Algorithmus:

(a) Einzug des Bildes über einen Videosensor.

(b) Segmentierung des Bildes.

(c) Kompakte Speicherung der Segmente mit Hilfe spezieller Datenstrukturen (z.B. *run-length*-Code).

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_26

(d) Für alle zu trainierenden Segmente:

(da) Auswahl eines Segments für das Training.

(db) Aufbau der Trainingsdaten.

(dc) Eintragen der Trainingsdaten in die Trainingsdatei.

(e) Training des neuronalen Netzes.

(f) Abspeichern des trainierten Netzes in einer Datei.

Ende des Algorithmus

Die Produktionsphase ist der Trainingsphase sehr ähnlich:

26.2: Segmenterkennung mit einem neuronalen Netz.

Algorithmus:

(a) Einzug des Bildes über einen Videosensor.

(b) Segmentierung des Bildes.

(c) Kompakte Speicherung der Segmente mit Hilfe spezieller Datenstrukturen (z.B. *run-length*-Code).

(d) Einlesen des trainierten Netzes aus einer Datei.

(e) Für alle zu erkennenden Segmente:

(ea) Auswahl eines Segments.

(eb) Aufbau des Merkmalsvektors .

(ec) Klassifizierung des Merkmalsvektors mit dem trainierten Netz.

Ende des Algorithmus

Der Punkt (db) im Algorithmus zum Training und der Punkt (eb) im Algorithmus zur Produktionsphase sind die wichtigsten Kernstücke des Verfahrens. Sie werden im Folgenden näher erläutert.

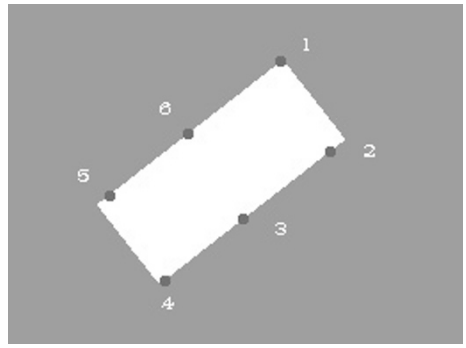


Bild 26.1: Gleichabständige Unterteilung des Randes eines Segments durch sechs Punkte.

26.3 Trainingsdaten und Training

Zu einem Segment, das trainiert werden soll, wird zunächst der Rand des Segments ermittelt. Hier sei darauf hingewiesen, dass in der vorliegenden Beschreibung nur Segmente mit einem zusammenhängenden Rand, also ohne Löcher innerhalb des Segments, betrachtet werden. Auf dem Rand des Segments werden dann $m = n + 1$ Punkte äquidistant verteilt. In der Praxis wird das nicht immer exakt möglich sein, da ja die Länge des Randes (in *Pixel*) meistens nicht durch m teilbar sein wird. Es sollen aber die m Punkte „so gut wie möglich“ gleichabständig verteilt sein (Bild 26.1).

Nun werden für jeden der $m = n + 1$ Punkte die n Distanzen zu den restlichen n Punkten berechnet. Die so entstehenden Distanzvektoren werden als Merkmalsvektoren verwendet. Im Beispiel von Bild 26.1 wurden $m = 6 = 5 + 1$ Punkte verteilt. Die sechs Merkmalsvektoren $\vec{g_1}$ bis $\vec{g_6}$ sind hier fünfdimensional:

$$\vec{g_1} = \begin{pmatrix} d_{12} \\ d_{13} \\ d_{14} \\ d_{15} \\ d_{16} \end{pmatrix}, \vec{g_2} = \begin{pmatrix} d_{23} \\ d_{24} \\ d_{25} \\ d_{26} \\ d_{21} \end{pmatrix}, \vec{g_3} = \begin{pmatrix} d_{34} \\ d_{35} \\ d_{26} \\ d_{31} \\ d_{32} \end{pmatrix}, ..., \vec{g_6} = \begin{pmatrix} d_{61} \\ d_{62} \\ d_{63} \\ d_{64} \\ d_{65} \end{pmatrix}. \tag{26.1}$$

Die Distanz des Punktes i mit den Koordinaten (x_i, y_i) zum Punkt j mit den Koordinaten (x_j, y_j) wird mit der euklidischen Abstandsformel berechnet:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{26.2}$$

Es liegen somit zu einem Segment m Merkmalsvektoren $\vec{g}$ mit der Dimension n vor. Man sagt auch: Zum Rand des Segments werden m *Ansichten* berechnet. In der späteren

Trainingsphase werden diese m Ansichten des Segments als charakterisierende Merkmale mit einem neuronalen Netz trainiert.

Da die Eingangswerte eines Backpropagation-Netzwerks zwischen 0 und 1 liegen müssen, werden die Distanzen normiert. Dazu wird während der Berechnung eines Merkmalsvektors die minimale (min) und die maximale (max) Komponente bestimmt. Die Normierung der Distanz d_{ij} kann dann nach der Formel

$$d_{ij}^{norm} = \frac{d_{ij} - min}{max - min} \tag{26.3}$$

erfolgen.

In einer Trainingsdatei werden alle Merkmalsvektoren aller Segmente gesammelt, für die das Netz trainiert werden soll. Ein Beispiel einer Trainingsdatei wird in Abschnitt 26.5 gegeben.

Mit diesen Daten wird das Backpropagation-Netz trainiert. Für die Netztopologie, die Anzahl der Neuronen in verdeckten Schichten, die Lernrate und den zu erzielenden minimalen Systemfehler gelten die Aussagen in Kapitel 21.

26.4 Die Produktions- (Recall-) Phase

Der Algorithmus zur *recall*-Phase in Abschnitt 26.2 zeigt, dass hier fast derselbe Weg beschritten wird. Im Gegensatz zur Trainingsphase wird hier zu den m äquidistanten Punkten nur *ein* Merkmalsvektor berechnet, z.B. der Vektor, der dem ersten Punkt zugeordnet ist. Das Segment wird hier also nur durch *eine* Ansicht charakterisiert. Dieser Merkmalsvektor wird dem neuronalen Netz zugeführt, und die Ausgabe des Netzes gibt die zugehörige Klasse an.

Dabei ist zu beachten, dass das Segment in einer anderen Drehlage vorliegen kann. Die äquidistanten Punkte werden dann sicher nicht genau so liegen wie in der Trainingsphase. Der grundlegende Gedanke ist dabei jedoch, dass die Ansicht einer der gelernten Ansichten ähnlich sein wird und das Netz durch den Generalisierungseffekt für die richtige Klasse entscheidet.

Hier wird das Generalisierungsverhalten eines Netzes als wesentliche Eigenschaft der Zuordnung zu den Klassen verwendet. Aus diesem Grund ist es sinnvoll, keine übertrainierten Netze (zu viele Neuronen in den verdeckten Schichten und zu kleiner Systemfehler) zu verwenden (Kapitel 21).

26.5 Ein Beispiel: Unterscheidung von Schrauben

In diesem abschließenden Beispiel zur translations-, rotations- und größeninvarianten Segmenterkennung sollen zwei verschiedene Schraubentypen unterschieden werden (Bild 26.2-a). In der Vorverarbeitung wurde das Bild invertiert und zweimal dilatiert, um die

Konturen der Schrauben etwas ausgeglichener zu gestalten (Bild 26.2-b). Die Segmentierung konnte durch eine einfache Binarisierung erzeugt werden. Zur Bezeichnung wurden die Segmente von 1 bis 6 numeriert (Bild 26.2-c).

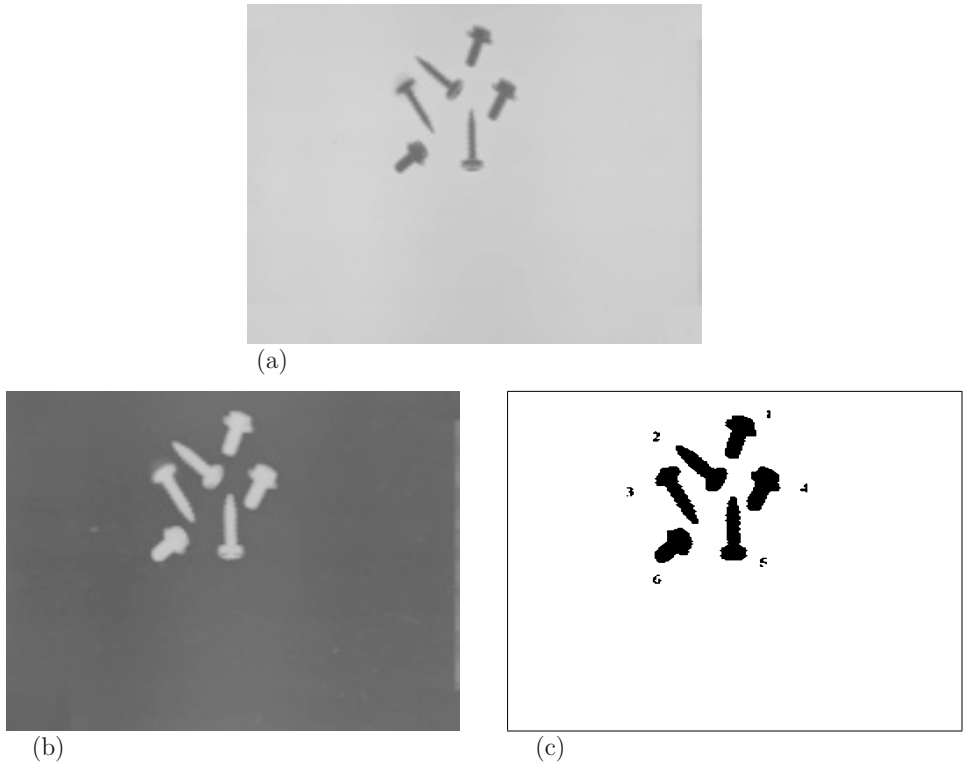


Bild 26.2: (a) Originalbild: Jeweils drei Schrauben eines bestimmten Schraubentyps. (b) Vorverarbeitung: Invertierung und einmalige Dilatation, um die Konturen etwas „runder“ zu gestalten. (c) Segmentierung durch Binarisierung. Mit den Segmenten vier und fünf wurde das neuronale Netz für die zwei Klassen trainiert.

Im Training wurden nur die Segmente vier und fünf verwendet. Es wurden auf der Kontur der Segmente elf Punkte gleichmäßig verteilt, so dass die Merkmalsvektoren zehndimensional sind. Das neuronale Netz hatte bei diesem Test, abhängig von den Ein-/Ausgabedaten, folgende Struktur: Zehn Eingabeneuronen, zwei Ausgabeneuronen und eine verdeckte Schicht mit zehn Neuronen. Die Trainingsdatei ist in folgender Tabelle wiedergegeben:

```
train
inpn 10
outpn 2
data
0.022891 0.297219 0.501564 0.499727 0.748440
1.000000 0.865390 0.583158 0.297538 0.053051    0.95 0.05
0.022891 0.098389 0.349881 0.417831 0.701165
0.980041 0.894565 0.619416 0.340513 0.212547    0.95 0.05
0.297219 0.098389 0.053051 0.195373 0.487907
0.780699 0.758973 0.518783 0.303894 0.354507    0.95 0.05
0.501564 0.349881 0.053051 0.000000 0.257634
0.547774 0.577370 0.389987 0.279645 0.461229    0.95 0.05
0.499727 0.417831 0.195373 0.000000 0.074066
0.365914 0.362515 0.172850 0.123398 0.376825    0.95 0.05
0.748440 0.701165 0.487907 0.257634 0.074066
0.073503 0.137564 0.108117 0.269947 0.568188    0.95 0.05
1.000000 0.980041 0.780699 0.547774 0.365914
0.073503 0.038732 0.234922 0.490473 0.786469    0.95 0.05
0.865390 0.894565 0.758973 0.577370 0.362515
0.137564 0.038732 0.062584 0.349881 0.621778    0.95 0.05
0.583158 0.619416 0.518783 0.389987 0.172850
0.108117 0.234922 0.062584 0.067808 0.344337    0.95 0.05
0.297538 0.340513 0.303894 0.279645 0.123398
0.269947 0.490473 0.349881 0.067808 0.080194    0.95 0.05
0.053051 0.212547 0.354507 0.461229 0.376825
0.568188 0.786469 0.621778 0.344337 0.080194    0.95 0.05
0.012551 0.241898 0.471953 0.709719 0.935187
1.000000 0.805930 0.595665 0.345771 0.095366    0.05 0.95
0.012551 0.070258 0.307902 0.546835 0.767430
0.850464 0.672995 0.455651 0.215990 0.000000    0.05 0.95
0.241898 0.070258 0.069366 0.307752 0.527592
0.617138 0.455651 0.236322 0.035707 0.033615    0.05 0.95
0.471953 0.307902 0.069366 0.069366 0.293825
0.380524 0.241032 0.033615 0.026119 0.227095    0.05 0.95
0.709719 0.546835 0.307752 0.069366 0.060871
0.156343 0.098571 0.029008 0.225476 0.458492    0.05 0.95
0.935187 0.767430 0.527592 0.293825 0.060871
0.060871 0.185213 0.238597 0.455651 0.687740    0.05 0.95
1.000000 0.850464 0.617138 0.380524 0.156343
0.060871 0.065763 0.237723 0.487742 0.737700    0.05 0.95
...
0.805930 0.672995 0.455651 0.241032 0.098571
0.687740 0.737700 0.540947 0.331763 0.081286    0.05 0.95
```

In der Trainingsdatei ist gespeichert, dass das Backpropagation-Netz in diesem Beispiel zehn Eingabeneuronen und zwei Ausgabeneuronen besitzt, entsprechend den zehndimensionalen Merkmalsvektoren und den beiden Ausgabeklassen (Segmentnummer 1 und 2). Ab dem Kennzeichen `data` sind die Trainingspaare abgespeichert. Nach dem ersten Merkmalsvektor des ersten Segments folgt die Sollausgabe 0.95 und 0.5. Aus numerischen Gründen wird hier nicht 1 und 0 verwendet. Dann folgen die weiteren Merkmalsvektoren des Segments, jedes Mal mit der Sollausgabe 0.95 und 0.05. Die Trainingsdaten eines zweiten Segments schließen sich an.

Im Training waren 224 Durchläufe notwendig, um einen Systemfehler von 0.05 zu erreichen. Das Ergebnis ist in folgender Tabelle zusammengefasst:

Segmentnr.	Ausgabeneuron 1	Ausgabeneuron 2	Klasse
1	0.981386	0.019592	1
2	0.101804	0.892246	2
3	0.050654	0.945478	2
4	0.937410	0.064300	1
5	0.057105	0.938721	2
6	0.937877	0.063785	1

Man sieht, dass alle Segmente richtig erkannt wurden. Wie schon im einleitenden Abschnitt zu dieser Thematik erwähnt, ist die Trainingsphase einfacher als beim Strahlenverfahren, da hier ein Segment nur einmal mit seinen verschiedenen Ansichten trainiert werden muss.

Kapitel 27

Kalman-Filter

27.1 Grundidee

Gegeben sei ein physikalisches System: z.B. eine Kugel, die durch die Luft fliegt und von Luftwirbeln ein wenig aus der Normal-Bahn abgelenkt wird (Bild 27.1). Die Bewegung dieses Systems kann durch das Newton'sche Gesetz ($F_a = ma$) beschrieben werden ($F_a = F_g + F_r$, wobei F_g die Gewichtskraft ist und F_r die Reibungskraft). Die Kräfte F_r, die durch die Luftwirbel auf die Kugel wirken, werden einfach als eine zufällige, normalverteilte Komponente betrachtet. Weiterhin gegeben sei eine physikalische Messvorrichtung: z.B. eine feststehende Kamera, die in bestimmten zeitlichen Abständen Bilder der Situation aufnimmt. Messbar sind dadurch die relative Position und die relative Größe der Kugel im Bild. Die Messgenauigkeit der Kamera wird nicht nur durch die Auflösung limitiert, sondern sei zusätzlichen Schwankungen durch die Luftwirbel oder anderen Störungen (z.B. Verdeckung) ausgesetzt.

Die Aufgabe besteht nun darin, den besten Schätzwert für die Position und die Geschwindigkeit der Kugel zum Zeitpunkt k unter Verwendung aller bis zu diesem Zeitpunkt angefallenen Messungen zu finden.

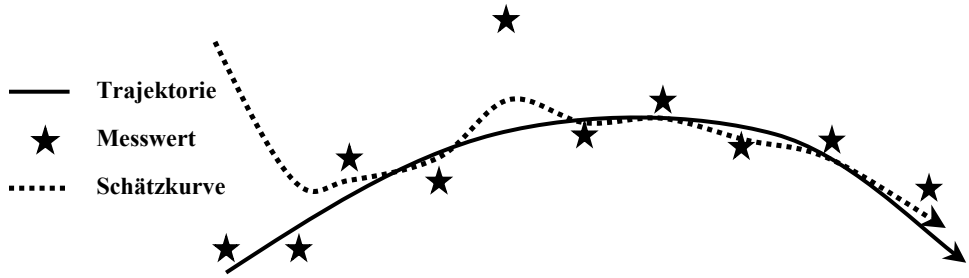


Bild 27.1: Die Grundidee des Kalman-Filters in der Bildverarbeitung: Prädiktion der Objekttrajektorie mit Hilfe von Bewegungsgleichungen und Korrektur der Trajektorie mit jeder neuen Messung. Die Messgenauigkeit bestimmt den Einfluss der jeweiligen Messung.

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_27

In der Lösungsidee werden alle verfügbaren Informationen sofort verarbeitet: Aus der bekannten Größe der Kugel und den Abbildungseigenschaften der Kamera kann die Entfernung Kugel – Kamera und somit auch die Position der Kugel (x, y, z) geschätzt werden. Aus zwei aufeinanderfolgenden Bildern kann auch die Geschwindigkeit (v_x, v_y, v_z) der Kugel geschätzt werden. Mit Hilfe der Systemgleichung (hier die Newton'sche Bewegungsgleichung) kann der Ort und die Geschwindigkeit der Kugel für zukünftige Zeitpunkte prädiziert werden. Die Abweichung zwischen prädiziertem und gemessenem Systemzustand (hier Ort und Geschwindigkeit) wird zur Verbesserung der Schätzung des Systemzustands verwendet. Messungen mit großer Genauigkeit sollen stark in die Berechnung einfließen, Messungen mit geringer Genauigkeit sollen nur wenig in die Berechnung einfließen. Das Kalman-Filter ist gerade so ausgelegt, dass es unter Berücksichtigung des Mess- und Systemrauschens (hier die Luftwirbel) aus den vorhandenen Messwerten den optimalen Schätzwert für den Systemzustand liefert.

27.2 Anwendungen

Seit der Originalarbeit von Kalman im Jahr 1960 [Kalm60] wurden zahlreiche Anwendungen für das gleichnamige stochastische Filter entwickelt. In der Regelungstechnik gehört das Kalman-Filter bereits seit längerem zu den Standardmethoden der Optimalen Regelung mit Zustandsbeobachter [Ludy95]. Eine wesentliche Rolle spielt das Kalman-Filter in modernen Navigationssystemen, in denen Messdaten verschiedener Navigations-Sensoren (z.B. Inertialsensor und GPS (Global Positioning System)) zur einer optimalen Positionsschätzung fusioniert werden [Jeke01]. Ganz allgemein ist das Kalman-Filter eines der am häufigsten eingesetzten Verfahren, mit denen verrauschte Messdaten verschiedener Sensoren zusammengeführt werden. Ein weiteres Einsatzgebiet des Kalman-Filters ist die Vorhersage ökonomischer Indikatoren. In diesem Fall hat man es mit sehr komplexen Systemen zu tun, bei denen systematische Zusammenhänge einzelner Größen von starken statistischen Einflüssen überlagert werden. Mit dem Kalman-Filter kann man die systematischen Zusammenhänge herausfiltern, falls sie richtig modelliert wurden.

Im Folgenden wird etwas detaillierter auf Anwendungen des Kalman-Filters in der Bildverarbeitung und Computergrafik eingegangen.

27.2.1 Tracking

Der häufigste Anwendungsfall des Kalman-Filters in der Bildverarbeitung ist die Objektverfolgung, auch *Tracking* genannt. In diesem Fall soll die Position und/oder die Geschwindigkeit von Objekten zu jedem Zeitpunkt optimal geschätzt werden. Als Eingabewerte liegen in der Regel Kamerabilder vor[1]. Mit Hilfe von Bildverarbeitungsverfahren wird eine Objekterkennung durchgeführt, so dass die Lage des Objektabbildes auf dem Kamerasensor bekannt ist. Aus der prädizierten Position und Lage des Objekts im 3-dimensionalen

[1] In speziellen Anwendungen können dies auch Positionsdaten anderer Sensoren sein, wie z.B. von einem Radar, Lidar oder Sonar.

Raum kann mit den vorab vermessenen Abbildungseigenschaften der Kamera die Lage des auf den Sensor projizierten Objekts vorhergesagt werden. Die Abweichung zwischen prädizierter und gemessener Lage des Objekts wird zur Verbesserung der Positions- und Geschwindigkeits-Schätzung des Objekts verwendet. Für eine erfolgreiche Filterung ist es erforderlich, sowohl die Genauigkeit der Positionsmessung als auch die Objekt- und Eigenbewegung zu berücksichtigen.

Für das Tracking gibt es eine große Bandbreite verschiedener Anwendungen, von denen hier nur einige Beispiele stellvertretend genannt werden. Ein vielversprechendes Gebiet sind Mensch-Maschine-Schnittstellen, wie z.B. die Gestenerkennung (z.B. Verfolgung einer bewegten Hand, Bild 27.3), die Verfolgung von Lippenbewegungen zur Spracherkennung, die Verfolgung von Augen-, Kopf- oder Körperbewegungen zur Steuerung von Automaten oder Virtual Reality Umgebungen [Dorf03]. In der Automobiltechnik werden neue Fahrerassistenzsysteme auf der Basis bildgebender Sensoren entwickelt, wie z.B. automatische Geschwindigkeits- und Abstandskontrolle, Warnung vor Hindernissen oder zu hoher Kurvengeschwindigkeit, automatische Fahrspurverfolgung bis hin zum automatischen Fahrer. In der Robotik geht es sowohl um die bildbasierte Navigation und Hinderniserkennung zur autonomen Bewegung, als auch um das Greifen oder Fangen von bewegten Objekten. Im Verteidigungsbereich wird das Kalman-Filter seit Langem zur automatischen Zielverfolgung eingesetzt.

Die prinzipiellen Ablaufschritte eines Kalman Trackers bei der Objektverfolgung sind:

- Aufschalten des Kalman-Filters (*Alignment*):
  Beim *Alignment* muss das gesamte Bild[2] abgescannt werden, da noch keine verlässliche Schätzung über die Lage des Objekts im Bild vorhanden ist.

  - Bildaufnahme:
    Mit dem Kamerasensor wird ein Bild eingezogen und digitalisiert.
  - Bildvorverarbeitung:
    Geometrische Entzerrung (z.B. wegen Kissenverzerrung der Optik), Kontrastoptimierung (z.B. Histogrammlinearisierung), Rauschunterdrückung (z.B. Median-Operator).
  - Merkmalsextraktion:
    Häufig sind dies Kantendetektion (z.B. Gradienten- oder Sobel-Operatoren) und Linienverfolgung (z.B. Bestimmung der Gradienten-Kammlinie). Ergebnis: Position und Orientierung der Kanten, sowie Geradheit, Ausgefranstheit von Linien.
  - Projektion:
    Die Kanten des gesuchten 3D-Objekts werden auf die 2D-Sensorbildebene projiziert.

[2] Bei einer beweglichen Kamera wird meist der gesamte Schwenkbereich der Kamera nach dem gesuchten Objekt abgescannt.

  - Matching-Verfahren:
    Korrelation des gemessenen Linienmusters mit dem erwarteten Muster. Das Ergebnis bei einer ausreichenden Korrelation ist eine Startschätzung des Systemzustands (z.B. Position, Lage und Geschwindigkeit des Objekts relativ zur Kamera).

Nach dem *Alignment* laufen abwechselnd die beiden folgenden Schritte ab, bis alle Bilder verwertet sind:

- Prädiktionsschritt:
  Vorhersage der relativen Position und Lage des Objekts zum nächsten Abtastzeitpunkt auf der Basis der zugrunde liegenden Systemdynamik (Position, Lage, Geschwindigkeit, Beschleunigung der eigenen Kamera und des Objekts, Kardanwinkel der Kamera usw.)

- Korrekturschritt:
  Korrektur der Prädiktion mit der Messung. Dabei werden ähnliche Schritte durchgeführt wie beim *Alignment*, d.h. Bildaufnahme, Bildvorarbeitung, Merkmalsextraktion, Objekterkennung mit eingeschränktem Suchbereich (je nach Qualität der Prädiktion): Dabei wird die Messung (Detektion des Objekts) dadurch erheblich erleichert, dass vom vorhergehenden Prädiktionsschritt ein Schätzwert für die Objektposition vorhanden ist. Deshalb genügt es, das Objekt in einem kleinen Bildausschnitt zu suchen - man muss nicht mehr das gesamte Bild bzw. den gesamten Schwenkbereich der Kamera absuchen (wie beim *Alignment*). Dieser Bildausschnitt wird „track gate“ genannt, und seine Größe wird bestimmt von der Genauigkeit der Prädiktion. Die Abweichung zwischen geschätzter und gemessener Objektposition auf der Sensorebene wird zur Verbesserung der Positions- und Geschwindigkeits-Schätzung für das Objekt benutzt.

Qualitätsmaße für die detektierten Kanten und Linien, wie z.B. der Kontrast der Kanten oder die Ausgefranstheit von Linien, werden als Maß für das Messrauschen benutzt und bestimmen somit, wie stark die Messung in die Schätzung des Systemzustands eingeht. Ist die Kantendetektion z.B. aufgrund eines sehr niedrigen Bildkontrasts sehr unsicher, geht die Messung kaum in die Zustandsschätzung ein. Auf diese Weise wird eine evtl. fehlerhafte Objekterkennung praktisch nicht berücksichtigt. Sind die Kanten dagegen klar zu erkennen, gehen sie mit höherem Gewicht in die Schätzung ein. In diesem Sinne werden die guten Messergebnisse herausgefiltert.

Die Wirkungsweise eines Kalman-Filters, bei dem die oben geschilderten Ablaufschritte durchgeführt werden, ist in Bild 27.2 dargestellt. In diesem Beispiel werden charakteristische Kanten eines Helicopterlandeplatzes verfolgt, so dass ein Helicopter mit Kamera automatisch an der vorgesehen Stelle landen kann. Die blau hervorgehobenen Kanten sind gemessen, d.h. durch Bildverarbeitungsalgorithmen aus dem aktuellen Bild extrahiert. Die rot eingezeichneten Kanten sind vom Kalman-Filter aus dem vorhergehenden Bild prädiziert. Wie man aus den Bildern sehen kann, liegen die prädizierten Kanten meistens relativ nahe an den gemessenen Kanten.

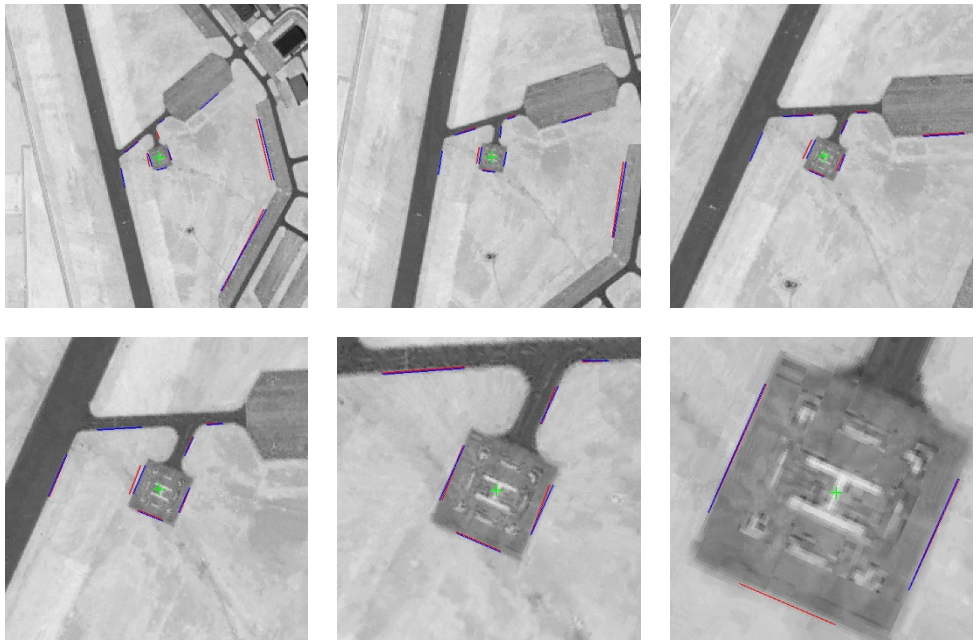


Bild 27.2: Kanten-basierter Kalman Tracker zur Detektion eines Hubschrauberlandeplatzes. In diesem Fall ruht das Objekt (der Landeplatz) und die Kamera bewegt sich. Die blauen Kanten sind gemessen, d.h. durch Bildverarbeitungsalgorithmen detektiert. Die roten Kanten sind vom Kalman-Filter prädiziert. Quelle: Lackner, EADS-LFK Lenkflugkörpersysteme GmbH.

Bei dem oben dargestellten Prinzip eines Kalman Trackers wird der Vergleich zwischen geschätzter und gemessener Objektposition auf der Basis von Kanten und Linien durchgeführt. Dies ist der häufigste Fall. Eine Alternative dazu sind segment-basierte Methoden, bei denen eine Szene nach bestimmten Kriterien in verschiedene Segmente unterteilt wird. Solche Kriterien können z.B. ähnliche Bewegungsvektoren oder Texturmerkmale sein. Das Kalman-Filter dient in diesem Fall dazu, die Verschiebung und evtl. die Verzerrung des Segments im nächsten Bild vorherzusagen. Das prädizierte Segment wird nun verglichen mit dem „gemessenen“ Segment, das sich aus der Einzelbild-Segmentierung des aktuellen Bildes ergibt. Die sich nicht überlappenden Teile des Segments sind ein Maß für die Abweichung der Prädiktion von der Messung und dienen zur Verbesserung der Schätzung der Objektbewegung. Ein Beispiel für einen segment-basierten Kalman Tracker, der in diesem Fall zur Verfolgung einer Handbewegung dient, ist in Bild 27.3 gegeben [Huan02].

Bild 27.3: Segment-basierter Kalman Tracker zur Verfolgung von Handbewegungen. In diesem Fall bewegt sich das Objekt (die Hand) und die Kamera ruht. Die grüne Ellipse wird anfangs manuell um das zu verfolgende Objekt (die Hand) gelegt. Das *Alignment* wird in diesem Fall also manuell durchgeführt. Die rote Kurve stellt die detektierte Kontur der Hand dar. Quelle: Yu Huang.

In den meisten Tracking-Anwendungen geht es um die Verfolgung von bewegten Objekten. „Bewegung“ bezieht sich in diesem Fall immer auf eine Relativbewegung zwischen Kamera und Objekt, d.h. es ist unerheblich, welches Koordinatensystem man wählt. Im Koordinatensystem der Kamera bewegt sich das Objekt (wie in Bild 27.3), im Koordinatensystem des Objekts bewegt sich die Kamera (wie in Bild 27.2) und in einem ungeschickt gewählten Koordinatensystem bewegen sich sowohl die Kamera als auch das Objekt. An dieser Stelle sei aber darauf hingewiesen, dass man auch andere Objekteigenschaften „verfolgen“ und „vorhersagen“ kann, wie z.B. die Orientierung (d.h. eine Rotationsbewegung), die Verformung oder Oberflächeneigenschaften (Textur, Farbe etc.) von Objekten.

Außerdem sollte an den bisher gezeigten Anwendungsbeispielen klar geworden sein, dass das Kalman-Filter kein Bildverarbeitungsfilter im herkömmlichen Sinn ist, wie z.B. ein Hoch- oder Tiefpassfilter, das direkt auf den pixel-basierten Grauwerten eines Bildes agiert, sondern ein Meta-Filter, das auf einer höheren Abstraktionsebene arbeitet, wie z.B. Kanten, Linien oder Segmenten. Das Kalman-Filter setzt also als Vorverarbeitungsschritte immer die klassischen Methoden der Bildverarbeitung und Mustererkennung voraus.

27.2.2 3D-Rekonstruktion aus Bildfolgen

Die Rekonstruktion der 3-dimensionalen Struktur einer Szene ausgehend von 2-dimensionalen Bildfolgen ist eines der zentralen Probleme der *Computer Vision*. Die 2-dimensionalen Bildfolgen können dabei sowohl von einer bewegten Videokamera als auch von einer Stereokamera (stehend oder bewegt) aufgezeichnet worden sein. Das wesentliche Problem dabei ist, korrespondierende Bildpunkte in aufeinander folgenden Bildern zu finden. Dieses sogenannte Korrespondenzproblem ist vor allem deshalb schwierig zu lösen, weil durch Bildrauschen und Verdeckungen eine Zuordnung von Bildpunkten zwischen zwei Bildern manchmal unmöglich ist, oder weil wegen Mehrdeutigkeiten falsche Zuordnungen erfolgen. Die Lösungansätze für das Korrespondenzproblem kann man einteilen in pixel-basierte Verfahren, bei denen Verschiebungsvektorfelder mit Hilfe des Blockmatchings (Abschnitt 14.5) oder differentieller Methoden (Abschnitt 14.4) bestimmt werden, sowie in merkmal-basierte Verfahren, bei denen versucht wird, Merkmale wie Kanten oder Linienzüge einander zuzuordnen. Für die weitere Verarbeitung dieser Informationen in einem Kalman-Filter wird davon ausgegangen, dass das Korrespondenzproblem gelöst ist. Fehlerhafte Korrespondenzen werden als Messrauschen aufgefasst.

Die 3D-Rekonstruktion aus Bildfolgen ist eine Erweiterung des Trackings, bei der von einer statischen Szene ausgegangen wird, durch die sich eine Kamera bewegt. Während beim Tracking nur die 3-dimensionale Bewegung der Kamera geschätzt wird, soll bei der 3D-Rekonstruktion sowohl die Bewegung der Kamera, als auch die 3-dimensionale Struktur der Szene optimal geschätzt werden. Dazu wird der Zustandsvektor, der beim Tracking nur die Bewegungsparameter enthält, um eine bestimmte Anzahl von Parametern erweitert, die die räumliche Tiefe der Merkmale beschreiben. Als Eingabewerte liegen wieder Bilder der Kamera aus verschiedenen Aufnahmepositionen vor. Die Aufgabe der Bildvorverarbeitung ist es, die Positionen einer bestimmten Anzahl von Merkmalen in jedem Bild der Sequenz zu berechnen. Aufgrund des oben geschilderten Korrespondenzproblems ist dies eine nichttriviale Aufgabe, deren Lösung in der Regel fehlerbehaftet ist. Aus der Position eines Merkmals in den Kamerabildern kann mit Hilfe der bekannten Projektionseigenschaften der Kamera auf die Position des Merkmals im 3-dimensionalen Raum geschlossen werden. Im Prinzip kann die räumliche Tiefe von Merkmalen bereits aus zwei Bildern (d.h. einem Stereobildpaar) durch Triangulation berechnet werden. Wegen der fehlerbehafteten Korrespondenzen ist es jedoch sinnvoll, Bilder aus weiteren Aufnahmepositionen zur Verbesserung der 3D-Struktur-Schätzung im Rahmen des Kalman-Filters heranzuziehen. Bei einer ausreichenden Anzahl an Bildern aus geeigneten Aufnahmepositionen kann bei nicht zu stark verwinkelten Szenen sowohl die 3D-Struktur der Szene als auch die Bewegung der Kamera rekonstruiert werden ([Ying04], [Alon00], [Azar95]). Falls bestimmte Merkmale nicht aus mehreren unterschiedlichen Kamerapositionen einzusehen sind, ist die räumliche Tiefe dieses Merkmals ein nicht beobachtbarer Zustand des Systems.

Das Kalman-Filter bietet sich in der 3D-Rekonstruktion von Szenen natürlich auch besonders gut an, wenn Daten von verschiedenen Sensoren gleichzeitig gemessen werden. Die 3D-Rekonstruktion ausschließlich aus Videobildsequenzen ist zwar, wie eben geschildert, durchaus möglich, aber verhältnismäßig schwierig und fehleranfällig. Wenn es die

Anwendung erlaubt, werden deshalb neben einer Videokamera häufig zusätzliche Sensoren, wie Laserentfernungsmesser, Radar oder Ultraschallentfernungsmesser eingesetzt, um direkt eine Tiefenkarte der Szene erzeugen zu können. Zur Fusion der verschiedenen Sensordaten kann ein Kalman-Filter eingesetzt werden, in dem die Beiträge der einzelnen Sensoren – gewichtet mit ihrer jeweiligen Messgenauigkeit – in eine optimale Schätzung der 3-dimensionalen Geometrie der Szene eingehen [Vanp93].

27.2.3 Bilddatencodierung

Die Grundidee moderner Standards zur Codierung von Bewegtbildern, wie MPEG-1/2 oder MPEG-4, besteht darin, die Ähnlichkeit zweier aufeinander folgender Bilder zur Datenreduktion auszunutzen. Häufig unterscheidet sich ein Bild zum Zeitpunkt t vom vorhergehenden Bild zum Zeitpunkt $t-1$ nur durch geringfügige Objekt- oder Kamerabewegungen. Aus den beiden Bildern lässt sich durch Blockmatching ein Bewegungsvektorfeld berechnen (Abschnitt 14.5), das angibt, wie sich die $8 \cdot 8$ Pixel großen Blöcke des Bildes von $t-1$ bis t bewegen. Folglich kann man aus dem Bild zum Zeitpunkt $t-1$ mit Hilfe der Bewegungsvektoren das Bild zum Zeitpunkt t bis auf gewisse Differenzen prädizieren. Dieses Bild bezeichnet man als *bewegungskompensiert*. Zur Codierung des Bildes t benötigt man also nur die Bewegungsvektoren, sowie das sehr gut komprimierbare Differenzbild zwischen dem bewegungskompensierten Bild und dem Originalbild. Auf diese Weise kann man häufig eine Datenkompression um den Faktor 10 oder besser erreichen. Zur Rekonstruktion des Originalbildes rechnet man zunächst aus dem Bild $t-1$ und den Bewegungsvektoren das bewegungskompensierte Bild aus. Durch die anschließende Addition des Differenzbildes erhält man wieder das Originalbild t (Bild 27.4).

Die erreichbare Kompressionsrate hängt bei dem geschilderten Verfahren von der Qualität der Bewegungsschätzung ab. Da das Kalman-Filter, wie in den vorigen Abschnitten dargestellt, hervorragend für die Bewegungsschätzung geeignet ist, können damit bessere Ergebnisse erzielt werden, als mit dem Blockmatching. Es gibt zahlreiche Vorschläge für den Einsatz des Kalman-Filters in der Bilddatencodierung, von denen einer stellvertretend kurz vorgestellt wird [Calv00]. Grundannahme bei diesem Verfahren, das sich insbesondere für Videokonferenzen sehr gut eignet, ist eine ruhende Kamera bzw. ein ruhender Hintergrund, vor dem sich einige starre Objekte (z.B. Kopf, Arm, Oberkörper) bewegen. Die Objektbewegungen besitzen sechs Freiheitsgrade, drei für die Translation und drei für die Rotation. Zusätzlich können noch einige spezielle Merkmale (wie z.B. Augen und Mund) als verformbare Objekte mit Hilfe von Formparametern beschrieben werden. Ein erweitertes Kalman-Filter (EKF, Abschnitt 27.3.6) wird zur Schätzung der Bewegungen und der Formparameter eingesetzt. Dieses Verfahren lässt sich auch sehr gut zur Animation von hybriden synthetisch/natürlichen Objekten einsetzen, die durch ein 3-dimensionales Polygonnetz beschrieben werden, auf das eine Foto-Textur gemappt wird (Band I Kapitel 13). Durch das Kalman-Filter wird in diesem Fall die Bewegung der Eckpunkte des Polygonnetzes geschätzt. Für das erste Bild muss die Foto-Textur und das Polygonnetz übertragen werden, für alle weiteren Bilder nur noch die Bewegung der Eckpunkte des Polygonnetzes.

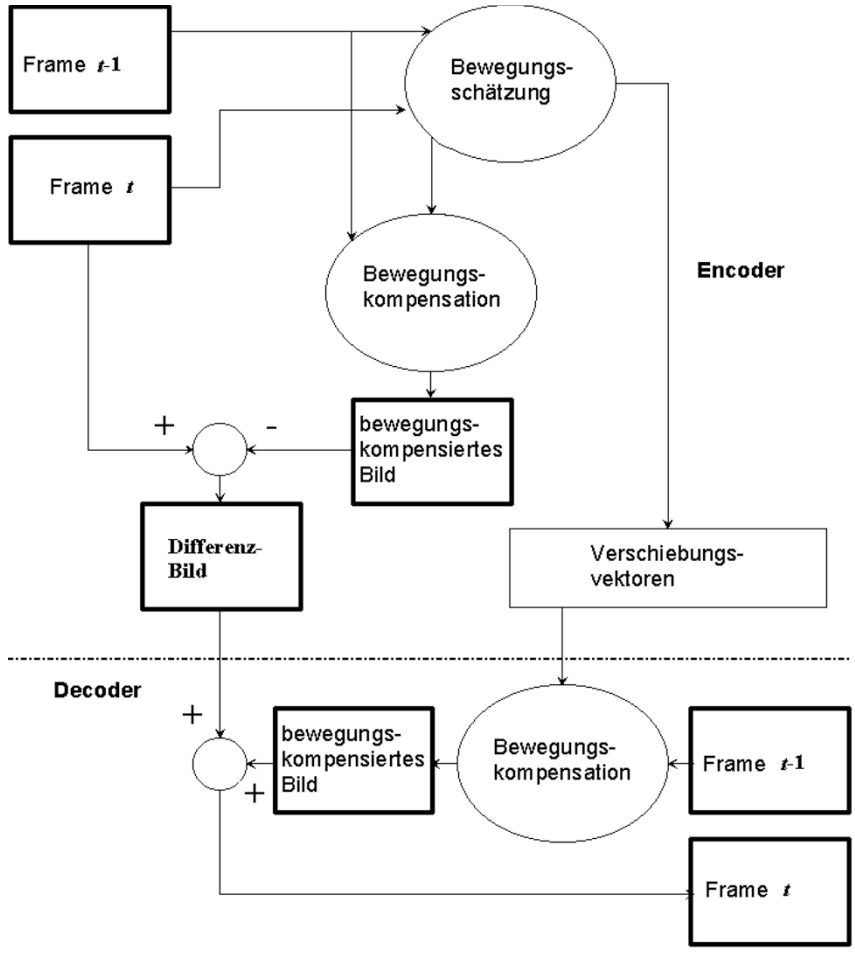


Bild 27.4: Das Blockschaltbild für einen Encoder/Decoder mit Bewegungsschätzung und Bewegungskompensation.

Auf der Empfängerseite werden die Bilder mit Hilfe einer schnellen Grafikhardware wieder erzeugt. Mit solchen hybriden Codierverfahren, wie sie seit dem Codierstandard MPEG-4 möglich sind, lassen sich sehr hohe Kompressionsfaktoren erzielen.

27.3 Theorie des diskreten Kalman-Filters

27.3.1 Das System

Das Kalman-Filter ist eine Lösung des Problems, einen optimalen Schätzwert für den Zustand $\mathbf{x}$ eines Systems zu liefern, welches durch folgende zeitdiskrete lineare stochastische Differenzengleichung beschreibbar ist:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \cdot \mathbf{x}_k + \mathbf{B}_k \cdot \mathbf{u}_k + \mathbf{w}_k \tag{27.1}$$

Dabei ist $\mathbf{x}_k$ der n-dimensionale Zustands-Vektor des Systems und $\mathbf{u}_k$ der l-dimensionale Steuer-Vektor zum Zeitpunkt k. Die $(n \cdot n)$-Matrix $\mathbf{A}_k$ in (27.1), auch Systemmatrix genannt, bildet den Systemzustand vom vorherigen Zeitschritt k auf den aktuellen Zeitschritt $k+1$ ab, und zwar ohne eine externe Kraft $\mathbf{u}_k$, sowie ohne Systemrauschen $\mathbf{w}_k$. Die $(n \cdot l)$-Matrix $\mathbf{B}_k$, auch Eingabematrix genannt, bildet den optionalen Steuer-Vektor $\mathbf{u}_k$ auf den Systemzustand ab. Die n-dimensionale, $N(0, \mathbf{Q}_k)$-verteilte Zufallsvariable $\mathbf{w}_k$ repräsentiert das Systemrauschen ($N(0, \mathbf{Q}_k)$ steht für Normalverteilung mit Mittelwert 0 und Varianz $\mathbf{Q}_k$).

27.3.2 Die Messung

Beobachtbar seien nur die Messungen $\mathbf{y_k}$, die über folgende Messgleichung aus dem Zustand des Systems hervorgehen:

$$\mathbf{y}_k = \mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{v}_k \tag{27.2}$$

$\mathbf{y}_k$ ist ein m-dimensionaler Mess-Vektor zum Zeitpunkt k. Die $(m \cdot n)$-Matrix $\mathbf{H}_k$ in (27.2), auch Messmatrix genannt, bildet den Systemzustand beim Zeitschritt k auf den Messwert $\mathbf{y}_k$ ab, und zwar ohne Messrauschen $\mathbf{v}_k$. Die m-dimensionale, $N(0, \mathbf{R}_k)$-verteilte Zufallsvariable $\mathbf{v}_k$ repräsentiert das Messrauschen ($N(0, \mathbf{R}_k)$ steht für Normalverteilung mit Mittelwert 0 und Varianz $\mathbf{R}_k$).
Es gelten folgende Annahmen:

- Systemrauschen $\mathbf{w}_k$ und Messrauschen $\mathbf{v}_k$ seien unabhängig voneinander.
- Es wird angenommen, dass die Größen $\mathbf{w}_k$, $\mathbf{v}_k$, $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{H}_k$ konstant sind. In der Praxis kann sich das Rauschen bzw. die System-, Eingabe- und Messmatrix mit jedem Zeitschritt ändern.

27.3.3 Die Schätzfehler-Gleichungen

Man definiert $\hat{\mathbf{x}}_k^-$ als den a priori Schätzwert für den Systemzustand zum Zeitschritt k, wobei alle Messungen bis zum Zeitschritt $k-1$ einfließen können. „A priori“ bedeutet in diesem Zusammenhang „im Vorhinein“, d.h. $\hat{\mathbf{x}}_k^-$ ist der prädizierte Schätzwert für den

Systemzustand. Man definiert $\hat{\mathbf{x}}_k^+$ als den a posteriori Schätzwert für den Systemzustand zum Zeitschritt k, wobei die aktuelle Messung zum Zeitschritt k eingeflossen ist. „A posteriori“ bedeutet in diesem Zusammenhang „im Nachhinein“, d.h. $\hat{\mathbf{x}}_k^+$ ist der mit der neuen Messung korrigierte Schätzwert für den Systemzustand.

Jetzt kann man die a priori und a posteriori Schätzfehler definieren als:

$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^- \tag{27.3}$$

$$\mathbf{e}_k^+ = \mathbf{x}_k - \hat{\mathbf{x}}_k^+ \tag{27.4}$$

sowie die a priori und a posteriori Schätzfehlerkovarianzen als:

$$\mathbf{P}_k^- = E\left[\mathbf{e}_k^- \cdot \mathbf{e}_k^{-T}\right] \tag{27.5}$$

$$\mathbf{P}_k^+ = E\left[\mathbf{e}_k^+ \cdot \mathbf{e}_k^{+T}\right] \tag{27.6}$$

Hinweis: „E“ steht für Erwartungswert. Bei einer diskreten Verteilung α_i bedeutet das: $E(\mathbf{x}) = \sum \alpha_i \cdot \mathbf{x}_i$. Bei einer Gleichverteilung entspricht der Erwartungswert dem Mittelwert.

27.3.4 Optimales Schätzfilter von Kalman

Zunächst muss eine Formel zur Berechnung des a posteriori Schätzwerts $\hat{\mathbf{x}}_k^+$ des Systemzustands hergeleitet werden, und zwar auf Basis des „a priori“ Schätzwerts $\hat{\mathbf{x}}_k^-$ und der gewichteten Differenz zwischen dem aktuellen Messwert $\mathbf{y}_k$ und dem vorhergesagten Messwert $\mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-$:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-\right) \tag{27.7}$$

Die Differenz $\left(\mathbf{y}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-\right)$ ist ein Maß für die Güte der Vorhersage des Systemzustands. Die Differenz wird null, falls der vorhergesagte Messwert exakt mit dem aktuellen Messwert übereinstimmt.

Die $(n \cdot m)$-Matrix $\mathbf{K}_k$ in (27.7), auch Matrix der Kalman-Verstärkungsfaktoren genannt, wird so gewählt, dass die a posteriori Schätzfehlerkovarianz $\mathbf{P}_k^+$ minimal wird. Eine mögliche Form von $\mathbf{K}_k$, die $\mathbf{P}_k^+$ minimiert ist:

$$\mathbf{K}_k = \mathbf{P}_k^- \cdot \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} = \frac{\mathbf{P}_k^- \cdot \mathbf{H}_k^T}{\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k} \tag{27.8}$$

Hinweis: Eine ausführliche Herleitung von Gleichung (27.8) ist in [Mayb79] zu finden.

Zur Vertiefung des Verständnisses betrachtet man die Grenzfälle der Gleichung (27.8):

- Falls die Varianz des Messfehlers $\mathbf{R}_k$ gegen null geht, wird die Matrix der Kalman-Verstärkungsfaktoren $\mathbf{K}_k$ maximal. Anders ausgedrückt: Falls die Varianz des Messfehlers $\mathbf{R}_k$ gegen Null geht, wird der aktuellen Messung $\mathbf{y}_k$ immer mehr „vertraut“, der prädizierten Messung $\mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-$ wird dagegen immer weniger „vertraut“.

- Falls die a priori Schätzfehlerkovarianz $\mathbf{P}_k^-$ gegen null geht, wird die Matrix der Kalman-Verstärkungsfaktoren $\mathbf{K}_k$ ebenfalls null. Anders ausgedrückt: Falls die a priori Schätzfehlerkovarianz $\mathbf{P}_k^-$ gegen null geht, wird der aktuellen Messung $\mathbf{y}_k$ immer weniger „vertraut“, der prädizierten Messung $\mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-$ wird dagegen immer mehr „vertraut“.

27.3.5 Gleichungen des Kalman-Filters

Das Kalman-Filter schätzt die Entwicklung eines Systemzustands in einer Art Regelschleife durch Rückkopplung über die Messwerte: das Filter schätzt den Systemzustand zu einem bestimmten Zeitpunkt ab und benützt anschließend die Rückkopplung in Form einer verrauschten Messung, um den Systemzustand im nächsten Zeitschritt etwas besser abzuschätzen. Danach kommt wieder die nächste Messung usw. in einer Rekursion. Die Gleichungen des Kalman-Filters kann man deshalb in zwei Gruppen einteilen:

- Die Gleichungen für die zeitliche Entwicklung des Systems (Prädiktion)
- Die Gleichungen für die Rückkopplung durch die Messung, d.h. die Berücksichtigung einer neuen Messung, um aus der a priori Schätzung eine verbesserte a posteriori Schätzung abzuleiten (Korrektur mit der Messung).

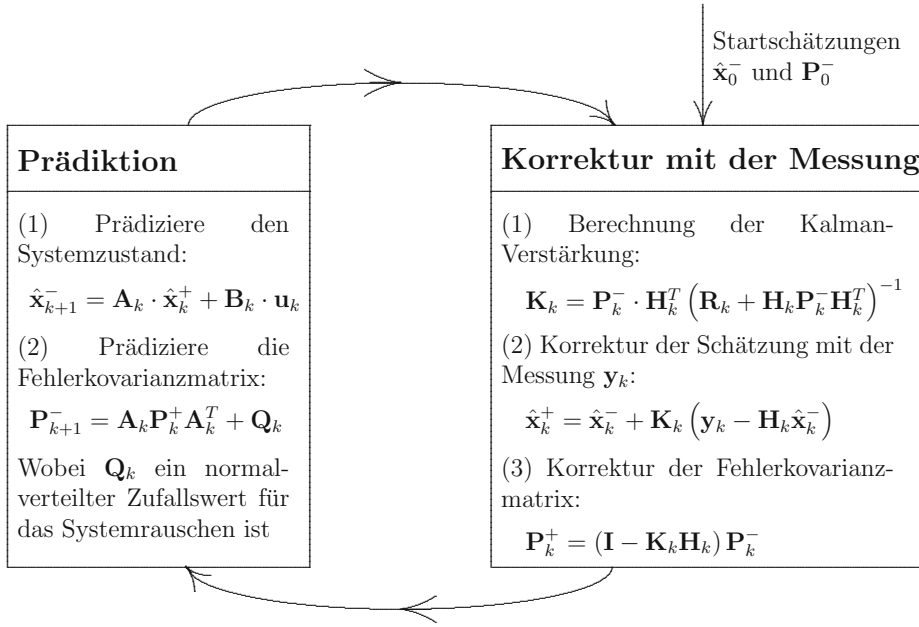


Bild 27.5: Die Regelschleife des Kalman-Filters: Prädiktion und Korrektur mit der Messung.

Eigenschaften des Kalman-Filters und Hinweise:

- Das Kalman-Filter ist rekursiv:
  nach jedem Prädiktions- und Messschritt wird der Vorgang wiederholt, wobei die vorhergehenden a posteriori Schätzungen benützt werden, um die nächsten a priori Schätzungen zu bestimmen.

- In der Praxis ist ein rekursives Filter sehr gut brauchbar, da es mit jeder neuen Messung auch einen neuen, d.h. aktuellen Schätzwert liefert (im Gegensatz z.B. zum Wiener Filter, das immer alle bisherigen Messungen gleichzeitig für die Schätzung braucht). Das Kalman-Filter ist daher prädestiniert für den Einsatz in Echtzeit-Anwendungen.

- Falls die Varianzen des Mess- bzw. Systemrauschens konstant sind, nähern sich sowohl die Matrix der Kalman-Verstärkungsfaktoren $\mathbf{K}_k$ als auch die (a priori und a posteriori) Kovarianzen des Schätzfehlers $\mathbf{P}_k$ sehr schnell konstanten Werten an (die auch vorab, d.h. offline berechnet werden können).

- In der Praxis sind die Messfehler $\mathbf{R}_k$ jedoch meist nicht konstant: falls sich z.B. der Kontrast in Videobildern vorübergehend abschwächt, wird die Objekterkennung entsprechend unsicherer, so dass die entsprechenden Messwerte im Kalman-Filter nicht so stark berücksichtigt werden sollten. Man kann also z.B. den (messbaren) Kontrast als Maß für die Varianz des Messfehlers benutzen und somit den gewünschten Effekt erzielen.

- Auch das Systemrauschen $\mathbf{Q}_k$ ändert sich in manchen Anwendungen dynamisch: z.B. kann beim Tracking einer Person die Größe von $\mathbf{Q}_k$ reduziert werden, falls die Person sich gerade langsam zu bewegen scheint, und falls sich die Bewegungen der Person schnell ändern, wird die Größe von $\mathbf{Q}_k$ erhöht.

- Eine wesentliche Voraussetzung für das Funktionieren eines Kalman-Filters ist die *Beobachtbarkeit* aller Systemzustände durch die Messung [Ludy95]. Falls versucht wird, nicht beobachtbare Systemzustände zu schätzen, divergiert das Kalman-Filter.

- Bei der praktischen Realisierung von Kalman-Filtern treten häufiger numerische Probleme auf. Ursachen dafür können z.B. Singularitäten bei der Matrix-Invertierung für die Berechnung der Kalman-Verstärkung $\mathbf{K}_k = \mathbf{P}_k^- \cdot \mathbf{H}_k^T \left(\mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T \right)^{-1}$ sein, unzureichende Maschinengenauigkeit oder ungünstige Implementierung. Abhilfe schaffen hier Methoden zur numerischen Stabilisierung, wie sie in [Scho03] beschrieben sind.

27.3.6 Das erweiterte Kalman-Filter (EKF)

Das bisher hergeleitete Kalman-Filter basiert auf einer linearen stochastischen Differenzengleichung. In der Praxis sind die System- und Messgleichungen jedoch meist nicht-linear ! Die allgemeine Form einer nicht-linearen Systemgleichung lautet:

$$\mathbf{x}_{k+1} = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right) \tag{27.9}$$

Die allgemeine Form einer nicht-linearen Messgleichung lautet:

$$\mathbf{y}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{v}_k\right) \tag{27.10}$$

wobei die nicht-lineare Übergangsfunktion $\mathbf{f}$ den Systemzustand vom vorherigen Zeitschritt k auf den aktuellen Zeitschritt $k+1$ abbildet, und zwar mit Systemrauschen $\mathbf{w}_k$. Die nicht-lineare Messfunktion $\mathbf{h}$ bildet den Systemzustand beim Zeitschritt k auf den Messwert $\mathbf{y}_k$ ab, und zwar mit Messrauschen $\mathbf{v}_k$.

In ähnlichem Sinne wie man eine nicht-lineare Funktion um einen bestimmten Punkt in eine Taylor-Reihe entwickeln und nach dem linearen Glied abbrechen kann, linearisiert man beim erweiterten Kalman-Filter (EKF) die nicht-lineare Übergangs- und Messfunktion jeweils um den aktuellen Schätzwert:

$\delta\mathbf{A}_k$ ist die Jakobi-Matrix der partiellen Ableitungen von $\mathbf{f}$ nach $\mathbf{x}$ zur Zeit k:

$$\delta\mathbf{A}_k = \frac{\partial}{\partial\mathbf{x}}\mathbf{f}\left(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, 0\right) \quad \text{bzw.} \quad \mathbf{A}_{k,[i,j]} = \frac{\partial}{\partial\mathbf{x}_{[j]}}\mathbf{f}_{[i]}\left(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, 0\right) \tag{27.11}$$

$\delta\mathbf{W}_k$ ist die Jakobi-Matrix der partiellen Ableitungen von $\mathbf{f}$ nach $\mathbf{w}$ zur Zeit k:

$$\delta\mathbf{W}_k = \frac{\partial}{\partial\mathbf{w}}\mathbf{f}\left(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, 0\right) \quad \text{bzw.} \quad \mathbf{W}_{k,[i,j]} = \frac{\partial}{\partial\mathbf{w}_{[j]}}\mathbf{f}_{[i]}\left(\hat{\mathbf{x}}_k^+, \mathbf{u}_k, 0\right) \tag{27.12}$$

$\delta\mathbf{H}_k$ ist die Jakobi-Matrix der partiellen Ableitungen von $\mathbf{h}$ nach $\mathbf{x}$ zur Zeit k:

$$\delta\mathbf{H}_k = \frac{\partial}{\partial\mathbf{x}}\mathbf{h}\left(\hat{\mathbf{x}}_k^-, 0\right) \quad \text{bzw.} \quad \mathbf{H}_{k,[i,j]} = \frac{\partial}{\partial\mathbf{x}_{[j]}}\mathbf{h}_{[i]}\left(\hat{\mathbf{x}}_k^-, 0\right) \tag{27.13}$$

$\delta\mathbf{V}_k$ ist die Jakobi-Matrix der partiellen Ableitungen von $\mathbf{h}$ nach $\mathbf{v}$ zur Zeit k:

$$\delta\mathbf{V}_k = \frac{\partial}{\partial\mathbf{v}}\mathbf{h}\left(\hat{\mathbf{x}}_k^-, 0\right) \quad \text{bzw.} \quad \mathbf{V}_{k,[i,j]} = \frac{\partial}{\partial\mathbf{v}_{[j]}}\mathbf{h}_{[i]}\left(\hat{\mathbf{x}}_k^-, 0\right) \tag{27.14}$$

Eine exakte Herleitung des erweiterten Kalman-Filters ist z.B. in [Welc95] zu finden.

Das erweiterte Kalman-Filter läuft nach der Linearisierung im Prinzip genauso ab, wie das Standard-Kalman-Filter für lineare Probleme. Einziger Unterschied: Die Jacobi-Matrizen müssen für jeden Zeitschritt neu berechnet werden, so dass ein entsprechend größerer Rechenaufwand nötig ist.

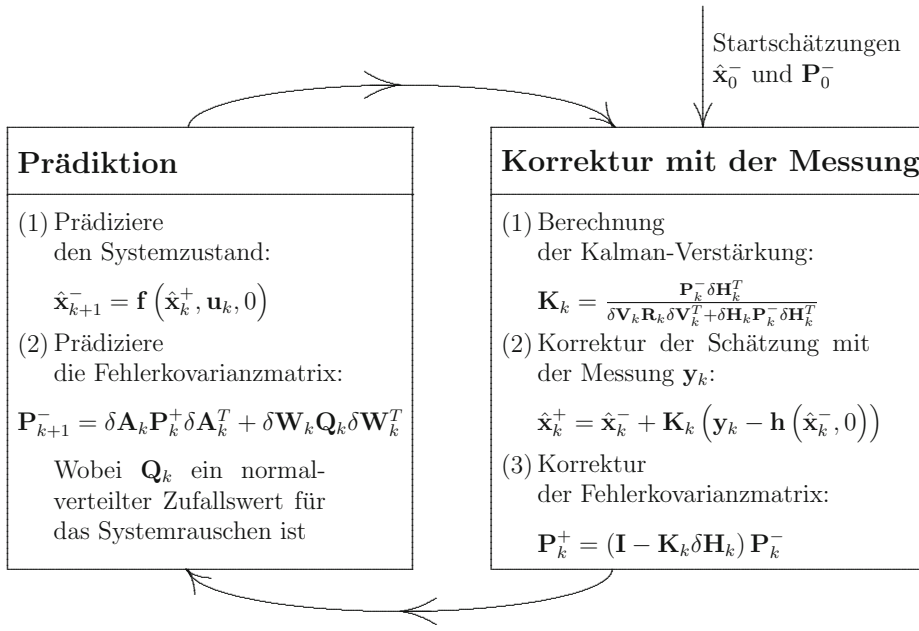


Bild 27.6: Die Regelschleife des erweiterten Kalman-Filters (EKF): Prädiktion und Korrektur mit der Messung. Die Jacobi-Matrizen müssen für jeden Zeitschritt neu berechnet werden.

27.4 Konkrete Beispiele

In diesem Abschnitt werden drei konkrete Beispiele für ein Kalman-Filter vorgestellt, um ein besseres Gefühl für die Auslegung, die Funktionsweise und die Leistungsfähigkeit des Konzepts zu vermitteln. Das erste Beispiel, die Schätzung eines konstanten Wertes auf der Basis verrauschter Messwerte, ist extrem einfach und dient dazu, ein Gefühl für das Grundprinzip des Kalman-Filters zu entwickeln. Es ist empfehlenswert, dieses Beispiel selber nachzurechnen. Das zweite Beispiel, die Positions- und Geschwindigkeitsschätzung einer konstant beschleunigten Bewegung, führt die im ersten Abschnitt dieses Kapitels anhand eines Ballwurfs vorgestellte Grundidee des Kalman-Filters aus. Solche Objektverfolgungsaufgaben, auch *Tracking* genannt, sind ein sehr häufiger Fall für die Anwendung des Kalman-Filters in der Bildverarbeitung. Außerdem wird an dem zweiten Beispiel noch die besondere Leistungsfähigkeit des Kalman-Filters im Hinblick auf Messstörungen dargestellt. Anhand des dritten Beispiels wird schließlich gezeigt, dass das Kalman-Filter auch in der umgekehrten Problemstellung hervorragend eingesetzt werden kann, nämlich bei der Bestimmung der Beobachterposition auf der Basis bekannter Objektpositionen, auch *Navigation* genannt.

27.4.1 Schätzung einer verrauschten Konstante

Der Systemzustand x wird durch einen einzigen Wert (z.B. eine eindimensionale Position) beschrieben, der zeitlich konstant sei. Die Systemmatrix A_k degeneriert in diesem Fall zu einem einzigen Wert, der wegen der zeitlichen Konstanz auch noch 1 ist, d.h. $A_k = 1$ für alle k. Ein Steuer-Vektor existiert ebenfalls nicht, d.h. $u_k = 0$. Damit vereinfacht sich die Systemzustandsgleichung zu:

$$x_{k+1} = A_k \cdot x_k + B_k \cdot u_k + w_k = x_k + w_k \tag{27.15}$$

Man kann die Konstante zu bestimmten Zeitpunkten direkt messen, allerdings sind die Messwerte verrauscht durch ein weißes Messrauschen mit einer Standardabweichung von 0,1. Die Messmatrix H_k degeneriert in diesem Fall zu einem einzigen Wert, der wegen der direkten Messbarkeit des Systemzustands auch noch 1 ist, d.h. $H_k = 1$ für alle k. Damit vereinfacht sich die Messgleichung zu:

$$y_k = H_k \cdot x_k + v_k = x_k + v_k \tag{27.16}$$

Die Kalman-Filter-Gleichungen für den Prädiktionsschritt und den Korrekturschritt durch die Messung sind in Bild 27.7 zusammengefasst:

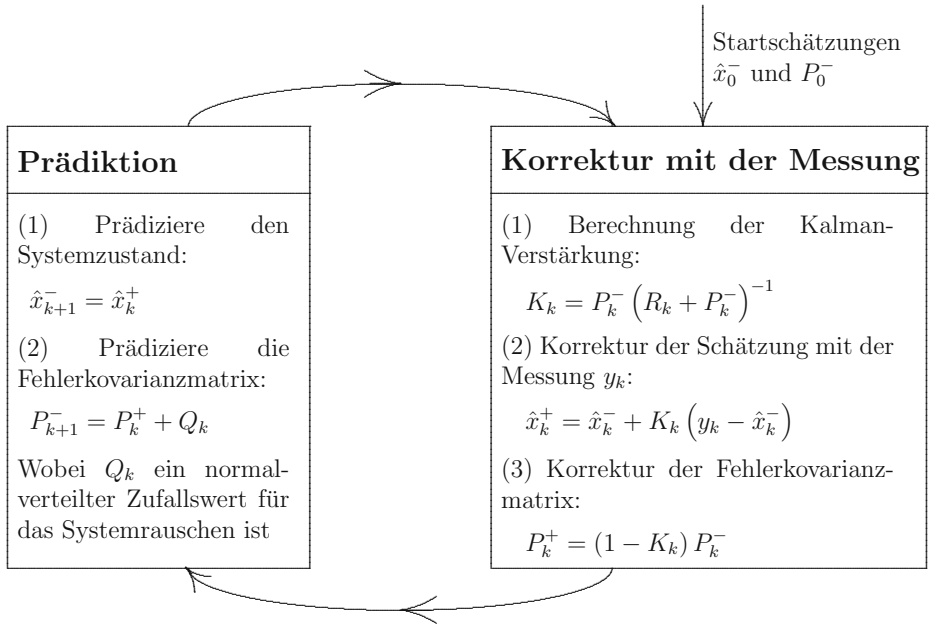


Bild 27.7: Die Regelschleife des Kalman-Filters für die Schätzung einer verrauschten Konstante. Man beachte, dass die Variablen nicht mehr fettgedruckt sind, d.h. es sind nur noch Skalare anstatt Vektoren oder Matrizen.

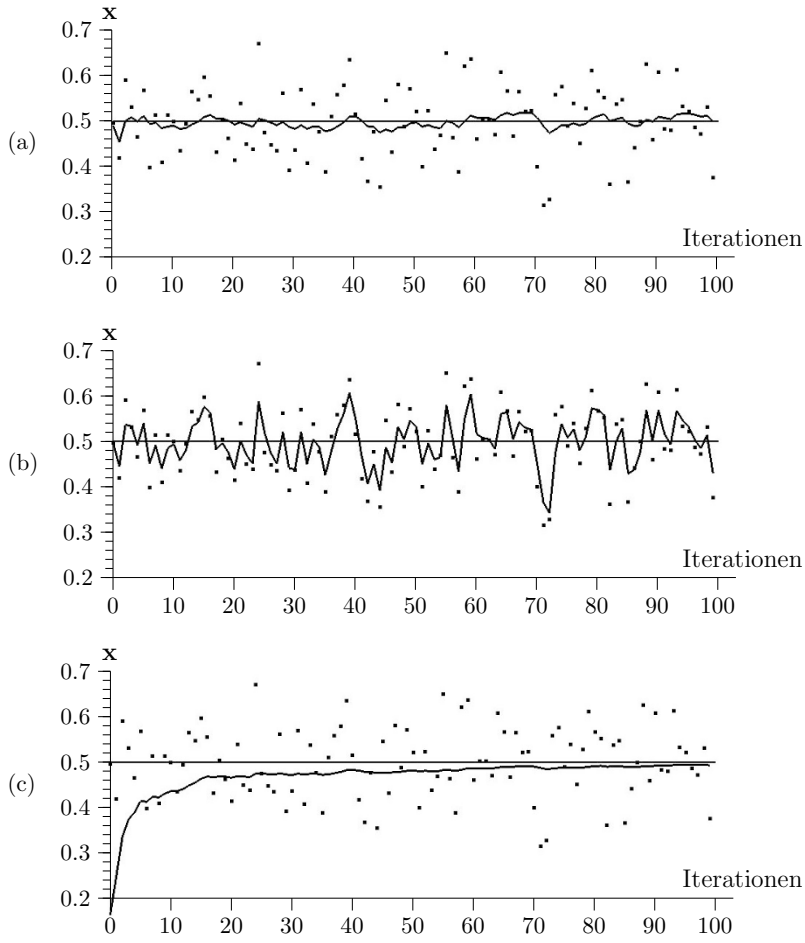


Bild 27.8: Schätzung einer verrauschten Konstante: (a) Simulation 1: $R_k = 0,1^2 = 0,01$. Der Systemzustand $\mathbf{x} = +0,5$ ist durch die gerade Linie dargestellt, die Punkte markieren die verrauschten Messwerte und der Schätzwert des Filters ist die verbleibende dicke Kurve. (b) Simulation 2: $R_k = 0,01^2 = 0,0001$. Das Filter reagiert sehr schnell auf die einzelnen Messwerte, woraus eine größere Varianz des Schätzwerts für den Systemzustand folgt. (c) Simulation 3: $R_k = 1$. Das Filter reagiert langsamer auf die einzelnen Messwerte, woraus eine geringere Varianz des Schätzwerts für den Systemzustand folgt.

Die Simulationsparameter:
Der Systemzustand x wird zufällig ausgewählt und beträgt: $x = +0,5$ (dies ist der „wahre“ Wert, deshalb ist kein „Dach“ auf dem x). Da man über den „wahren“ Systemzustand nichts weiß, beginnt man mit einer beliebigen Startschätzung des Systemzustands

z.B. $\hat{x}_0^- = 0$. Aus dem selben Grund, nämlich der Unkenntnis des „wahren“ Systemzustands, beginnt man mit einer beliebigen Fehlerkovarianz größer Null, z.B. $P_0^- = 1$. Da man weiß, dass es sich beim Systemzustand um eine Konstante handelt, könnte man für das Systemrauschen Null annehmen. Dies würde allerdings bei einem niedrigen Messrauschen zum „Einschlafen“ des Kalman-Filters führen (d.h. neue Messwerte würden kaum mehr in den Schätzwert einfließen). Deshalb wählt man einen sehr kleinen Wert für das Systemrauschen, z.B. $Q_k = 10^{-5}$.

Jetzt werden 100 verschiedene „Messwerte“ y_k simuliert, indem man zum wahren Systemzustand jeweils einen normalverteilten Zufallswert mit einer Standardabweichung von 0,1 addiert. Diese 100 „Messwerte“ werden für die Simulationen 1,2,3 benutzt, um die Auswirkungen unterschiedlicher Parametereinstellungen des Kalman-Filters besser vergleichen zu können.

Simulation 1: Varianz des Messfehlers $R_k = (0,1)^2 = 0,01$ (Bild 27.8-a).
Dies ist das „wahre“ Messrauschen, da der Messfehler eine Standardabweichung von 0,1 hat. Deshalb erwartet man für diese Parametereinstellung die beste Leistung des Kalman-Filters im Sinne eines Gleichgewichts zwischen Reaktionsvermögen auf neue Messwerte und Varianz des Schätzfehlers (dies wird offensichtlich bei den Simulationen 2 und 3, bei denen Abweichungen vom Gleichgewicht in beide Richtungen auftreten).

Simulation 2: Varianz des Messfehlers $R_k = (0,01)^2 = 0,0001$ (Bild 27.8-b).
d.h. die angenommene Varianz des Messfehlers ist um Faktor 100 kleiner als bei Simulation 1 (dem „wahren“ Wert). Es wird also (zu Unrecht) angenommen, dass die Messungen sehr genau sind und man deshalb jeder einzelnen Messung relativ gut vertrauen kann. Folglich reagiert das Kalman-Filter sehr schnell auf die einzelnen Messwerte, die jetzt zu einem großen Anteil in den neuen Schätzwert für den Systemzustand einfließen.

Simulation 3: Varianz des Messfehlers $R_k = 1$ (Bild 27.8-c).
d.h. die angenommene Varianz des Messfehlers ist um Faktor 100 größer als bei Simulation 1 (dem „wahren“ Wert). Es wird also (zu Unrecht) angenommen, dass die Messungen sehr stark verrauscht sind und man deshalb jeder einzelnen Messung relativ wenig Vertrauen entgegen bringen kann. Folglich lässt das Kalman-Filter die einzelnen Messwerte nur sehr langsam, also nur mit einem geringen Anteil in den neuen Schätzwert für den Systemzustand einfließen.

Die Wahl der anfänglichen Fehlerkovarianz P_0^- ist unkritisch (solange $P_0^- \neq 0$), da der Wert der Fehlerkovarianz sehr schnell von 1 auf ca. 0,0002 konvergiert.

27.4.2 Schätzung einer Wurfparabel

Im ersten Abschnitt dieses Kapitels wurde anhand eines Ballwurfs die Grundidee des Kalman-Filters vorgestellt. Jetzt wird für dieses Beispiel, in dem es um die Positions- und Geschwindigkeitsschätzung einer konstant beschleunigten Bewegung geht, ein Kalman-Filter ausgelegt. Als Input liegen Messdaten zur Position des Objekts zu diskreten Zeitpunkten vor. Die Aufgabe, die Bahn eines Objekts im Raum zu verfolgen, auch *Tracking* genannt, ist ein sehr häufiger Fall für die Anwendung des Kalman-Filters in der Bildverarbeitung.

Hier wird der Fall der Bewegung eines Massepunkts mit konstanter Beschleunigung (der Erdbeschleunigung g) behandelt. Der Massepunkt habe eine bestimmte Anfangsgeschwindigkeit $\mathbf{v}_0 > 0$, die Anfangsposition sei im Ursprung des Koordinatensystems. Die ideale Bahn des Massepunkts, eine Wurfparabel, ist aus der Physik bekannt. Die Bewegung lässt sich also in zwei Dimensionen x, y beschreiben, wobei x nach rechts und y nach oben bzgl. des Beobachters zeigen soll. Als Systemzustände werden die Position und die Geschwindigkeit des Massepunkts gewählt, d.h. $\mathbf{x} = (x, y, v_x, v_y)^T$, die treibende Kraft ist die Erdbeschleunigung, d.h. $\mathbf{u} = (0, -g)^T$. Die stochastische Systemzustandsgleichung lautet daher:

$$\begin{array}{rcccccccc} \mathbf{x}_{k+1} & = & \mathbf{A}_k & \cdot & \mathbf{x}_k & + & \mathbf{B}_k & \cdot & \mathbf{u}_k + \mathbf{w}_k \end{array} \quad (27.17)$$

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_k + \begin{pmatrix} 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 \\ 0 & \Delta t \end{pmatrix} \cdot \begin{pmatrix} 0 \\ -g \end{pmatrix} + \mathbf{w}_k$$

Man kann die Position des Objekts zu bestimmten Zeitpunkten direkt messen, allerdings sind die Messwerte in y-Richtung verrauscht durch ein weißes Messrauschen mit einer Standardabweichung von 0,1. Die Messgleichung lautet in diesem Fall:

$$\begin{array}{rcccc} \mathbf{y}_k & = & \mathbf{H}_k & \cdot & \mathbf{x}_k & + \mathbf{v}_k \end{array} \quad (27.18)$$

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_k + \mathbf{v}_k$$

Die Kalman-Filter-Gleichungen für den Prädiktionsschritt und den Korrekturschritt durch die Messung lauten genau so, wie die in Bild 27.5 dargestellten.

Die Simulationsparameter:
Der Systemanfangszustand lautet: $\mathbf{x} = (x, y, v_x, v_y)^T = (0, 0, 0.5 \cdot v_0, 0.86 \cdot v_0)^T$ (dies ist der „wahre“ Wert, deshalb ist kein „Dach“ auf dem x). Da man über den „wahren“ Systemzustand nichts weiß, beginnt man mit irgend einer Startschätzung des Systemzustands, z.B. $\hat{\mathbf{x}}_0^- = (0, 0, 1, 1)^T$. Aus dem selben Grund der Unkenntnis des „wahren“ Systemzustands beginnt man mit einer Fehlerkovarianzmatrix $\mathbf{P}_0^-$, bei der alle Komponenten 1 sind. Kleine

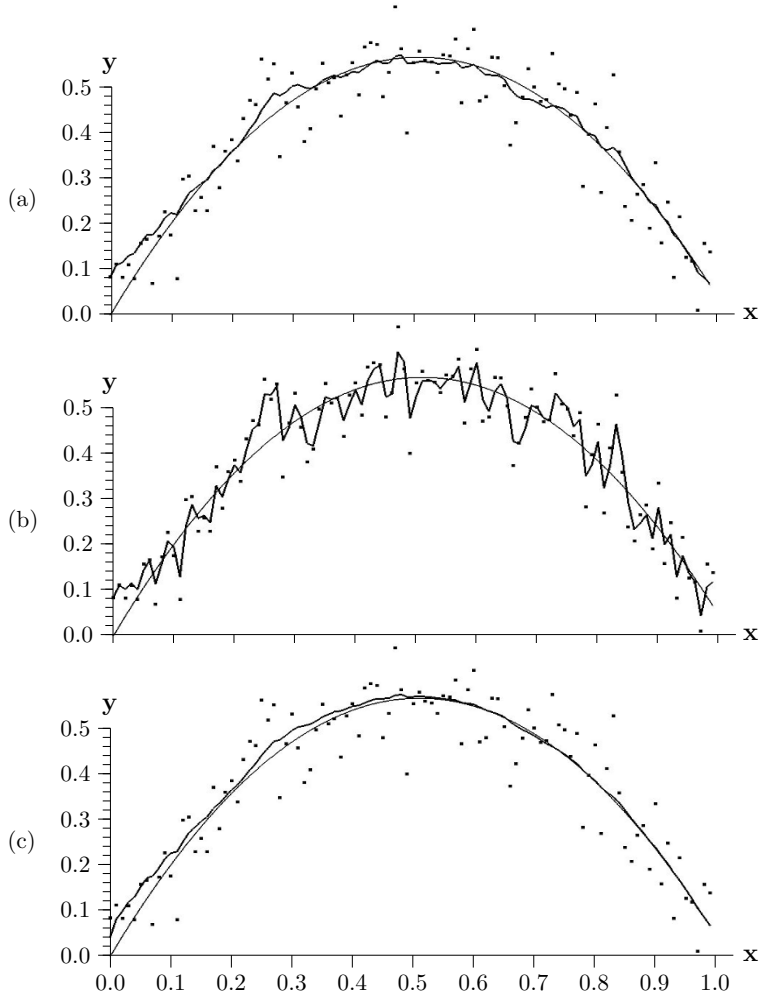


Bild 27.9: Schätzung einer Wurfparabel. (a) Simulation 1: $R_k = 0,1^2 = 0{,}01$. Die Position des Massepunkts ist durch die glatte Parabel dargestellt, die Punkte markieren die verrauschten Messwerte, und der Schätzwert des Filters ist die verbleibende dicke Kurve. (b) Simulation 2: $R_k = 0,01^2 = 0{,}0001$. Das Messrauschen wird um einen Faktor 100 zu klein angenommen, so dass den einzelnen Messwerten zu sehr „vertraut“ wird. Das Filter reagiert daher sehr schnell auf die einzelnen Messwerte, woraus eine größere Varianz des Schätzwerts für den Systemzustand folgt. (c) Simulation 3: $R_k = 1$. Das Messrauschen wird um einen Faktor 100 zu groß angenommen, so dass den einzelnen Messwerten zu wenig „vertraut“ wird. Das Filter reagiert zu langsam auf die einzelnen Messwerte, woraus zwar eine geringere Varianz des Schätzwerts für den Systemzustand folgt, aber auch eine große Trägheit des Filters.

Luftwirbel stören die Bewegung des Massepunkts etwas, deshalb wählt man einen sehr kleinen Wert für die Komponenten der Systemrauschmatrix, z.B. 10^{-5}.

Jetzt werden 100 verschiedene „Messwerte“ $\mathbf{y}_k$ simuliert, indem man zur y-Koordinate des wahren Systemzustands jeweils einen normalverteilten Zufallswert mit einer Standardabweichung von 0,1 addiert. Diese 100 „Messwerte“ werden für die Simulationen 1,2,3 benutzt, um die Auswirkungen unterschiedlicher Parametereinstellungen des Kalman-Filters besser vergleichen zu können.

Die Simulationen 1,2,3 laufen bis auf das geänderte Systemmodell genau so ab, wie im vorigen Abschnitt 27.4.1 beschrieben. Bei der Simulation 1 entspricht die angenommene Varianz des Messfehlers $R_k = (0,1)^2 = 0,01$ (Bild 27.9-a) dem „wahren“ Messrauschen. Bei der Simulation 2 ist die angenommene Varianz des Messfehlers $R_k = 0,0001$ (Bild 27.9-b) um den Faktor 100 kleiner als das „wahre“ Messrauschen. Bei der Simulation 3 ist die angenommene Varianz des Messfehlers $R_k = 1$ (Bild 27.9-c) um den Faktor 100 größer als das „wahre“ Messrauschen.

Besonders interessant ist die Leistungsfähigkeit des Kalman-Filters im Hinblick auf Messstörungen. Eine typische Problematik beim Tracking ist die kurzzeitige Verdeckung der Objektbahn durch irgend welche Hindernisse, wie z.B. Häuser, Bäume oder Hügel. Das Objekt kann daher im Suchfenster, dessen Position und Größe durch das Kalman-Filter vorgegeben wird, nicht detektiert werden. Da das Kalman-Filter ein Modell der zeitlichen Entwicklung des Systems, sowie Daten über die letzte gemessene Position und Geschwindigkeit besitzt, kann die Objektbahn für einige Zeitschritte auch ohne Messwerte prädiziert werden. Nachdem das Objekt im prädizierten Suchfenster wieder detektierbar ist und somit neue (bzw. verlässliche) Messwerte zur Verfügung stehen, kann das Kalman-Filter wieder wie gewohnt abwechselnd Korrektur- und Prädiktionschritte durchführen.

Zur Simulation einer Messstörung wurden die simulierten Messwerte im Intervall $x \in [0.6, 0.7]$ stark verfälscht, d.h. die y-Komponente der Position wurde einfach auf den Wert 0.1 gesetzt und anschließend verrauscht. Bei dieser Messstörung liegen also zu jedem Zeitschritt Messwerte vor, was die Sache eher noch erschwert. In der Simulation 4 (Bild 27.10-a) ist die Varianz des Messfehlers $R_k = 0,01$ konstant, da kein Suchfenster verwendet wurde, in dem die Messwerte liegen müssen. Dies führt dazu, dass die geschätzte Position des Objekts den gestörten Messwerten relativ rasch folgt und anschließend mit einer gewissen Trägheit wieder auf die ungestörte Objektbahn einschwenkt. Ein sehr viel besseres Verhalten des Filters kann man erzielen, wenn man ein Suchfenster definiert, das umso kleiner wird, je verlässlicher die Messwerte sind (d.h. je kleiner die Fehlerkovarianz ist). In der Simulation 5 (Bild 27.10-b) liegen die gestörten Messwerte außerhalb des Suchfensters und werden deshalb nicht berücksichtigt. Das Kalman-Filter läuft deshalb im Prädiktionsmodus, Korrekturen durch die Messwerte gehen praktisch nicht in das Filter ein. Da vor der Störung bereits gute Schätzungen für die Position und die Geschwindigkeit des Objekts vorliegen, kann die Bahn des Objekts auch ohne Messwerte für einen gewissen Zeitraum gut prädiziert werden. Nachdem die Messstörung vorüber ist (Position $x = 0.7$), liegen wieder Messwerte innerhalb des (evtl. vergrößerten) Suchfensters vor, und die Schätzung der Objektbahn kann wieder durch Messwerte gestützt werden.

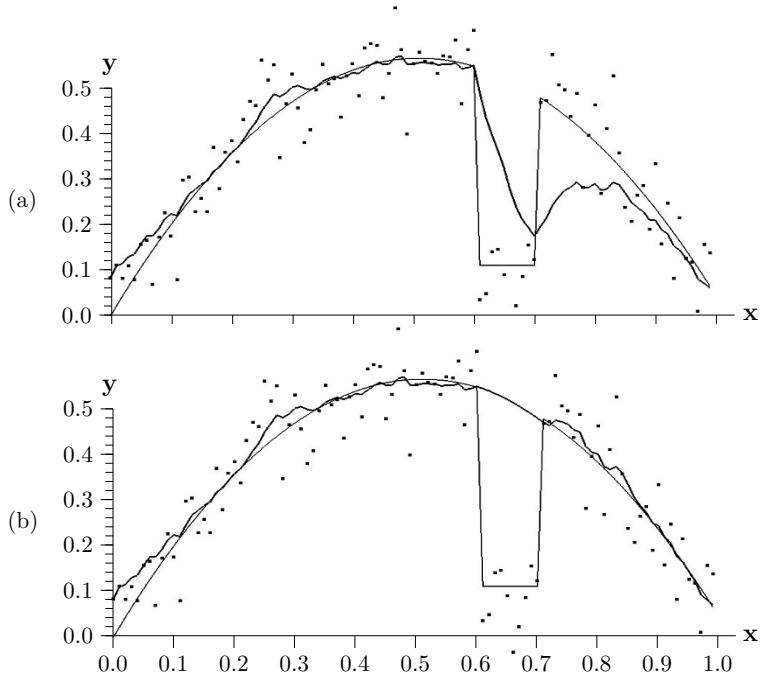


Bild 27.10: Schätzung einer Wurfparabel mit Störung (bei den simulierten Messwerten im Intervall $x \in [0.6, 0.7]$ wurden die y-Komponenten der Position auf den Wert 0.1 gesetzt). (a) Simulation 4: $R_k = 0,01$ konstant auch während der Störung. Die Schätzwerte für die Objektposition werden durch die Störung stark beinflusst. (b) Simulation 5: $R_k = 1$ während der Störung, $R_k = 0,01$ sonst. Die Störung wird detektiert und die Varianz des Messfehlers wird um einen Faktor 100 hochgesetzt, so dass die (falschen) Messwerte im Kalman-Filter praktisch nicht mehr berücksichtigt werden. Das Kalman-Filter läuft deshalb im Prädiktionsmodus und kann nach dem Ende der Störung wieder an der richtigen Stelle weiterarbeiten.

27.4.3 Bildbasierte Navigation

Bei dem vorherigen Beispiel ist man davon ausgegangen, dass der Beobachter bzw. die Kamera an einem bekannten Ort fest steht. Geschätzt wurde die Position des bewegten Objektes. Das Kalman-Filter kann man auch bei der umgekehrten Problemstellung hervorragend einsetzen: Der Schätzung der Beobachter- bzw. Kameraposition auf der Basis bekannter Positionen von Objekten bzw. Landmarken. Aufgaben dieser Art werden als *Navigation* bezeichnet, speziell als *Bildbasierte Navigation*, da als Sensordaten Kamerabilder dienen.

Navigation ist eine der grundlegenden Funktionen eines autonomen mobilen Roboters. Bestückt mit einer kalibrierten Kamera soll die Position der Kamera und damit auch des Roboters aus den abgebildeten Landmarken berechnet werden. Voraussetzung dafür ist,

dass die Positionen der Landmarken vorab bekannt sind. Dies ist in einem kontrollierbaren Umfeld, wie z.B. in einer Fabrikhalle durchwegs gegeben. Um das Beispiel möglichst einfach zu gestalten, geht man von einer statischen Situation aus, d.h. weder die Kamera noch die Landmarken bewegen sich. Damit wird die stochastische Systemzustandsgleichung trivial, da die Systemmatrix zur Einheitsmatrix wird ($\mathbf{A} = \mathbf{I}$) und die Steuermatrix Null ist ($\mathbf{B} = 0$):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{w}_k \tag{27.19}$$

Im Gegensatz zu dem vorigen Beispiel, der Schätzung einer Wurfparabel, steckt die Komplexität bei diesem Beispiel allein in der Messung. Deshalb wurde es ausgewählt. Bei einer realen Kamera als Sensor hat man es mit einer perspektivischen Projektion zu tun, die eine nicht-lineare Abbildung darstellt (Band I Abschnitt 7.6.2). Aufgrund dieser Nichtlinearität muss hier das erweiterte Kalman-Filter (EKF, Abschnitt 27.3.6) zum Einsatz kommen. Im Folgenden wird zunächst die Bildvorverarbeitung kurz dargestellt und anschließend die Herleitung der linearisierten Messmatrix beschrieben.

Bildvorverarbeitung

In der Arbeit von Lohmann [Lohm88], von der dieses Beispiel abgeleitet wurde, bedient man sich einiger einfacher Tricks, um die Landmarken vom Hintergrund und voneinander zu unterscheiden. Als Landmarken werden Kombinationen von jeweils sechs blinkenden Leuchtdioden eingesetzt, die äquidistant auf einer dunklen Box angebracht sind. Zieht man die Grauwerte zweier Bilder pixelweise von einander ab, so wird das entstehende Differenzbild überall dort dunkel sein, wo die beiden verwendeten Bilder übereinstimmen. Nimmt man bei ruhender Kamera und ruhender Umgebung eine Anordnung von blinkenden Leuchtdioden auf, so werden bei Differenzbildung der zeitlich nacheinander gewonnenen Bilder die Leuchtdioden sehr deutlich hervortreten (die Frequenz der blinkenden Leuchtdioden muss in sinnvoller Relation zur Bildrate der Kamera stehen). Die LED-Landmarken können dann mit einfachsten Mitteln der Bildverarbeitung, wie z.B. Schwellwertbildung, erkannt werden. Die LED 1 und 6 blinken im Gleichtakt bei gleicher Hell- wie Dunkelzeit. LED 2 bis LED 5 blinken dazu gleich- oder gegenphasig. Dadurch lassen sich $2^4 = 16$ verschiedene Codierungen einstellen, so dass eine entsprechende Anzahl an Landmarken identifiziert werden kann. Die Abstände der 6 LEDs untereinander, sowie die Position der LEDs im Raum sind vorab bekannt.

Herleitung der linearisierten Messmatrizen

Falls eine Landmarke im Bild (mit bekannter Position) erkannt wird, kann über die Größe der LED-Kette im Bild der Abstand zu dieser LED-Kette berechnet werden. Folglich kann sich die Kamera in einem Kreis[3] um die Landmarke befinden (Bild 27.11-a). Werden zwei

[3]Im dreidimensionalen Raum wäre es eine Kugeloberfläche. Hier wird aber nur das etwas einfachere zweidimensionale Navigationsproblem behandelt. Die Erweiterung auf drei Dimensionen erfordert vor allem mehr Schreibarbeit, das prinzipielle Vorgehen ist aber das Gleiche

Landmarken erkannt, ergeben sich normalerweise zwei Kreise, die sich schneiden (Bild 27.11-b). Einer der beiden Schnittpunkte ist der Standort der Kamera $\mathbf{x}_C = (x_C, y_C)^T$. Durch weitere Landmarken kann ein Schnittpunkt ausgeschlossen werden. Allerdings werden sich wegen der unvermeidlichen Messfehler nicht alle Kreise in exakt einem Punkt schneiden. Weitere Informationen über den Standort liefert der Peilwinkel ϑ zwischen zwei erkannten Landmarken, der aus dem Bild der Kamera ebenfalls ermittelt werden kann. Man hat es also bei einer ausreichenden Zahl erkannter Landmarken mit einem überbestimmten Problem zu tun. Das Kalman-Filter dient zur Fusion dieser Informationen und liefert neben der besten Schätzung für die Kameraposition $\mathbf{x}_C$ auch eine Kovarianzmatrix $\mathbf{P}$ des Schätzfehlers. Zunächst werden die für die Schätzung der Abstände l_i und Peilwinkel ϑ nötigen Gleichungen hergeleitet. Anschließend werden diese Gleichungen linearisiert, damit sie in den Messmatrizen verwendbar sind.

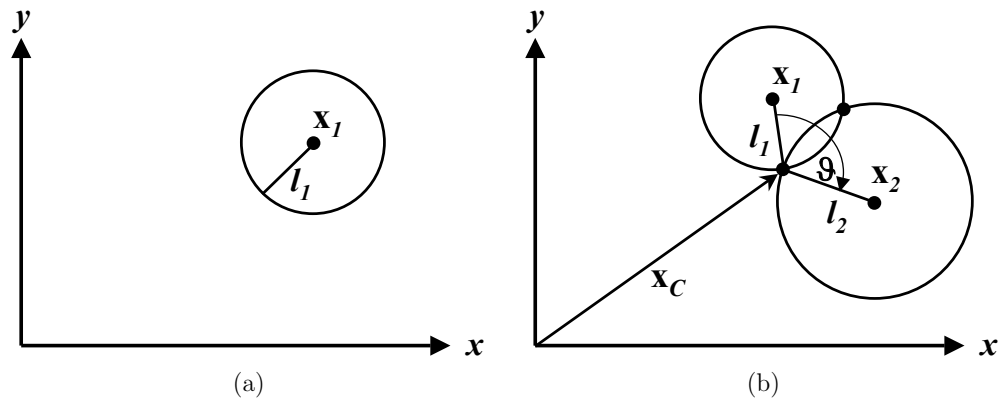


Bild 27.11: (a) Eine Landmarke: Die Kamera kann sich auf einem Kreis mit Radius l_1 um die Landmarke befinden. (b) Zwei Landmarken: Die Kamera kann sich an einem der beiden Kreis-Schnittpunkte befinden. Aus je zwei Landmarken kann ein Peilwinkel ϑ aus dem Bild berechnet werden.

Verwertung der Längenmessung

Zur Längenmessung mit der kalibrierten Kamera wird der Strahlensatz verwendet: Das Verhältnis zwischen der Bildweite f und dem Abstand l vom Brennpunkt der Kamera zu einem Objekt ist gleich dem Verhältnis zwischen der Länge des Objekts z und der Länge z', unter der das Objekt in der Bildebene erscheint (Bild 27.12):

$$\frac{f}{l} = \frac{z'}{z} \tag{27.20}$$

In (27.20) ist die Bedingung eingeflossen, dass z (d.h. die Verbindungsgerade der sechs Leuchtdioden) senkrecht auf der optischen Achse steht. Da z und f vorab bekannt sind,

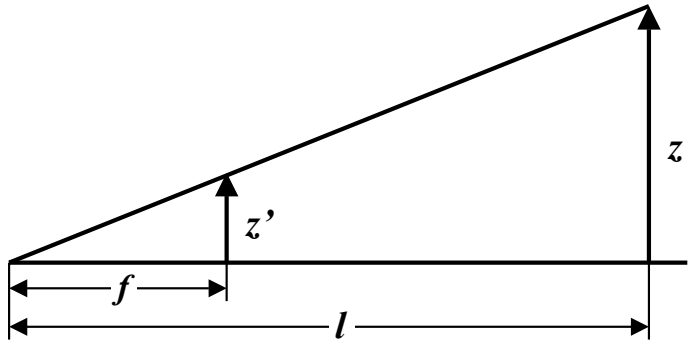


Bild 27.12: Abstandsmessung mit der kalibrierten Kamera: l = Abstand vom Brennpunkt der Kamera zu einer Landmarke. f = Bildweite in Pixel. z = Länge des Objekts (Abstand von der ersten zur sechsten Leuchtdiode) z' = Länge des Objekts in der Bildebene in Pixel.

kann aus der Messung von z' der Abstand der Kamera zu einer erkannten Landmarke aus (27.20) errechnet werden.

Die Abstandsmessungen werden nun zur Schätzung der Kameraposition im Rahmen des Kalman-Filters verwertet. Bei n erkannten Landmarken liegen n Größen z'_k, $k = 1, .., n$ vor, die die Höhen der LED-Ketten in der Bildebene beschreiben. Daraus resultieren nach (27.20) n indirekte Abstandsmessungen:

$$l_k = \frac{f}{z'_k} z \tag{27.21}$$

Die Kameraposition $\mathbf{x}_C = (x_C, y_C)^T$ und die Landmarkenpositionen $\mathbf{x}_k = (x_k, y_k)^T$ stehen über die Kreisgleichung in Beziehung zu den Abstandsmessungen:

$$l_k^2 = (x_k - x_C)^2 + (y_k - y_C)^2 \quad , \qquad k = 1, .., n \tag{27.22}$$

Mit Gleichung (27.21) folgt daraus die nicht-lineare Messgleichung für z'_k:

$$z'_k = \frac{z \cdot f}{\sqrt{(x_k - x_C)^2 + (y_k - y_C)^2}} + v_k = h_z(x_C, y_C) + v_k \tag{27.23}$$

Hierbei ist v_k eine Zufallsvariable, die das $N(0, R_k)$-verteilte Messrauschen repräsentiert. Die nicht-lineare Funktion h_z bildet den Systemzustand (hier die Kameraposition $\mathbf{x}_C$) auf den Messwert ab.

Voraussetzung für den Einsatz des Kalman-Filters ist eine lineare Messgleichung. Deshalb wird (27.23) um einen Schätzwert $\hat{\mathbf{x}}_C = (\hat{x}_C, \hat{y}_C)^T$ für die Kameraposition $\mathbf{x}_C$ linearisiert:

$$z'_k \approx h_z(\hat{\mathbf{x}}_C) + \left.\frac{\partial h_z}{\partial \mathbf{x}_C}\right|_{\hat{\mathbf{x}}_C} \cdot (\mathbf{x}_C - \hat{\mathbf{x}}_C) + v_k \tag{27.24}$$

Mit der Definition der Messmatrix $\mathbf{H}_k$:

$$\begin{aligned}\mathbf{H}_k &= \left.\frac{\partial h_z}{\partial \mathbf{x}_C}\right|_{\hat{\mathbf{x}}_C} = \left.\left(\frac{\partial h_z}{\partial x_C}, \frac{\partial h_z}{\partial y_C}\right)\right|_{\hat{\mathbf{x}}_C} \\ &= \left(\frac{z \cdot f \cdot (x_k - \hat{x}_C)}{\sqrt{(x_k - \hat{x}_C)^2 + (y_k - \hat{y}_C)^2}^3}, \frac{z \cdot f \cdot (y_k - \hat{y}_C)}{\sqrt{(x_k - \hat{x}_C)^2 + (y_k - \hat{y}_C)^2}^3}\right)\end{aligned} \tag{27.25}$$

und einer Umformung von Gleichung (27.24) folgt:

$$z'_k - h_z(\hat{\mathbf{x}}_C) + \mathbf{H}_k \cdot \hat{\mathbf{x}}_C = \mathbf{H}_k \cdot \mathbf{x}_C + v_k \tag{27.26}$$

Mit der Definition von

$$y_k = z'_k - h_z(\hat{\mathbf{x}}_C) + \mathbf{H}_k \cdot \hat{\mathbf{x}}_C \tag{27.27}$$

lässt sich (27.26) in der Form einer linearen Messgleichung schreiben, wie im normalen Kalman-Filter[4] üblich (27.2):

$$y_k = \mathbf{H}_k \cdot \mathbf{x}_C + v_k \tag{27.28}$$

Der Kalman-Filter Algorithmus zur Schätzung der Kameraposition $\mathbf{x}_C$ mit Hilfe der Abstandsmessungen ist in **A27.1** zusammengefasst.

[4]Bis auf den gesuchten Systemzustand $\mathbf{x}_C$ (Position der Kamera), der im Allgemeinen veränderlich ist und somit $\mathbf{x}_{C,k}$ heißen müsste, aber im Spezialfall einer ruhenden Kamera konstant ist, so dass $\mathbf{x}_{C,k} = \mathbf{x}_C$.

A27.1: Algorithmus zur Schätzung der Kameraposition mit Hilfe der Abstandsmessungen.

Voraussetzungen und Bemerkungen:

◇ Bestimmung einer Startschätzung für die Kameraposition $\hat{\mathbf{x}}_{C,0}$ und einer Fehlerkovarianzmatrix $\mathbf{P}_0^-$ dieser Schätzung.

Algorithmus:

(a) Für alle erkannten Landmarken:

(aa) Berechnung der Messmatrix $\mathbf{H}_k$ nach Gleichung (27.25)

(ab) Berechnung der Differenz $(y_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_{C,k}^-)$ nach Gleichung (27.27). Der aktuelle Messwert z_k' und der vorhergesagte Messwert $h_z(\hat{\mathbf{x}}_{C,k}^-)$ sind bekannt.

(ac) Durchführung des Korrekturschritts:

(aca) Berechnung der Kalman-Verstärkung:
$\mathbf{K}_k = \mathbf{P}_k^- \cdot \mathbf{H}_k^T \left(\mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T\right)^{-1}$

(acb) Korrektur der Schätzung mit der Messung:
$\hat{\mathbf{x}}_{C,k}^+ = \hat{\mathbf{x}}_{C,k}^- + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{C,k}^-\right)$

(acc) Korrektur der Fehlerkovarianzmatrix:
$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_k^-$

(ad) Durchführung des Prädiktionsschritts:

(ada) Prädiktion des Systemzustands nach (27.19):
$\hat{\mathbf{x}}_{C,k+1}^- = \hat{\mathbf{x}}_{C,k}^+$

(ada) Prädiktion der Fehlerkovarianzmatrix:
$\mathbf{P}_{k+1}^- = \mathbf{P}_k^+ + \mathbf{Q}_k$
Wobei $\mathbf{Q}_k$ ein normal-verteilter Zufallswert für das Systemrauschen ist.

Ende des Algorithmus

Verwertung der Peilwinkelmessung

In Bild 27.13 ist das Prinzip dargestellt, nach dem der horizontale Peilwinkel ϑ, unter dem zwei Landmarken erscheinen, berechnet wird.

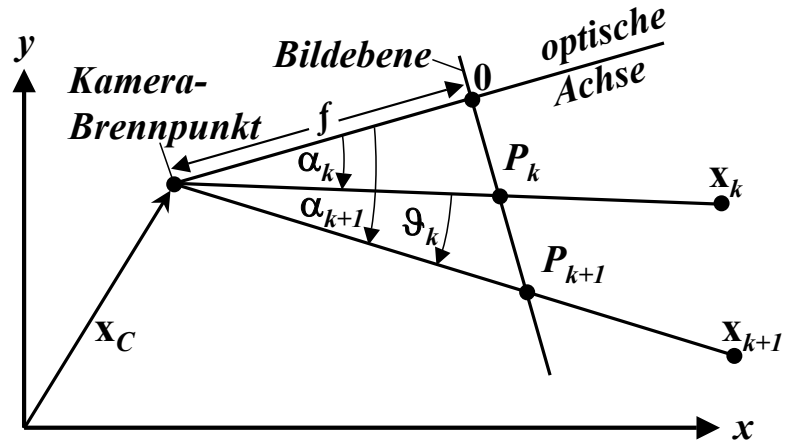


Bild 27.13: Peilwinkelmessung mit der kalibrierten Kamera: aus den Abständen P_k und P_{k+1} der Landmarken von der Bildmitte lässt sich der Peilwinkel ϑ_k bestimmen.

Aus den Abständen P_k und P_{k+1} der auf die Bildebene projizierten Landmarken von der Bildmitte lassen sich zunächst die Winkel α_k und α_{k+1} bestimmen:

$$\alpha_k = \arctan \frac{P_k}{f} \tag{27.29}$$

Wie aus Bild 27.13 ersichtlich, ist der Peilwinkel ϑ_k der Betrag der Differenz zwischen den Winkeln α_{k+1} und α_k:

$$\vartheta_k = |\alpha_{k+1} - \alpha_k| = \left| \arctan \frac{P_{k+1}}{f} - \arctan \frac{P_k}{f} \right| \tag{27.30}$$

Auf diese Weise kann man aus den n Landmarken $n-1$ Winkel ϑ_k, $k = 1, ..., n-1$ ermitteln, die zur Ortsbestimmung der Kamera beitragen sollen.

Als Nächstes wird der Zusammenhang zwischen dem horizontalen Peilwinkel ϑ und dem Systemzustand $\mathbf{x}_C$ (Position der Kamera) abgeleitet. Die Positionen der Landmarken $\mathbf{x}_k = (x_k, y_k)^T$ sind im Weltkoordinatensystem bekannt. $\mathbf{x}_C = (x_C, y_C)^T$ ist der unbekannte Ortsvektor des Kamerabrennpunktes.

Für das Skalarprodukt zweier Vektoren gilt per Definition:

$$\begin{aligned}\vartheta_k &= \arccos \frac{(\mathbf{x}_k - \mathbf{x}_C) \cdot (\mathbf{x}_{k+1} - \mathbf{x}_C)}{|\mathbf{x}_k - \mathbf{x}_C| \cdot |\mathbf{x}_{k+1} - \mathbf{x}_C|} \\ &= \arccos \frac{(x_k - x_C) \cdot (x_{k+1} - x_C) + (y_k - y_C) \cdot (y_{k+1} - y_C)}{\sqrt{(x_k - x_C)^2 + (y_k - y_C)^2} \cdot \sqrt{(x_{k+1} - x_C)^2 + (y_{k+1} - y_C)^2}} \\ &= h_\vartheta(\mathbf{x}_C)\end{aligned} \tag{27.31}$$

womit der Zusammenhang zwischen Messgröße ϑ_k und Zustandsgröße $\mathbf{x}_C$ hergestellt ist. Die nicht-lineare Funktion h_ϑ bildet den Systemzustand (hier die Kameraposition $\mathbf{x}_C$) auf den Messwert ab.

Das weitere Vorgehen ist das gleiche wie bei der Verwertung der Längenmessung. Zunächst erfolgt die Linearisierung der Messgleichung (27.31) um einen Schätzwert $\hat{\mathbf{x}}_C = (\hat{x}_C, \hat{y}_C)^T$ für die Kameraposition. Die Messmatrix $\mathbf{H}_k$ für die Peilwinkelmessung lautet:

$$\mathbf{H}_k = \left.\frac{\partial h_\vartheta}{\partial \mathbf{x}_C}\right|_{\hat{\mathbf{x}}_C} = \left.\left(\frac{\partial h_\vartheta}{\partial x_C}, \frac{\partial h_\vartheta}{\partial y_C}\right)\right|_{\hat{\mathbf{x}}_C} \tag{27.32}$$

Damit lässt sich (27.31) in eine lineare Messgleichung vom Typ (27.2) umformen. Nun kann der Kalman-Filter Algorithmus **A27.1** zur Verbesserung der Schätzung der Kameraposition $\mathbf{x}_C$ mit Hilfe der Peilwinkelmessungen gestartet werden. Als Startschätzung geht man hier vom Ergebnis $\hat{\mathbf{x}}_{C,n}^-$ der letzten Schätzung auf Basis der Längenmessungen aus. Als Messmatrix muss (27.32) anstatt (27.25) verwendet werden. Der Algorithmus **A27.1** wird dieses Mal für die $(n-1)$ Peilwinkelmessungen durchgeführt. Damit sind alle Informationen für die Schätzung der Kameraposition ausgewertet, die die ruhende Kamera liefert.

Das dargestellte Prinzip ist natürlich erweiterbar:

- Auf Situationen, bei denen der Roboter (bzw. die Kamera) bewegt wird.
- Auf Situationen, bei denen sich sowohl der Roboter also auch die Landmarken (bzw. Objekte) bewegen. Falls Bewegung im Spiel ist, erreicht die Zustandsübergangsgleichung wieder die volle Komplexität (27.1), da jetzt die Eigen- bzw. Objektdynamik in der Systemmatrix $\mathbf{A}_k$ berücksichtigt werden muss. Entsprechend komplexer wird auch die Gleichung für die Prädiktion der Fehlerkovarianzmatrix.
- Auf Situationen, bei denen eine Stereo-Kamera (d.h. zwei Kameras oder auch mehr) eingesetzt wird.
- Auf Situationen, bei denen andere Navigations-Sensoren (z.B. Beschleunigungsmesssysteme, Odometer, GPS (Global Positioning System) oder Höhenmesser) eingesetzt werden.
- Auf kombinierte Situationen, bei denen mehrere Sensoren gleichzeitig eingesetzt werden [Scho03].

Kapitel 28

Zusammenfassen von Segmenten zu Objekten

28.1 Bestandsaufnahme und Anwendungen

In diesem abschließenden Kapitel werden einige Verfahren erläutert, wie in einem segmentierten Bild die Segmente zu Objektstrukturen zusammengefasst werden können. Dies ist bei vielen Anwendungen notwendig, bei denen aus komplexen Bildinhalten eine Reaktion abgeleitet werden soll. Einige Beispiele dazu werden weiter unten gegeben.

Zunächst erfolgt eine Bestandsaufnahme der Informationen, die mit den bisher diskutierten Verfahren aus der Originalszene abgeleitet wurden. Aus den Originaldaten wurden, nach einer eventuellen Korrektur systematischer oder statistischer Fehler (z.B. Kapitel 4 bis 9), weitere Merkmale berechnet (Kapitel 12 bis 19). Mit Verfahren zur Segmentierung (Kapitel 20 bis 22) wurden Bildpunkte (Merkmalsvektoren), die einander ähnlich sind, d.h. die im N-dimensionalen Merkmalsraum benachbart liegen, zu Segmenten zusammengefasst. Hier spielt sich auch der Übergang von der bildpunktorientierten Bildverarbeitung zur listenorientierten Bildverarbeitung ab (Beispiele dazu in Kapitel 23). Im nächsten Schritt werden zu den Segmenten Parameter berechnet, die sie näher beschreiben. Diese Verarbeitungsstufe wurde in den Kapiteln 24 bis 26 eingehend behandelt.

Häufig ist diese Verarbeitungsstufe aber aus der Sicht der digitalen Bildverarbeitung und Mustererkennung nicht die letzte Stufe. Vielmehr müssen jetzt einzelne Segmente, die auf Grund bekannter Struktureigenschaften zusammengehören, zu Objekten zusammengefasst werden. Es interessiert nämlich letztlich nicht die Information, ob eine kreisförmige Anordnung von Bildpunkten vorliegt, sondern ob es sich dabei um eine Beilagscheibe oder ein Kugellager handelt.

Im Folgenden werden einige Beispiele gegeben, die die Bandbreite der möglichen Interessenlagen etwas illustrieren sollen.

Bei einem segmentierten Luft- oder Satellitenbild ist eine Zusammenfassung von Segmenten zu Objekten meistens nicht gefordert. Nach der Segmentierung liegt die Information vor, dass unterschiedlich codierte Segmente verschiedene Landnutzungsklassen repräsen-

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4_28

tieren. Parameter wie z.B. zur Form oder Orientierung der Segmente werden hier nicht benötigt. Allenfalls ist noch die Größe der Segmente einer Klasse von Belang. Die weitere Verarbeitung kann die Bereinigung des segmentierten Bildes sein (z.B. die Zuweisung von sehr kleinen Segmenten zu den sie umgebenden Segmentklassen), die Überführung der Segmente in kompaktere Datenstrukturen und die Integration dieser Daten in Informationssysteme, um sie mit anderen Daten (z.B. politische Grenzen, statistische Erhebungen, digitale Geländemodelle, usw.) verknüpfen zu können.

Wenn im Rahmen einer Qualitätskontrolle Oberflächen auf Fehlerstellen oder Störungen untersucht werden sollen, so kann der Segmentierungsschritt das Endergebnis, nämlich die Markierung der fehlerhaften Stelle, liefern. Beispiele dazu wurden in Kapitel 17, Bilder 17.18, 17.17 und 17.22 und in Kapitel 18, Bildfolge 18.9 gegeben. Die Segmentierung besteht hier aus einer Einteilung in die Klassen „fehlerhafter Bereich“ und „nicht fehlerhafter Bereich“. Meistens genügt hier eine Positionsangabe (z.B. Schwerpunktskoordinaten). Manchmal wird noch eine Auswahl der markierten Bereiche getroffen, etwa dadurch, dass alle Fehlerstellen am Rand ignoriert werden oder dass nur Störungen innerhalb einer festgelegten *area-of-interest* akzeptiert werden. Weitere Beispiele zu Anwendungen aus diesem Bereich sind die Inspektion von lackierten Oberflächen auf Unebenheiten und Kratzer oder die Untersuchung von Pressteilen an sicherheitsrelevanten Stellen auf Einschnürungen oder Risse.

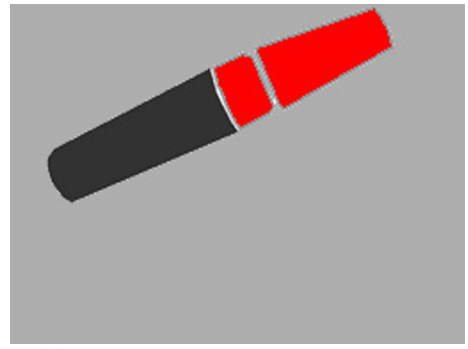
Bei einer anderen Gruppe von Anwendungen wird segmentiert, und danach werden die Segmente (oder das Segment, da hier häufig nur ein relevantes Segment vorliegt) näher untersucht. Beispiele dazu wurden in der Einleitung zu Kapitel 11 (Bildfolge 11.1, Kleberaupen) gegeben. Bei der Anwendung in Bild 11.1-a und 11.1-b ist zu untersuchen, ob die Kleberaupe unterbrochen ist. Das Segmentierungsergebnis ist hier nur in der festgelegten *area-of-interest* relevant. Falls in diesem Bereich, nach der Elimination von Segmenten mit weniger als c_1 Pixel, nur ein Segment übrig bleibt, so kann daraus die Schlussfolgerung abgeleitet werden, dass die Kleberaube vorhanden und nicht unterbrochen ist. Falls z.B. zwei Segmente auftreten, ist noch zu prüfen, ob die Unterbrechung größer als c_2 Pixel ist und somit außerhalb der Toleranz liegt. Auf eine Untersuchung, ob die Orientierung der Segmente zusammenpasst, kann hier verzichtet werden, da nur die Segmente innerhalb der *area-of-interest* verarbeitet werden und stillschweigend die Annahme gemacht wird, dass segmentierte Bereiche automatisch zur Kleberaupe gehören.

Etwas anders ist die Situation bei der Kleberaupenvermessung in den Bildbeispielen 11.1-c und 11.1-d. Hier dient die Segmentierung nur zur Ermittlung des Bildbereichs, in dem das Lichtband verläuft. Die eigentliche Auswertung, die die gewünschte Information liefert, ob die Kleberaupe die richtige Form hat, erfolgt erst anschließend. Dazu wird der segmentierte Bereich z.B. skelettiert und die Skelettlinie analysiert, um die markanten Punkte zu erhalten. Aus der relativen oder absoluten Lage dieser Punkte zueinander kann dann die Entscheidung abgeleitet werden.

Ähnlich sind Anwendungen, bei denen das Ergebnis der Segmentierung als Maske verwendet wird, um Bildbereiche festzulegen, in denen eine weitere Verarbeitung stattzufinden hat. Ein Beispiel hierzu wurde in Kapitel 7, Bildfolge 7.34, gegeben. Hier wird durch die Segmentierung ein Bereich markiert, in dem eine Beschriftung vorliegt. In weiteren Verarbei-

(a)

(b)



(c)

Bild 28.1: (a) Rotauszug des Originalbilds: Das Objekt „Filzstift“ besteht aus den drei Komponenten „Griff“ (schwarz), „Halterung“ (rot) und „Verschlusskappe“ (rot). (b) Vorverarbeitung: Zwei Erosionen, gefolgt von zwei Dilatationen (*opening*), um die Reflexionen zu entfernen. (c) Segmentiertes Bild. Zur Segmentierung wurden alle drei Farbkanäle verwendet. Es wurden die Klassen „schwarze Bildpunkte“ (Griff), „rote Bildpunkte“ (Halterung und Verschlusskappe) und „sehr helle Bildpunkte“(Hintergrund) klassifiziert.

tungsschritten werden nur mehr die Bildpunkte in diesem markierten Bereich untersucht, um z.B. die Schriftzeichen zu erkennen und dadurch die Bezeichnung eines Teils zu erfassen. Dieser zweite Verarbeitungsschritt ist vom ersten Segmentierungsschritt weitgehend unabhängig. Er wird aber wieder aus einer Elimination von Störungen, einer Berechnung von Merkmalen und einer Segmentierung (möglicherweise mit ganz anderen Verfahren als im ersten Schritt) bestehen.

Komplexe Objektstrukturen können nach der Segmentierung in mehrere Segmente zergliedert sein. Es ist jetzt die Aufgabe eines weiteren Verarbeitungsschrittes, die zu einem Objekt gehörigen Segmente zu finden, zu untersuchen, ob sie „zusammenpassen“ und letzt-

lich zu entscheiden, ob es sich um ein Objekt eines bestimmten Typs handelt. Ein einfaches Beispiel hierzu ist in der Bildfolge 28.1 zusammengestellt. Das Objekt ist hier ein Filzstift, der sich aus einem schwarzen Griff (links unten), einer Halterung für den Faserstift (in der Mitte) und der Verschlusskappe (rechts oben) zusammensetzt. Nach einer Vorverarbeitung, die hier aus der Elimination von Reflexionen bestehen könnte (z.B. zwei Erosionen gefolgt von zwei Dilatationen, *opening*), werden zur Segmentierung die Farbmerkmale verwendet: Griff - Schwarz, Halterung und Verschlusskappe - Rot, Hintergrund - Hellgrau. Die drei für die Segmentierung maßgeblichen Klassen (die Klasse der schwarzen und der roten Bildpunkte des Objekts und die Klasse der sehr hellen Bildpunkte des Hintergrundes) verteilt sich im Segmentierungsergebnis auf die vier Segmente mit den Bedeutungen „Hintergrund“, „Griff“, „Halterung“ und „Verschlusskappe“.

Aber erst eine genauere Untersuchung der Objektsegmente liefert die Information, ob es sich dabei tatsächlich um die entsprechende Struktur handelt: Man muss z.B. prüfen, ob die Flächeninhalte innerhalb der Toleranz liegen und ob die Formen stimmen. Wenn diese Untersuchungen ein positives Ergebnis liefern, kann versucht werden, die Segmente zusammenzufassen. Dazu wird man prüfen, ob die Lage der Schwerpunkte zusammenpasst, was hier bedeutet, dass sie auf einer Geraden liegen und dass die Abstände innerhalb gewisser Toleranzen liegen. Wenn dies der Fall ist, kann man entscheiden, dass es sich um einen Filzstift des Typs XYZ handelt.

Diese Beispiele haben gezeigt, dass die nach einer Segmentierung folgenden Schritte einfach, aber auch sehr komplex sein können. In den weiteren Abschnitten werden nun einige Vorgehensweisen für diese Verarbeitungsschritte diskutiert.

28.2 Einfache, heuristische Vorgehensweise

Im vorhergehenden Abschnitt ist bereits angeklungen, dass bei vielen Implementierungen heuristisch, pragmatisch vorgegangen wird. Wenn z.B. auf einer texturierten Oberfläche Störungen der Textur gefunden werden sollen, so könnte die Vorgehensweise wie folgt aussehen:

- Segmentierung nach Maßgabe bestimmter Texturmerkmale.
- Alle Segmente, die kleiner als c_1 Pixel sind, werden ignoriert.
- Alle Segmente, die außerhalb einer festgelegten *area-of-interest* liegen, werden ignoriert.
- Alle Segmente, die vom Rand der *area-of-interest* berührt werden, werden ignoriert.
- Aufgrund von apriori-Wissen sei bekannt, dass die Störungen immer länglich ausgeprägt sind. Deshalb werden Segmente, bei denen das Verhältnis der Eigenwerte der Kovarianzmatrix kleiner als c_2 ist, nicht berücksichtigt.

- Falls die Orientierung des Segments mit der (bekannten) Texturhauptrichtung übereinstimmt, wird das Segment als Markierung einer Texturstörung akzeptiert.

Man sieht an diesem Beispiel, dass eine sequentielle Vorgehensweise gewählt wurde: Es wird versucht, durch apriori-Wissen Schritt für Schritt Segmente auszuschließen. Dabei wird man, um Rechenzeit zu sparen, die Berechnung nur auf benötigte Segmentparameter beschränken. Wenn sich die abgebildeten Objekte bereits durch den Flächeninhalt unterscheiden, so kann man auf aufwändige Verfahren, wie z.B. das Strahlenverfahren (Kapitel 25) oder eine Segmentbeschreibung mit neuronalen Netzen (Kapitel 26) verzichten.

Wenn die interessierenden Objekte nach der Segmentierung in mehrere Segmente zergliedert sind, so muss anhand von apriori-Wissen geprüft werden, welche Segmente zusammengehören. Dies sei am Beispiel des Objekts „Filzstift“ von Bild 28.1 erläutert. Dieses Objekt setzt sich aus drei Segmenten zusammen:

- Das erste Segment ist der „Griff“. Es gehört zur Klasse der schwarzen Bildpunkte. Der Flächeninhalt dieses Segments muss bei etwa 13500 Bildpunkten liegen. Der Parameter für die Länglichkeit (λ_1/λ_2) muss bei etwa 10 liegen.

- Das zweite Segment ist die „Halterung“. Es gehört zur Klasse der roten Bildpunkte. Flächeninhalt: Etwa 3600 Bildpunkte. Das Segment ist nicht ausgeprägt länglich $(\lambda_1/\lambda_2 = 2)$.

- Das dritte Segment ist die „Verschlusskappe“. Dieses Segment gehört ebenfalls zur Klasse der roten Bildpunkte. Flächeninhalt: Etwa 9000 Bildpunkte. Das Segment ist ausgeprägt länglich $(\lambda_1/\lambda_2 = 5)$.

Auf eine genaue Untersuchung der Form der Segmente kann in diesem Beispiel verzichtet werden, da die Flächeninhalte, das Länglichkeitsmerkmal und die Klassenzugehörigkeit die Unterscheidung gewährleisten. Falls das Problem größeninvariant behandelt werden soll, muss eine Flächennormierung, etwa über eine Division aller Segmentflächen durch die größte Segmentfläche, erfolgen. Dann treten an die Stelle der absoluten Flächeninhalte die Verhältnisse der Flächeninhalte der Segmente.

Wenn nun drei Segmente extrahiert wurden, auf die die obigen Beschreibungen zutreffen, so kann man daraus noch nicht schließen, dass diese drei Segmente zusammengesetzt das abgebildete Objekt „Filzstift“ ergeben. Um zu diesem Schluss zu kommen, muss man in diesem Beispiel weiter fordern:

- Die drei Schwerpunkte der Segmente müssen näherungsweise auf einer Geraden liegen.

- Die Orientierungen der Segmente müssen übereinstimmen. Das ist bei nicht länglichen Segmenten oft schwer zu bestimmen. Im vorliegenden Beispiel könnte man die Tatsache verwenden, dass das Segment „Halterung“ rechteckig ist und die längere Rechteckseite senkrecht auf der Orientierungsachse des Objekts stehen muss.

- Die Reihenfolge der Segmente entlang der Orientierungsachse muss stimmen: Ausgehend vom Segment „Griff“ muss das Segment „Halterung“ folgen und dann das Segment „Verschlusskappe“.

- Die Abstände der Segmentschwerpunkte müssen innerhalb bestimmter Toleranzen liegen.

Diese Forderungen werden in der Bildfolge 28.2 verdeutlicht.

Es ist offensichtlich, dass selbst bei diesem sehr einfachen Beispiel vielfältige Überprüfungen durchgeführt werden müssen, um das gewünschte Ergebnis ableiten zu können. Ungleich schwieriger wird es natürlich, wenn die Objekte die Struktur von komplexen Bauteilen, Fahrzeugen oder Personen haben. Hier muss im jeweiligen Anwendungsfall eine besondere Strategie festgelegt werden. Sehr wichtig ist hier die sorgfältige Zerlegung des komplexen Objekts in einfache Bestandteile, die mit den Methoden der Segmentierung erkannt werden können (siehe Kapitel 20). Bei der Synthese sollte man soweit wie möglich apriori-Wissen in die Entscheidungen mit einbeziehen.

28.3 Strukturelle Verfahren

Die Verfahren des vorhergehenden Abschnitts haben den Nachteil, dass sie zwangsläufig sehr stark auf eine bestimmte Anwendung zugeschnitten sind. Das bedeutet, dass bei Änderungen der Systemkonstellation oder beim Übergang zu einer anderen, ähnlichen Anwendung viele Teile modifiziert werden müssen. Da bei Implementierungen dieser Art das apriori-Wissen und die Auswertungsstrategie fest in den Programmcode eingebunden sind, wird die Umkonfiguration einer Anwendung nur mit einer Umprogrammierung zu lösen sein.

Wünschenswert wäre ein System, das allgemeingültige Inferenzmethoden zur Verfügung stellt, die in den verschiedensten Anwendungsfällen eingesetzt werden können. Ein derartiges System sollte folgende Komponenten enthalten:

- Eine (*Bildverarbeitungs-* und) *Mustererkennungskomponente* (MK), die die Methoden der digitalen Bildverarbeitung und Mustererkennung bereitstellt.

- Eine *statische Wissenbasis* (WB) für das gesamte Wissen, das der Anwendung zugrunde liegt.

- Eine *dynamische Wissensbasis* für das während einer Auswertung anfallende Wissen.

- Eine *Reaktionskomponente* (*Exekutive*, RK), hinter der sich die spezielle Anwendung verbirgt.

- Eine *Verwaltungskomponente* (VK), die die Ablaufsteuerung der Anwendung übernimmt.

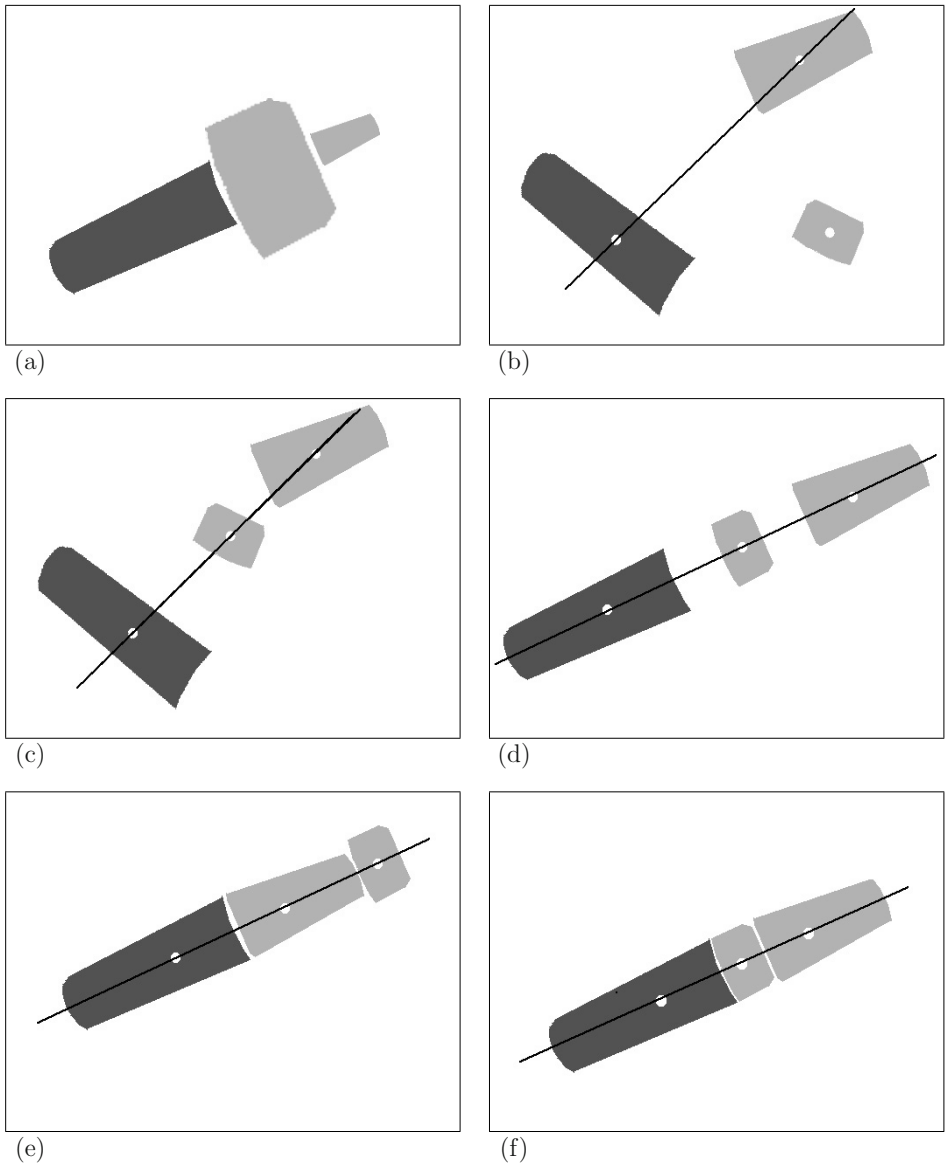


Bild 28.2: Zusammenfassen von Segmenten zu einem Objekt. (a) Bei den drei Segmenten stimmen die Größenverhältnisse nicht. (b) Die Segmentschwerpunkte liegen nicht auf einer Geraden. (c) Die Orientierungen der Segmente passen nicht zusammen. (d) Die Abstände der Segmentschwerpunkte stimmen nicht. (e) Die Reihenfolge der Segmente entlang der Orientierungsachse stimmt nicht. (f) Korrekte Konstellation: Hier kann auf das Objekt „Filzstift“ geschlossen werden.

- Eine *Interaktionskomponente* (IK), durch die der Benutzer mit dem System kommunizieren kann.
- Eine *Dokumentationskomponente* (DK), die die gesamte Beschreibung des Systems enthält.

In den folgenden Abschnitten werden einige dieser Komponenten ausführlicher erläutert.

28.3.1 Die Mustererkennungskomponente

Die Mustererkennungskomponente kann man sich als eine Methodenbasis mit Modulen (Prozeduren, Prozessen) vorstellen, die die Gebiete der digitalen Bildverarbeitung und Mustererkennung so weit wie möglich abdecken. Wenn eine neue Anwendung entworfen wird, so wählt der Designer des Systems aus der Methodenbasis die benötigten Module aus. Bei der Untersuchung der Kleberaupen (Bildbeispiel in Kapitel 11, Bild 11.1-c und 11.1-d) werden etwa folgende Module benötigt:

- Ein Modul zum zur Aufnahme eines Digitalbildes.
- Ein Modul zur Festlegung einer *area-of-interest*, in der das Lichtband liegt. Alle weiteren Module arbeiten nur mehr in dieser *area-of-interest*.
- Ein Modul zur Entfernung von Bildstörungen, etwa durch einen bewegten Mittelwert.
- Ein Modul zur Binarisierung: Hintergrund / Lichtband.
- Ein Modul zur Skelettierung des segmentierten Lichtbandes.
- Ein Modul, das die Skelettlinie auswertet, die markanten Punkte ermittelt und die Koordinaten dieser Punkte in geeigneter Weise bereitstellt.

In diesem Beispiel könnte es vorkommen, dass die Methodenbasis kein spezielles Modul zur Auswertung der Skelettlinie enthält. Dann muss dieses Modul neu implementiert werden und steht ab diesem Zeitpunkt in der Methodenbasis auch für spätere Anwendungen zur Verfügung.

Die Module der Methodenbasis sollten so implementiert werden, dass ihre Ein-/Ausgabeschnittstellen, soweit sinnvoll, miteinander kompatibel sind. So sollte z.B. das Binarisierungsmodul direkt die Ergebnisse des Moduls für den bewegten Mittelwert übernehmen können. Anschaulich gesprochen sollte die Methodenbasis so implementiert werden, dass die Module, vergleichbar mit den Steinen eines Dominospiels, zu längeren Verarbeitungsfolgen kombiniert werden können.

Ob die Module als Prozeduren (Unterprogramme) oder als selbständige Prozesse ablaufen können, muss beim Design der Methodenbasis entschieden werden. Auf jeden Fall

ist eine saubere prozedurale Lösung, eventuell in Verbindung mit objektorientierter Programmierung, immer vorteilhaft. Die Implementierung einzelner Module als Prozesse ist dann mit geringem Aufwand zu erreichen.

Bei einer Konstellation, in der einzelne Module oder Kombinationen von Modulen zu Prozessen zusammengefasst werden, ist die Möglichkeit der Parallelisierung von Abläufen gegeben. Dieser Aspekt gewinnt bei komplexen Anwendungen immer mehr an Bedeutung und sollte auf jeden Fall bedacht werden.

28.3.2 Die statische und dynamische Wissensbasis

In diesen Komponenten wird das gesamte Wissen der Anwendung in einer strukturierten Form bereitgestellt. Für das Beispiel des Problems der Erkennung eines Objekts vom Typ „Filzstift“ aus dem vorhergehenden Abschnitt könnten das etwa Informationen zu folgenden Fragen sein:

- Welche Module aus der Methodenbasis werden benötigt?
- In welcher Reihenfolge müssen die Module ablaufen?
- Welche äußeren Voraussetzungen erwarten bestimmte Module? Hier könnte z.B. die Position von Beleuchtungseinrichtungen festgelegt werden, die beim Bildeinzug notwendig sind.
- Mit welchen Parameterwerten müssen die Module ablaufen? Für das Klassifizierungsmodul sind das z.B. die Dimension des Merkmalsraums, die Anzahl der Klassen, die trainierte Datenbasis für den Klassifikator, usw.
- Mit welchen Parametern und welchen Toleranzen werden die Segmente beschrieben? In diesem Beispiel könnte das für das Segment „Griff“ lauten: „Das Segment muss zur Klasse der schwarzen Bildpunkte gehören. Der Flächeninhalt muss zwischen 13000 *Pixel* und 14000 *Pixel* und die Maßzahl für die Länglichkeit muss zwischen 9.5 und 10.5 liegen.“
- Welche Segmentkonstellationen müssen vorliegen, dass auf ein bestimmtes Objekt geschlossen werden kann? Die für das gewählte Beispiel notwendigen Informationen hierzu wurden bereits im Kapitel 11 und in der Bildfolge 28.2 zusammengestellt.
- Welche Reaktion hat zu erfolgen, falls ein bestimmtes Objekt erkannt wird?

Die Antworten auf diese Fragen werden in der statischen Wissensbasis gespeichert. Sie stehen während der gesamten „Lebenszeit“ einer Anwendung zur Verfügung. Sollten Änderungen notwendig werden, so ist das meistens durch eine Modifikation der Wissensbasis und einen Neustart des Systems zu erreichen.

Die dynamische Datenbasis nimmt diejenigen Informationen auf, die während eines Ablaufs der Anwendung anfallen. Das könnte z.B. die Information sein, dass nach einer

Segmentierung 25 Segmente gefunden wurden, deren Positionen durch die Segmentschwerpunkte festgehalten wurden. Die Segmente sind jeweils als Listenstrukturen abgelegt, auf die über Pointer zugegriffen werden kann.

Die dynamische Datenbasis ist nur während eines Arbeitsvorgangs der Anwendung gültig. Nach dessen Abschluss wird sie wieder gelöscht.

Zu einzelnen Teilbereichen der hier angeschnittenen Datenbasenproblematik liegen brauchbare Lösungsmöglichkeiten vor. Ohne auf Details einzugehen seien hier erwähnt: Syntaktische Beschreibung, relationale Graphen, semantische Netze oder Expertensysteme.

28.3.3 Die Reaktionskomponente

Die Reaktionskomponente, die auch häufig Exekutive genannt wird, enthält die Module, die für die spezielle Anwendung notwendig sind. Dazu wieder Beispiele: Wenn ein sichtgesteuerter Industrieroboter Bauteile erkennen und manipulieren soll, so wird die Exekutive alle Programme enthalten, die zur Steuerung des Roboters notwendig sind. Wenn es sich dagegen um die Auswertung mikroskopischer Zellabstriche handelt, so sind in der Reaktionskomponente alle Module zusammengefasst, die benötigt werden, um z.B. das Mikroskop und die Ver- und Entsorgung des Mikroskops mit Messpräparaten zu steuern.

Die Exekutive ist immer problemabhängig. Wenn eine neue Anwendung implementiert wird, so werden die Programme für die Exekutive in der Regel neu zu erstellen sein. Die Module der anderen Komponenten sollten aber möglichst unabhängig von der Reaktionskomponente sein.

28.3.4 Die Verwaltungskomponente

Diese Komponente ist für die gesamte Ablaufsteuerung des Systems verantwortlich. Sie besorgt z.B. die Initialisierung des Systems (der einzelnen Komponenten). Auch die Gewährleistung der Betriebssicherheit, die Ablaufsteuerung und die korrekte Beendigung des Systems fällt in den Aufgabenbereich der Verwaltungskomponente. Wenn parallele Prozesse aktiv sind, so werden die Botschaften zwischen den einzelnen Komponenten über die Verwaltungskomponente abgewickelt.

28.3.5 Die Interaktionskomponente

Die Interaktionskomponente ist die Schnittstelle des Systems zum Benutzer. In diesen Bereich fällt die gesamte Benutzeroberfläche, aber auch die Steuerung der möglichen Interaktionen mit dem System. Bei einem Industrieroboter könnte z.B. eine natürlichsprachliche Anweisung lauten: „Bauteile des Typs X sind defekt, wenn sie keine Bohrung oder mehr als zwei Bohrungen haben. Defekte Bauteile sind an der Position Y abzulegen.“ Die Interaktionskomponente hat die Aufgabe, eine Kommandosprache bereitzustellen, die eingegebenen Befehle geeignet aufzubereiten und sie an die Verwaltungskomponente weiterzuleiten.

Zu Testzwecken und zur Fehlersuche sollte die Interaktionskomponente auch Möglichkeiten bereit stellen, mit denen man in das (laufende) System „hineinschauen“ kann.

28.3.6 Die Dokumentationskomponente

Die Dokumentationskomponente enthält die gesamte Beschreibung des Systems sowie die *online*-Hilfe und möglicherweise auch die Entwicklungsdokumentation des Systems.

28.3.7 Ein Beispiel

In Bild 28.3 ist in Form eines einfachen Petri-Netzes ein System zur Robotersteuerung schematisch dargestellt. Es besteht aus Ereignis- und Zustandsknoten. Ein Ereignis kann stattfinden, wenn die für dieses Ereignis notwendigen Zustände eingetreten sind. Wenn ein Zustand eingetreten ist, wird das durch eine Markierung dieses Zustands dargestellt.

Im Beispiel sind die Zustände „IK: Bereit zur Entgegennahme von Aufträgen“ und „RK: Roboter bereit zur Initialisierung“ Anfangszustände, die nach dem Start des Systems markiert werden. Wenn ein Ereignis stattfindet, werden alle Markierungen der vorausgesetzten Zustände abgezogen und die nachfolgenden Zustände markiert. Ein Ablauf könnte etwa wie folgt aussehen:

- Da das System nach der Startphase zur Entgegennahme von Aufträgen bereit ist, wird an die Interaktionskomponente folgender Auftrag gegeben: „Alle Bauteile vom Typ X mit ungerader Lochanzahl sind defekt“.
- Die Interaktionskomponente bereitet diesen Auftrag auf und leitet ihn an die Verwaltungskomponente weiter.
- Die Verwaltungskomponente extrahiert aus dem Auftrag die für die einzelnen Komponenten notwendigen Informationen und benachrichtigt diese Komponenten. Die weiteren Schritte können teilweise parallel ablaufen.
- Die Reaktionskomponente (der Roboter) wird beauftragt, eine bestimmte Grundposition einzunehmen.
- Parallel dazu führt die Mustererkennungskomponente den Bildeinzug durch, macht bestimmte Vorverarbeitungen, segmentiert das Bild und legt die Segmente in einer passenden Datenstruktur ab.
- Parallel dazu ermittelt die Wissenskomponente, anhand der Informationen der Verwaltungskomponente und der Wissensbasis, welche Aktionen durchzuführen sind. Das könnte z.B. sein: Flächenzählung (Bauteil Typ X liegt zwischen ... und ... Bildpunkten), Formerkennung (Bauteil vom Typ X besteht aus nur einem quadratischen Segment), Ermittlung der Lochanzahl (Bauteil vom Typ X muss zwei Bohrungen haben), Ermittlung der Lage (der Greifpunkt von Bauteil Typ X ist in der Position ... und der Orientierung ...), usw.

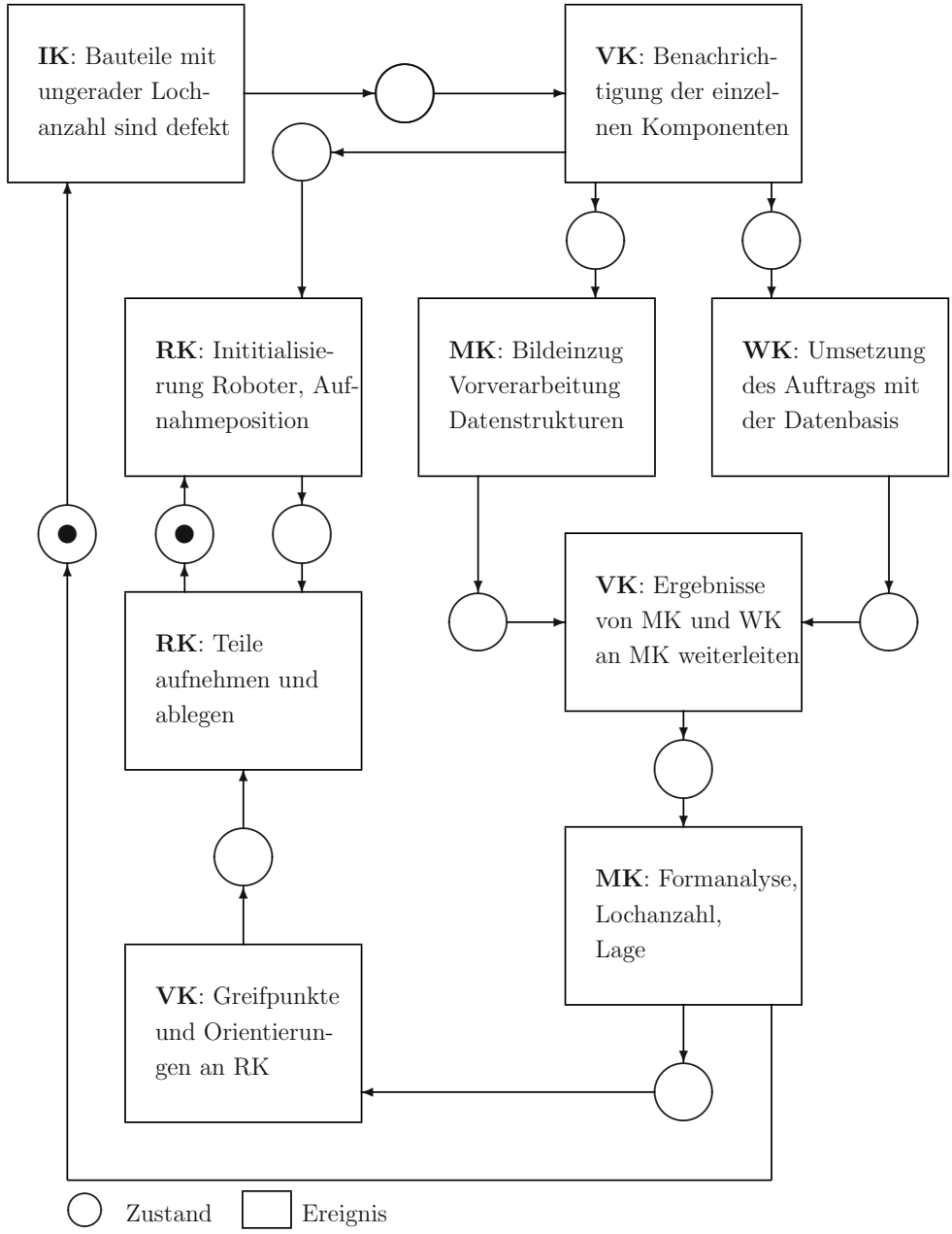


Bild 28.3: Beispiel eines einfachen Systems zum sichtgesteuerten Betrieb eines Roboters. Darstellung als Petri-Netz.

- Die beiden letzten Ereignisse müssen synchronisiert werden, da die weitere Verarbeitung erst fortgesetzt werden kann, wenn beide Ereignisse abgeschlossen sind.
- Nach einer Benachrichtigung durch die Verwaltungskomponente führt die Mustererkennungskomponente die von der Wissenskomponente erhaltenen Überprüfungen durch.
- Nach Abschluss dieses Ereignisses wird über die Verwaltungskomponente die Reaktionskomponente aktiviert, falls diese nach der obigen Initialisierungsphase bereit ist (Synchronisation). Die Reaktionskomponente nimmt nun die gefundenen Bauteile auf, legt sie entsprechend ab und ist dann wieder bereit für den nächsten Auftrag.
- Parallel dazu benachrichtigt die Verwaltungskomponente die Interaktionskomponente, dass das System zur Entgegennahme eines neuen Auftrags bereit ist.

Dieses Beispiel ist hier zwar weitgehend vereinfacht, es zeigt aber doch gut das Zusammenspiel der einzelnen Systemkomponenten.

28.4 Bildverarbeitungssysteme im Einsatz

Abschließend sind in diesem Abschnitt einige Bemerkungen zusammengestellt, die die manchmal leidvollen Erfahrungen bei der Implementierung von Systemen mit bildverarbeitenden Komponenten widerspiegeln.

Bei der Planung des Systems sollte der Auftraggeber soweit wie möglich mit einbezogen werden, um sicherzustellen, dass das gewünschte System entsteht. Oft weiß der Auftraggeber nicht genau, was er will.

Häufig werden derartige Systeme aus Gründen der Rationalisierung geplant. Wenn ein System mehrere Arbeiter ersetzt und vielleicht noch eine bessere Kontinuität liefert, so wird es schwerlich für einige tausend Mark zu realisieren sein. Sollte der Auftraggeber hier nicht einsichtig sein, so verzichtet man besser auf den Auftrag.

Der größere Kostenanteil liegt meistens nicht bei der Hardware, sondern bei der Softwareerstellung. Diese Seite kann etwas geringer gehalten werden, wenn der Auftragnehmer über eine modulare Methodenbasis (siehe Bemerkungen in Abschnitt 28.3.1) verfügt.

Die Erfahrung zeigt, dass der Implementierungsanteil für die Bildverarbeitungs- und Mustererkennungskomponenten oft nicht der umfangreichste Teil des gesamten Systems ist. Viel aufwändiger (und damit teurer) ist z.B. die gesamte Benutzeroberfläche, die zur einfachen Bedienung des Systems geschaffen werden muss. Ein System, das in einer Werkhalle nur von einem promovierten Ingenieur betrieben werden kann, ist wertlos.

Wenn man sich bei der Planung des Systems an einigen Stellen nicht ganz sicher ist, wie sie aus algorithmischer Sicht mit den Methoden der Bildverarbeitung und Mustererkennung zu lösen sind, sollte eine Durchführbarkeitsstudie vereinbart werden. Im Rahmen dieser Studie ist zu prüfen:

- Wie lassen sich die Teilprobleme prinzipiell lösen?

- Welche der gefundenen Verfahren lassen sich im gegebenen Zeitrahmen für die Systemreaktion realisieren?
- Lassen sich einzelne Algorithmen so vereinfachen, dass sie die Aufgabe noch lösen und in den Zeitrahmen passen?
- Sind die gefundenen Verfahren mit dem finanziellen Rahmen verträglich (Kosten der Hardware, Kosten der Implementierung)?

Wenn das System als Labormuster läuft, so ist es noch ein weiter Weg zum operationellen System in der Werkhalle. Das System sollte nach Möglichkeit von einem Team von Fachleuten aus verschiedenen Bereichen erstellt werden. Ein Informatiker, der Spezialist auf dem Sektor der Bildverarbeitung ist, wird in der Regel nicht in der Lage sein, ein System erfolgreich in eine Fertigungslinie zu integrieren.

Das System muss Komponenten zur Selbstkontrolle enthalten. Dazu ein Beispiel: Bei einem in eine Fertigungslinie integrierten System wird im Lauf der Zeit die Linse des Sensors verschmutzen oder die Lichtquellen werden an Leuchtkraft verlieren. Ab einer gewissen Stufe sind dann die Voraussetzungen für die Funktionsfähigkeit nicht mehr gegeben und das System wird fehlerhaft arbeiten. Wenn das System Kontrollen für diesen Fall selbst durchführt, kann rechtzeitig ein Alarm ausgelöst werden.

Da in der Industrie zunehmend flexible Fertigungssysteme zum Einsatz kommen, muss das Inspektionssystem ebenfalls flexibel auf den jeweils zu fertigenden Auftrag reagieren. Es wäre sehr behindernd, wenn für die unterschiedlichen Fertigungsaufträge z.B. die Lichtquellen für die Sensoren mit der Hand neu justiert werden müßten.

Es ist auch zu beachten, dass für das Inspektionssystem ein geeigneter Ort gewählt wird. Dafür kommen vor allem Positionen in Frage, an denen die Teile für gewisse Taktzeiten unbewegt liegen, was den Inspektionsvorgang wesentlich erleichtert. Wenn es möglich ist, sollte man auch darauf achten, dass bei einem Ausfall des Systems nicht die gesamte Produktionslinie zum Stehen kommt. Ein mehrstündiger Stillstand kann sehr teuer sein!

Literaturverzeichnis

[Abma94] Abmayr W.: *Einführung in die digitale Bildverarbeitung.* Teubner Verlag, Stuttgart, 1994

[Alon00] Alon J., Sclaroff S.: *Recursive Estimation of Motion and Planar Structure.* Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2000

[Azar95] Azarbayejani A., Pentland A.: *Recursive estimation of motion, structure, and focal length.* IEEE Trans. on Pattern Analysis and Machine Intelligence 17, 562-575, 1995

[Barn92] Barnsley M.F., Hurd L.P.: *Fractal Image Compression.* Wellesley, AK Peters Ltd., 1992

[Bay06] Bay H., Tuytelaars T., Van Gool L.: *SURF: Speeded Up Robust Features.* Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008

[Baye76] Bayer B.E.: *Color Imaging Array.* U.S. Patent 3.971.065, 1976

[Boeh93] Böhme G.: *Fuzzy-Logik, Einführung in die algebraischen und logischen Grundlagen.* Springer Verlag, Berlin Heidelberg New York, 1993

[Bose04] Bose T.: *Digital Signal and Image Processing.* John Wiley & Sons, 2004

[Brow07] Brown M., Lowe, D.G.: *Automatic Panoramic Image Stitching using Invariant Features.* International Joural of Computer Vision, Volume 74, Number 1, pp. 59-73, August 2007

[Burg06] Burger W., Burge, M.J.: *Digitale Bildverarbeitung: Eine Einführung mit Java und ImageJ.* 2. überarbeitete Auflage, Springer-Verlag Berlin Heidelberg, 2006

[Burt83] Burt P.J., Adelson E.H.: *A Multiresolution Spline With Applications to Image Mosaicing.* ACM Transactions on Graphics, Vol. 2, No. 4, 217-236, 1983

[Burt84] Burt P.J.: *The Pyramid as a Structue for Efficient Computation.* Multiresolution Image Processing and Analysis, Rosenfeld A., ed., Springer-Verlag, Berlin Heidelberg New York, 6-35, 1984

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4

[Calv00] Calvagno G., Mian G.A., Rinaldo R.: *3D Motion Estimation for Frame Interpolation and Video Coding.* Proc. Int. Workshop on Packet Video, Cagliari, 2000

[Calo12] Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: *BRIEF: computing a local binary descriptor very fast.* IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 1281–1298, 2012

[Cann83] Canny J.: *Finding edges and lines in images.* Technical Report TR-720, MIT, Cambridge, MA, USA, 1983

[Cann86] Canny J.: *A computational approach to edge detection.* IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986

[Dala05] Dalal N., Triggs B.: *Histograms of oriented gradients for human detection.* IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, Vol. 1, pp. 886–893, 2005

[Deni11] Déniz O., Bueno G., Salido J., and De la Torre F.: *Face recognition using histograms of oriented gradients.*, Pattern Recognition Letters, Vol. 32, no. 12, pp. 1598–1603, 2011

[DIN6174] DIN 6174: *Farbmetrische Bestimmung von Farbmaßzahlen und Farbabständen im angenähert gleichförmigen CIELAB-Farbenraum.* DIN Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 2007.

[DIN6176] DIN 6176: *Farbmetrische Bestimmung von Farbabständen bei Körperfarben nach der DIN99-Formel.* DIN Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 2001.

[Dorf03] Dorfmüller-Ulhaas K.: *Robust Optical User Motion Tracking Using a Kalman Filter.* VRST'03, Osaka, 2003

[Ekel92] Ekeland I.: *Zufall, Glück und Chaos. Mathematische Expeditionen.* Carl Hanser Verlag, München Wien, 1992

[Enge96] Engeln-Müllges G., Reutter F.: *Numerik-Algorithmen.* VDI Verlag, Düsseldorf, 1996

[Erns91] Ernst H.: *Einführung in die digitale Bildverarbeitung.* Franzis-Verlag, München, 1991

[Ever90] Evers Ch.: *Erweiterung der Hough-Transformation für die Erkennung von Objekten, die nicht durch ihre Konturkurve charakterisiert werden können.* Mustererkennung 1990, 209-219, Springer Verlag, Berlin Heidelberg New York, 1990

[Faug93] Faugeras O.: *Three-Dimensional Computer Vision.* The MIT Press, Cambridge Massachusetts, 1993

[Fail04] Faille F.: *Comparison of demosaicking methods for color information extraction.* Proceedings of „Computer Vision and Graphics International Conference“, ICCVG 2004, p. 820-825, Warsaw, Poland. ISBN: 978-1-4020-4178-5, 2004

[Felz10] Felzenszwalb P.F., Girshick R. B. , McAllester D. and Ramanan D.: *Object Detection with Discriminatively Trained Part-Based Models.* In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627-1645, Sept. 2010, doi: 10.1109/TPAMI.2009.1679

[Fisc81] Fischler M.A., Bolles R.C.: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Communications of the ACM, Volume 24, Number 6, pp. 381-395, 1981

[Fish92] Fisher Y.: *Fractal Image Compression.* Siggraph 92, Course Notes, 1992

[Fung05] Fung J.: *Computer Vision on the GPU.* in GPU Gems 2, ed. by Matt Pharr, 649-666, Addison-Wesley, Reading, 2005

[Gonz18] Gonzalez R., Woods E.: *Digital Image Processing.* 4th Edidtion, Pearson Prentice Hall, New Jersey, 2018

[Gonz09] Gonzalez R., Woods E., Eddins S.: *Digital Image Processing using MATLAB.* Second Edidtion, Gatesmark Publishing, 2009

[Grün02] von Grünigen D.Ch.: *Digitale Signalverarbeitung.* Fachbuchverlag Leipzig im Carl Hanser Verlag, 2002

[Habe83] Haberäcker P., Thiemann R.: *Szenenanalytische Auswertung von digitalisierten Luftbildern mit Baumstrukturen.* Mustererkennung 83, 5. DAGM-Symposium, VDE-Verlag, Berlin, 1983

[Habe84] Haberäcker P., Thiemann R.: *Klassifizierung von Siedlungen in digitalisierten Luftbildern, die als Quad-Trees codiert sind.* Mustererkennung 84, 49-54, Springer, 1984

[Harr88] Harris C., M. Stephens M.: *A combined corner and edge detector.* Proceedings of the 4th Alvey Vision Conference. pp. 147-151, 1988

[Hart04] Hartley R., Zisserman A.: *Multiple View Geometry in Computer Vision.* 2. Auflage, Cambridge University Press, Cambridge, UK, 2004

[Hami97] Hamilton J.F.Jr., Adams J.E.Jr.: *Adaptive color plane interpolation in single sensor color electronic camera.* U.S. Patent 5.629.734, 1997

[Haus19] Hausdorff F.: *Dimension und äußeres Maß.* Mathematische Annalen 79, 157-179, 1919

[Heyn03] Heyna A. et.al.: *Datenformate im Medienbereich.* Fachbuchverlag Leipzig im Carl Hanser Verlag, 2003

[Hira05] Hirakawa K., Parks T.W.: *Adaptive homogeneity-directed demosaicing algorithm.* IEEE Transactions on Image Processing, Vol. 14, No.3, 2005

[Horn81] Horn B.K.P., Schunck B.G.: *Determining Optical Flow.* Artificial Intelligence 17, pp. 185-203, 1981

[Houg62] Hough P.V.C.: *Methods and Means for Recognizing Complex Patterns.* U.S. Patent 3,069,654, 1962

[Huan92/1] Huang C.L., Cheng T.Y., Chen C.-C.: *Color image segmentation using scale space filtering and Markov random field.* Pattern Recognition, Vol. 25, No. 10, 1217-1230, 1992

[Huan94] Huang Q., Lorch J.R., Dubes R.C.: *Can the fractal dimension of images be measured.* Pattern Recognition, Vol. 27, No. 3, 339-350, 1994

[Huan02] Huang Y., Huang T.S., Niemann H.: *Segmentation-based Object Tracking Using Image Warping and Kalman Filtering.* Proc. IEEE Int. Conference on Image Processing, Rochester, New York, 2002

[Hu62] Hu M.K.: *Visual Pattern Recognition by Moment Invariants.* IRE Transactions on Information Theory, Vol. IT-8, 179-187, 1962

[Jaeh11] Jähne B.: *Digital Image Processing.* Springer Verlag, Berlin; 7th Edition, 2011

[Jeke01] Jekeli C.: *Inertial Navigation Systems with Geodetic Applications.* Walter de Gruyter, Berlin, 2001

[Kalm60] Kalman R.E.: *A New Approach to Linear Filtering and Prediction Problems.* Trans. of the ASME 82, 35-45, 1960

[Ke04] Ke Y., Sukthankar R.: *PCA-SIFT: A More Distinctive Representation for Local Image Descriptors.* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 506-513, Los Alamitos, CA, 2004

[Koho88] Kohonen T.: *Self-Organization and Associative Memory.* Springer Verlag, Berlin Heidelberg New York, 1988

[Kosk92] Kosko B.: *Neural Networks and Fuzzy Systems.* Prentice-Hall, Englewood Cliffs, 1992

[Krat90] Kratzer K.P.: *Neuronale Netze.* Carl Hanser Verlag, München Wien, 1990

[Leut11] Leutenegger, S., Chli, M., Siegwart, R.: *BRISK: binary robust invariant scalable keypoints.* In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2548–2555, 2011

[Lohm88] Lohmann B.: *Entwicklung von Bildverarbeitungsverfahren zur Positionsbestimmung eines mobilen Roboters.* Zwischenbericht 6/88 SFB 314 V2, Universität Bremen, 1988

[Lowe04a] Lowe D.G.: *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, Volume 60, Number 2, pp. 91-110, 2004

[Lowe04b] Lowe D.G.: *Method and Apparatus for Identifying Scale Invariant Features in an image and use of same for locating an object in an image.* U.S. Patent 6,711,293 B1, 2004

[Ludy95] Ludyk G.: *Theoretische Regelungstechnik 2: Zustandsrekonstruktion, optimale und nichtlineare Regelungssysteme.* Springer Verlag, Berlin, 1995

[Malv04] Malvar H.S., He Li-Wei, Cutler R.: *High-quality linear interpolation for demosaicing of Bayer-patterned color images.* IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004 (ICASSP 04), p.485-488, ISBN: 0-7803-8484-9, 2004

[Mand87] Mandelbrot B.B.: *Die fraktale Geometrie der Natur.* Birkhäuser Verlag, Basel Boston Berlin, 1987

[Mayb79] Maybeck P.S.: *Stochastic Models, Estimation and Control, Vol. 1.* Academic Press, London, 1979

[Mess05] Messom C. H., Gupta G. S. und Demidenko S.: *Hough transform run length encoding for real-time image processing.* In Instrumentation and Measurement Technology Conference, Ottawa, Canada, 2005

[Miko05] Mikolajczyk K., Schmid C.: *A performance evaluation of local descriptors.* IEEE Transactions on Pattern Analysis & Machine Intelligence, PAMI, Volume 27, Number 10, pp. 1615–1630, Oktober 2005

[Nage92] Nagel H.-H.: *Direct Estimation of Optical Flow and its Derivatives.* Artificial and Biological Vision Systems, G.A. Orban and H.-H. Nagel (eds.), Springer Verlag Berlin etc., pp. 193-224, 1992

[Nisc10] Nischwitz A.: *Verfahren zur Interpolation von Farbwerten an Bildpunkten eines Bildsensors (Demosaicing).* Patent Nr. DE102008063970A1 beim Deutschen Patent- und Markenamt veröffentlicht am 24.06.2010

[Owen05] Owens J.: *Streaming Architectures and Technology Trends.* in GPU Gems 2, ed. by Matt Pharr, 457-470, Addison-Wesley, Reading, 2005

[Peit86] Peitgen H.-O., Richter P.H.: *The Beauty of Fractals.* Springer Verlag, Berlin Heidelberg New York, 1986

[Pere02] Pereira F.C.N., Ebrahimi T.: *The MPEG-4 book.* IMSC Press multimedia series, New Jersey, 2002

[Peso16] Peso D., Nischwitz A., Ippisch S., Obermeier P.: *Kernelized Correlation Tracker on Smartphones.* Journal of Pervasive and Mobile Computing. Elsevier. PMCJ719, 2016

[Rost06] Rosten E., Drummond T.: *Machine learning for high-speed corner detection.* In European Conference on Computer Vision, Volume 1, pages 430–443, May 2006

[Same82] Samet H.: *Neighbor Finding Techniques for Images Represented by Quad-Trees.* Computer Graphics and Image Processing, Vol. 18, No. 1, 37-57, 1982

[Scho03] Schöttl A.: *Kalman Filterung: Grundlagen, Anwendungen, neue Trends.* Carl Cranz Gesellschaft e.V., Weßling, 2010

[Serr82] Serra J.: *Image Analysis and Mathematical Morphology, Vol. 1.* Academic Press, London, 1982

[Serr88] Serra J.: *Image Analysis and Mathematical Morphology, Vol. 2: Theoretical Advances.* Academic Press, London, 1988

[Shan48] Shannon C.E.: *A mathematical theory of communication .* Bell Systems Technical Journal, Volume 27, pp. 379-423,623-656, 1948

[Smit78] Smith A.R.: *Color Gamut Transform Pairs.* SIGGRAPH, 12-19, 1978

[Strz03] Strzodka R., Ihrke I., Magnor M.: *A Graphics Hardware Implementation of the Generalized Hough Transform for Fast Object Recognition, Scale, and 3D Pose Detection.* in Proceedings of IEEE International Conference on Image Analysis and Processing, 188-193, 2003

[Suma05] Sumanaweera T., Liu D.: *Medical Image Reconstruction with the FFT.* in GPU Gems 2, ed. by Matt Pharr, 765-784, Addison-Wesley, Reading, 2005

[Tarj75] Tarjan R. E.: *Efficiency of a good but not linear set union algorithm.* Journal of the ACM, 22(2):215–225, 1975

[Toen05] Tönnies K.: *Grundlagen der Bildverarbeitung.* Pearson Studium, München, 2005

[Vanp93] Van Pabst J.V.L., Krekel P.F.C.: *Multi Sensor Data Fusion of Points, Line Segments and Surface Segments in 3D Space.* 7th International Conference on Image Analysis and Processing, Singapore, 174-182, 1993

[Viol95] Viola P., Wells, III W.M.: *Alignment by maximization of mutual information.* International Journal of Computer Vision 24, 2, pp. 137-154, September 1997

[Wass89] Wassermann P.D.: *Neural Computing.* Van Nostrand Reinhold, New York, 1989

[Watk01] Watkinson J.: *MPEG Handbook.* Focal Press, 2001

[Watt02] Watt A.: *3D-Computergrafik. 3. Auflage.* Pearson Studium, München, 2002

[Welc95] Welch G., Bishop G.: *An Introduction to the Kalman Filter.* TR 95-041, University of North Carolina at Chapel Hill, 1995

[Wilk00] Wilkinson M. und Roerdink J.: *Fast morphological attribute operations using tarjan's union-find algorithm.* In J. Goutsias, L. Vincent, and D. S. Bloomberg, Hrsg., Mathematical Morphology and Its Application to Image and Signal Processing. Kluwer, 2000

[Witk83] Witkin A.P.: *Scale Space Filtering.* Proc. International Joint Conference on Artificial Intelligence, Karlsruhe, 1983

[YiTr16] Yi K.M., Trulls E., Lepetit V., Fua P.: *LIFT: Learned Invariant Feature Transform.* In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, Vol. 9910. Springer, Cham, 2016

[Ying04] Ying-Kin Y., Kin-Hong W., Ming-Yuen Chang M.: *Recursive 3D Model Reconstruction Based on Kalman-Filtering.* IEEE Transactions on Systems, Man and Cybernetics – Part B (akzeptiert), 2004

[Zimm92] Zimmermann H.-J.: *Fuzzy Set Theory and its Applications.* Kluwer Academic Publishers, Boston, 1992

Sachverzeichnis

A. Nischwitz et al., *Bildverarbeitung*,
https://doi.org/10.1007/978-3-658-28705-4