

LEHRBUCH

Herbert Süße
Erik Rodner

Bildverarbeitung und Objekterkennung

Computer Vision in Industrie und Medizin



Springer Vieweg

Bildverarbeitung und Objekterkennung

Herbert Süße · Erik Rodner

Bildverarbeitung und Objekterkennung

Computer Vision in Industrie und Medizin



Springer Vieweg

Dr. Herbert Süße
Dr. Erik Rodner
Lehrstuhl für Digitale Bildverarbeitung
Institut für Mathematik und Informatik
Friedrich-Schiller-Universität Jena
Jena, Deutschland

ISBN 978-3-8348-2605-3
DOI 10.1007/978-3-8348-2606-0

ISBN 978-3-8348-2606-0 (eBook)

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

Springer Vieweg
© Springer Fachmedien Wiesbaden 2014
Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.

Springer Vieweg ist eine Marke von Springer DE. Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-vieweg.de

Vorwort

Ein Lehrbuch über die Bildverarbeitung zu schreiben ist genauso unmöglich wie ein Lehrbuch der Mathematik oder Physik herauszugeben. Die Bildverarbeitung ist inzwischen so spezialisiert, dass man über jedes Teilgebiet ein eigenes Lehrbuch herausgeben könnte. Im folgenden Buch versuchen wir trotzdem einen relativ breiten Überblick über das Gebiet zu geben, welcher aber nicht im geringsten den Anspruch auf Vollständigkeit erhebt.

Insbesondere Studenten und Anwender, welche das erste Mal mit dem Thema der Bildverarbeitung in Berührung kommen, verwechseln dieses mit der Bildbearbeitung. Bei der Bildbearbeitung beschäftigt man sich hingegen mit der *manuellen* Modifikation von Bildern zum Zwecke der „optischen Verbesserung“ oder um Fotomontagen zu erstellen. Im Gegensatz dazu steht die Bildverarbeitung, welche sich mit der *automatischen maschinellen Analyse* von Bildern beschäftigt. Trotz der bereits schon seit ein paar Jahrzehnten betriebenen Forschung, ist in den letzten Jahren erst ein unglaublicher Boom möglicher Anwendungen zu beobachten. An dieser Stelle sei zum Beispiel die automatische Fussgängererkennung im Automobilbereich oder die Gesichtsdetektion bei aktuellen Digitalkameras und Mobiltelefonen genannt. Diese Anwendungen basieren auf Algorithmen der Bildverarbeitung und zeigen nur einen kleinen Bereich der Möglichkeiten der automatischen Bildanalyse auf.

Unser Buch wendet sich an Studenten, Anwender und Spezialisten der Bildverarbeitung. Studenten sollen mit dem Buch an die Bildverarbeitung herangeführt werden um das Interesse an diesen Themenbereichen zu wecken. Für Anwender und Spezialisten sind diejenigen Abschnitte geeignet, in denen ausführlich einzelne spezifische Problemstellungen dargelegt werden, welche Bestandteil der jahrelangen Forschung der Autoren waren und immer noch sind. Wir hoffen, dass die Spezialisten in diesen Abschnitten einige Anregungen finden. Insgesamt kann das vorliegende Buch als eine Art Zusammenfassung der Lehr- und Forschungstätigkeit der Autoren am Lehrstuhl Digitale Bildverarbeitung der Friedrich-Schiller-Universität Jena angesehen werden.

Eine besondere und spannende Herausforderung der Bildverarbeitung ist die Tatsache, dass diese Disziplin auf vielen anderen Wissenschaftsdisziplinen basiert. So sind Aspekte aus der Mathematik (Optimierung, Numerik, lineare Algebra, ...) genauso wichtig wie Konzepte aus der Physik aber vor allem auch der Informatik. Die aktuelle Forschung in diesem Bereich durchläuft oft alle Stufen der Entwicklung, von der mathematischen Mo-

dellierung der Problemstellung über die algorithmische Konzeption bis hin zur fertigen Software. Dadurch entsteht eine Einheit aus Theorie und Praxis, welche in wenigen anderen Wissenschaften in dieser Form zu finden ist. Die Bildverarbeitung ist eng verwandt mit den folgenden Gebieten:

1. Computergrafik
2. Mustererkennung, maschinelles Lernen und künstliche Intelligenz
3. Digitale Geometrie
4. Berechenbare Geometrie
5. Stereologie
6. Stochastische Geometrie.

Wir werden bestimmte thematische Schnittmengen mit diesen Gebieten auch in diesem Buch näher erläutern.

Ein besonderer Schwerpunkt des Buches wird es sein Beziehungen und Parallelen zwischen Problemstellungen und Lösungen zu ziehen. Oft wird ein sehr allgemeines mathematisches Konzept bei unterschiedlichen Problemstellungen zur Lösung herangezogen, aber von der Notation und Motivation ganz anders dargestellt. Im Buch werden wir daher bei bestimmten Themengebieten nicht die algorithmischen Details beleuchten, sondern eher die mathematische Modellierung und deren Motivation betrachten. Da die Bildverarbeitung auf vielen Wissenschaftsdisziplinen basiert, ist ein zusätzliches Problem die Verwendung einheitlicher Begriffe. So werden manchmal die gleichen Konzepte mit unterschiedlichen Begriffen bezeichnet. Beispiele dafür sind die Begriffe Varianzen und Kovarianzen, welche in der Physik als Momente zweiter Ordnung bezeichnet werden. Zusätzliche Verwirrung entsteht durch die inflationäre Verwendung von Schlagworten, welche mit der ursprünglichen Definition nichts mehr gemein haben, als Beispiel sei an dieser Stelle der Begriff des „Strukturtensors“ erwähnt.

Da die Bildverarbeitung eine relativ junge Wissenschaft ist, haben sich viele Begriffe in englischer Notation eingebürgert. Oft geben wir in Klammern den englischen Begriff an. Die digitale Bildverarbeitung selbst wird als *Computer Vision* bezeichnet, wobei man oft zwischen 2D- und 3D-*Computer Vision* unterscheidet. Im Buch werden wir den Begriff *Computer Vision* als Synonym für die Bildverarbeitung verwenden. Weiterhin werden wir nur vereinzelt auf wichtige Werke der Bildverarbeitung verweisen und haben bewusst auf umfangreiche Literaturlisten verzichtet. Dem interessierten Leser sei an dieser Stelle empfohlen auf einschlägige Suchdatenbanken für wissenschaftliche Veröffentlichungen zurückzugreifen, wie etwa scholar.google.com. Sinnvolle Schlagworte für die Suche sind in jedem Abschnitt hervorgehoben. Wir möchten an dieser Stelle auch auf die Internetseite des Buches hinweisen:

<http://www.dbvbuch.de>

Auf dieser Seite werden wir Links und Ergänzungen zur Verfügung stellen.

Die Entstehung des Buches verdanken wir gemeinsamen Forschungsarbeiten und regen Diskussionen mit vielen Kollegen. Insbesondere gilt unser Dank *Prof. Klaus Voss, Dr. Wolfgang Ortmann* und *Prof. Joachim Denzler*. Besonderer Dank geht auch an *Dr. Marcel Brückner, Dr. Martin Rapus, Dr. Michael Kemmler, Dr. Björn Fröhlich* und *Daniel Haase* für das Bereitstellen einiger Abbildungen. Herrn *Dr. Christian Lautenschläger* vom Klinikum Jena danken wir für die erfolgreiche Zusammenarbeit auf dem Gebiet der medizinischen Bildverarbeitung. Weiterhin sind wir dankbar für das unermüdliche Korrekturlesen vieler Kollegen: *Paul Bodesheim, Alexander Freytag, Daniel Haase, Christoph Käding, Marco Körner, Mahesh Venkata Krishna, Sven Sickert* und *Johannes Rühle*.

Zu guter Letzt danken wir Irene Süße und Wiebke Rodner für Geduld, Aufmunterung, Verständnis und Motivation während der gesamten Zeit und insbesondere in den langen Nächten des Schreibens.

Jena, im März 2014

Herbert Süße und Erik Rodner

Inhaltsverzeichnis

Teil I 2D-Bildverarbeitung

1	Elementare Grundlagen	3
1.1	Abtastung	3
1.2	Diskrete Natur von digitalen Bildern	6
1.2.1	Gittertypen	7
1.2.2	Kuriositäten im Quadratgitter	10
1.2.3	Digitale Konvexität	10
1.2.4	Silhouetten	12
1.2.5	Perkolation	16
1.3	Bildmodelle	17
1.4	Bildtransformationen	18
1.4.1	„Reine“ Grauwertransformationen	18
1.4.2	Polar- und Log-Polar-Darstellungen	20
1.4.3	Globale geometrische Transformationen	21
1.4.4	Praktische Durchführung der Transformationen	23
1.4.5	Beispiel OCR	24
2	Die Operationen Faltung und Korrelation	27
2.1	Faltung und Korrelation	27
2.1.1	Faltung	27
2.1.2	Korrelation	31
2.1.3	Ableitungen der Faltung	36
2.1.4	<i>Wrap-around effect</i> und <i>zero padding</i>	37
2.2	Zirkularmatrizen	38
2.2.1	Elementare Eigenschaften	38
2.2.2	Normabschätzungen und Stabilität	41
2.3	LSI-Operatoren	43
2.4	Faltungsgleichungen im Ortsraum	48
2.4.1	Iterative Entfaltung	48
2.4.2	Beschränkte Entfaltungskerne	49

2.5	FIR und IIR Filter	49
2.6	Schnelle Faltungen im Ortsraum	51
3	Bildtransformationen	53
3.1	Grundlagen und Basissysteme	53
3.2	Analoge Fouriertransformation im endlichen Intervall (AFT)	57
3.3	Die endliche diskrete Fouriertransformation (DFT)	60
3.4	Die Fourier-Integraltransformation (IFT)	62
3.4.1	Integraltransformation als Grenzwert	62
3.4.2	Integraltransformation mit stetigen Basen	64
3.5	Die unendliche diskrete Fouriertransformation (UDFT)	65
3.6	Fourier-Mellin-Transformation (FMT)	66
3.7	Hilbert-Transformation (HIT)	67
4	Grundlegende Eigenschaften der Fouriertransformation	71
4.1	Bezeichnungen	71
4.2	Nullter Fourierreduktionsfaktor, Gleichanteil	72
4.3	Konvergenzverhalten, Gibbssches Phänomen	72
4.4	Linearität der Fouriertransformation	74
4.5	Periodizität der Fourierreduktionsfaktoren	75
4.6	Spektrum reeller Funktionen	77
4.7	Parsevalsche Gleichung	77
4.8	Faltungstheorem	78
4.9	Spektrum der Korrelationsfunktion	80
4.10	Kohärenz	80
4.11	Eigenwerte und Eigenfunktionen bezüglich der Faltung	81
4.12	Normabschätzungen und Kondition	82
4.13	Faltungs-Iterationen	83
4.14	Reelle diskrete Fouriertransformation – Hartleytransformation	84
4.15	Separierbarkeit der mehrdimensionalen Fouriertransformation	86
4.16	Implementierung der inversen Fouriertransformation	86
4.17	Verschiebungstheorem	87
4.18	Fouriertransformierte spezieller Funktionen	88
4.19	Spektrum der Ableitungen	93
4.20	Periodische Bilder	94
4.21	Fouriertransformation für komplexe Funktionen	96
4.21.1	Blockmatching	96
4.21.2	Trigonometrische Interpolation	97
4.21.3	Trigonometrische Approximation	98
4.21.4	Trigonometrisches Ellipsenfitting	99
4.21.5	Normalisierung von Punktmengen	101
4.21.6	Fourierdeskriptoren und Translationen	104

4.21.7	Fourierdeskriptoren und isotrope Skalierungen	104
4.21.8	Fourierdeskriptoren für Rotation und Startpunktwahl	105
4.21.9	Fourierdeskriptoren in der Praxis	107
4.22	Cepstrum	109
4.23	Modifizierte diskrete Fouriertransformationen (MDFT)	110
4.24	Diskrete Kosinus Transformation (DCT)	112
4.25	Unschärferelation	115
4.26	Affine Transformationen	116
4.27	Lokales Energiemodell	118
4.27.1	Lokale Energie	118
4.27.2	Phasenkongruenz	119
4.27.3	Funktionen mit hoher Phasenkongruenz	121
5	Abtasttheoreme	123
5.1	Abtasttheorem im endlichen Intervall	123
5.2	Abtasttheorem für das unendliche Bildmodell	129
5.3	Abtasttheorem für 2D-Signale	133
5.4	Anwendungen des Abtasttheorems	134
5.4.1	Verkleinern von Bildern (Shrinking)	134
5.4.2	Vergrößern von Bildern (Zooming, Superresolution)	134
5.4.3	Superresolution und Bildrestaurierung	139
5.4.4	Bemerkungen zum Abtasttheorem	142
5.4.5	Pyramiden	145
6	Orts-Frequenz-Darstellungen	147
6.1	Der Leck-Effekt (Leakage effect)	147
6.2	Das komplexe Spektrogramm	148
6.3	Wigner-Ville-Orts-Frequenz-Darstellung	151
6.4	Gabor-Filter	152
6.5	Quadratur-Filter (Energie-Filter)	153
7	Filterentwurf im Frequenzraum	155
7.1	Tiefpassfilter	155
7.2	Unscharfe Maskierung (Unsharp Masking)	156
7.3	Pruning-Filter	158
7.4	Homorphe Filter	158
7.5	DoB-Filter	158
7.6	Scharfe Bilder (Fokussierung)	161
8	Filter im Ortsraum	165
8.1	Lineare und verschiebungsinvariante Filter	165
8.1.1	Berechnung der Filterkoeffizienten	165
8.1.2	Implementierung von LSI-Filtren	169

8.2	Nichtlineare Filter	170
8.2.1	Funktionen von LSI-Filttern	171
8.2.2	Richtungsfilter	174
8.2.3	Relaxationsfilter	176
8.2.4	Morphologische Filter	176
8.2.5	Rangordnungsfilter	180
9	Stochastische Bildsignale	183
9.1	Grundbegriffe der Informationstheorie	183
9.2	Statistiken n -ter Ordnung	189
9.3	Rauschmodelle	190
9.3.1	Rauschminderung mit mehreren Aufnahmen	192
9.3.2	Rauschen und Bilddatenkompression	193
9.4	Energiefunktionen	193
9.5	Bildrestaurierung	195
9.5.1	Invers-Filter	196
9.5.2	Restaurierung unter Zwang	196
9.5.3	Wiener-Hellstrom-Optimal-Filter	199
9.5.4	Richardson-Lucy-Algorithmus	202
9.5.5	Bewegungsunschärfe (<i>motion blur</i>)	203
9.5.6	Wellenfront Kodierung (<i>wavefront coding</i>)	209
9.5.7	Kodierte Apertur (<i>coded aperture imaging</i>)	209
10	Bildsegmentierung	211
10.1	Thresholding	212
10.2	Konturfolgeverfahren bei binärer Quantisierung	213
10.3	Konturfolgeverfahren bei lokaler ternärer Quantisierung	215
10.4	Konturfolgeverfahren bei beliebigen Punktmengen	218
10.5	<i>Seeded region growing</i>	220
10.6	Anwendung in der Biofilmanalyse	221
10.7	Liniendetektion	222
10.7.1	Dynamische Programmierung	222
10.7.2	Dijkstras Algorithmus	226
10.7.3	Graphbasierte Methoden	229
10.8	Akkumulations- oder Votingmethoden	231
10.8.1	Primale Voting-Methode zur Detektion von Geraden	231
10.8.2	Duale Voting-Methode: Hough-Transformation	233
10.8.3	Zeit- und Speicher-Effizienz	234
10.8.4	Hough-Transformation und Konvexgeometrie	235
10.8.5	Verallgemeinerungen	237

11	Farbbildverarbeitung	241
11.1	Spektrale Farben	242
11.2	Visuelle Farbwahrnehmung	243
11.3	Farbtafeln	244
11.4	Farbmodelle	246
11.4.1	Technikorientierte Farbmodelle	246
11.4.2	Wahrnehmungsorientierte Farbmodelle	249
11.5	Operationen auf Farbbildern	250
12	Texturen	253
12.1	Einführung	253
12.2	Elementare statistische Merkmale	255
12.3	Autoregressive Prozesse (AR)	256

Teil II 3D-Bildverarbeitung

13	3D-Geometrie	261
13.1	Geometrische Transformationen und homogene Koordinaten	261
13.1.1	Geometrische Transformationen in der Ebene	261
13.1.2	Homogene Koordinaten	263
13.2	Beschreibung von 3D-Rotationen	269
13.2.1	Beschreibung durch Rotationsmatrizen	269
13.2.2	Bestimmung der Drehachse und des Drehwinkels	272
13.2.3	Drehmatrix aus Drehachse und Drehwinkel	273
13.2.4	Beschreibung von 3D-Rotationen durch Quaternionen	273
13.2.5	Exponentielle Form einer Quaternion	279
13.2.6	Quaternionen über komplexen Zahlen	280
13.2.7	Dot-Produkte	281
13.2.8	Beispiele	281
13.2.9	Kanonisch exponentielle Darstellung einer Drehmatrix	283
13.2.10	Vor- und Nachteile	286
13.2.11	Eine Anwendung zur Bestimmung einer Rotation	286
13.2.12	Duale Zahlen, Duplex-Zahlen	288
13.2.13	Duale Quaternionen	289
13.2.14	Euklidische Transformationen und duale Quaternionen	290
14	Geometrie der Abbildungsprozesse	293
14.1	Kameramodelle	293
14.1.1	Orthografische Projektion	294
14.1.2	Skalierte orthografische Projektion	294
14.1.3	Anisotrop skalierte orthografische Projektion	295
14.1.4	Affine Kamera	296

14.1.5	Spezielle affine Kameras	296
14.1.6	Pinhole camera	296
14.1.7	Pinhole camera – Erweiterung 1	302
14.1.8	Pinhole camera – allgemeiner Fall	302
14.1.9	Normalisierte Koordinaten	303
14.1.10	Homographien	304
14.1.11	Kamerainvarianten	304
14.1.12	Homogene Koordinaten via Kartesische Koordinaten	308
14.2	Bewegung der Kamera	310
14.3	Invarianten	312
14.3.1	Punktinvarianten	312
14.3.2	Flächeninvarianten	317
14.3.3	Invarianten von Punkten, Geraden, Kurven	319
14.4	Epipolargeometrie	322
14.4.1	Zusammenhang über die Projektionsmatrizen	324
14.4.2	Zusammenhang über die Epipolartransformation	326
14.4.3	Zusammenhang über die <i>Essential Matrix</i>	328
14.4.4	Spezialfall: 3D-Punkte liegen in einer Ebene	334
14.4.5	Berechnung der Fundamentalmatrix	335
15	Kamerakalibrierung	339
15.1	Direkte Kalibrierung	339
15.2	Selbstkalibrierung	343
15.2.1	Allgemeines	343
15.2.2	Planare Kalibrierung nach Zhang	347
15.2.3	Tomasi und Kanade: Schwache Perspektive	352
15.2.4	Tomasi und Kanade: Perspektive	355
15.3	Kalibrierung mit der E-Matrix	356
15.4	Kameraparameter aus der Projektionsmatrix	356
15.5	Verzeichnungen	357
16	3D-Rekonstruktion	363
16.1	Rekonstruktion aus Projektionen	363
16.2	Rekonstruktion der 3D-Struktur von polyedrischen Objekten	367
16.2.1	<i>Weak perspective camera</i>	367
16.2.2	Affine Kamera	369
16.2.3	<i>Pinhole camera</i>	370
16.3	Triangulation von Raumpunkten	376
16.4	Bestimmung der Punktcorrespondenzen	378
16.5	Eine einfache, praktische 3D-Vermessungsaufgabe	380
16.6	Aktives Sehen	383
16.6.1	Projektion von Lichtstrahlen	383

16.6.2	Projektion mit Lichtebenen	383
16.6.3	Codierter Lichtansatz (Coded Light Approach)	384
16.6.4	Moire-Technik	387
16.6.5	Phasenshift-Verfahren	390
16.6.6	Profilometrie	393
16.6.7	Statistische Muster	394
16.6.8	Shape from focus/defocus	395
16.7	Photometrische Methoden der Rekonstruktion	398
16.7.1	Beleuchtung und Reflexion	398
16.7.2	Shape from Shading	400
16.8	Monokulare Rekonstruktion	403
16.8.1	Siebanalyse	403
16.8.2	Planare Vierecke	406
16.8.3	Rechtecke	408
16.8.4	Kugeln	410
16.8.5	Kreisscheiben	411
17	Tensoren in der Bildverarbeitung	415
17.1	Grundbegriffe der Tensoralgebra	415
17.1.1	Vektorraum V	415
17.1.2	Einsteinsche Summenschreibweise	416
17.1.3	Dualer Vektorraum V^*	416
17.1.4	Multilineare Vektorfunktionen	417
17.1.5	Multilineare Vektorabbildungen	418
17.1.6	Wechsel des Koordinatensystems	419
17.1.7	Tensoren	420
17.2	Allgemeine Tensoren	423
17.2.1	Tensoralgebra	423
17.2.2	m -Vektoren	432
17.2.3	Rang von Tensoren	433
17.2.4	Zerlegung von Tensoren	435
17.3	Der Euklidische Raum	436
17.4	Tensorfelder	438
17.5	Anwendungen in der Bildverarbeitung	440
17.5.1	Der trifokale Tensor	440
17.5.2	Der Strukturtensor	443
 Teil III Objekterkennung		
18	Maschinelles Lernen	449
18.1	Grundprinzipien des maschinellen Lernens	449
18.1.1	Funktionale Modellierung	451

18.1.2	Stochastische Modellierung	451
18.1.3	Diskriminative und generative Klassifikatoren	452
18.2	Herausforderungen des Maschinellen Lernens	454
18.2.1	Modellkomplexität, Überanpassung und Generalisierung	454
18.2.2	Lernen mit wenigen Beispielen	455
18.3	Statistische Schätzer	456
18.3.1	Maximum-Likelihood und Maximum-A-Posteriori-Schätzung	456
18.3.2	Bayes-Ansätze und Marginalisierung	458
18.3.3	MMSE-Schätzer	459
18.3.4	Kombination von Modellen mit Bagging	460
18.4	Klassifikatoren	461
18.4.1	Nächster-Nachbar-Klassifikator	462
18.4.2	Normalverteilungsklassifikator	464
18.4.3	Entscheidungsbäume	466
18.4.4	Randomisierte Entscheidungsbäume	468
18.4.5	Boosting	470
18.4.6	Support-Vektor-Klassifikator	473
18.5	Kernelbasierte Klassifikation	478
18.5.1	Kernelfunktionen und nichtlineare Klassifikation	478
18.5.2	Mercer-Bedingung und Konstruktion von Kernelfunktionen .	481
18.5.3	Kerneldichteschätzung und Parzen-Klassifikator	484
18.5.4	Support-Vektor-Klassifikator mit Kernelfunktionen	486
18.5.5	Modellierung mit Gauß-Prozessen	490
18.5.6	Gauß-Prozess-Regression	495
18.5.7	Klassifikation mit Gauß-Prozessen	498
18.5.8	Laplace-Approximation	499
18.5.9	Zusammenhang zum SVM Ansatz	500
18.5.10	Mehrklassen-Klassifikation	502
18.5.11	Hyperparameter-Schätzung	503
18.6	Bewertung von Klassifikatoren	504
18.6.1	Wahl der Testbeispiele	504
18.6.2	Erkennungsraten bei mehreren Klassen	505
18.6.3	Bewertung von binären Klassifikatoren	506
18.7	Unüberwachte Verfahren und Gruppierung	508
18.7.1	Gruppierung mit k -Means	509
18.7.2	Schätzung von Mischverteilungen	510
19	Momente, Matching und Merkmale	515
19.1	Momente	515
19.2	Transformation der Momente	520
19.3	Normalisierung der Momente	522
19.4	Kovariante Merkmale, Verfahren und Umgebungen	524

19.5	Invarianten und Momente	527
19.5.1	Erweiterte Hu-Invarianten (Rotationen)	527
19.5.2	Invarianten für Euklidische Transformationen	528
19.5.3	Invarianten für Ähnlichkeitstransformationen	529
19.5.4	Invarianten für affine Transformationen	529
19.6	Gestaltsanalyse mit momentenbasiertem Fitting	530
19.7	Effiziente Berechnung von Konturmerkmalen	535
19.7.1	Elementare Konturmerkmale	535
19.7.2	Berechnung der Momente	539
19.8	Matching	541
19.8.1	Matchingmaße	542
19.8.2	Histogrammbasierte Matchingmaße	544
19.8.3	Signaturen	546
19.8.4	Matching und Registrierung	547
19.9	Lokale Merkmale und Häufigkeitshistogramme	571
19.9.1	Detektor nach Kadir und Brady	572
19.9.2	SIFT-Merkmale	572
19.9.3	HOG-Merkmale	575
19.9.4	Local Binary Patterns (LBP)	576
19.9.5	Statistiken von lokalen Merkmalen	577
19.9.6	Schnelle Berechnungen mit Integralbildern	579
19.10	Transformation von Merkmalen	580
19.10.1	<i>Principal Component Analysis (PCA)</i>	580
19.10.2	Faktorenanalyse	585
20	Visuelle Erkennungsaufgaben	589
20.1	Herausforderungen der visuellen Objekterkennung	589
20.2	Bildklassifikation	591
20.2.1	Kategorisierung mit globalen Merkmalen	591
20.2.2	Kernelbasierte Kombination von mehreren Merkmalen	595
20.2.3	Kombinierte Merkmalsberechnung und Klassifikation	596
20.2.4	Erkennung auf Instanzebene	597
20.3	Objektlokalisation	598
20.3.1	Sliding-Window Klassifikation	599
20.3.2	Viola-und-Jones-Verfahren	600
20.3.3	Fusion von Objekthypothesen	604
20.3.4	HOG Detektor	605
20.3.5	Teilebasierte Verfahren	606
20.4	Semantische Segmentierung	607
20.4.1	Klassifikation lokaler Deskriptoren	607
20.4.2	Klassifikation mit effizienten Pixelmerkmalen	609
20.4.3	Iterative Verwendung von Kontextmerkmalen	611

Teil IV Mathematische Hilfsmittel

21	Ausgleichsrechnung	617
21.1	Quadratische Formen und Eigenwertprobleme	617
21.2	Ausgleichsrechnung	620
21.2.1	Lineare Ausgleichsrechnung	620
21.2.2	Nichtlineare Ausgleichsrechnung	625
21.2.3	Kovariantes Fitting	630
21.2.4	Ausreißer-Probleme (Outlier)	633
21.2.5	Andere Schätzer (LAD)	636
22	Lineare Algebra und Stochastik	639
22.1	Pseudoinverse	639
22.2	Regularisierung nach Tichonov und Arsenin	644
22.3	Singulärwertzerlegung (SVD)	645
22.3.1	Grundlagen	645
22.3.2	Anwendungen	647
22.4	Blockmatrizen	649
22.5	Normalverteilungen	651
	Literatur	653
	Sachverzeichnis	659

Teil I

2D-Bildverarbeitung

In den folgenden Abschnitten werden elementare Grundlagen und Probleme der 2D-Bildverarbeitung angesprochen. Von den Autoren erfolgte dabei die Auswahl der Probleme subjektiv und erhebt nicht den Anspruch irgendeiner Vollständigkeit. Ziel dieser Abschnitte ist es zu erkennen, wo die wesentlichen Unterschiede zwischen „analoger“ und „digitaler“ Bildverarbeitung liegen.

1.1 Abtastung

Bilder werden durch die vielfältigsten Aufnahmetechniken und Sensoren gewonnen, eine Übersicht dazu findet man in J. Beyerer et al. [2]. Um ein mit Hilfe der Sensoren gewonnenes analoges Bild oder ein analoges Signal im Rechner abzuspeichern, müssen wir das analoge Signal in die diskrete Form überführen. Dazu benötigen wir eine **Rasterung** des Definitionsbereiches und eine **Quantisierung** des Wertebereiches, beides zusammen nennen wir **Digitalisierung** des analogen Signals. Wenn wir keine zusätzlichen Bemerkungen machen, dann erfolgt die Rasterung immer äquidistant, zweidimensional bedeutet dies die Abtastung im Quadratgitter. Nach der Abtastung nummerieren wir abstrakt die Abtastpunkte mit ganzen Zahlen durch, so dass z. B. f_n nicht unbedingt der Funktionswert an der Stelle n der analogen Funktion $f(x)$ bedeuten muss, es ist aber der n -te abgetastete, diskrete Funktionswert. Die Quantisierung der Funktionswerte werden wir hier nicht untersuchen, sondern nur die reine Abtastung. Wir gehen also davon aus, dass f_n beliebige reelle oder komplexe Zahlen sein können und exakt die analogen Funktionswerte an den Abtastpunkten darstellen.

Später werden wir noch das sogenannte Abtasttheorem beweisen und zwar für verschiedene Bildmodelle, siehe Kap. 5. Das Abtasttheorem sagt etwas über die Rasterung aus, jedoch nichts über die Quantisierung. Was besagt eigentlich das Abtasttheorem? Wenn man eine analoge Funktion an diskreten Stützstellen abtastet und anschließend nur die Abtastwerte zur Verfügung hat, dann kann man allein aus den Abtastwerten die analog-

ge Funktion nicht mehr rekonstruieren. Wir wissen nämlich nicht, wie sich die Funktion zwischen den Stützstellen verhält. Haben wir jedoch ein geeignetes A-priori-Wissen über die analoge Funktion, dann können wir die analoge Funktion rekonstruieren. Wie könnte dieses A-priori-Wissen aussehen? Da gäbe es sehr viele Möglichkeiten. Wir könnten z. B. wissen, dass die Funktion Ableitungen bis zur n -ten Ordnung besitzt und alle Ableitungen höherer Ordnung Null sind. Es wird sich aber in der Praxis nur das A-priori-Wissen durchsetzen, dass verständlich und praktisch handhabbar ist. Das ist oft das Wissen, aus welchen Frequenzen sich das Signal zusammensetzt und aus welchen nicht. Zunächst formulieren wir das bekannte Abtasttheorem (nach Shannon, Whittaker und Kotelnikov) verbal, um ein „Gefühl“ für diese Aussage zu bekommen.

Satz 1.1 (Abtasttheorem) *Eine analoge Funktion sei bandbegrenzt, d. h. die größte im Signal auftretende Frequenz v_g sei endlich. Dies ist das A-priori-Wissen, was wir haben müssen. Wenn wir dann mit der äquidistanten Abtastschrittweite Δt*

$$\Delta t \leq \frac{1}{2v_g} \quad (1.1)$$

das Signal abtasten, dann können wir anschließend allein mit den diskreten Abtastwerten und dem A-priori-Wissen über die Bandbegrenzung das analoge Signal wieder vollständig rekonstruieren.

Das heißt nun für die Abtastfrequenz $v_{ab} \geq 2v_g$. Ist sie größer als $2v_g$, dann spricht man von **Überabtastung**. Das führt zu keiner Informationserhöhung, wird aber aus Sicherheitsgründen oft gemacht. Die halbe Abtastfrequenz $v_{Nyquist} = \frac{1}{2}v_{ab}$ wird auch **Nyquist-Frequenz** genannt, damit gilt $v_{Nyquist} \geq v_g$.

Eine andere Formulierung bezieht sich auf die Wellenlänge oder Periode der Schwingung mit der höchsten Frequenz: Bezuglich der Periode der Schwingung mit der Grenzfrequenz muss man mindestens zwei Abtastpunkte wählen. Zum Verständnis geben wir jetzt zwei einfache Beispiele an:

Beispiel 1 Für ein gewöhnliches Fernsprechsignal zur Übertragung normal gesprochener Sprache reicht eine Übertragung des Frequenzbereiches von 300 bis 3400 Hertz aus. Es liegt also eine Grenzfrequenz von 3400 Hertz vor. Wie viele Signale/Sekunde müssen wir nun übertragen? Wir müssen das Signal im Abstand von

$$\Delta t \leq \frac{1}{2 \cdot 3400} = 147 \cdot 10^{-6} \text{ s} \quad (147 \mu\text{s}) \quad (1.2)$$

abtasten. Man wählt aus Sicherheitsgründen 125 μs , damit erhalten wir als eigentliche Abtastfrequenz:

$$v_{ab} = 2 \cdot v_{Nyquist} = \frac{1}{125 \mu\text{s}} = 8000 \text{ Hz.} \quad (1.3)$$

Beispiel 2 Betrachten wir einmal ein BAS-Signal (Bild-, Austast- und Synchron-Signal) nach der „alten“ CCIR-Norm, charakterisiert durch

- *Interlace Mode 2 : 1,*
- Teilbildfrequenz 50 Hz,
- Vollbildfrequenz 25 Hz,
- Zeit zum Aufbau eines Vollbildes 40 ms,
- Anzahl Zeilen 625,
- Sichtbare Zeilen 576,
- Verhältnis horizontale zu vertikale Auflösung 4 : 3.

Es handelt sich somit nicht um eine „echte“ 2D-Funktion. Bezuglich der y -Richtung liegt eine Rasterung in den Zeilen schon vor – wir müssen also im Prinzip nur noch horizontal diskretisieren. Wir gehen einmal ideal von den 625 Zeilen aus. Wenn wir abtasten mit dem Ziel quadratische Pixel zu erhalten, so müssen wir $625 \cdot \frac{4}{3} = 835$ Bildpunkte pro Zeile abtasten. Damit müssen wir pro Sekunde $25 \cdot 625 \cdot 835 \approx 13 \cdot 10^6$ Signale abtasten oder übertragen. Die Abtastfrequenz muss also 13 MHz betragen. Sie beträgt aber beim Fernsehen nach der CCIR-Norm nur 10 MHz, daher ist die vertikale Auflösung besser als die horizontale Auflösung. Durch den Interlace-Modus haben wir z. B. unschöne Effekte bei Videosequenzen. Wenn wir ein Standbild als Vollbild auswerten wollen und in der Szene hat sich einiges bewegt, dann kann es beim Zusammensetzen „Versätze“ geben. Diese werden oft durch Interpolation ausgeglichen, dies nennt man **Deinterlacing**. Aufnahmen, die direkt auf Vollbildern beruhen, nennt man **Progressive Scan**. Die Regel ist dann die direkte Abtastung über die geometrisch angeordneten Sensoren, in der Regel CCD- oder CMOS-Sensoren. Wie ordnet man nun diese Sensoren auf einem Chip an? Standard ist das Rechteck- bzw. Quadratgitter. Was gibt es eigentlich in der Ebene noch für Möglichkeiten Sensoren so anzurichten, dass das Muster total „regelmäßig“ ist? Dazu muss man erst den Begriff „regelmäßig“ definieren. Regelmäßige oder homogene Gitter sind die, bei denen jeder Gitterpunkt die gleiche Anzahl von unmittelbaren Nachbarn hat und die entstehende, elementare „Gitterzelle“ immer die gleiche Anzahl von Kanten hat. Solche regelmäßigen Strukturen in der Ebene gibt es nur drei, das **Rechteckgitter**, das **Dreiecksgitter** und das **Hexagonalgitter**. Durchgesetzt hat sich in der Bildverarbeitung aber nur das Rechteckgitter, siehe auch Abschn. 1.2.

Wir kehren noch einmal zurück zum Abtasttheorem. Wenn wir weniger Abtastpunkte haben als laut Theorem nötig ist, dann spricht man von **Unterabtastung**. Wir können dann das Signal in der vollen Auflösung nicht mehr rekonstruieren. Bei periodischen Strukturen tritt dann oft ein Nebeneffekt auf, man glaubt Strukturen zu sehen, die es gar nicht gibt. Diesen Effekt nennt man **Aliasing**. In Abb. 5.4 sind „echte“ Bilder mit Aliasing-Effekten zu sehen. In Abb. 1.1 sind zwei simulierte Abtastungen des Siemens-Sternes zu sehen, es entstehen Aliasing-Effekte, die an die Computer-Grafik erinnern. Schon das simple Problem der Bildverkleinerung berührt erneut das Abtasttheorem. Verkleinern wir ein Bild, indem wir einfach jeden zweiten Bildpunkt weglassen, dann tasten wir mit einer höhe-

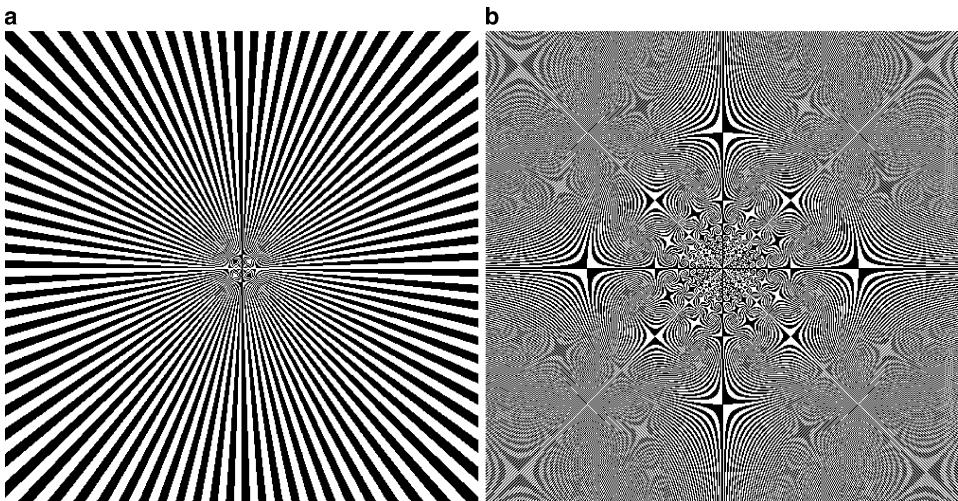


Abb. 1.1 Abtastung des Siemens-Sterns: **a** Aliasing mit größerer Abtastschrittweite, **b** Aliasing mit kleinerer Abtastschrittweite

ren Schrittweite ab. Dies kann funktionieren, muss aber nicht, je nach der Grenzfrequenz des Bildes. Eine Unterabtastung bewirkt Störungen des Bildes, indem zum Beispiel kleine Strukturen verschwinden. Der markanteste Hinweis auf eine Unterabtastung ist der „Gartenauneffekt“, bei dem in periodischen Strukturen langwellige „Schwierigkeiten“ sichtbar werden. Durch die Unterabtastung gibt es Bereiche, in denen nur die Latten oder nur die Zwischenräume im Bild übrigbleiben.

1.2 Diskrete Natur von digitalen Bildern

Digitale Bilder entstehen aus analogen durch den Prozess der Digitalisierung, d. h. Rasterung der x, y -Koordinaten und Quantisierung der Funktionswerte, siehe Abschn. 1.1. In der Bildverarbeitung bedient man sich deshalb zweier grundlegender Strategien:

- Man modelliert das analoge Bild mit den Mitteln der nichtdiskreten Mathematik. Dabei benutzt man die Geometrie und die Analysis. Erst zum Schluß diskretisiert man entsprechend der gegebenen Auflösung des Bildes. Man kennt dann in der Regel nicht die Diskretisierungsfehler.
- Man benutzt direkt mathematische Modelle der diskreten Mathematik. Falls diese Modelle praxistauglich sind, entfallen die anschließenden Diskretisierungsfehler.

In der Bildverarbeitung benutzt man je nach Aufgabenstellung das diskrete oder analoge Modell, oft auch Mischformen aus beiden. Betrachten wir z. B. Rotationen. Diese lassen

sich im diskreten Gitter nicht direkt beschreiben. Man benutzt folglich analoge Rotationen und diskretisiert anschließend. Möchte man z. B. im Bild mit einem Algorithmus die Kontur eines Objektes bestimmen, so muss man den diskreten Konturfolgealgorithmus direkt benutzen. Einen sinnvollen Algorithmus zur Bestimmung einer analogen Kontur gibt es nicht. Je nach Aufgabenstellung unterscheiden wir folglich Bildmodelle, siehe Abschn. 1.3.

1.2.1 Gittertypen

Wenn man die Pixel in ein Koordinatensystem einträgt, entsteht ein Gitter. Das bekannteste ist das Quadratgitter, eine Unterlage von $\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}$. Noch allgemeiner stellen die Pixel die Knoten und die Verbindungen zu den Nachbarn die Kanten in einem Nachbarschaftsgraphen dar. Könnte man nun auch andere Gitter als das Quadratgitter verwenden, die vielleicht sogar Vorteile bieten? Welche Anforderungen stellt man nun an ein Gitter aus praktischen Gründen der Bildverarbeitung?

- Jedes Pixel sollte bezüglich einer Nachbarschaft stets die gleiche Anzahl Nachbarn haben. Dies nennt man den Nachbarschaftsgrad v .
- Jede „Elementarzelle“, genannt Masche, sollte stets vom gleichen Typ sein, d. h. die gleiche Maschenlänge λ besitzen.

Nachbarschaftsgraphen mit $v = \text{konstant}$ und $\lambda = \text{konstant}$ nennt man homogen oder regulär. Man kann nun zeigen, dass es in der Ebene mit unendlich vielen Pixeln nur drei solcher homogener Gitter gibt:

- Quadratgitter ($v = 4, \lambda = 4$), siehe Abb. 1.2a,
- Hexagonalgitter ($v = 3, \lambda = 6$), siehe Abb. 1.2b,
- Dreiecksgitter ($v = 6, \lambda = 3$), siehe Abb. 1.2c.

Dabei spielt es keine Rolle, ob beim Quadratgitter die Elementarzellen tatsächlich Quadrate sind, es können auch beliebige Vierecke sein, Analoges gilt bei den anderen Gittern. Diese drei Gitter mit einer endlichen Anzahl von Pixeln sind nur regulär auf einem Torus (Schwimmreifen), nicht in der Ebene. Dasselbe liegt vor, wenn man sich ein endliches Pixelbild periodisch fortgesetzt denkt. Gibt es nun gar keine endliche, reguläre Nachbarschaftsgraphen in der Ebene? Die gibt es tatsächlich, aber auf Grund der geringen Anzahl von Pixeln, haben sie in der Bildverarbeitung keinerlei Bedeutung. Ein simples Beispiel sind die geschlossenen Ringstrukturen mit n Pixeln, dann ist $v = 2, \lambda = n$. Es gibt aber noch andere. Dazu müssen wir erst einen Zusammenhang zwischen Polyedern im Raum und Nachbarschaftsgraphen in der Ebene herstellen. Man kann zeigen:

Ein endliches, nichtentartetes 3D-Polyeder kann man stets als endlichen, planaren Graphen darstellen und umgekehrt. Endlich bedeutet beim Polyeder eine endliche Anzahl von Ecken und beim Graphen eine endliche Anzahl von Knoten.

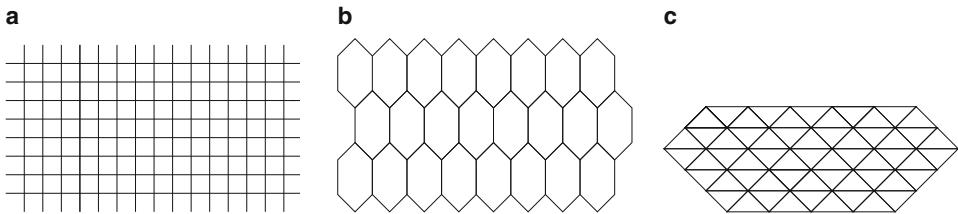


Abb. 1.2 a Quadratgitter, b Hexagonalgitter, c Dreiecksgitter

Dann werden die Ecken des Polyeders die Knoten des Graphen, die Kanten des Polyeders werden die Kanten des Graphen und die Seitenflächen des Polyeders werden die Maschen des Graphen. Folglich kann man (nichtentartete) endliche, planare Graphen mit (nichtentarteten) Polyedern identifizieren. Umgedreht repräsentieren damit solche Polyeder planare Graphen. Nun gibt es die berühmten fünf Platonischen Körper (dies sind 3D-Polyeder), die folglich fünf endliche, reguläre Nachbarschaftsgraphen bilden. Es bezeichnen α_0 die Anzahl der Ecken (Pixel, Knoten), α_1 die Anzahl der Kanten, α_2 die Anzahl der Seitenflächen (Maschen). Dann lauten die fünf Platonischen Körper:

- Hexaeder (Würfel) mit $v = 3, \lambda = 4, \alpha_0 = 8, \alpha_1 = 12, \alpha_2 = 6$,
- Tetraeder mit $v = 3, \lambda = 3, \alpha_0 = 4, \alpha_1 = 6, \alpha_2 = 4$,
- Oktaeder mit $v = 4, \lambda = 3, \alpha_0 = 6, \alpha_1 = 12, \alpha_2 = 8$,
- Dodekaeder mit $v = 3, \lambda = 5, \alpha_0 = 20, \alpha_1 = 30, \alpha_2 = 12$,
- Ikosaeder mit $v = 5, \lambda = 3, \alpha_0 = 12, \alpha_1 = 30, \alpha_2 = 20$.

Auf Grund der geringen Anzahl von Pixeln sind sie für die Bildverarbeitung bedeutungslos.

In regulären Gittern kann man eine **Digitale Geometrie** im Gegensatz zur Euklidischen Geometrie aufbauen, siehe Klette und Rosenfeld [35]. Eine ähnliche Fragestellung bezüglich regulärer Gitter gibt es in der Geometrie und nennt sich *Parkettierung* der Ebene: Kann man mit einem vorgegebenen Polygon die Ebene vollständig parkettieren? Mit beliebigen Dreiecken und Vierecken geht dies natürlich. Aber schon mit einem regelmäßigen Fünfeck ist dies nicht mehr möglich. Mit Hexagonen muss dies möglich sein, da das Hexagonalgitter eines der drei regulären Gitter ist.

Da sich in der praktischen Bildverarbeitung nur das Quadratgitter durchgesetzt hat, werden wir uns im Folgenden immer auf das metrische Gitter $\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}$ bzw. in 3D auf \mathbb{Z}^3 beziehen. Dabei nehmen wir immer an, die Abstände zwischen den Pixelkoordinaten in x , y und z -Richtung sind gleich. In der Praxis stimmt das auch fast immer, es gibt aber Ausnahmen. So z. B. bei 3D-Bildern in der

- Magnetresonanztomographie (MRT), Computertomographie (CT)
- Konfokale Laser Scanning Mikroskopie (CLSM).

Hier ist oft der Abstand in z -Richtung anders (oft größer) als der in den beiden x - und y -Richtungen. Dies muss insbesondere bei Visualisierungen beachtet werden.

Wichtig in einer gegebenen Menge von Pixeln im Sinne von Nachbarschaftsgraphen ist der Begriff der Nachbarschaftsrelation, man spricht auch von Nachbarschaftssystemen oder Nachbarschaftsstrukturen. In diesem Sinne hat im Quadratgitter jedes Pixel vier Nachbarn, im Dreiecksgitter sechs Nachbarn und im Hexagonalgitter drei Nachbarn. Führt man im Quadratgitter eine beliebiges Nachbarschaftssystem ein, dann meint man eigentlich nicht das Quadratgitter, sondern eine Menge von Pixeln, die im Koordinaten- system quadratisch angeordnet sind und deren Nachbarn durch das Nachbarschaftssystem bestimmt sind. Diese bilden dann i. Allg. keine regulären Nachbarschaftsgraphen.

Es sei die Menge aller Pixel eine Punktmenge S und R eine Relation auf S , d. h. $R \subseteq S \times S$. Eine **Nachbarschaftsrelation** ist dann eine

- *irreflexive* und
- *symmetrische*

Relation. Prätgt man dem Nachbarschaftsgraphen noch eine Orientierung auf, dann redet man von orientierten Nachbarschaftsgraphen und kann den Begriff der Maschen einführen. Oft benötigt man den Begriff *clique*. Eine Teilmenge $C_S \subseteq S$ heißt *clique*, wenn jedes Paar verschiedener Elemente aus C_S Nachbarn sind. C bezeichne die Menge aller cliquen C_S . Damit sind auch die Teilmengen cliquen, die nur ein einziges Element von S enthalten.

Im metrischen Gitter \mathbb{Z}^2 sind die gebräuchlichsten Nachbarschaften die Vierernachbarschaft (City-Block-Metrik)

$$R_4(x, y) = \{(u, w) : |u - x| + |w - y| = 1\} \quad (1.4)$$

und die Achternachbarschaft (Chessboard-Metrik)

$$R_8(x, y) = \{(u, w) : \max(|u - x|, |w - y|) = 1\}. \quad (1.5)$$

Manchmal benutzt man auch sogenannte Gitterzellenmodelle. Man stellt sich dabei in \mathbb{Z}^2 ein Pixel als Quadrat, in \mathbb{Z}^3 ein Pixel als Würfel vor. Dann ist die Vierernachbarschaft in \mathbb{Z}^2 durch die 4 Kanten eines Quadrates erklärbar, die Achternachbarschaft durch die 4 Kanten und die 4 Ecken des Quadrates. Mit dieser Interpretation können wir die Vierer- und Achternachbarschaft sofort auf \mathbb{Z}^3 übertragen. Die Vierernachbarschaft wird zur 6-er Nachbarschaft, da ein Würfel 6 Flächen besitzt. Die Achternachbarschaft wird zur 26-er Nachbarschaft, da ein Würfel 6 Flächen, 12 Kanten und 8 Ecken besitzt.

Redet man von einer Umgebung oder Nachbarschaft U eines Pixels (x, y) , dann gehören die Nachbarn und der Aufpunkt (x, y) zur Umgebung U . Die nächst wichtige Relation ist nun die Verbundenheitsrelation, die sich auf eine Nachbarschaftsrelation bezieht. Zwei Pixel p und q stehen in Relation (sind verbunden, es gibt einen Weg von p nach q), wenn es eine benachbarte Folge von Pixeln gibt, so dass p der Startpunkt ist und q der Endpunkt. Wenn wir noch definieren, dass jedes Pixel mit sich selbst verbunden ist, dann ist die Verbundenheitsrelation eine Äquivalenzrelation. Die Verbundenheitsrelation teilt folglich alle schwarzen (oder weißen) Pixel in Klassen ein, die wir Objekte nennen wollen.

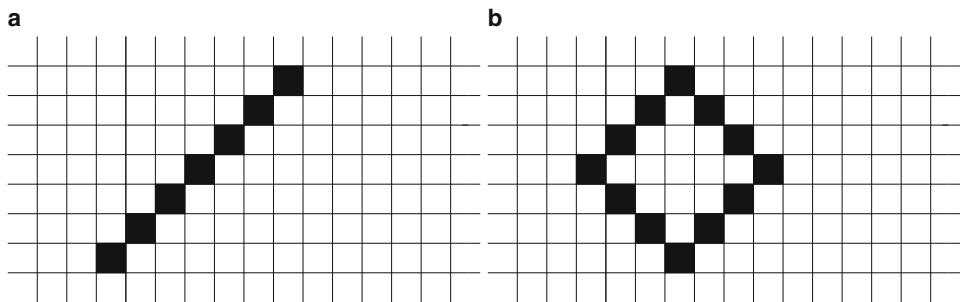


Abb. 1.3 Interpretation eines Binärbildes: **a** Eine Linie oder viele Objekte? **b** Relation zwischen Innen und Außen eines Objektes bezüglich der Vierer- oder Achternachbarschaft?

1.2.2 Kuriositäten im Quadratgitter

In \mathbb{Z}^2 gibt es bezüglich der Nachbarschaften und Verbundenheitsrelation einige „Kuriositäten“ im Gegensatz zur Euklidischen Ebene. Dazu betrachten wir einige Beispiele, dabei sind schwarze Quadrate Objektpunkte und weiße sind Hintergrundpunkte. Wenn wir in Abb. 1.3a die Vierernachbarschaft benutzen, dann liegen sieben Objekte mit je einem Pixel vor. Benutzen wir dagegen die Achternachbarschaft, dann liegt nur ein Objekt vor, das als flächiges Objekt oder auch als Linie gedeutet werden kann. Echt kurios wird es in Abb. 1.3b. Wenn wir die Achternachbarschaft zugrunde legen, dann haben wir eine geschlossene Linie, also ein Objekt mit einem Innern. Andererseits hängt das Innere über die Achternachbarschaft mit dem Äußeren zusammen, was der Euklidischen Geometrie völlig widerspricht. Mathematisch ist das kein Widerspruch, aber sehr unbefriedigend. Daher wählt man oft zwei Nachbarschaften, für den Objektzusammenhang die Vierer oder Achternachbarschaft, und dann aber umgedreht für den Hintergrund die Achter oder Vierernachbarschaft.

Weiterhin kann es sein, dass „übliche“ Objekte der Euklidischen Ebene in \mathbb{Z}^2 überhaupt nicht existieren. So gibt es kein einziges gleichseitiges Dreieck, dessen Eckpunkte Gitterpunkte aus \mathbb{Z}^2 sind. Weiterhin müssen grundlegende Definitionen in der Euklidischen Ebene in \mathbb{Z}^2 neu durchdacht werden. So ist ein sehr wichtiger Begriff die Konvexität von Mengen, d. h. in \mathbb{Z}^2 die Konvexität von Binärobjekten.

1.2.3 Digitale Konvexität

Ausführliche Darstellungen zur Digitalen Konvexität findet man in Klette [35]. Die Konvexität ist in \mathbb{R}^2 folgendermaßen definiert:

Definition 1.2 (Konvexe Punktmengen) *Eine Punktmenge $M \subset \mathbb{R}^2$ heißt konvex, wenn für zwei beliebige Punkte $x_1, x_2 \in M$ auch alle Punkte der Verbindungsstrecke in M liegen.*

Zwischen zwei Gitterpunkten brauchen aber gar keine weiteren Gitterpunkte liegen. Daher müssen wir im Gitter \mathbb{Z}^2 eine andere, möglichst mit der Euklidischen Ebene einigermaßen verträgliche Definition finden. Dazu definieren wir zunächst den Begriff der Kollinearität von drei Gitterpunkten $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ und $P_3 = (x_3, y_3)$. Diese drei Gitterpunkte heißen kollinear, wenn die Fläche des von ihnen aufgespannten Dreiecks gleich Null ist, d. h. wenn

$$2 \cdot F_{\Delta} = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) = 0 \quad (1.6)$$

gilt.

Definition 1.3 (Konvexe Linearkombination) *Gegeben seien m beliebige Punkte \mathbf{x}_i , $i = 1, \dots, m$ in der Euklidischen Ebene. Dann nennt man*

$$\mathbf{x} = \sum_{i=1}^m \lambda_i \mathbf{x}_i, \quad \lambda_i \geq 0, \quad \sum_{i=1}^m \lambda_i = 1 \quad (1.7)$$

eine konvexe Linearkombination der m Punkte \mathbf{x}_i .

Für drei Punkte $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ werden damit alle Punkte desjenigen Dreiecks dargestellt, welches die drei Punkte aufspannen. Wenn die drei Eckpunkte des Dreiecks nun selbst Gitterpunkte sind, dann können natürlich auch alle Gitterpunkte innerhalb des Dreiecks als konvexe Linearkombination der Eckpunkte dargestellt werden.

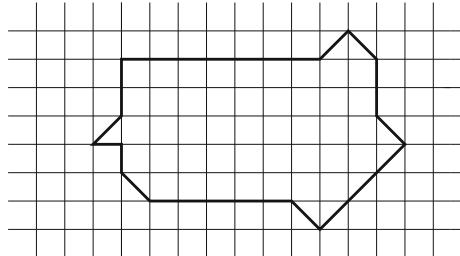
Definition 1.4 (Triangulare Gitterpunkte) *Ein beliebiger Gitterpunkt P heißt bezüglich dreier nichtkollinearer Gitterpunkte triangular, wenn er als konvexe Linearkombination der drei Gitterpunkte darstellbar ist, d. h. wenn er selbst in dem Dreieck liegt, das durch die drei Gitterpunkte aufgespannt wird.*

Damit können wir nun die „digitale Konvexität“ einführen:

Definition 1.5 (Konvexe Menge von Gitterpunkten) *Eine Menge M von Gitterpunkten heißt konvex, wenn für beliebige drei Punkte $P_1, P_2, P_3 \in M$ alle bezüglich dieser drei Punkte triangularen Punkte auch zur Menge M gehören.*

In der Abb. 1.4 ist ein digital konvexas Objekt zu sehen, welches in der Euklidischen Ebene nicht konvex ist (wenn man das Objekt durch das Konturpolygon beschreibt). Da wir nur nichtkollineare Punkte zugelassen haben, erhalten wir wieder kuriose Konvexitätsbegriffe bei speziellen Punktmengen: Besteht die Menge M in \mathbb{Z}^2 nur aus Punkten, die auf

Abb. 1.4 Kontur eines digital-konvexen Objektes, welches in \mathbb{R}^2 nicht konvex ist



einer Geraden liegen, dann ist diese Menge M immer konvex, egal ob diese Punkte zusammenhängend sind oder nicht. Wenn die Menge M aus zwei Punkten besteht, ist diese folglich immer konvex. Dies ist kein mathematischer Widerspruch zur Konvexität in \mathbb{R}^2 , aber eben „seltsam“. Man könnte nun die Restriktion an die Kollinearität fallen lassen und die triangularen Punkte zu beliebigen, auch entarteten Dreiecken betrachten. Es bleiben dann immer noch kuriose Fälle übrig: Die Menge M bestehe aus zwei Gitterpunkten. Liegt auf der Verbindungsstrecke dieser beiden Gitterpunkte kein weiterer Gitterpunkt, ist die Menge M digital konvex. Der Begriff der digitalen Konvexität zeigt auch, dass es natürlich viele gemeinsame Probleme mit der Computergrafik und der Berechenbaren Geometry (*Computational Geometry*) gibt. Da insbesondere eine digitale Kontur ein Polygon darstellt, kann man viele Erkenntnisse von Polygonen aus diesen beiden Gebieten nutzen. Im Zusammenhang mit der Konvexität spielt der Begriff der **Konvexen Hülle (KH)** P_H einer beliebigen Punktmenge P . Dabei ist P_H die „kleinsten“ konvexe Menge, die P enthält. Dies gilt sowohl in der Euklidischen Ebene als auch im Gitter. Zur Berechnung der Konvexen Hülle gibt es sowohl in der Bildverarbeitung als auch in der *Computational Geometry* Standardalgorithmen. In der Bildverarbeitung ist die konvexe Hülle oft eine zu grobe Abschätzung der Punktmenge P , man möchte dies oft etwas „weicher“ haben. Daher führen wir jetzt den Begriff der **Limitierten Konvexen Hülle (LKH)** ein, die gewissermaßen die **Silhouette** einer diskreten Punktmenge M darstellen soll, siehe dazu auch [74]. Die Silhouettenberechnung ist ein schönes Beispiel für die Gemeinsamkeiten aus der Bildverarbeitung (Segmentierung einer Punktmenge M) und der Berechenbaren Geometrie (Triangulierung einer Punktmenge M).

1.2.4 Silhouetten

Gegeben sei eine Punktmenge P , entweder in der Euklidischen Ebene oder im Gitter. Im Gitter ist diese im Regelfall endlich. Weiter sei eine **Strukturkonstante** $limit$ gegeben. Zwei Punkte aus P seien benachbart, wenn bezüglich einer Metrik ihr Abstand kleiner gleich (\leq) $limit$ ist. Wir betrachten nun alle möglichen Dreiecke von drei beliebigen Punkten aus P , deren drei Seitenlängen kleiner gleich (\leq) $limit$ sind. In \mathbb{R}^2 müssen wir auch alle entarteten

Dreiecke zulassen, wenn wir die Kompatibilität zur Konvexität beibehalten wollen. In \mathbb{Z}^2 lassen wir dagegen nur dann die entarteten Dreiecke zu, wenn dies auch bei der Definition der digitalen Konvexität getan wurde. Für jedes Dreieck betrachten wir

- In der Euklidischen Ebene: Alle Punkte P_D des Dreiecks, d. h. alle durch eine konvexe Linearkombination der drei Eckpunkte des Dreiecks erzeugten Punkte.
- Im Gitter: Alle triangularen Punkte P_T , die durch die drei Gitterpunkte des Dreiecks erzeugt werden.

Nun können wir den Begriff der Limitierten Konvexen Hülle (*LKH*) einführen.

- In der Euklidischen Ebene: Die *LKH* P'_H ist die Vereinigung aller Punktmengen P_D mit P .
- Im Gitter: Die *LKH* P'_H ist die Vereinigung aller triangularen Punkte P_T mit P .

Die *LKH* hängt natürlich direkt vom Strukturparameter *limit* ab. Ist *limit* = 0, dann ist die $LKH(P) = P$, also die Menge P selbst. Ist *limit* → ∞, dann gilt $LKH(P) \rightarrow KH(P)$, d. h. mit größer werdendem *limit* nähern wir uns der Konvexen Hülle $KH(P)$. Die *LKH*(P) vergrößert die Menge P , folglich gilt:

$$P \subseteq LKH(P) \subseteq KH(P). \quad (1.8)$$

Während die Bildung der Konvexen Hülle ein Projektionsoperator darstellt, d. h. es gilt $KH(KH(P)) = KH(P)$, ist dies bei der *LKH* nicht mehr der Fall. Die *LKH*(*LKH*(P)) wird „größer“ als die *LKH*(P), ähnlich der morphologischen Operation Dilation, siehe Abschn. 8.2.4. Die *LKH* kann benutzt werden, um **Silhouetten** von Punktmengen in \mathbb{Z}^2 zu berechnen. Unter der Silhouette verstehen wir alle Randkurven (Außen- und Innenkonturen) der *LKH* bezüglich der Vierernachbarschaft in \mathbb{Z}^2 als Zusammenhangsrelation in der *LKH*.

Wenn wir die *LKH* in einem Binärbild darstellen, dann stellen wir alle Punkte aus der *LKH* schwarz dar und nichts weiter. Die Zusammenhangseigenschaft bez. *limit* haben wir dann vergessen und betrachten das Binärbild nur noch bezüglich der Vierernachbarschaft. Mit dem Konturfolgeverfahren bezüglich der Vierernachbarschaft als Objektzusammenhang bestimmen wir alle Außen- und Innenkonturen und nennen alle diese Konturen die Silhouette der Punktmenge M . Nun gibt es aber ähnlich wie bei der digitalen Konvexität „kuriose“ Fälle. Es kann durchaus vorkommen, dass wir zwei Außenkonturen und damit zwei Objekte im Binärbild bestimmen, obwohl beide Objekte bezüglich *limit* in der *LKH* logisch zusammenhängend sind. Dann beschreibt die Silhouette die *LKH* eigentlich nicht korrekt. Dies ist ein Kompromiss und ist für praktische Aufgaben nicht bedeutsam, da es doch „seltene“ oder „kuriose“ Fälle sind.

In Abb. 1.5a ist das Bild einer Nervenzelle zu sehen. Entsprechend einem A-priori-Wissen über *limit* soll die Silhouette bestimmt werden. Dazu werden mit klassischen

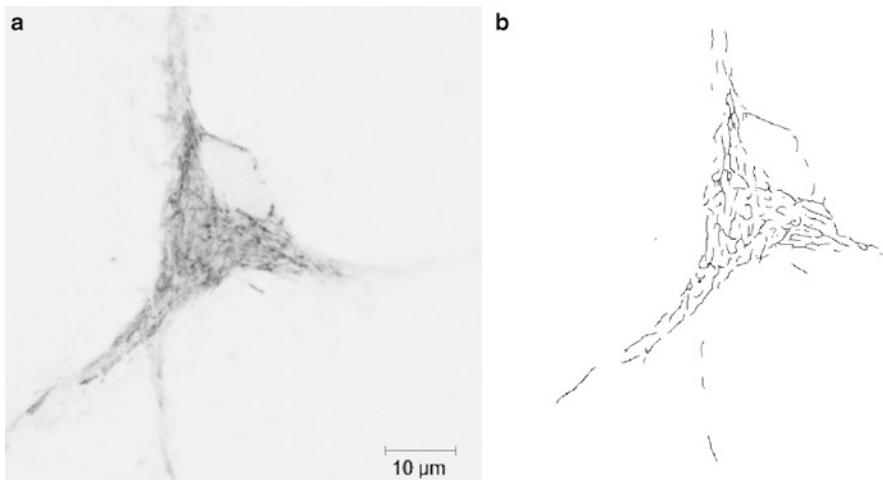


Abb. 1.5 a Eine Schicht der Aufnahme einer Nervenzelle mit einem konfokalen Laserscanning-Mikroskop, b Detektionierte, typische Punkte aus a

Vorverarbeitungsschritten der Bildverarbeitung typische Punkte der Zelle ermittelt, siehe Abb. 1.5b. Alle schwarzen Pixel in Abb. 1.5b bestimmen nun die Punktmenge P . Entsprechend dem Parameter $limit$ sind in Abb. 1.6a, b die LKH für zwei Werte von $limit$ zu sehen. In Abb. 1.6c, d sind LKH für größere Werte von $limit$ zu sehen. Aus allen vier Abb. 1.6a–d ist zu sehen, wie sich die LKH mit wachsendem $limit$ entsprechend (1.8) verhält. Außer der Silhouettenbestimmung von Punktmengen kann die LKH auch zur Konturglättung erfolgreich eingesetzt werden. Zu diesem Zwecke bestimmen alle Punkte der Kontur eines Objektes die Punktmenge P . Nun wird wieder mit einem $limit$ die LKH von P bestimmt. Von dieser $LKH(P)$ wird nun mit dem Konturfolgeverfahren (siehe Abschn. 10.2) wieder die Kontur bestimmt. Diese neue Kontur ist nun entsprechend $limit$ geglättet, d. h. konkave Stellen werden „abgeschwächt“ bzw. Lücken womöglich ganz geschlossen, siehe Abb. 1.7.

Die dunklen Linien sind die originalen Konturen und bilden jeweils die Punktmenge P . Die grauen Flächen bilden die LKH der Konturen. Von diesen grauen Flächen muss man nun die Konturen bestimmen und erhält die neuen, geglätteten Konturen.

Für praktische Anwendungen benötigen wir einen schnellen Algorithmus zur Berechnung der LKH. Als Approximation der LKH benutzen wir eine Modifikation der Delaunay-Triangulation der endlichen, diskreten Punktmenge M . Wenn $n = \text{card}(M)$ ist, dann gibt es $n \log n$ -Algorithmen zur Delaunay-Triangulation, die wir aber noch bezüglich $limit$ modifizieren müssen, siehe dazu auch Abschn. 10.4.

Bemerkung Die Idee, Dreiecke zur Konstruktion der LKH zu benutzen, liegt auf der Hand, wenn man die innere und äußere Struktur einer Punktmenge beschreiben möchte. Eine völlig andere Frage ist es, wie man im diskreten Gitter \mathbb{Z}^2 diese Dreiecke darstellt, welche Diskretisierungsstrategie man wählt. So gibt es z. B. die Gaußsche Digitalisierung,

Abb. 1.6 Von der Punktmenge aus 1.5b wurde die LKH mit **a** $limit = 10$, **b** $limit = 30$, **c** $limit = 60$, **d** $limit = 150$ berechnet

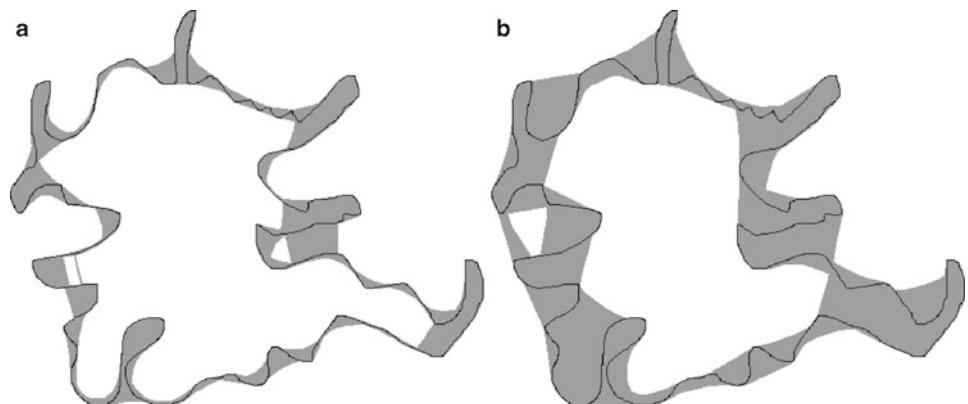
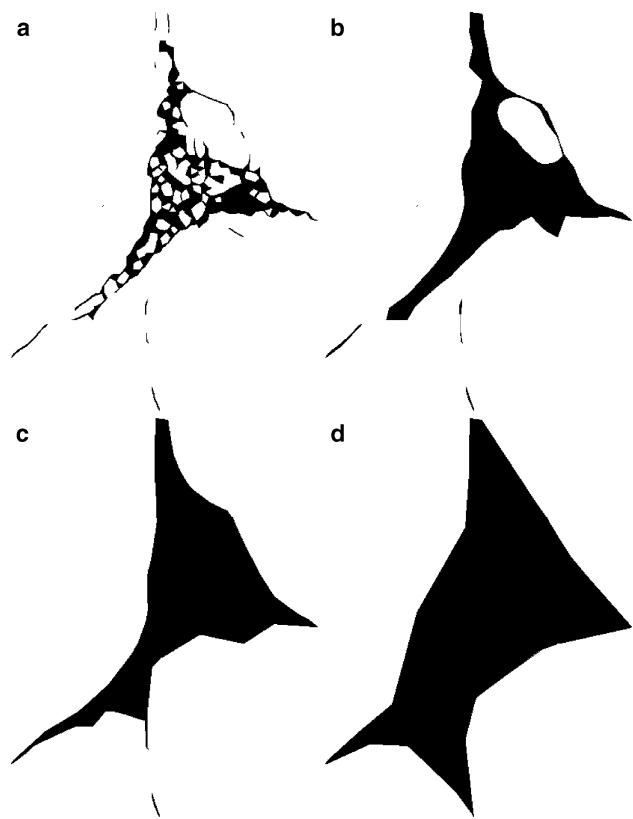


Abb. 1.7 Eine diskrete Kontur wird als diskrete Punktmenge aufgefasst und von dieser Punktmenge wird die LKH mit **a** $limit = 30$, **b** $limit = 50$ berechnet

die Gitterlinienschnittpunkt-Digitalisierung usw., siehe dazu Klette [35]. Wir hatten als Digitalisierung die triangularen Punkte gewählt, damit die Kompatibilität zur Digitalen Konvexität bestehen bleibt. Wählt man eine andere Diskretisierungsstrategie, so erhält man im diskreten Gitter \mathbb{Z}^2 auch eine andere *LKH* oder andere Silhouetten. Allerdings gilt dann die Beziehung (1.8) bezüglich der digitalen konvexen Hülle nicht mehr. Dazu ein

Beispiel Es soll von einer Punktmenge nur die Außenkontur bestimmt werden, die innere Struktur interessiert überhaupt nicht. Die Silhouette soll also lediglich die Außenkontur sein. Dazu wählen wir alle Paare von Punkten aus P . Wir verbinden die beiden Punkte derjenigen Paare durch ein Geradensegment, deren Punktabstand $\leq \text{limit}$ sind. Wenn wir dies nun im Gitter \mathbb{Z}^2 tun, wählen wir als Diskretisierung die „Gitterlinienschnittpunkt-Digitalisierung“. Dazu können wir den bekannten Bresenham-Algorithmus benutzen. Wir erhalten damit in \mathbb{Z}^2 eine Menge P' . Von dieser Menge wollen wir nun die Außenkontur bestimmen. Da wir aber auch die „äußersten“ Punkte von P nur durch Geradensegmente verbunden haben, liefert der klassische Konturfolgealgorithmus bezüglich der Vierernachbarschaft nicht die gewünschte Außenkontur, er „bricht durch“. Wir müssen die Geradensegmente „flächig“ verdicken. Dies geschieht durch eine Dilation mit einem 3×3 Strukturelement. Nun erhalten wir eine Menge P'' . Von dieser Menge P'' bestimmen wir mit dem Konturfolgealgorithmus die Außenkontur und damit die Silhouette der Punktmenge P . Die Innenkonturen beschreiben allerdings nicht die Struktur der Punktmenge und sind daher ohne A-priori-Wissen nicht nutzbar. Für einen „großen“ Wert von limit erhalten wir nicht mehr die digitale konvexe Hülle. Da wir Paare von Punkten gewählt haben, hat der Algorithmus eine quadratische Komplexität.

1.2.5 Perkolation

In den Gittern gibt es auch sehr interessante Beziehungen zur Physik. So gibt es die sogenannte Perkolationstheorie (Durchsickerungstheorie von Körpern). Die Perkolation kann man schön in den Gittern simulieren. Dazu betrachten wir ein genügend großes Bild mit einer Gitterstruktur, z. B. das Quadratgitter. Wir geben eine beliebige Wahrscheinlichkeit p vor und besetzen jedes Pixel entsprechend dieser Wahrscheinlichkeit mit 1, also schwarz, ansonsten mit weiß. Dies tun wir unabhängig voneinander für jedes Pixel. Nun suchen wir in der Menge der schwarzen Pixel bezüglich der Vierernachbarschaft einen Weg von der ersten Bildzeile zur letzten Bildzeile, bzw. von der ersten Bildspalte zur letzten Bildspalte. Diese Verbundenheit drückt die Perkolation, folglich die Durchsickerung aus. Wir nehmen einmal an, wir hätten theoretisch die Wahrscheinlichkeit q ermittelt, mit der bei gegebener Wahrscheinlichkeit p ein Weg existiert, die Durchdringung also möglich ist. Diese Kurve muss trivialerweise monoton sein, d. h. je größer p wird, desto größer muss ja wohl auch q werden, zumindestens kann q nicht kleiner werden. Deshalb wird man vermuten, dass die Kurve langsam ansteigt. Die Überraschung ist groß: Diese sieht völlig anders aus. Von $p = 0$ bis p_c ist sie Null und dann springt sie bis $p = 1$ auf 1, folglich eine 0-1-Sprungfunktion. Bis

p_c gibt es keine Wege und ab p_c gibt es immer Wege. Ab der kritischen Wahrscheinlichkeit p_c tritt sozusagen ein Phasenwechsel auf. Das Interessante daran ist, dass diese kritischen Wahrscheinlichkeiten für die Gittertypen unterschiedlich sind:

Gittertyp	Kritische Wahrscheinlichkeit p_c
Hexagonalgitter	0,6962
Quadratgitter	0,5927
Dreiecksgitter	0,5000

Wenn man praktisch Gitter-Körper mit geringen Durchdringungseigenschaften haben möchte, dann wäre das Hexagonalgitter bevorzugt.

1.3 Bildmodelle

Ein Bild stellt mathematisch eine Funktion $f(x)$ von einer, meistens aber von zwei unabhängigen Variablen $f(x, y)$ dar. Natürlich können es auch Funktionen von drei oder mehreren Variablen sein. Oft sind es skalare Funktionen, es können aber auch vektorwertige Funktionen sein, wie z. B. Farbbilder. Wir werden dabei immer ein Koordinatensystem zugrundelegen. Man könnte z. B. ein diskretes Grautonbild auch als Matrix auffassen, dann müsste man die Variablen x und y vertauschen. Dies hätte den „kleinen Vorteil“, dass man alle Matrixoperationen auf Bilder anwenden könnte. Dabei ist aber Vorsicht geboten, was ist z. B. die Determinante eines Bildes? Ist dies der Fall, dann werden wir explizit darauf hinweisen. Je nach Anwendung bzw. mathematischer Behandlung kann man ein Bild auffassen als diskrete Funktion oder als nichtdiskrete Funktion, die wir im Folgenden als analoge Funktion bezeichnen wollen. Eine zweite sehr wichtige Unterscheidung liegt im Definitionsbereich des Bildes, ist dieser endlich oder unendlich? Dies kann man wieder entsprechend dem mathematischen Apparat so oder so tun. Und genau darin liegt die Verwirrung in vielen Büchern, es wird einfach ein Modell ohne Begründung gewählt. Wir wollen im Folgenden also genau unterscheiden und bezeichnen dies als **Bildmodell**. Häufig beschränken wir uns aus schreibtechnischen Gründen auf Funktionen einer Veränderlichen. Ein kleines Problem bei Bildern ist der Bereich außerhalb des Bildes, was ist dort? So könnte man einfach schließen, wir wählen eben einen endlichen Definitionsbereich von der Größe des Bildes, dann haben wir dieses Problem nicht mehr. Dann entstehen aber neue Probleme, die es vorher nicht gab: Bei einem endlichen Definitionsbereich gibt es einen Rand, und manche Operationen (z. B. wie die Faltung) verlangen aber, dass man über den Rand hinaus zugreift. Also muss man außerhalb erklären, was dort vorzufinden ist. Eine Möglichkeit ist, dort alles als Null anzunehmen. Eine zweite Möglichkeit ist die periodische Fortsetzung. Eine weitere Möglichkeit ist die gespiegelte Fortsetzung usw. In den meisten Fällen werden wir auf die periodische Fortsetzung zurückgreifen. Dies entspricht dann dem regulären Quadratgitter auf dem Torus. Es gibt auch genügend Aufgabenstellungen, wo wir uns das Bildmodell nicht auswählen können, sondern wo es völlig klar ist,

welches Modell genommen werden muss. Für periodische Fortsetzungen gilt:

$$f(x + kX) = f(x) \quad \forall x, k \in \mathbb{Z} \quad (1.9)$$

$$f(n + kN) = f(n) \quad \forall n, k \in \mathbb{Z} \quad (1.10)$$

$$f(x + kX, y + lY) = f(x, y) \quad \forall x, y, k, l \in \mathbb{Z} \quad (1.11)$$

$$f(m + kM, n + lN) = f(m, n) \quad \forall m, k, n, l \in \mathbb{Z}. \quad (1.12)$$

Bei Funktionen einer Variablen kann man die Periode X bzw. N genau benennen, für Funktionen zweier Variablen müsste man dann als Periode X, Y schreiben bzw. M, N im diskreten Fall.

Wir wollen im Folgenden vereinbaren bei diskreten Funktionen $f(n)$ den Index tief zu stellen, d. h. $f_n = f(n)$ und $f_{m,n} = f(m, n)$. Wir legen weiterhin Abkürzungen für die Bildmodelle fest:

- A1[X], A2[X,Y] Eindimensionale bzw. zweidimensionale analoge Funktionen über dem Intervall X bzw. X, Y mit periodischer Fortsetzung
- A1 $[-\infty, +\infty]$, A2 $[-\infty, +\infty]$ entsprechend von Minus Unendlich bis plus Unendlich und schließlich das diskrete Analogon bezeichnen wir mit
- D1[N], D2[M,N] bzw.
- D1 $[-\infty, +\infty]$, D2 $[-\infty, +\infty]$.

Bei den unendlichen Modellen muss man das Bild „außerhalb“ des eigentlichen Bildes irgendwie auffüllen, dann sind Nullen durchaus sinnvoll.

1.4 Bildtransformationen

Aus den verschiedensten Gründen müssen Bilder transformiert werden, z. B. bei der Bildregistrierung. Dies können geometrische Transformationen sein, aber auch nur andere Koordinatendarstellungen.

1.4.1 „Reine“ Grauwertransformationen

Natürlich werden bei allen Bildtransformationen immer Grauwerte transformiert. Mit „reinen“ Grauwertransformationen meint man Grauwertransformationen, die völlig unabhängig vom Ort im Bild sind. Ein Grauwert wird immer nach derselben Vorschrift transformiert, egal an welcher Pixelposition er sich im Bild befindet. Die einfachsten Transformationen sind lineare Transformationen oder stückweise lineare Transformationen, wie z. B. die Binarisierung oder andere Quantisierungen der Grauwerte. So simpel wie mathematisch eine Binarisierung auch sein mag, die praktische Bestimmung einer

„optimalen“ Binarisierungsschwellen ist ein ganz anderes und schwieriges Problem, siehe Abschn. 10.1. Mathematisch interessanter sind dagegen die Transformationen, die auf Verteilungen der Grauwerte beruhen. Dazu nehmen wir ein analoges Modell an, die Grauwerte seien $g \in R$. Das Grauerthistogramm ist dann die Dichtefunktion $h_g(x)$ von g und $h'_{g'}(x)$ die Dichte der transformierten Grauwerte g' , wobei die Transformation $g' = f(g)$ laute. Dann gilt:

$$H(g) = \int_0^g h_g(x) dx = H'(g') = \int_0^{g'} h'_{g'}(x) dx \quad \text{mit } 0 = f(0). \quad (1.13)$$

Dies sind aber gerade die Verteilungsfunktionen, woraus man die Transformation der Dichten ablesen kann:

$$h'_{g'}(g') = h_g(f^{-1}(g')) \cdot \frac{df^{-1}}{dg'}. \quad (1.14)$$

Damit die inverse Transformation f^{-1} existiert, fordern wir die strenge Monotonie von f . Wir können nun f so bestimmen, dass die Zieldichte h' oder die Zielverteilung H' eine bestimmte Form oder Funktion annehmen soll. Der einfachste Fall ist sicher der, dass h' konstant bzw. H' linear sein soll. Diesen Fall nennt man *Histogrammausgleichung* (histogram equalization). Ist z. B. $g \in [0, G]$, so können wir h' als Konstante nicht beliebig vorgeben, es muss immer noch eine Dichte sein, daher muss $h' = \frac{1}{G}$ oder $H' = \frac{g'}{G}$ sein. Folglich muss gelten:

$$H(g) = \int_0^g h_g(x) dx = H'(g') = \int_0^{g'} h'_{g'}(x) dx = \frac{g'}{G}. \quad (1.15)$$

Daraus können wir die Transformation

$$g' = f(g) = G \cdot \int_0^g h_g(x) dx \quad (1.16)$$

ablesen. Da die Grauwerte in der Praxis diskret (sogar ganz) sind, müssen wir das Integral in (1.16) durch

$$g' = f(g) = G \cdot \sum_{k=0}^g h(k) \quad (1.17)$$

approximieren, wobei $h(k)$ aus dem Grauerthistogramm zu entnehmen ist. Dadurch lässt sich praktisch keine ideale Gleichverteilung erreichen. Bei Bildern erhöht sich dadurch der Kontrast. Manchmal treten dann auch unerwünschte Effekte auf. Wenn im Bild große

Flächen mit einem geringen Grauwertbereich auftreten (z. B. der Himmel oder Wolken über einer Landschaft), dann werden auch diese Grauerbereiche gestreckt und erscheinen verrauschter oder gestörter als vor der Transformation. Dies lässt sich immer dann nicht verhindern, wenn eine Transformation unabhängig vom Bildinhalt oder vom Ort im Bild durchgeführt wird.

Das Zielhistogramm muss nicht unbedingt konstant sein, wir können beliebige Funktionen vorgeben. Es sei z. B. das Zielhistogramm linear, dann ist die Zielverteilungsfunktion eine Parabel, folglich:

$$h'(g') = 2ag' + b \rightarrow H'(g') = ag'^2 + bg' \quad \text{mit } H'(0) = 0. \quad (1.18)$$

Daraus folgt

$$ag'^2 + bg' = f^*(g) = (aG^2 + bG) \cdot \int_0^g h_g(x) dx \quad (1.19)$$

und damit:

$$g'^2 + \frac{b}{a}g' - f^*(g) = 0 \rightarrow g'_{1,2} = f(g) = -\frac{b}{2a} \pm \sqrt{\frac{b^2}{4a^2} + \frac{f^*(g)}{a}}. \quad (1.20)$$

Es ist diejenige Lösung zu wählen bei der die Ableitung von $f(g)$ positiv ist. Die Parameter a, b der Zielverteilung müssen natürlich bekannt sein, man könnte sie durch Fitting an ein Zielhistogramm ermitteln.

Das Interessante an diesen histogrammbasierten Transformationen ist, dass man sie überall dort einsetzen kann, wo die Histogramme eine Rolle spielen. Man ist überhaupt nicht an Grauerhistogramme gebunden. Ein Beispiel dazu. Es sollen Unterschriften durch *matching* verglichen werden. Dazu muss man die Unterschriften irgendwie normalisieren, damit ein Vergleich auch sinnvoll erscheint. Dazu richten wir die Unterschriften horizontal aus und binarisieren die Unterschriften-Bilder. Dann bilden wir sogenannte Projektionshistogramme. Das x -Projektionshistogramm wird folgendermaßen gebildet: Pro x -Koordinate zählt man die Anzahl der schwarzen Pixel. Mit der Histogrammausgleichung werden anschließend die x -Koordinaten transformiert, die y -Koordinaten ändern sich nicht. Die Binärwerte an den Pixelpositionen werden natürlich „mitgenommen“. Eigentlich liegt nun keine „reine“ Grauwerttransformation vor, sondern eine geometrische Koordinatentransformation, die auch noch nichtlinear ist.

1.4.2 Polar- und Log-Polar-Darstellungen

- Bei der Polar-Darstellung werden die kartesischen Koordinanten (x, y) in Polarkoordinaten (r, φ) umgerechnet, wobei $x = r \cos \varphi$ und $y = r \sin \varphi$ gilt.

- Bei der Log-Polar-Darstellung ist der einzige Unterschied zur Polar-Darstellung die Logarithmierung des Radius. Anstelle (r, φ) werden die Koordinaten $(\log r, \varphi)$ benötigt. Dadurch ergeben sich bei der Umrechnung einige kleine Schwierigkeiten. Für $r = 0$ ist der Logarithmus nicht definiert, man fängt daher erst mit $r \geq 1$ an umzurechnen.

1.4.3 Globale geometrische Transformationen

- Die einfachsten Transformationen sind Translationen:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_{10} \\ a_{20} \end{pmatrix} \Leftrightarrow \mathbf{x}' = \mathbf{x} + \mathbf{a}_0. \quad (1.21)$$

- Die nächste Stufe sind Euklidische Transformationen:

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{a}_0. \quad (1.22)$$

\mathbf{R} ist dabei eine orthogonale Matrix. Diese Transformationen bestehen also aus Translationen, Rotationen und eventuell Spiegelungen. Wenn man Spiegelungen ausschließt, dann ist \mathbf{R} eine Rotationsmatrix, d. h.:

$$\mathbf{R} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}. \quad (1.23)$$

Manchmal betrachten wir auch nur „reine“ Rotationen:

$$\mathbf{x}' = \mathbf{R}\mathbf{x}. \quad (1.24)$$

Wenn wir einen Punkt (x, y) in der Ebene als komplexe Zahl $z = x + iy$ auffassen, dann können wir reine Rotationen in der komplexen Ebene kompakt formulieren:

$$z' = z \cdot e^{i\varphi}. \quad (1.25)$$

Statt reinen Rotationen sind manchmal auch x -Scherungen oder y -Scherungen von Interesse. Eine x -Scherung ist:

$$\begin{aligned} x' &= x + s_x y \\ y' &= y \end{aligned} \quad (1.26)$$

Eine y -Scherung ist:

$$\begin{aligned} x' &= x \\ y' &= s_y x + y \end{aligned} \quad (1.27)$$

Manchmal benötigen wir auch noch eine anisotrope Skalierung:

$$\begin{aligned} x' &= c \cdot x \\ y' &= d \cdot y. \end{aligned} \quad (1.28)$$

- Die nächste Stufe sind Ähnlichkeitstransformationen:

$$\mathbf{x}' = s \cdot \mathbf{R}\mathbf{x} + \mathbf{a}_0 s > 0. \quad (1.29)$$

Es kommt zusätzlich zu den Euklidischen Transformationen noch eine isotrope Skalierung s hinzu.

- Eine weitere Steigerung sind die affinen Transformationen:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{a}_0. \quad (1.30)$$

Dabei ist:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}. \quad (1.31)$$

Die lineare affine Transformation \mathbf{A} (ohne Spiegelungen) lässt sich separieren in eine Hintereinanderausführung einer Rotation, einer anisotropen Skalierung und einer weiteren Rotation:

$$\mathbf{A} = \begin{pmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{pmatrix} \begin{pmatrix} c & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}. \quad (1.32)$$

Dies folgt aus der Singulärwertzerlegung von \mathbf{A} , siehe Abschn. 22.3. Man kann auch gleichwertig die erste Rotation durch eine x -oder y -Scherung ersetzen:

$$\mathbf{A} = \begin{pmatrix} \cos \gamma & -\sin \gamma \\ \sin \gamma & \cos \gamma \end{pmatrix} \begin{pmatrix} e & 0 \\ 0 & f \end{pmatrix} \begin{pmatrix} 1 & s_x \\ 0 & 1 \end{pmatrix}. \quad (1.33)$$

Eine ähnliche Separation gelingt durch Hintereinanderausführung einer x -Scherung, einer y -Scherung und einer anisotropen Skalierung:

$$\mathbf{A} = \begin{pmatrix} g & 0 \\ 0 & h \end{pmatrix} \begin{pmatrix} 1 & 0 \\ s_y & 1 \end{pmatrix} \begin{pmatrix} 1 & s_x \\ 0 & 1 \end{pmatrix}. \quad (1.34)$$

Dabei muss allerdings $a_{11} \neq 0$ vorausgesetzt werden, was aber praktisch nicht relevant ist. Man kann auch die affine Transformation in der komplexen Ebene kompakt darstellen. Mit $z = x + iy$ gilt:

$$z' = a \cdot z + b \cdot \bar{z} + c \quad (1.35)$$

mit

$$\begin{aligned} a &= \frac{1}{2}(a_{11} + a_{22}) + \frac{i}{2}(a_{21} - a_{12}), & b &= \frac{1}{2}(a_{11} - a_{22}) + \frac{i}{2}(a_{21} + a_{12}), \\ c &= a_{10} + ia_{20}. \end{aligned} \quad (1.36)$$

- Die letzte Steigerung, die von praktischem Interesse ist, sind die projektiven Transformationen:

$$x' = \frac{a_{11}x + a_{12}y + a_{10}}{a_{31}x + a_{32}y + a_{30}}, \quad y' = \frac{a_{21}x + a_{22}y + a_{20}}{a_{31}x + a_{32}y + a_{30}}. \quad (1.37)$$

- Eine allgemeine geometrische Transformation schreiben wir einfach als Vektorfunktion:

$$\mathbf{x}' = \mathbf{g}(\mathbf{x}). \quad (1.38)$$

Bei Integralsubstitutionen mit geometrischen Transformationen schreibt man oft:

$$f(\mathbf{x}) = f(\mathbf{g}^{-1}(\mathbf{x}')) = f'(\mathbf{x}'). \quad (1.39)$$

Dies ist keine Bildtransformation, sondern nur der Übergang zu neuen Koordinaten. Eine „echte“ Bildtransformation bezüglich der Koordinaten $\mathbf{x}' = \mathbf{g}(\mathbf{x})$ wird durch

$$f'(\mathbf{x}) = f(\mathbf{g}^{-1}(\mathbf{x})) \quad (1.40)$$

beschrieben. Wir müssen folglich die inverse Transformation \mathbf{g}^{-1} benutzen.

Wollen wir z. B. ein Bild nach „rechts“ verschieben, so müssen wir das verschobene Bild mit $f'(\mathbf{x}) = f(\mathbf{x} - \mathbf{a}_0)$ bezeichnen, es ist folglich die inverse Translation in dieser Notation zu benutzen. Das ist aber nur die mathematische Notation, gedanklich nehmen wir natürlich die Kooordinaten-Transformation $\mathbf{x}' = \mathbf{g}(\mathbf{x})$ und übertragen die Funktionswerte.

1.4.4 Praktische Durchführung der Transformationen

Wenn wir ein diskretes Bild $f(i, j)$ mit einer Transformation $\mathbf{g}(\mathbf{x})$ in ein diskretes Bild $f'(k, l)$ transformieren, so sind ein paar Dinge zu beachten. Wenn wir ganzzahlige Pixel transformieren muss das Ergebnis nicht ganzzahlig sein, wo schreiben wir dann den Grauwert hin? Es könnte dann im Zielbild auch Lücken geben, die überhaupt nicht mit Grauwerten belegt sind. Daher müssen wir umgekehrt vorgehen. Wir geben die Pixel aus dem Zielbild vor und berechnen mit der inversen Transformation die Koordinaten im Quellbild, diese werden i. Allg. auch nicht ganzzahlig sein. Die einfachste Variante ist nun, dass wir den Grauwert eines Nachbarpixels nehmen. Die bessere Variante ist, wenn wir interpolieren. Die häufigste Interpolationsvariante ist die *bilineare* Interpolation mit den vier

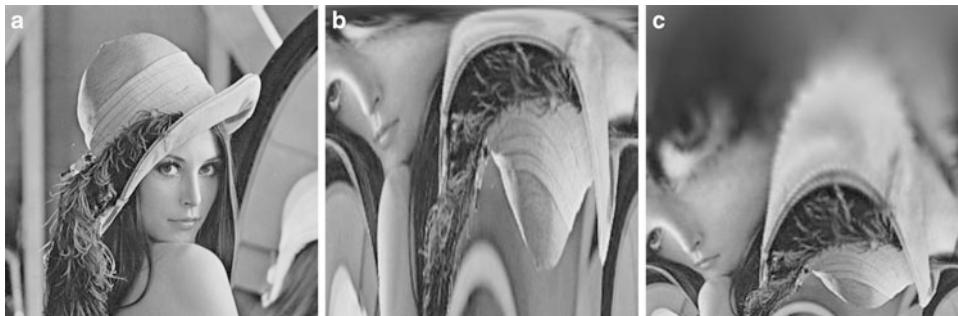


Abb. 1.8 Lenna, **a** Original-Bild, **b** Polar-Bild mit bilinearer Interpolation, $r \geq 1$, **c** Log-Polar-Bild mit bilinearer Interpolation, $r \geq 1$

benachbarten Grauwerten. Weiterhin ist bei der Polar- und Log-Polartransformation noch etwas zu beachten. Da für kleine r wenige Pixel mit Grauwerten zur Verfügung stehen, tragen im Zielbild kleine r wenige Informationen, so dass aus praktischen Gründen r nicht bei Null bzw. Eins, sondern bei größeren Werten beginnen sollte. In der Abb. 1.8c wurde die Log-Polar-Darstellung mit $r = 1$ begonnen. Man sieht den auffälligen homogenen Bereich, der kaum eine Bildinformation trägt, deshalb sollte man nicht mit $r = 1$ beginnen, sondern mit größeren Werten.

1.4.5 Beispiel OCR

Es gibt eine Fülle von Anwendungen, bei denen Bilder transformiert werden müssen. Wir geben einmal ein Beispiel aus der OCR (Optical Character Recognition) an. In der Regel wird ein Dokument/Bild so segmentiert, dass man für jedes Zeichen das umschreibende Rechteck erhält, siehe auch Abschn. 19.7.1. Das umschreibende Rechteck mit den darin enthaltenen Grauwerten ist ein Bild, wenn auch in der Regel ein kleines Bild. Von dem Zeichen sollen nun Merkmale berechnet werden, die für eine anschließende Klassifikation genutzt werden können. Da man in der Vorverarbeitung davon ausgeht, dass anschließend das Dokument horizontal ausgerichtet ist, sind alle Merkmale bezüglich der Rotation normiert. Von affinen Verzerrungen gehen wir auch nicht aus, also bleiben typischerweise noch anisotrope Skalierungen übrig. Die Merkmale sollten also invariant bezüglich dieser Skalierungen sein. Die einfachste und auch beste Methode zur Merkmalsauswahl ist: wir nehmen als Merkmalsvektor einfach alle Grauwerte innerhalb des umschreibenden Rechteckes. Nun müssen wir zur Normalisierung zwei Transformationen durchführen:

- Zur Normalisierung der anisotropen Skalierung geben wir ein Zielbild einer bestimmten Dimension vor, z. B. 10×12 Pixel. Nun transformieren wir die Grauwerte des umschreibenden Rechteckes mittels bilinearer Interpolation auf dieses Zielbild.

- Die Grauwerte des Zielbildes können wir noch nicht als Merkmale benutzen, da Beleuchtungsschwankungen, verschiedener Kontrast usw. noch nicht normalisiert sind. Wir werden die Grauwerte linear transformieren, aber wie? Dazu binarisieren wir das Zielbild, wobei wir eine Schwelle z. B. nach Otsu bestimmen. Da sich im Zielbild ein Zeichen befindet, müsste das Zeichen als Vordergrund schwarz werden und der Rest als Hintergrund weiß. Dies muss nicht genau unseren Vorstellungen entsprechen, aber größtenteils wird es stimmen. Nun bestimmen wir das arithmetische Mittel aller Grauwerte dieser Vordergrundpixel \bar{v} und das arithmetische Mittel aller Grauwerte der Hintergrundpixel \bar{h} . Das binarisierte Zielbild wird nicht mehr benötigt. Die Idee ist nun eine lineare Transformation, wobei \bar{h} auf den kleinsten Grauwert g_{\min} (z. B. Null) und \bar{v} auf den größten Grauwert g_{\max} (z. B. 255) abgebildet wird. Dann wird es passieren, dass einige transformierte Grauwerte kleiner g_{\min} und einige größer als g_{\max} werden. Um diesen Effekt etwas zu lindern, kann man statt g_{\min} nun $g_{\min} + \text{threshold}_1$ und statt g_{\max} nun $g_{\max} - \text{threshold}_2$ benutzen, wobei diese Schranken empirisch zu wählen sind.

Nun können die Grauwerte des Zielbildes als Merkmale benutzt werden, in diesem Falle wäre dies ein 120-dimensionalmer Merkmalsvektor. Diesen können wir noch komprimieren mit der Karhunen-Loeve-Transformation, auch PCA genannt, siehe Abschn. 19.10.1.

2.1 Faltung und Korrelation

Die Operationen Faltung und Korrelation sind in vielen Wissenschaften bekannte und weit verbreitete Operationen. Die Korrelation ist in der Stochastik, Physik, Bildverarbeitung, Signaltheorie usw. eine weit verbreitete Operation, während die Faltung typischerweise in der Signalttheorie angesiedelt ist. Die Korrelation ist auf die Faltung zurückführbar, sodass vom mathematischen Standpunkt aus die Faltung als Operation ausreichen würde. Während die Faltung alle „schönen“ algebraischen Eigenschaften besitzt, hat die Korrelation kaum „vernünftige“ algebraische Eigenschaften, sie ist nicht einmal assoziativ. Da sie aber große Bedeutung in Anwendungen hat, ist es besser sie direkt zu benutzen als sie auf die Faltung zurückzuführen. In der Bildverarbeitung wird die Faltung häufig mit linearen Filtern identifiziert. Dies ist natürlich nicht ausreichend. Zusätzlich werden in diesem Zusammenhang oft nicht ganz korrekte Formulierungen verwendet. So ist die Aussage: „Lineare Filter werden durch eine Faltung beschrieben“ falsch. Die Faltung ist eine lineare Operation, aber zusätzlich noch verschiebungsinvariant. Daher ist richtig: die Faltung beschreibt ein lineares Filter, aber nicht jedes lineare Filter wird durch eine Faltung beschrieben.

2.1.1 Faltung

Die Faltung ist eine der wichtigsten Operationen in der Signalverarbeitung (lineare Systemtheorie) und der Bildverarbeitung, sie ist eine spezielle lineare Operation. In der englischsprachigen Literatur heißt diese **convolution**, manchmal auch **folding**. Für eindimensionale und zweidimensionale, diskrete Funktionen ist die Operation Faltung $*$ definiert zu:

$$(f * g)_n = \sum_i f_i g_{n-i}, (f * g)_{m,n} = \sum_i \sum_j f_{i,j} g_{m-i,n-j}. \quad (2.1)$$

Für analoge Funktionen gilt entsprechend:

$$(f * g)(t) = \int_B f(\xi)g(t - \xi)d\xi, (f * g)(t_1, t_2) = \int_{B_1} \int_{B_2} f(\xi, \eta)g(t_1 - \xi, t_2 - \eta)d\xi d\eta. \quad (2.2)$$

Die Summationsgrenzen oder die Integrationsbereiche sind entsprechend dem Bildmodell zu wählen. Das Ergebnis der Faltung zweier Funktionen ist also wieder eine Funktion, zwei Bilder gefaltet ergibt wieder ein Bild. Häufig findet man in vielen Büchern die nicht ganz korrekte Bezeichnung:

$$f(t) * g(t) = \int_B f(\xi)g(t - \xi)d\xi. \quad (2.3)$$

Es werden nicht zwei Funktionswerte $f(t)$ und $g(t)$ gefaltet, sondern zwei Funktionen f und g und anschließend wird das Ergebnis an der Stelle t betrachtet. Wir werden diese Schreibweise aus praktischen Gründen zum Teil auch benutzen, sind uns aber bewusst, was eigentlich gemeint ist.

Stellvertretend für alle Definitionen werden wir oft die eindimensionale, diskrete Version bevorzugen. Manchmal im Zusammenhang mit der Interpolation wird eine „gemischte“ Faltungsoperation bezüglich der äquidistanten Stützstellen t_k

$$(f * g)(t) = \sum_k f(t_k)g(t - t_k) \quad (2.4)$$

angegeben. Dies ist weder die diskrete noch die analoge Faltung, also keine „echte“ Faltung. Wir werden später zeigen, wie wir diese auf die analoge Faltung zurückführen. Dagegen fasst man die folgende „gemischte“ Faltung

$$h(t_k) = (f * g)(t_k) = \int_B f(\tau)g(t_k - \tau)d\tau \quad (2.5)$$

einfach als Hintereinanderausführung der analogen Faltung

$$h(t) = (f * g)(t) = \int_B f(\xi)g(t - \xi)d\xi \quad (2.6)$$

und der anschließenden Abtastung dieser Funktion $h(t)$ an den diskreten Stellen t_k auf. Die Operation Faltung besitzt alle „schönen“ algebraischen Eigenschaften, sie ist linear, kommutativ, assoziativ, distributiv und vieles mehr. Wir wollen einmal als einfache Übung die Kommutativität zeigen:

$$(f * g)_n = \sum_i f_i g_{n-i} = \sum_{n-l} f_{n-l} g_l = \sum_l g_l f_{n-l} = (g * f)_n. \quad (2.7)$$

Die anderen Eigenschaften lassen sich ebenso einfach zeigen. Eine 2D-Faltung kann manchmal auf die Anwendung von zwei 1D-Faltungen zurückgeführt werden, dann lässt sich die 2D-Faltung effektiv implementieren, siehe Abschn. 8.1.2. Man spricht dann von der Separierbarkeit der Faltung. Wann ist diese nun gegeben? Unter der Separierbarkeit der 2D-Faltung wollen wir verstehen, dass sich ein Operand h der Faltung $g = h * f$ als Produkt $h_{m,n} = c_m \cdot d_n \forall m, n$ schreiben lässt und damit $h = x * y$ ist. Die 2D-Funktionen x und y lassen sich dann durch 1D-Funktionen $x_{i,j} = c_i \delta_j$, $y_{i,j} = d_j \delta_i$ darstellen. Die 1D-Funktion c ist in x als Zeile bzw. d als Spalte in y eingebettet. Dann gilt

$$(x * y)_{m,n} = \sum_{i,j} x_{i,j} y_{m-i,n-j} = \sum_{i,j} c_i \delta_j d_{n-j} \delta_{m-i} = c_m \cdot d_n. \quad (2.8)$$

Nun erkennen wir leicht die Hintereinanderausführung zweier 1D-Faltungen

$$g = h * f = x * y * f = x * (y * f). \quad (2.9)$$

Die Faltung hat viele interessante Anwendungen in der Bildverarbeitung, wird aber auch in anderen Gebieten benutzt. Zum besseren Verständnis zeigen wir zunächst eine einfache Anwendung in der Wahrscheinlichkeitsrechnung.

Beispiel 1 Gegeben seien zwei diskrete Zufallsvariablen X und Y , die der Einfachheit halber nur ganze Zahlen annehmen sollen, ein typisches Beispiel ist das Würfeln. Die Wahrscheinlichkeiten $P(X = i) = p_i$ und $P(Y = j) = q_j$ seien gegeben. Die Zufallsvariablen X und Y seien unabhängig. Gesucht ist nun die Verteilung der Summe der Zufallsvariablen, also $Z = X + Y$. Beim Würfeln würde das bedeuten, wir würfeln zweimal unabhängig voneinander und suchen die Wahrscheinlichkeiten für die Summe der gewürfelten Augen. Da die Zufallsvariablen unabhängig sind, brauchen wir nur die Wahrscheinlichkeiten aller Kombinationen von Augen zu addieren, die die Gesamtsumme der Augen ergeben, also

$$P(Z = l) = \sum_{i,j, i+j=l} p_i \cdot q_j = \sum_i p_i q_{l-i} = (p * q)_l. \quad (2.10)$$

Wenn Y die gleiche Verteilung wie X hat, d.h. $p \equiv q$, dann ist $P(Z = l) = (p * p)_l$, d. h. die Faltung von p mit sich selbst. Dies wird in Analogie zur Autokorrelation als **Autofaltung** bezeichnet. Sind X und Y stetig verteilt und unabhängig mit den Dichten $f(x)$ und $g(y)$, dann ist die Dichte für die Summe ebenfalls

$$h(z) = (f * g)(z) = \int_{-\infty}^{\infty} f(x)g(z-x)dx \quad (2.11)$$

als Faltung beschrieben. Aus der Wahrscheinlichkeitsrechnung ist bekannt: sind X und Y zusätzlich normalverteilt, so ist auch Z normalverteilt. Da die Dichte $h(z)$ von Z sich als Faltung der Dichten schreiben lässt, sagt man auch: „die Normalverteilung ist invariant

gegenüber der Faltung“. Immer wenn man normalverteilte Dichten faltet, ist das Ergebnis eine Dichte der Normalverteilung. Da die Gaußfunktion als Dichte der Normalverteilung in der Bildverarbeitung in verschiedenen Zusammenhängen, insbesondere bei Filtern, eine große Bedeutung besitzt, schreiben wir die Faltung noch einmal für die isotrope 2D-Gauß-Funktion $G(x, y; 0, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$ auf:

$$G(x, y; 0, \sqrt{\sigma_1^2 + \sigma_2^2}) = G(x, y; 0, \sigma_1) * G(x, y; 0, \sigma_2). \quad (2.12)$$

Betrachten wir nochmals zweimal Würfeln mit einem klassischen Würfel, dann gilt $p_i = q_j = \frac{1}{6}$, $i = 1, 2, 3, 4, 5, 6$, $p_i = q_j = 0 \forall i, j < 1, i, j > 6$, welche zweifelsfrei Rechteckfunktionen darstellen. Dies können wir auffassen als ein eindimensionales Bild mit unendlicher Ausdehnung, ein endlich ausgedehntes Bild mit periodischer Fortsetzung ist logischer Weise hier verboten. Nach der Faltung mit sich selbst „dehnt“ sich das Bild weiter aus, es gibt also mehr als sechs Funktionswerte ungleich Null, es entsteht eine Dreiecksfunktion als Verteilungsfunktion. Dabei ist $P(Z = 7) = \sum_l p_l \cdot p_{7-l} = \frac{1}{6}$ der größte Wert der Faltung.

Beispiel 2 Ein fast identisches Beispiel ist die Multiplikation zweier Polynome. Gegeben seien zwei „gewöhnliche“ Polynome

$$p_1(x) = \sum_{i=0}^n a_i x^i, p_2(x) = \sum_{k=0}^m b_k x^k. \quad (2.13)$$

Die Koeffizienten fassen wir als zwei diskrete Funktionen a und b auf, wobei $a_i = 0, i \notin \{0, \dots, n\}$ und $b_k = 0, k \notin \{0, \dots, m\}$ ist. Wir betrachten nun das Produktpolynom $p(x)$ und erhalten

$$p(x) = p_1(x) \cdot p_2(x) = \sum_{l=0}^{m+n} (a * b)_l \cdot x^l. \quad (2.14)$$

Die Koeffizienten des Produktpolynomes ergeben sich als Faltung der beiden diskreten Funktionen, gebildet aus den Koeffizienten der einzelnen Polynome. Dieses Beispiel hat darüberhinaus eine praktische Bedeutung. Wenn wir $x = 10$ setzen, dann würden wir zwei ganze dezimale Zahlen multiplizieren. Das Ergebnispolynom ist das Resultat der Multiplikation, allerdings ohne Übertrag. Wenn wir folglich auf einem Rechner die Multiplikation zweier großer ganzer Zahlen effizient implementieren sollen, dann relaisieren wir dies über die diskrete Faltung, die sich „schnell“ implementieren lässt. Anschließend reichen wir die Überträge von „Ziffer“ zu „Ziffer“ durch und sind fertig.

Beispiel 3 Nun noch ein Beispiel aus der Physik. Wir betrachten einen homogenen, isotropen Diffusionsprozess, der durch folgende partielle Differentialgleichung beschrieben wird:

$$u_t = \operatorname{div}(\nabla u) = \nabla^2 u = \Delta u = u_{xx} + u_{yy}, u(x, y, 0) = f(x, y), \quad (2.15)$$

wobei $u(x, y, t)$ die zeitabhängige Konzentration beschreibt und $f(x, y)$ die Anfangskonzentration festlegt. Diese Gleichung entspricht auch der sogenannten Wärmeleitungsgleichung. Wenn man diese Differentialgleichung löst, erhält man die erstaunliche Lösung

$$u(x, y, t) = f(x, y) * G(x, y; 0, \sqrt{2t}) \quad \text{mit } t > 0. \quad (2.16)$$

Dabei ist $G(x, y; 0, \sigma)$ die isotrope Gauß-Funktion

$$G(x, y; 0, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (2.17)$$

mit dem Erwartungswert 0 und der zeitabhängigen Varianz $\sigma^2 = 2t$. Die Anfangskonzentration $f(x, y)$ wird also mit einer sich ausbreitenden Gaußfunktion gefaltet. Wenn wir die Substitution $2t = \sigma^2$ betrachten und u nach σ ableiten anstatt nach t , dann geht diese Differentialgleichung über in

$$u_\sigma = \sigma \nabla^2 u = \sigma \Delta u. \quad (2.18)$$

Wie oben schon erwähnt ist die Gauß-Funktion selbst Lösung dieser partiellen Differentialgleichung (dies kann man leicht nachrechnen), d. h. es gilt

$$\frac{\partial G(x, y; 0, \sigma)}{\partial \sigma} = \sigma \nabla^2 G(x, y; 0, \sigma) = \sigma \Delta G(x, y; 0, \sigma). \quad (2.19)$$

2.1.2 Korrelation

Als nächste Operation führen wir die Korrelation \circ ein:

$$(f \circ g)_n = \sum_i f_{n+i} \overline{g_i}, \quad (f \circ g)_{m,n} = \sum_i \sum_j f_{m+i, n+j} \overline{g_{i,j}}. \quad (2.20)$$

Für analoge Funktionen gilt:

$$(f \circ g)(t) = \int_B f(t + \xi) \overline{g(\xi)} d\xi,$$

$$(f \circ g)(t_1, t_2) = \int_{B_1} \int_{B_2} f(t_1 + \xi, t_2 + \eta) \overline{g(\xi, \eta)} d\xi d\eta. \quad (2.21)$$

Überstreichungen bedeuten immer die Operation *konjugiert komplex*, konjugiert komplex von Funktionen bedeutet im Folgenden die elementweise Operation. Bei „normalen“ (reellen) Bildern entfällt diese Operation. Verbal heißt dies: wir verschieben f um n Positionen nach „links“ und bilden das Skalarprodukt oder wir verschieben g um n Positionen nach

„rechts“ und bilden das Skalarprodukt. Die Korrelation ist nicht identisch mit der Faltung, ist dieser aber sehr ähnlich. Leider hat sie keinerlei „vernünftige“ algebraische Eigenschaften, weder Kommutativität noch Assoziativität. Anstatt der Kommutativität gilt z. B.:

$$(f \circ g)_m = \overline{(g \circ f)_{-m}}. \quad (2.22)$$

Sie hat aber auch in der Bildverarbeitung große Bedeutung. Weiterhin können wir ablesen:

$$(f \circ g)_n = \sum_i f_{n+i} \overline{g_i} = \sum_l f_{n-l} \overline{g_{-l}}. \quad (2.23)$$

Das ist aber die Faltung von f mit der konjugiert komplexen, am Koordinatenursprung gespiegelten Funktion g . Daher führen wir den Spiegelungsoperator R (**Reflection**) $g_n^R = (Rg)_n := g_{-n}$ ein. Eine erste Rechenregel für die Spiegelung bezüglich der Faltung ist $(f * g)^R = f^R * g^R$. Damit können wir nun die Korrelation auf die Faltung zurückführen:

$$f \circ g = f * \overline{g^R} = f * (\overline{Rg}). \quad (2.24)$$

Dies hat nun den Vorteil, dass wir mit der Faltung rechnen können, bei der ja viele Rechenregeln gelten. Zum Beispiel gilt für reellwertige Funktionen:

$$f \circ g = f * g^R = (g * f^R)^R = (g \circ f)^R. \quad (2.25)$$

Für reelle, gerade Funktionen g gilt $g = g^R$. Ist zusätzlich auch f reell und gerade, so ist

$$f \circ g = f * g^R = f * g = g * f^R = g \circ f. \quad (2.26)$$

Nur in diesem Spezialfall ist die Korrelation kommutativ. Wir sehen, dass durch die Zurückführung der Korrelation auf die Faltung, Ausübung einiger Rechenregeln bezüglich der Faltung und dann wieder Rückübersetzung in die Korrelation, wir Rechenregeln für die Korrelation ableiten können.

Als nächstes führen wir den allgemein bekannten Begriff eines linearen und verschiebungsinvarianten Filters in der Bildverarbeitung ein.

Lineares und verschiebungsinvariantes Filter (LSI-Filter) Gegeben sei ein Bild f und eine Filtermaske g einer ungeraden Dimension mit fest vorgegebenen Zahlenwerten. Die Filtermaske erweitern wir nun gedanklich auf die Bildgröße von f indem wir Nullen auffüllen. Die Filterung bedeutet nun, wir setzen das Filter g mit seinem Mittelpunkt gedanklich über ein Pixel in f . Nun multiplizieren wir alle Grauwerte mit den darüberliegenden Zahlen des Filters und addieren alle diese Produkte. Diese Summe weisen wir dem entsprechenden Pixel in einem dritten Bild h zu. Dies tun wir für alle Pixel des Bildes f . Am Rande setzen wir periodisch fort. Diesen Prozess wollen wir im Folgenden immer als LSI-Filter bezeichnen.

Wir erkennen nun, dieses LSI-Filter wird genau durch die Korrelation beschrieben. Da wir die Korrelation auf die Faltung zurückführen können, heißt dies, ein LSI-Filter anwenden bedeutet die Korrelation von f mit g oder die Faltung von f mit der gespiegelten Filtermaske g^R . Wenn das Filter reell und symmetrisch ist (viele Filtermasken sind symmetrisch), dann ist ein LSI-Filter identisch mit der Korrelation und der Faltung zugleich.

Will man nun ein Bild f mit zwei Filtern g und h nacheinander filtern, also exakt $(f \circ g) \circ h$ ausführen, dann möchte man sicher aus Effektivitätsgründen die beiden Filter g und h zu einem Filter zusammenfassen. Leider gilt aber das Assoziativgesetz für die Korrelation nicht. Wie fasst man aber korrekt die Filter zusammen? Wir wenden jetzt unsere Rechenregeln an:

$$(f \circ g) \circ h = f * g^R * h^R. \quad (2.27)$$

Andererseits gilt aber für die Zusammenfassung der Filter

$$f \circ (g \circ h) = f * (g * h^R)^R = f * g^R * h. \quad (2.28)$$

Wir vergleichen und fassen zusammen

$$(f \circ g) \circ h = f \circ (g \circ h^R). \quad (2.29)$$

Wir müssen also das Filter g mit dem gespiegelten Filter h filtern. Die Korrelation hat nicht nur Bedeutung als lineares Filter, sondern sie ist ein Übereinstimmungsmaß (Matchingmaß) zweier Funktionen bei einer Verschiebung. Der Zahlenwert $(f * g)_n$ gibt an, wie gut f und g miteinander korrelieren, wenn man g um n Positionen nach „rechts“ verschiebt, oder was dasselbe ist, wenn man f um n Positionen nach „links“ verschiebt. Das Matchingmaß ist das Skalarprodukt.

Im Gegensatz zur Autofaltung hat die **Autokorrelation** große Bedeutung, wir korrelieren also eine Funktion mit sich selbst. Sie ist deutbar an einer Stelle n wie gut die Funktion mit sich selbst korreliert wenn man sie um n Positionen verschiebt. Trivialerweise muss die Autokorrelation bei $n = 0$ immer ihr Maximum haben, da f hier mit sich selbst am besten korreliert. Wir schreiben nochmals:

$$f \circ f = f * \bar{f}^R = \bar{f}^R * f. \quad (2.30)$$

Die Autokorrelation ist stets für reelle f eine **gerade** (symmetrische) Funktion, d. h. es gilt immer für reelle f

$$(f \circ f)^R = f^R \circ f^R = f^R * f = f * f^R = f \circ f. \quad (2.31)$$

Da Korrelationen und Faltungen eng mit Verschiebungen verbunden sind, definieren wir nun einen Verschiebungsoperator (*shift operator*) S_m bzw. S_u mit

$$(S_m f)_n := f_{n-m}, (S_u f)(t) := f(t-u). \quad (2.32)$$

Die nächste interessante Eigenschaft lautet: Die Autokorrelation ist **verschiebungsinvariant**:

$$(S_m f \circ S_m f)_n = \sum_i f_{n+i-m} \bar{f}_{i-m} = \sum_l f_{n+l} \bar{f}_l = (f \circ f)_n. \quad (2.33)$$

Gibt es für die Faltung oder Korrelation eigentlich ein neutrales Element *neu*, sodass $f * \text{neu} = \text{neu} * f = f$ bzw. $f \circ \text{neu} = \text{neu} \circ f = f$ gilt? Für die Faltung gibt es dieses Element tatsächlich, es ist der Einheitsimpuls δ , für die Korrelation ist dies nicht der Fall. So gilt bei der Korrelation $\delta \circ f = \delta * f^R = f^R$. Für die Faltung gilt stets $f * \delta = \delta * f = f$. Wie sieht dieser Einheitsimpuls, diese Funktion nun aus? Für den diskreten Fall ist dies einfach $\delta_0 = 1$, $\delta_i = 0 \forall i \neq 0$. Mit dem Verschiebungsooperator S_m können wir nun die „Eins“ des Einheitsimpulses an eine beliebige Stelle m platzieren $e^{(m)} := S_m \delta$. Wenn wir diskrete Funktionen mit N Stützstellen betrachten, so sind diese „Einheitsfunktionen“ trivialerweise Basisfunktionen aller diskreten Funktionen, diese sind vergleichbar mit den „Einheitsbasisvektoren“ im N -dimensionalen Vektorraum. Für analoge Funktionen ist dies schwieriger, hier ist δ die Dirac-Funktion, sie ist im Ursprung nicht eins, sondern unendlich. Dies ist folglich keine „normale“ Funktion mehr, daher nennt man sie auch verallgemeinerte Funktion. Verallgemeinerte Funktionen werden in der Mathematik als **Distributionen** bezeichnet. Der Einheitsimpuls muss also im analogen Fall die Eigenschaft

$$(f * \delta)(t) = (\delta * f)(t) = \int_B f(\tau) \delta(t - \tau) d\tau = \int_B \delta(\xi) f(t - \xi) d\xi = f(t) \quad (2.34)$$

erfüllen. Aus dieser Forderung für beliebige f können wir den Spezialfall $f \equiv 1$ einsetzen und erhalten eine notwendige Bedingung für die Diracfunktion $\int_B \delta(\xi) d\xi = 1$. Setzen wir z. B. für f selbst δ ein, dann folgt

$$\delta(0) = (\delta * \delta)(0) = \int_B \delta(\xi) \delta(0 - \xi) d\xi = \int_B \delta^2(\xi) d\xi = \infty. \quad (2.35)$$

Das Integral über das Quadrat des Einheitsimpulses existiert also nicht mehr. Da wir nun ein neutrales Element δ bezüglich der Faltung gefunden haben, können wir auch inverse Elemente bezüglich der Faltung beschreiben. Falls ein inverses Element f^{-1} zur Funktion f existiert, dann muss es die Bedingung

$$f * f^{-1} = f^{-1} * f = \delta \quad (2.36)$$

erfüllen. In der Signalverarbeitung spielen Funktionen, die der Gleichung

$$f \circ f = f * f^R = \delta \quad (2.37)$$

genügen, eine bedeutende Rolle. Die Gleichung besagt: Die Autokorrelation dieser Funktion ist gleich dem Einheitsimpuls, andererseits ist gleichzeitig die gespiegelte Funktion f^R das inverse Element bezüglich der Faltung. Funktionen, die (2.37) erfüllen, nennt man „weiß gemachte“ Funktionen (*whitening*), siehe Abschn. 4.20. Wenn für f eine inverses Element f^{-1} bezüglich der Faltung existiert, dann ist $(f^{-1})^R$ gleichzeitig rechtsinvers und linksinvers bezüglich der Korrelation, also inverses Element bezüglich der Korrelation, obwohl δ kein neutrales Element der Korrelation ist. Denn es ist

$$\begin{aligned} f * f^{-1} &= \delta \rightarrow f * ((f^{-1})^R)^R = f \circ (f^{-1})^R = \delta = \delta^R = (f^{-1})^R * f^R \\ &= (f^{-1})^R \circ f. \end{aligned} \quad (2.38)$$

Man spricht dann trotzdem von der Korrelationsinversen, obwohl ein neutrales Element bezüglich der Korrelation nicht existiert.

In speziellen Anwendungen, z. B. der *coded aperture imaging*, (siehe auch Abschn. 9.5.7) möchte man binäre Masken haben, wobei ca. 50 Prozent Nullen sein sollen und der Rest Einsen. Die Faltungsinverse dazu sollte auch binär sein, z. B. aus -1 und $+1$ bestehen. Dann kommt man beim Invertieren der Faltung ohne Multiplikationen aus. Gibt es denn solche Faltungsmasken?

Beispiel 1

- $f = (0, 1, 0, 0, 1)$, $N = 5$
- $f^{-1} = (1, 1, -1, -1, 1)$
- $f * f^{-1} = (2, 0, 0, 0, 0) = 2 \cdot \delta$
- Wenn f die Dimension $N = 5$ hat, dann muss f^{-1} auch die Dimension $N = 5$ haben. Sollte aber f als Filtermaske in einem Großen Bild benutzt werden, dann hat f^{-1} die Dimension des grossen Bildes und sieht völlig anders aus.

Wenn man Faltungsmasken entwirft und deren Inverse aber benötigt, dann sollten sie zumindestens näherungsweise (2.37) erfüllen, da dann die inverse Faltung numerisch besonders stabil ist, siehe Abschn. 9.5.1. Die Faltungsmaske stellt dann eine „weiß“ gemachte Funktion dar. Dazu ein

Beispiel 2

- $f = (0, 1, 1, 0, 1, 0, 0)$, $N = 7$.
- Es ist $f \circ f = f * f^R = (3, 1, 1, 1, 1, 1, 1) = 2\delta + 1$.
- Damit muss $f \circ (c \cdot f + d) = \delta$ gelten.
- Es ist $f \circ (cf + d) = cf \circ f + f \circ d = c(2\delta + 1) + f \circ d = \delta$.
- Daraus folgt $c = \frac{1}{2}$, $d = -\frac{1}{6}$.
- Damit ist $(f^{-1})^R = \frac{1}{2}f - \frac{1}{6} = \frac{1}{3}(-\frac{1}{2}, 1, 1, -\frac{1}{2}, 1, -\frac{1}{2}, -\frac{1}{2})$ die Korrelationsinverse und f^{-1} die Faltungsinverse.

Kommen wir noch einmal auf die „gemischte“ Faltungsoperation (2.4) zurück. Diese wird oft im Zusammenhang mit der Interpolation benutzt. Wir wollen diese nun exakt auf die analoge Faltung zurückführen. Es gilt

$$\begin{aligned}
 [(f * g)(t)]_{\text{gemischte Faltung}} &= \sum_k f(t_k)g(t - t_k) = \sum_k f(t_k)g(t - t_k) * \delta(t_k) \\
 &= \sum_k \int_{-\infty}^{\infty} f(\xi)g(t - \xi)\delta(t_k - \xi)d\xi \\
 &= \int_{-\infty}^{\infty} g(t - \xi) \left(\sum_k f(\xi)\delta(t_k - \xi) \right) d\xi \\
 &= \int_{-\infty}^{\infty} g(t - \xi)f_a(\xi)d\xi = (f_a * g)(t).
 \end{aligned} \tag{2.39}$$

Folglich ist die „gemischte“ Faltung die analoge Faltung von g mit der Funktion $f_a(t) = \sum_k f(t)\delta(t_k - t) = \sum_k f(t)\delta(t - t_k)$. Die Summe $\delta_a = \sum_k \delta(t - t_k)$ nennt man auch *ideale Abtastfunktion*. Wir multiplizieren somit f mit der idealen Abtastfunktion. Mit der „gemischten“ Faltung kann man eine beliebige Interpolationsfunktion erzeugen. Wir suchen eine Interpolationsfunktion, die durch die diskreten Abtastwerte $f(t_k)$ von $f(t)$ geht. Dann brauchen wir an die Funktion $g(t)$ nur die Forderung stellen: An den äquidistanten Abtaststellen t_k muss $g(t_k)$ den diskreten Einheitsimpuls darstellen. Zwischen den Abtaststellen kann $g(t)$ ein beliebiges analoges Verhalten aufweisen.

2.1.3 Ableitungen der Faltung

Wir betrachten die Faltung von differenzierbaren Funktionen, also $g(x) = h(x) * f(x)$ und wollen die Regel zur Ableitung von $g(x)$ bestimmen. Es gilt:

$$\begin{aligned}
 g'(x) &= \left(\int_0^x h(\tau)0f(x - \tau)d\tau \right)' = \int_0^x h(\tau)f'(x - \tau)d\tau \\
 &= \int_0^x f(\tau)h'(x - \tau)d\tau.
 \end{aligned} \tag{2.40}$$

Damit gilt die Regel:

$$g'(x) = h'(x) * f(x) = h(x) * f'(x). \tag{2.41}$$

Es wird also nur ein Operand abgeleitet, egal welcher. Eine praktische Anwendung dieser Regel (2.41) wollen wir nun zeigen. Man glättet ein Bild mit der Gauß-Funktion $G(\mathbf{x})$

als Filter und anschließend filtert man das erhaltene Bild mit dem Laplace-Filter Δ . Nach obiger Ableitungsregel gilt dann aber:

$$\Delta(G(\mathbf{x}) * f(\mathbf{x})) = (\Delta G(\mathbf{x})) * f(\mathbf{x}). \quad (2.42)$$

Wir brauchen nur die Gauß-Funktion ableiten und erhalten ein Filter, das sogenannte LoG-Filter (*Laplacian of Gaussian*) oder *Mexican hat*. Mit der isotropen Gauß-Funktion $G(\mathbf{x}) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ bilden wir

$$LoG = \Delta G(x, y) = -\frac{1}{\pi\sigma^4} \left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.43)$$

und erhalten das *LoG*-Filter. Für praktische Anwendungen muss dieses nun noch diskretisiert werden. Dieses LoG-Filter kann man auch als DoG-Filter (*Difference of Gaussian*) näherungsweise darstellen, dazu benutzen wir (2.19), nochmals aufgeschrieben gilt $\frac{\partial G}{\partial \sigma} = \sigma \Delta G$. Wir ersetzen nun näherungsweise die Ableitung $\frac{\partial G}{\partial \sigma}$ durch den Differenzenquotienten und benutzen dabei eine skalierte Standardabweichung $k\sigma$ mit einem Faktor k :

$$\sigma \Delta G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, 0, k\sigma) - G(x, y, 0, \sigma)}{k\sigma - \sigma}. \quad (2.44)$$

Damit gilt:

$$G(x, y, 0, k\sigma) - G(x, y, 0, \sigma) \approx (k - 1)\sigma^2 \Delta G. \quad (2.45)$$

2.1.4 Wrap-around effect und zero padding

In jedem Bildmodell ist die Faltung und Korrelation definiert und das Ergebnis diesbezüglich korrekt. Oft hat man aber Funktionen/Bilder theoretisch im Modell $A1[-\infty, +\infty]$ gegeben, praktisch müssen wir aber letztlich das Modell $A1[X]$ bzw. $D1[N]$ benutzen. Dann können aber unterschiedliche Ergebnisse auftreten. Bei der Anwendung der zyklischen Faltung und später der DFT muss man also genau prüfen ob das Intervall X korrekt ist oder nicht. Dazu ein Beispiel:

Im Abschn. 2.1.1 hatten wir die Verteilung der Summe zweier unabhängiger Zufallsvariablen untersucht. Nehmen wir einen Würfel mit den Augen 1 bis 6 und den Wahrscheinlichkeiten $p_i = 1/6$, $i = 1, 2, 3, 4, 5, 6$ und $p_i = 0$, $i < 1, i > 6$. Diese Verteilung p ist zweifelsohne eine Rechteckfunktion. Die Verteilung der Summe der Augen, wenn wir zweimal würfeln, ist die Autofaltung dieser Rechteckfunktion im Modell $A1[-\infty, +\infty]$, damit stellt $p * p$ eine Dreiecksfunktion in diesem Modell dar. Nun stellen wir uns vor, $p * p$ auszurechnen ist uns numerisch zu kompliziert (was es natürlich nicht ist) und wir nutzen das Faltungstheorem mit der DFT (siehe Fouriertransformation, Abschn. 4.8), dann müssen wir ein endliches Intervall mit den diskreten Wahrscheinlichkeiten $1/6$ und entsprechend viel Nullen festlegen. Nehmen wir das Intervall mit den Werten $p_0 = 0$, $p_1 = 1/6$, $p_2 = 1/6$,

..., $p_6 = 1/6$, $p_7 = 0$, dann ist das auch zweifelsfrei eine Rechteckfunktion, allerdings mit der Periode $N = 8$. Nun rechnen wir im Modell D1[8] die Verteilungsfunktion $p * p$ aus und sehen, dass das Ergebnis falsch ist. Durch die periodische Fortsetzung stimmen die Werte nicht, dies nennt man ***wrap around effect***. Wir haben also das Intervall X dem Problem falsch angepasst. Wie muss man dies nun richtig tun? Die Rechteckfunktion darf sich bei Verschiebung durch die periodische Fortsetzung nicht „echt“ überlappen. Dies erreichen wir, indem wir einfach das Intervall verdoppeln und mit Nullen auffüllen. Das Ergebnisintervall wird trotzdem periodisch fortgesetzt, aber innerhalb einer Periode stimmt das Ergebnis. Das Auffüllen mit Nullen zur korrekten Anpassung des Intervall X nennt man ***zero padding***. Beim Würfeln würden wir also (wenn wir bei $i = 0$ beginnen) die Funktion $p_0 = 0$, $p_1 = p_2 = p_3 = p_4 = p_5 = p_6 = 1/6$, $p_7 = p_8 = p_9 = p_{10} = p_{11} = p_{12} = p_{13} = 0$ benutzen mit $N = 14$. ($N = 12$ würde eigentlich auch reichen, wenn wir bei $i = 1$ beginnen). Nun dürfen wir nicht den Unsinn schlussfolgern, dass bei der Faltung (oder DFT) stets ein Bild oder Block usw. mit Nullen auf die doppelte Größe aufzufüllen ist. Dies kommt immer auf die konkrete Aufgabenstellung an. Nehmen wir einmal zwei gewöhnliche Grauwertbilder gleicher Größe und wir falten diese beiden. Dann brauchen wir nicht mit Nullen auffüllen, da wir ja nicht wissen, was „außerhalb“ der Bilder die tatsächlichen Grauwerte sind. Anzunehmen diese sind Null, ist genauso fragwürdig wie die periodische Fortsetzung.

2.2 Zirkularmatrizen

2.2.1 Elementare Eigenschaften

In diesem Abschnitt wollen wir ausschließlich die endliche, diskrete Faltung mit periodischer Fortsetzung näher untersuchen. Unsere diskreten Funktionen können wir deshalb auch als Vektoren in einem n -dimensionalen Vektorraum auffassen. Daher werden wir manchmal die Funktionen als Spaltenvektoren schreiben bzw. auffassen. Aus dem Zusammenhang soll stets hervorgehen, was exakt gemeint ist. Wenn auf der einen Seite z. B. eine diskrete Funktion steht und auf der anderen Seite ein Spaltenvektor, dann ist die Gleicheit im Sinne zweier diskreter Funktionen zu verstehen. Wenn diskrete periodische Funktionen voraussetzen, dann können wir die Faltung auch ausdrücken als das Produkt einer Matrix \mathbf{C}_h mit dem Spaltenvektor \mathbf{f} :

$$g = h * f \triangleq \mathbf{C}_h \cdot \mathbf{f}. \quad (2.46)$$

Diese Matrix \mathbf{C}_h können wir leicht ausrechnen durch Anwendung der Definition der Faltung, sodass wir leicht finden:

$$\begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{pmatrix} = \begin{pmatrix} h_0 & h_{N-1} & h_{N-2} & \dots & h_1 \\ h_1 & h_0 & h_{N-1} & \dots & h_2 \\ h_2 & h_1 & h_0 & \dots & h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \dots & h_0 \end{pmatrix} \cdot \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{pmatrix}. \quad (2.47)$$

Da in der Matrix \mathbf{C}_h die Zeilen zyklisch verschoben sind, folgt der Name Zirkularmatrix (*circulant matrices*). Wenn wir nicht die periodische Fortsetzung voraussetzen, sondern „außerhalb“ des Intervalles darf „irgendetwas“ stehen, dann schiebt sich das „irgendetwas“ in den nächsten Zeilen ein, es entstehen die sogenannten **Toeplitz-Matrizen**. Zirkularmatrizen sind also spezielle Toeplitz-Matrizen. Allgemein können wir für eine beliebige Toeplitzmatrix \mathbf{T} die Bedingung $t_{i,j} = t_{i+m,j+m}, \forall m$ formulieren. Gilt zusätzlich noch $t_{i,j} = t_{i+N,j}$ bzw. $t_{i,j} = t_{i,j+N}$ (Translationsinvarianz), dann wird die Toeplitzmatrix zur Zirkularmatrix \mathbf{Z} . Man kann durch eine geschickte Erweiterung der Toeplitzmatrix eine Zirkularmatrix erzeugen, allerdings verdoppeln sich dann die Anzahl der Zeilen und Spalten. Eine Toeplitzmatrix \mathbf{T} wird meist in der Form

$$\mathbf{T} = \begin{pmatrix} t_0 & t_1 & t_2 & \dots & \dots & t_{N-1} \\ t_{-1} & t_0 & t_1 & \dots & \dots & t_{N-2} \\ t_{-2} & t_{-1} & t_0 & \dots & \dots & t_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ t_{1-N} & t_{2-N} & t_{3-N} & \dots & \dots & t_0 \end{pmatrix} \quad (2.48)$$

angegeben. Eine typische Toeplitzmatrix, die in der Regel keine Zirkularmatrix ist, tritt z. B. in der Stochastik auf. Dazu fassen wir ein Bild als ein stochastisches Feld auf, d. h. das Bild \mathbf{F} stellt einen Zufallsvektor dar:

$$\mathbf{F} = (X_0, X_1, \dots, X_{N-1})^T. \quad (2.49)$$

Die Kovarianzmatrix oder Momentenmatrix zweiter Ordnung lautet dann:

$$\Sigma = \begin{pmatrix} E(X_0^2) & E(X_0 X_1) & \dots & E(X_0 X_{N-1}) \\ E(X_1 X_0) & E(X_1^2) & \dots & E(X_1 X_{N-1}) \\ \vdots & \vdots & \vdots & \vdots \\ E(X_{N-1} X_0) & E(X_{N-1} X_1) & \dots & E(X_{N-1}^2) \end{pmatrix}. \quad (2.50)$$

Wenn man fordert, dass das Feld *stationär* im erweiterten Sinne sein soll, d. h. $E(X_i X_j) = r_{i-j}$, dann ist die Kovarianzmatrix eine Toeplitzmatrix.

Nun können wir jeder Funktion h bezüglich der Faltung eine Zirkularmatrix \mathbf{C}_h zuordnen und die Faltung mit dieser Matrix ausdrücken. Damit können wir nun automatisch alle Eigenschaften der Faltung auf die Operationen mit diesen Zirkularmatrizen übertragen bzw. wieder „zurückübersetzen“ in die Faltung. Wie sieht z. B. die Zirkularmatrix des Einheitsimpulses aus? Dies ist die Einheitsmatrix, also $\mathbf{C}_\delta = \mathbf{E}$. Spiegeln wir eine Funktion, dann wird die Zirkularmatrix transponiert, folglich $\mathbf{C}_{f^R} = \mathbf{C}_f^T$. Die inverse Funktion f^{-1} bezüglich der Faltung existiert genau dann, wenn die Zirkularmatrix \mathbf{C}_f invertierbar ist und es gilt $\mathbf{C}_{f^{-1}} = (\mathbf{C}_f)^{-1}$. Dies heißt wiederum, die Inverse einer Zirkularmatrix muss selbst wieder eine Zirkularmatrix sein. Da die Faltung kommutativ ist $f * g = g * f$, muss das Matrixprodukt kommutativ sein $\mathbf{C}_f \cdot \mathbf{C}_g = \mathbf{C}_g \cdot \mathbf{C}_f$ und das Produkt muss selbst wieder

eine Zirkularmatrix sein. Für zweidimensionale diskrete Funktionen (Bilder) können wir genauso die Zirkularmatrizen bilden. Dazu ordnen wir alle Zeilen des Bildes nacheinander in einem Spaltenvektor an, wir nehmen an, die Dimension des Bildes sei M, N , dann hat dieser die Länge $L = M \cdot N$ und die Zirkularmatrix die Dimension (L, L) , wobei diese sich zirkular zusammensetzt aus den Zirkularmatrizen der Zeilenvektoren $h^j, j = 0, \dots, M - 1$ des Bildes h und daher auch **Blockzirkularmatrizen** genannt werden:

$$\mathbf{C}_h = \begin{pmatrix} \mathbf{C}_{h^0} & \mathbf{C}_{h^{M-1}} & \mathbf{C}_{h^{M-2}} & \dots & \mathbf{C}_{h^1} \\ \mathbf{C}_{h^1} & \mathbf{C}_{h^0} & \mathbf{C}_{h^{N-1}} & \dots & \mathbf{C}_{h_2} \\ \mathbf{C}_{h^2} & \mathbf{C}_{h^1} & \mathbf{C}_{h^0} & \dots & \mathbf{C}_{h^3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{h^{N-1}} & \mathbf{C}_{h^{N-2}} & \mathbf{C}_{h^{N-3}} & \dots & \mathbf{C}_{h^0} \end{pmatrix}. \quad (2.51)$$

Mit den Zirkularmatrizen haben wir eine erste numerische Möglichkeit, Faltungsgleichungen zu lösen bzw. die Faltung zu invertieren. Auf Grund der großen Dimension der Zirkularmatrizen (bei einem $(1000, 1000)$ -Pixel-Bild hat die Zirkularmatrix die Dimension $(1.000.000, 1.000.000)$) werden wir dies aber praktisch nicht tun. Dennoch sind die Zirkularmatrizen theoretisch sehr hilfreich, da wir damit Wissen aus der Matrixalgebra auf die Faltung recht einfach übertragen können. Dazu ein

Beispiel Wie wir später sehen werden, hat die inverse Faltung große praktische Bedeutung. Wir wollen also eine Faltungsgleichung $h * f = g$ lösen, wobei g und h gegeben sind und die Funktion f gesucht ist. Diese Gleichung resultiert oft aus verrauschten Daten, d. h. es ist sehr wahrscheinlich, dass diese Gleichung überhaupt keine Lösung hat. Dann wollen wir sie eben im Sinne der kleinsten Quadrate lösen, d. h. wir suchen ein f , sodass $\|h * f - g\|^2 \rightarrow \text{Minimum}$ gilt. Offensichtlich ist $\|\mathbf{C}_h \cdot \mathbf{f} - \mathbf{g}\|^2 \rightarrow \text{Minimum}$ die Formulierung mit der Zirkularmatrix \mathbf{C}_h . Wir haben also ein lineares Gleichungssystem im Sinne der kleinsten Quadrate zu lösen. Dies ist ein Standardproblem der linearen Algebra und führt auf die *Gaußschen Normalengleichungen* (21.32) bzw. (22.16):

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \leftrightarrow \mathbf{A}^T \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{A}^T \cdot \mathbf{b}. \quad (2.52)$$

Wir brauchen also nur mit der transponierten Koeffizientenmatrix \mathbf{A}^T die linke und rechte Seite zu multiplizieren. Dies tun wir nun auch mit der transponierten Zirkularmatrix:

$$\mathbf{C}_h \cdot \mathbf{f} = \mathbf{g} \leftrightarrow \mathbf{C}_h^T \cdot \mathbf{C}_h \cdot \mathbf{f} = \mathbf{C}_h^T \cdot \mathbf{g}. \quad (2.53)$$

Nun „übersetzen“ wir wieder diese beiden Gleichungen in die Faltungsgleichungen

$$h * f = g \leftrightarrow h^R * h * f = h^R * g. \quad (2.54)$$

Diese Gaußschen Faltungs-Normalengleichungen können wir auch in der Form

$$(h \circ h) * f = g \circ h \quad (2.55)$$

schreiben. Wir sehen, die Autokorrelation wird auf der linken Seite verwendet und auf der rechten Seite die Kreuzkorrelation. Wir könnten (2.55) auch als die Gaußschen Faltungs-Normalengleichungen bezeichnen.

2.2.2 Normabschätzungen und Stabilität

In vielen Anwendungen spielt die Abschätzung einer Norm $\|f * g\|_p$ eine große Rolle. Allgemein gilt folgende Normabschätzung:

$$\|f * g\|_r \leq \|f\|_p \cdot \|g\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1 + \frac{1}{r}, \quad p, q, r \geq 1. \quad (2.56)$$

Wir sehen, für die L_1 -Norm ist die Abschätzung korrekt, für die Euklidische Norm gilt dies nicht. Häufig benötigen wir aber die Euklidische Norm. Die Zirkularmatrizen helfen uns hier weiter, weil wir hier die Hilfsmittel aus der Numerik und linearen Algebra benutzen können. Gegeben sei eine Matrix \mathbf{A} und ein Vektor \mathbf{x} , wenn dann die Ungleichung

$$\|\mathbf{A} \cdot \mathbf{x}\|_l \leq \|\mathbf{A}\|_k \cdot \|\mathbf{x}\|_l \quad (2.57)$$

gilt, dann heißt die Matrixnorm $\|\mathbf{A}\|_k$ zur Vektornorm $\|\mathbf{x}\|_l$ verträglich. Für $l = 2$, d. h. zur Euklidischen Norm sind z. B. die Frobenius-Norm und die Spektralnorm verträglich, wobei die Spektralnorm die bessere Abschätzung liefert. Wenn wir dies nun für die Zirkularmatrizen nutzen, dann müssen wir die Matrixnorm der Zirkularmatrix wieder „zurückübersetzen“ und zwar in dem Sinne, was dies für die Funktion bedeutet. Für die Frobenius-Norm bedeutet dies z. B. $\|\mathbf{C}_f\|_2 = \sqrt{N}\|\mathbf{f}\|_2$, wodurch wir die Abschätzung

$$\|f * g\|_2 \leq \sqrt{N}\|f\|_2 \cdot \|g\|_2 \quad (2.58)$$

für die Euklidische Norm erhalten. Eine bessere Abschätzung liefert allerdings die Spektralnorm $\|\mathbf{A}\|_S$. Diese ist der betragsgrößte Eigenwert der Matrix \mathbf{A} . Der betragsgrößte Eigenwert einer Zirkularmatrix \mathbf{C}_f ist der betragsgrößte Fourierkoeffizient der Funktion f , diesen bezeichnen wir mit $\|f\|_S$. Daher gilt:

$$\|f * g\|_2 \leq \|f\|_S \cdot \|g\|_2. \quad (2.59)$$

In den Abschn. 4.11 und 4.12 kann man nachlesen, dass für die dort benutzte Fouriertransformation die Beziehung

$$\|f\|_S = \sqrt{N} \max_k |\alpha_k(f)| \quad (2.60)$$

gilt. Für andere Fouriertransformationen könnte sich der Faktor ändern. Zwischen Euklidischer Norm und Spektralnorm gilt dann die Abschätzung:

$$\|f\|_2 = \|f * \delta\|_2 \leq \|f\|_S \cdot \|\delta\|_2 = \|f\|_S. \quad (2.61)$$

In vielen wichtigen Anwendungen spielt die Lösung einer Faltungsgleichung eine große Rolle. In der Gleichung $h * f = g$ seien h und die rechte Seite g gegeben. Gesucht ist die Funktion f . Wenn die Faltungsinverse h^{-1} existiert, ist die Lösung formal $f = h^{-1} * g$. Oft existiert die Inverse nicht und die rechte Seite ist mit Fehlern behaftet, sei es durch Rauschen oder andere Störungen. Im Abschn. 9.5 wird dies ausführlich diskutiert. Die Lösung einer Faltungsgleichung nennt man auch *deconvolution*. Jetzt wollen wir aber den einfachen Fall betrachten, dass die Faltungsinverse h^{-1} tatsächlich existiert. Dann interessiert nur noch folgendes Problem:

Die rechte Seite g sei mit Fehlern behaftet und nicht g , sondern eine Näherung \tilde{g} ist gegeben. Dann lösen wir die Faltungsgleichung $h * \tilde{f} = \tilde{g}$ und erhalten die Lösung \tilde{f} . Es entsteht sofort die Frage, wie sich der Fehler $\|g - \tilde{g}\|$ auf den Fehler $\|f - \tilde{f}\|$ der Lösung auswirkt.

Definition 2.1 (Stabile Systeme) Ein System heißt stabil, wenn „kleine“ Änderungen des Inputs (hier die rechte Seite) nur „kleine“ Änderungen des Outputs (hier die Lösung) bewirken.

Das Wort „kleine“ ist sehr relativ und sollte irgendwie quantitativ beurteilbar sein. Da wir unsere Faltungsgleichungen auch mit Zirkularmatrizen beschreiben können und die Faltungsgleichung damit weiter nichts als ein lineares Gleichungssystem darstellt, entnehmen wir dazu die Hilfsmittel aus der numerischen linearen Algebra. Gegeben sei statt der Faltungsgleichung ein lineares Gleichungssystem:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \leftrightarrow \mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}. \quad (2.62)$$

Daraus folgt sofort:

$$\mathbf{x} - \tilde{\mathbf{x}} = \mathbf{A}^{-1} \cdot (\mathbf{b} - \tilde{\mathbf{b}}). \quad (2.63)$$

Unter Benutzung der Rechenregel (2.57) für verträgliche Matrixnormen erhalten wir sofort die Abschätzung:

$$\|\mathbf{x} - \tilde{\mathbf{x}}\| \leq \|\mathbf{A}^{-1}\| \cdot \|(\mathbf{b} - \tilde{\mathbf{b}})\|. \quad (2.64)$$

Wir haben mit der Norm der inversen Matrix eine quantitative Abschätzung gefunden. Allerdings haben wir den absoluten Fehler abgeschätzt, besser ist es, dafür den relativen Fehler zu benutzen. Wir benutzen dazu

$$\|\mathbf{b}\| = \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\| \rightarrow \|\mathbf{x}\| \geq \frac{\|\mathbf{b}\|}{\|\mathbf{A}\|} \quad (2.65)$$

und schätzen Ungleichung (2.64) weiter ab:

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}^{-1}\| \cdot \|(\mathbf{b} - \tilde{\mathbf{b}})\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}\| \|\mathbf{A}^{-1}\| \cdot \|(\mathbf{b} - \tilde{\mathbf{b}})\|}{\|\mathbf{b}\|}. \quad (2.66)$$

Die quantitative Abschätzung wird nun durch das Maß

$$K(\mathbf{A}) = \|\mathbf{A}\|_S \|\mathbf{A}^{-1}\|_S \quad (2.67)$$

beschrieben und heißt **Konditionszahl** der Matrix \mathbf{A} . Dabei haben wir unter allen verträglichen Normen die Spektralnorm als optimale Norm gewählt. Je kleiner $K(\mathbf{A})$ ist, desto stabiler verhält sich das System. Nun können wir dies mit den Zirkularmatrizen in die „Faltung übersetzen“. Folglich können wir bezüglich der Stabilität der Faltungsgleichung $h * f = g$ die Faltungs-Konditionszahl

$$K(h) = \|h\|_S \cdot \|h^{-1}\|_S \quad (2.68)$$

als Maß benutzen.

Im Abschn. 4.12 wird die Konditionszahl einer Funktion bezüglich der Faltung mit Hilfe der Fourierkoeffizienten dargestellt, siehe (4.48). Die Gaußschen Faltungs-Normalengleichungen (2.55) sind zwar immer lösbar, aber oft instabil. Des Weiteren braucht auch die Faltungsinverse von $h \circ h$ nicht zu existieren. Dann kann man sich der Methode der Regularisierung bedienen. Im Abschn. 22.2 wurde die Regularisierung eines linearen Gleichungssystems beschrieben. Dies führte auf die Ersatzaufgabe (22.39). Wenn wir dafür Zirkularmatrizen einsetzen, können wir wieder diese Gleichung (22.39) in die Faltung „übersetzen“ und erhalten die regularisierten Faltungs-Normalengleichungen:

$$(h \circ h + \lambda \delta) * f = g \circ h. \quad (2.69)$$

Diese regularisierten Faltungs-Normalengleichungen (2.69) treten wieder auf im Zusammenhang mit der Bildrestaurierung und sind Bestandteil der Methode „Restaurierung unter Zwang“, siehe Abschn. 9.5.2.

2.3 LSI-Operatoren

Im Folgenden wollen wir LSI-Operatoren (*Linear Shift Invariant*) untersuchen. In der Signaltheorie werden diese oft in Bezug auf die Zeit als LTI-Systeme (*Linear Time Invariant*) bezeichnet. Ein Operator L ist linear, wenn

$$L(\lambda \cdot f + \mu \cdot g) = \lambda \cdot Lf + \mu \cdot Lg \quad \forall \lambda, \mu \in \mathbb{C} \quad (2.70)$$

für beliebige Funktionen f und g gilt. Wir erinnern im Unterschied dazu daran, dass es auch lineare Funktionen gibt. Dabei meint man bei Polynomen, dass die höchste Potenz

eins ist. Lineare Funktionen stellen aber nicht unbedingt lineare Operatoren da. So ist $y = ax + b$ eine lineare Funktion, aber kein linearer Operator. Das additive Glied stört die Linearität. Ebenso ist eine affine Abbildung $\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$ linear im Sinne einer Funktion, aber kein linearer Operator, da die Verschiebung \mathbf{b} stört. Ohne die Verschiebung sind affine Abbildungen auch lineare Operatoren. Ein simples Beispiel eines linearen Operators ist der bereits eingeführte Verschiebungsoperator S_m bzw. S_u . Ein weiteres einfaches Beispiel ist der Faltungsoperator L_g :

$$L_g f := g * f. \quad (2.71)$$

Wenn wir einmal die „Nullfunktion“ einsetzen, z. B. $L0 = L(0 + 0) = L0 + L0 = 2L0$, so sehen wir, dass immer $L0 = 0$ gelten muss. Ein linearer Operator bildet also immer das Nullelement auf das Nullelement ab. Wann ist nun ein linearer Operator L noch zusätzlich verschiebungsinvariant? Wir benötigen dazu den Verschiebungsoperator, siehe (2.32). Ein beliebiger Operator L ist genau dann verschiebungsinvariant, wenn gilt:

$$L(S_m f) = S_m(Lf), \quad L(S_u f) = S_u(Lf). \quad (2.72)$$

Wenn wir den Operator L mit dem Verschiebungsoperator S_m bzw. S_u vertauschen dürfen, die Reihenfolge also keine Rolle spielt, dann nennen wir L verschiebungsinvariant. Ist ein Operator linear und verschiebungsinvariant, dann nennen wir L einen LSI-Operator. Welche Operatoren sind nun LSI-Operatoren? Dazu formulieren wir einen wichtigen Satz, den wir auch Hauptsatz der LSI-Theorie nennen könnten:

Satz 2.2 *Ein beliebiger Operator L ist genau dann ein LSI-Operator, wenn er ein Faltungsoperator ist.*

Beweis Der Satz zeigt uns zwei Richtungen auf. Zuerst wollen wir den einfachen Teil zeigen, wobei wir uns auf diskrete Funktionen im Beweis beschränken:

1. *Ein Faltungsoperator ist ein LSI-Operator.* Es ist nahezu trivial zu zeigen, dass die Faltung linear ist. Bleibt nur die Verschiebungsinvarianz der Faltung zu zeigen:

$$\begin{aligned} (L_f S_k g)_n &= (f * S_k g)_n = \sum_m f_m (S_k g)_{n-m} = \sum_m f_m g_{n-k-m} \\ &= (f * g)_{n-k} = (S_k(f * g))_n = (S_k L_f g)_n. \end{aligned} \quad (2.73)$$

Damit folgt

$$L_f S_k g = S_k L_f g \quad (2.74)$$

die Verschiebungsinvarianz der Faltung. Außer den elementaren algebraischen Eigenschaften hat nun die Faltung auch noch die LSI-Eigenschaft.

2. Ein beliebiger LSI-Operator L ist ein Faltungsoperator.

Eine beliebige diskrete Funktion g mit N Stützstellen entwickeln wir nach einer Basis, wir nehmen die „triviale“ Basis, also die N Einheits-Basisfunktionen $e^{(k)} = S_k \delta$ und schreiben

$$g = \sum_{k=0}^{N-1} g_k \cdot e^{(k)}. \quad (2.75)$$

Nun wenden wir den beliebigen LSI-Operator L auf g an und nutzen natürlich dabei die LSI-Eigenschaft des Operators aus:

$$(Lg)_n = \sum_k g_k (LS_k \delta)_n = \sum_k g_k (S_k L \delta)_n = \sum_k g_k (L \delta)_{n-k} = (g * L \delta)_n. \quad (2.76)$$

□

Als Folgerung aus dem Beweis lesen wir eine Grundgleichung ab:

$$Lg = g * L\delta \quad \forall g. \quad (2.77)$$

Da die Faltung verschiebungsinvariant ist, sind lineare Filter, so wie wir sie eingeführt hatten, eigentlich LSI-Filter, also spezielle lineare Filter. Folglich ist der allgemeine Ausdruck „lineares Filter“ nicht falsch, aber es ist ein spezielles lineares Filter.

Jeder LSI-Operator L lässt sich also darstellen als Faltung des Inputs g mit der Funktion $L\delta$. Die Funktion $L\delta$ heißt PSF (Point Spread Function) oder Impulsantwort, welche das Ergebnis der Anwendung des Operators L auf den Einheitsimpuls δ darstellt. Dies ist ein grundlegendes Ergebnis der linearen Systemtheorie. Nun folgen einige Beispiele von LSI-Operatoren.

Beispiel 1 Die Filtermasken der linearen Filterung sind als Impulsantwort eines LSI-Operators interpretierbar. Das bekannte Sobelfilter zur Bestimmung horizontaler und vertikaler Kanten können wir nun auch so schreiben:

$$h_x = R(L\delta) = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad h_y = R(L\delta) = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}. \quad (2.78)$$

Diese linearen Filter sind somit LSI-Operatoren.

Die Korrelation (lineare Filterung) ist also die Faltung mit den gespiegelten Filtermasken. Wie sehen dann aber lineare Filter aus, die wirklich nur lineare Operatoren darstellen und nicht verschiebungsinvariant sind? Dazu greifen wir die Stelle aus dem Beweis heraus, bei dem wir die Linearität genutzt haben, aber noch nicht die Verschiebungsinvarianz:

$$(Lg)_n = \sum_k g_k (LS_k \delta)_N = \sum_k g_k (Le^{(k)})_n. \quad (2.79)$$

Das gefilterte Bild Lg lässt sich mit $h_{i,k} := (Le^{(k)})_i$ darstellen als

$$y = Lg = \sum_k g_k Le^{(k)} \Leftrightarrow y_i = \sum_k g_k h_{i,k} \quad (2.80)$$

Linearkombination der Impulsantworten $Le^{(k)}$. Mit $h(t, u) := (LS_u\delta)(t)$ ist dies im analogen Fall

$$y(t) = \int_B g(u)h(t, u)du. \quad (2.81)$$

Die Impulsantworten hängen jetzt vom Ort ab. Das Wesentliche der Verschiebungsinvarianz ist also die Ortsunabhängigkeit dieser Impulsantworten. Ohne Verschiebungsinvarianz bedeutet also lineare Filterung: Wir konstruieren aus den ortsabhängigen PSF $Le^{(k)}$ pro Pixel i eine Filtermaske, die von Pixel zu Pixel verschieden sein kann. Bei linearen Filtern, die LSI-Operatoren darstellen, haben wir also eine einzige Filtermaske, und bei linearen Filtern ohne LSI-Eigenschaft, haben wir für jedes Pixel eine mögliche andere Filtermaske. Nochmals zur Klarstellung: Die Filtermaske pro Pixel i ist nicht identisch mit der PSF $Le^{(i)}$, sondern setzt sich aus der i -ten Stelle aller PSF $Le^{(k)}$ zusammen. Stillschweigend haben wir immer vorausgesetzt: die Filtermaske, egal ob sie ein LSI-Filter oder nur ein lineares Filter darstellt, darf niemals vom konkreten Bild selbst abhängen, ansonsten stellen sie nichtlineare Filter dar, siehe z. B. Abschn. 8.2.2.

Beispiel 2 Das optische System einer Kamera ist auf jeden Fall als Operator deutbar, der das „tatsächliche“ Bild der Umwelt als Input in das aufgenommene, im Rechner abgespeicherte Bild transformiert. Da sich Lichtintensitäten addieren, können wir von einer Linearität tatsächlich ausgehen. Verschiebungsinvarianz heißt, die PSF ist unabhängig von der Lage des Einheitsimpulses, also ortsunabhängig ist. Dazu nehmen wir das Modell der dünnen Linse an, siehe Abb. 14.2. Weiterhin nehmen wir an, dass die Blende kreisförmig ist. Dann hat die PSF in grober Näherung die Form einer Pillendose (*pill box*) mit einem bestimmten Radius und einer bestimmten Höhe. Man kann nun in der geometrischen Optik zeigen, dass dieser Radius unabhängig von der Lage des Objektpunktes in einer Objektebene, aber abhängig von der Tiefe der Objektebene ist. Folglich gilt die Verschiebungsinvarianz nur für den Bildbereich, in dem die Objekte gleich weit entfernt sind. Wenn alle Objekte im Bild sehr weit entfernt sind, können wir näherungsweise Verschiebungsinvarianz annehmen. Weiterhin kann man dies näherungsweise lokal im Bild annehmen. Damit realisiert das optische System (zumindestens näherungsweise, lokal im Bild) einen LSI-Operator und wir wissen sofort, dass dann

$$g = h * f, \quad h = L\delta \quad (2.82)$$

die Bildaufnahme bezüglich der Unschärfe beschreiben muss. Wenn man also g im Rechner abgespeichert hat und $h = L\delta$ die gemessene (oder berechnete) PSF ist, dann möchte

man das „scharfe“ Bild f bestimmen. Es liegt also mathematisch das Problem der inversen Faltung vor. Wie sieht eine solche optische PSF eigentlich aus? In der Abb. 14.2 ist zu sehen: Wenn man die Bildebene verschiebt, wird der gleiche Objektpunkt unscharf abgebildet. Wenn wir annehmen, dass die Apertur (Blende) kreisförmig ist, dann wird der Objektpunkt auch in einen Kreis „verschmiert“, zumindestens folgt dies aus der geometrischen Optik. Infolge anderer optischer Effekte ist dies aber in der Praxis etwas komplizierter. Im einfachsten Fall modelliert man die PSF als Pillendose (*pill box*), d. h. als eine Kreisfläche mit einem konstanten Funktionswert über der Kreisfläche. Häufig modelliert man sie aber als eine rotationssymmetrische Gaußfunktion

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (2.83)$$

wobei σ als Radius des Kreises zu deuten wäre.

Beispiel 3 Wir nehmen einmal den Verschiebungsoperator S_m und untersuchen, ob dieser ein LSI-Operator ist. Dass dieser linear ist, ist trivial. Verschiebungsinvarianz bedeutet $S_k S_m f = S_m S_k f$, was auch trivialerweise gilt. Damit ist der Verschiebungsoperator tatsächlich ein LSI-Operator und muss sich mit der Faltung darstellen lassen:

$$S_m f = f * S_m \delta. \quad (2.84)$$

Beispiel 4 Jetzt wollen wir ein Beispiel angeben für einen linearen Operator, der nicht verschiebungsinvariant ist. Wir betrachten den Spiegelungsoperator $Rf = f^R$, dessen Linearität trivial zu zeigen ist. Die Verschiebungsinvarianz bedeutet $S_k Rf = RS_k f$. Dies ist aber falsch, stattdessen gilt $S_{-k} Rf = RS_k f$.

Jetzt wollen wir einen LSI-Operator L auf eine Faltung selbst anwenden:

$$L(f * g) = (f * g) * L\delta = f * (g * L\delta) = f * (Lg) = (f * L\delta) * g = (Lf) * g. \quad (2.85)$$

Wir erkennen als Rechenregel, dass man den Operator an genau einen Operanden der Faltung „hineinziehen“ kann. Damit gilt auch

$$Lg = L(g * \delta) = (Lg) * \delta = g * (L\delta), \quad (2.86)$$

was wir aber schon kannten. Da der Verschiebungsoperator auch ein LSI-Operator ist, gilt somit auch $S_m g = g * S_m \delta$. Diese Faltungsgleichung werden wir noch zur Bestimmung der Verschiebung nutzen. Beim Spiegelungsoperator hatten wir die Eigenschaft $R(f * g) = (Rf) * (Rg)$ gezeigt, folglich kann er kein LSI-Operator sein. Mit den nun hergeleiteten Rechenregeln können wir noch einmal die Verschiebungsinvarianz der Autokorrelation beweisen, dazu sei $g = S_m f$ gegeben, dann gilt:

$$\begin{aligned} g \circ g &= S_m f \circ S_m f = S_m f * R(S_m f) = S_m(f * R(S_m f)) \\ &= S_m(f * S_{-m} Rf) = S_m S_{-m}(f * Rf) = f \circ f. \end{aligned} \quad (2.87)$$

Beispiel 5 Wie verhält sich die Autokorrelation bei einem LSI-System $g = h * f$? Dies ist mit unseren Rechenregeln sehr einfach herzuleiten:

$$\begin{aligned} g \circ g &= g * Rg = h * f * R(h * f) = h * f * Rh * Rf \\ &= (h * Rh) * (f * Rf) \\ &= (h \circ h) * (f \circ f). \end{aligned} \quad (2.88)$$

Diese formale Analogie ist verblüffend und wird als *Wiener-Lee-Beziehung* bezeichnet.

2.4 Faltungsgleichungen im Ortsraum

2.4.1 Iterative Entfaltung

Wenn eine Faltungsgleichung $h * f = g$ gegeben ist, h und g gegeben sind und f gesucht ist, dann liegt ein Problem der inversen Faltung oder *deconvolution* vor. Im Abschn. 9.5 werden wir dies als Restaurationsproblem ausführlich behandeln. Im Wesentlichen werden alle Methoden im Fourieraum durchgeführt. Kann man aber auch im Ortsraum die Funktion f berechnen? Mit Zirkularmatrizen ist dies möglich, aber der praktische Aufwand ist zu enorm. Dennoch gibt es einige numerische Hilfsmittel, um auch im Ortsraum die Faltung zu invertieren. Dazu formen wir unsere Gleichung $h * f = g$ in eine Fixpunktgleichung um:

$$f = f + \alpha(g - h * f) \Leftrightarrow f = (\delta - \alpha h) * f + \alpha g, \quad \alpha \in \mathbb{R} \quad (2.89)$$

und versuchen durch Iteration diese Fixpunktgleichung zu lösen. In der Bildverarbeitung werden Modifikationen dieser Iterationen als Van Cittert-, Bially- oder Landweber-Iterationen bezeichnet. In Abschn. 4.13 werden wir nochmals eine Faltungs-Iteration betrachtet, allerdings formulieren wir dann ein Konvergenzkriterium im Frequenzraum. Jetzt wollen wir ein ganz simples, hinreichendes Kriterium im Ortsraum formulieren, so dass die inverse Funktion h^{-1} existiert und die Iteration konvergiert:

Satz 2.3 *Die Bedingung*

$$\sum_{l=1}^{N-1} |h_l| < |h_0| \quad (2.90)$$

ist hinreichend dafür, dass h^{-1} bezüglich der Faltung existiert. Weiterhin erzeugt die Faltungs-Iteration

$$f^{(j+1)} = \left(\delta - \frac{h}{h_0} \right) * f^{(j)} + \frac{g}{h_0}, \quad j = 0, 1, 2, \dots \quad (2.91)$$

mit einer beliebigen Startfunktion $f^{(0)}$ eine Funktionenfolge $f^{(j)}$, die

$$\lim_{j \rightarrow \infty} f^{(j)} = f \quad \text{und} \quad g = h * f \quad (2.92)$$

erfüllt. Mit $g = \delta$ konvergiert die Folge gegen das inverse Element h^{-1} von h . Mit

$$q = \sum_{i=1}^{N-1} \frac{|h_i|}{|h_0|} \quad (2.93)$$

folgt die *a posteriori* Fehlerabschätzung bezüglich der l_1 oder l_∞ -Norm

$$\|f^{(m)} - f\| \leq \frac{q}{1-q} \|f^{(m-1)} - f^{(m)}\| \leq \frac{q^m}{1-q} \|f^{((0))} - f^{(1)}\|. \quad (2.94)$$

Wenn wir folglich eine örtlich beschränkte PSF h kennen, dann ist für die Konvergenz entscheidend das Verhältnis der Filterkoeffizienten außerhalb des Koordinatenursprungs zu dem Koeffizient im Koordinatenursprung h_0 .

2.4.2 Beschränkte Entfaltungskerne

Eine weitere Möglichkeit zur Entfaltung $h * f = g$ im Ortsraum ist die folgende. Wir betrachten zunächst nur die Invertierung der PSF h , also die Gleichung $h * q = \delta$. Da die Inverse nicht zu existieren braucht, lösen wir diese im Sinne der kleinsten Quadrate, also $\|\delta - h * q\|^2 \rightarrow \text{Minimum}$. Da q aber den gesamten Ortsraum ausfüllen wird, haben wir numerisch immer noch einen großen Aufwand im Ortsraum, daher beschränken wir nun den Ortsraum. Wir setzen von vornherein q so an, dass q eine bestimmte Filtergröße q_B hat und außerhalb setzen wir die Koeffizienten von vornherein zu Null und bestimmen nur die Koeffizienten innerhalb der Filtermaske q_B und lösen nun das numerische Problem $\|\delta - h * q_B\|^2 \rightarrow \text{Minimum}$. Nach aufstellen der Gaußschen Normalengleichungen erhalten wir ein lineares Gleichungssystem von der Dimension der Filtermaske q_B . Wenn wir dieses Gleichungssystem gelöst haben, können wir approximativ entfalten und erhalten als Näherungslösung $\hat{f} = q_B * g$, was numerisch eine gewöhnliche Filterung mit einer beschränkten Filtermaske darstellt.

2.5 FIR und IIR Filter

Ein LSI-Operator ist durch die PSF $L\delta$ eindeutig bestimmt. Je nach endlicher oder unendlicher Ausdehnung der PSF unterscheidet man

- FIR-Filter (**finite impulse response**) und
- IIR-Filter (**infinite impulse response**).

Für das endliche, periodische Bildmodell hat diese Unterscheidung überhaupt keinen Sinn, da die Ausdehnung der PSF niemals unendlich sein kann. Daher können wir für diese Unterscheidung nur das unendliche, diskrete oder analoge Bildmodell heranziehen. Unsere „üblichen“ LSI-Filter, bei denen man die endlichen Masken angibt, sind also immer FIR-Filter $h = L\delta$ und beziehen sich auf die Faltung $y = Lx = x * L\delta = x * h$. Wenn der Input x gegeben ist, dann bekommen wir einfach durch die Faltung des Inputs x mit h den Output y . Die Faltung drückt also eine LSI-Abbildung $x \rightarrow y$ direkt aus. Vielleicht kann man aber die LSI-Abbildung $x \rightarrow y$ auch noch anders beschreiben? Dazu betrachten wir eine sogenannte lineare Differenzengleichung mit konstanten Koeffizienten:

$$\sum_{k=-\infty}^{\infty} a_k y_{n-k} = \sum_{l=-\infty}^{\infty} b_l x_{n-l}. \quad (2.95)$$

Wir nehmen an, die diskreten Funktionen a und b haben endliche viel Funktionswerte ungleich Null:

$$a = (\dots, 0, a_{-N_1}, \dots, a_{-1}, a_0, a_1, \dots, a_{N_2}, 0, \dots) \quad (2.96)$$

$$b = (\dots, 0, b_{-M_1}, \dots, b_{-1}, b_0, b_1, \dots, b_{M_2}, 0, \dots). \quad (2.97)$$

Wir sehen sofort, auf der linken und rechten Seite stehen Faltungen, d. h.

$$a * y = b * x \quad (2.98)$$

beschreibt die Differenzengleichung. Wenn wir nun einen Input x vorgeben, dann müssen wir den Output y erst aus der Differenzengleichung ausrechnen. Wenn diese Lösung eindeutig ist, dann haben wir wieder einen Operator $y = Lx \leftrightarrow x \rightarrow y$ definiert. Die Lösung ist z. B. eindeutig, wenn die Inverse von a bezüglich der Faltung existiert. Welche Eigenschaften hat nun der Operator L , ist er etwa auch ein LSI-Operator?

Satz 2.4 Ist die Lösung y von (2.98) für jedes Input x eindeutig, dann ist der Operator L tatsächlich ein LSI-Operator.

Beweis Der Beweis ist nahezu trivial. Da auf der linken und rechten Seite Faltungen stehen und diese selbst die LSI-Eigenschaft haben, ist auch der Operator L ein LSI-Operator. \square

Dieser Satz impliziert nun Folgendes: Da jeder LSI-Operator durch die PSF $L\delta$ eindeutig bestimmt ist, müssen wir nur diese PSF aus der Differenzengleichung bestimmen. Wir lösen für $x = \delta$ explizit oder numerisch die Differenzengleichung $a * y = b * \delta = b$ und erhalten als Lösung $y_\delta = L\delta$. Die Gesamtlösung ergibt sich dann einfach zu $y = x * y_\delta = x * L\delta$.

Die nächste wichtige Frage ist nun: Sind denn die meisten Differenzengleichungen $a * y = b * x$ auch eindeutig lösbar? Einen Fall der eindeutigen Lösbarkeit kennen wir schon: Ist $a = \delta$, dann erhalten wir unsere bekannte Faltung $y = b * x$ und bei dieser ist der Output y immer eindeutig. Ist dagegen $a \neq \delta$, dann muss das inverse Element von a bezüglich

der Faltung existieren, wovon wir auf keinen Fall ausgehen können. Wir benötigen aber die Eindeutigkeit der Lösung, da sonst L kein Operator ist. Wir müssen also unter den vielen, möglichen Lösungen immer eine „herausgreifen“, dies geschieht durch sogenannte Anfangsbedingungen. Das ist die gleiche Situation wie bei linearen Differentialgleichungen, bei denen man auch Anfangs- oder Randbedingungen stellt. Nun müssen aber diese Bedingungen konsistent zur Linearität und Verschiebungsinvarianz sein, ansonsten kann es sein, dass das System kein LSI-Operator mehr ist. Für einen linearen Operator wissen wir, dass das Nullelement auf das Nullelement abgebildet werden muss, d. h. es muss immer $0 = L0$ gelten. Diese Gleichung darf durch die gestellten Anfangsbedingungen nicht verletzt werden. Dazu betrachten wir ein simples Beispiel. Die Gleichung $y_n - y_{n-1} = x_n$ ist nicht eindeutig lösbar. Wir nehmen als Anfangsbedingung $y_0 = c$, $c \in \mathbb{R}$ und berechnen rekursiv die Lösung für den Input $x = 0$. Dann ist $y_1 = y_0 + 0 = c$, $y_2 = y_1 + 0 = c$ usw., d. h. $y_n = c$, $n \geq 0$. Damit ist $0 = L0$ nur möglich für $c = 0$.

Für die Verschiebungsinvarianz müssen auch ähnliche Überlegungen durchgeführt werden, was wir aber nicht mehr tun werden.

2.6 Schnelle Faltungen im Ortsraum

Wie implementiert man nun schnelle Faltungen im Ortsraum? Dazu hat es nur Sinn das Modell D2 $[I, J]$ zu verwenden. Gegeben sei die 2D-Faltung:

$$g_{m,n} = (h * f)_{m,n} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} h_{i,j} f_{m-i,n-j}. \quad (2.99)$$

Das Filter h heißt *separierbar*, wenn $h_{i,j} = x_i \cdot y_j$ gilt, d. h. wenn sich die Filtermatrix als Tensorprodukt zweier Vektoren darstellen lässt. In diesem Falle gilt dann:

$$\begin{aligned} g_{m,n} &= (h * f)_{m,n} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x_i y_j f_{m-i,n-j} = \sum_{i=0}^{M-1} x_i \left(\sum_{j=0}^{N-1} y_j f_{m-i,n-j} \right) \\ &= \sum_{i=0}^{M-1} x_i z_{m-i,n}. \end{aligned} \quad (2.100)$$

Praktisch heißt dies: Wir filtern zunächst die Spalten von f mit den Filterkoeffizienten y_j und anschließend filtern wir das erhaltene Bild z wieder zeilenweise mit den Filterkoeffizienten x_i . Wenn wir die Filtermaske h als Matrix \mathbf{H} auffassen, die Werte x_i und y_j jeweils in Spaltenvektoren \mathbf{x} und \mathbf{y} schreiben, dann können wir die Separierbarkeit auch als dyadiisches Produkt (Tensorprodukt) $\mathbf{H} = \mathbf{x} \cdot \mathbf{y}^T$ schreiben. Diese Form der Separierbarkeit hatten wir schon in (2.8) und (2.9) behandelt, indem wir unter bestimmten Voraussetzungen eine 2D-Faltung auf die Hintereinanderausführung zweier 1D-Faltungen zurückgeführt hatten. Nun einige Beispiele:

- Zwei bekannte Beispiele für separierbare Filtermasken sind das Mittelwertfilter und das isotrope Gaußfilter:

$$h^M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad (2.101)$$

$$h^G = \begin{pmatrix} 0,11 & 0,32 & 0,11 \\ 0,32 & 1 & 0,32 \\ 0,11 & 0,32 & 0,11 \end{pmatrix} = \begin{pmatrix} 1 \\ 2,95 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0,11 & 0,32 & 0,11 \end{pmatrix}. \quad (2.102)$$

Auf Grund der speziellen Struktur des Mittelwertfilters kann dieses noch effizienter implementiert werden, siehe Abschn. 8.1.2.

- Zur Rauschunterdrückung werden oft Binomialfilter verwendet. Binomialfilter approximieren Gaußfilter im diskreten Gitter und stellen somit eine optimale Quantisierungsstrategie von Gaußfiltrern dar. Bei einem eindimensionalen Filter der Ausdehnung $N + 1$ laufen die Pixelpositionen von 0 bis N , daher ist:

$$P(X = k) = p_k = \binom{N}{k} p^k q^{N-k}. \quad (2.103)$$

Wir setzen $p = 1/2$ und erzeugen die zweidimensionalen Filterkoeffizienten durch Multiplikation, d. h. $p_{i,j} = p_i \cdot p_j$. Damit ist die Filtermaske schon von der Konstruktion her separierbar. Es gilt z. B. für ein 5×5 -Binomialfilter:

$$h^{Bin} = \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix}. \quad (2.104)$$

- Wenn wir nun andere Filter betrachten, wie ermitteln wir, ob diese separierbar sind. Wenn diese es nicht sind, was macht man dann? Dazu benutzen wir den Approximationssatz bezüglich der Singulärwertzerlegung einer Matrix $\mathbf{B} = \sum_{j=1}^r \sigma_j \mathbf{u}_j \cdot \mathbf{v}_j^T$, siehe Abschn. 22.3. Wenn diese Summe nur genau einen Summanden besitzt, dann ist \mathbf{B} separierbar, ansonsten ist \mathbf{B} die gewichtete Summe von separierbaren Filtermasken. Wenn wir den Approximationssatz beachten, dann können wir also die Filtermaske durch eine geringe Anzahl separierbarer Filtermasken approximieren. Dazu sind in Abschn. 22.3 einige Beispiele von Filtermasken aufgeführt. Weiterhin werden im Abschn. 8.1.2 weitere Überlegungen zur schnellen Implementierung ausgeführt.

3.1 Grundlagen und Basissysteme

Das Wort Fouriertransformation(en) werden wir im Folgenden mit FT abkürzen, sie ist benannt nach dem französischen Mathematiker Jean Baptiste Joseph Fourier (1768–1830). In Abhängigkeit vom verwendeten Bildmodell gibt es vier FT, unterscheidet man noch eine oder mehrere unabhängige Veränderliche, so erhöht sich dies noch. Wir werden stets die komplexe Schreibweise der FT benutzen, da sich dadurch viele Eigenschaften elegant und kompakt formulieren lassen, gleichzeitig erweitern wir die Anwendbarkeit auf komplexwertige Funktionen. Komplexwertige Funktionen werden wir insbesondere bei Konturen verwenden. Um die komplexe Schreibweise besser zu verstehen, ist die Eulersche Formel eigentlich die Grundlage, siehe (4.1). Die folgenden Ausführungen dienen dazu, bekannte mathematische Grundlagen aufzuschreiben, die wir im Folgenden benötigen. Wir betrachten einen linearen Raum H mit einem komplexwertigen Skalarprodukt:

$$\langle a, b \rangle, \quad a, b \in H: \tag{3.1}$$

- $\langle a, a \rangle \geq 0, \langle a, a \rangle = 0 \Leftrightarrow a = 0,$
- $\langle a, b \rangle = \overline{\langle b, a \rangle},$
- $\langle \alpha \cdot a + \beta \cdot b, c \rangle = \alpha \langle a, c \rangle + \beta \langle b, c \rangle, \alpha, \beta \in \mathbb{C},$
- $\langle a, \alpha \cdot b + \beta \cdot c \rangle = \overline{\alpha} \langle a, b \rangle + \overline{\beta} \langle a, c \rangle, \alpha, \beta \in \mathbb{C}.$

Bekannte Beispiele für Skalarprodukte sind:

- Der Raum aller über B quadratisch integrierbaren, analogen Funktionen:

$$\langle a, b \rangle = \int_B a(t) \overline{b(t)} dt, \tag{3.2}$$

- Der Vektorraum aller N -dimensionalen Vektoren oder diskreten Funktionen mit N -Stützstellen

$$\langle a, b \rangle = \sum_{i=0}^{N-1} a_i \bar{b}_i. \quad (3.3)$$

Wenn $\langle a, b \rangle$ ein Skalarprodukt ist, dann ist auch $\langle a, b \rangle := c \cdot \langle a, b \rangle$, $c > 0$, $c \in \mathbb{R}$ ein Skalarprodukt. Normen von Elementen aus H kann man viele angeben, aber eine wichtige Norm wird durch das Skalarprodukt $\|a\|^2 = \langle a, a \rangle$ induziert. Zwei Vektoren $a, b \in H$ heißen orthogonal, wenn $\langle a, b \rangle = 0$ erfüllt ist. Der nächste wichtige Begriff ist der einer Basis des Raumes. Wir nennen die Menge $\{\varphi^k, k \in \mathbb{Z}\}$ eine Basis, wenn durch sie der ganze Raum aufgespannt wird. Die orthogonalen Basen spielen dabei die zentrale Rolle, weil bei diesen vieles einfacher wird. Die Basis heißt orthogonal, wenn

$$\langle \varphi^k, \varphi^l \rangle = \delta_{k-l} = \delta_{l-k} \quad (3.4)$$

gilt, wobei δ_n der Einheitsimpuls ist. Wir entwickeln einmal ein Element f nach einer endlichen, orthonormalen Basis

$$f = \sum_{k=1}^{N-1} \alpha_k \varphi^k, \quad (3.5)$$

dann können wir die Koeffizienten α_k sofort mit den Rechenregeln des Skalarproduktes angeben:

$$\begin{aligned} \langle f, \varphi^l \rangle &= \left\langle \sum_{k=0}^{N-1} \alpha_k \varphi^k, \varphi^l \right\rangle = \sum_{k=0}^{N-1} \alpha_k \langle \varphi^k, \varphi^l \rangle \\ &= \sum_{k=0}^{N-1} \alpha_k \delta_{l-k} = (\alpha * \delta)_l = \alpha_l. \end{aligned} \quad (3.6)$$

Die Koeffizienten $\alpha_l = \langle f, \varphi^l \rangle$ nennt man verallgemeinerte Fourierkoeffizienten, die Entwicklung nach der Basis ist dann die verallgemeinerte inverse Fouriertransformation. Nun bilden wir das Skalarprodukt von f mit sich selbst:

$$\begin{aligned} \|f\|^2 &= \langle f, f \rangle = \left\langle \sum_{k=0}^{N-1} \alpha_k \varphi^k, \sum_{l=0}^{N-1} \alpha_l \varphi^l \right\rangle = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha_k \bar{\alpha}_l \delta_{k-l} \\ &= \sum_{k=0}^{N-1} |\alpha_k|^2 = \langle \alpha, \alpha \rangle = \|\alpha\|^2 \end{aligned} \quad (3.7)$$

und erhalten die Grundgleichung

$$\|f\|^2 = \|\alpha\|^2, \quad (3.8)$$

welche als *Parsevalsche Gleichung* bezeichnet wird. Oft wird diese auch *Energiegleichung* genannt, im Prinzip stellt sie die abstrakte Verallgemeinerung des Satzes des Pythagoras im rechtwinkligen Dreieck dar. Wenn keine besonderen Bemerkungen angebracht werden, dann gelten prinzipielle Aussagen für alle Bildmodelle, manchmal gibt es aber kleine Unterschiede, nehmen wir einmal Modell $D1[N]$. Die Basisfunktionen $\varphi^k(n)$ fassen wir als Spaltenvektoren auf und schreiben sie in eine Matrix $\boldsymbol{\varphi}$. Ebenso fassen wir alle Koeffizienten α_l als Elemente einer Spaltenvektors $\boldsymbol{\alpha}$ auf, ebenso die Elemente f_i als Spaltenvektor \mathbf{f} . Dann können wir schreiben:

$$\mathbf{f} = \boldsymbol{\varphi} \cdot \boldsymbol{\alpha} \leftrightarrow \boldsymbol{\alpha} = \overline{\boldsymbol{\varphi}}^T \cdot \mathbf{f}. \quad (3.9)$$

Wir können also für das Modell $D1[N]$ die Hin- und Rücktransformation kompakt in Matrixschreibweise notieren, wobei $\boldsymbol{\varphi}$ eine *unitäre* Matrix ist. Alle bisherigen Ausführungen gelten auch für unendlichdimensionale, orthogonale Basen, d. h. abzählbar viele Basiselemente. Nun kann man dies speziell für analoge Funktionen weitertreiben für überabzählbar viele Basisfunktionen, das i -te Basiselement wird zum τ -ten Basiselement:

$$\varphi^i(t) \leftrightarrow \varphi(t, \tau). \quad (3.10)$$

Die Summe wird zum Integral, so dass wir formal erhalten:

$$f = \sum_i \alpha_i \varphi^i \leftrightarrow f = \int_B \alpha(\tau) \varphi(t, \tau) d\tau. \quad (3.11)$$

Die Basisfunktionen $\varphi(t, \tau)$ nennt man dann *Basiskern* oder *Transformationskern*. Die Koeffizienten $\alpha(\tau)$ erhalten wir dann wieder durch Analogiebetrachtungen:

$$\alpha_i = \langle f, \varphi^i \rangle \leftrightarrow \alpha(\tau) = \int_B f(\eta) \overline{\varphi(\eta, \tau)} d\eta. \quad (3.12)$$

Die Koeffizienten $\alpha(v)$ nennt man nun *Spektrum* bezüglich des Transformationskernes oder der stetigen (überabzählbaren) Basis $\varphi(t, v)$.

Wir setzen nun einmal $f(t)$ in $\alpha(v)$ ein, dann ergibt sich mit einer orthonormalen stetigen Basis:

$$\begin{aligned} \alpha(v) &= \int_B \alpha(\tau) \left\{ \int_B \varphi(\eta, \tau) \overline{\varphi(\eta, v)} d\eta \right\} d\tau = \int_B \alpha(\tau) \delta(v - \tau) d\tau \\ &= (\alpha * \delta)(v) = \alpha(v). \end{aligned} \quad (3.13)$$

Umgedreht, setzen wir nun $\alpha(v)$ in $f(t)$ ein:

$$\begin{aligned} f(t) &= \int_B f(\eta) \left\{ \int_B \varphi(t, \tau) \overline{\varphi(\eta, \tau)} d\tau \right\} d\eta = \int_B f(\eta) \delta(t - \eta) d\eta \\ &= (f * \delta)(t) = f(t). \end{aligned} \quad (3.14)$$

Damit erhalten wir für stetige Basen zwei Orthogonalitätsbedingungen:

$$\delta(t - \eta) = \int_B \varphi(t, \tau) \overline{\varphi(\eta, \tau)} d\tau \quad (3.15)$$

$$\delta(t - \eta) = \int_B \varphi(\tau, t) \overline{\varphi(\tau, \eta)} d\tau. \quad (3.16)$$

Die zweite Bedingung entspricht sonst der üblichen Forderung. Wir können diese beiden Bedingungen als „Zeilenorthogonalität“ und „Spaltenorthogonalität“ interpretieren. Nehmen wir als Basiselemente einmal n -dimensionale Vektoren, die wir als Spaltenvektoren aufschreiben, dann ist Bedingung zwei die Spaltenorthogonalität, Bedingung eins die Zeilenorthogonalität. Der Unterschied besteht aber darin, dass in diesem Falle eine Bedingung reicht, weil die andere stets daraus folgt. Nehmen wir einmal die Spaltenorthogonalität einer quadratischen Matrix A an, dann gilt $A^T A = E$. Da die inverse Matrix existiert gilt weiter $A^T = A^{-1}$ und damit wieder $A A^T = E$, was aber die Zeilenorthogonalität darstellt. Bei stetigen Basen können wir dies nicht einfach so aufschreiben, dann haben wir zwei Orthogonalitätsbedingungen, in diesem Falle nennt man den Kern $\varphi(t, \tau)$ *selbstadjungiert*.

Mit diesen beiden Bedingungen und durch Einsetzen folgt sofort

$$\int_B f_1(t) f_2(t) dt = \int_B \alpha_1(\nu) \alpha_2(\nu) d\nu \quad (3.17)$$

und damit speziell

$$\int_B |f(t)|^2 dt = \int_B |\alpha(\nu)|^2 d\nu, \quad (3.18)$$

welches die Parsevalsche Gleichung (Energiegleichung, verallgemeinerter Pythagoras) für stetige Basen darstellt. Nun einige Beispiele für Integraltransformationen:

Beispiel 1 Mit $\varphi(t, \nu) = e^{2\pi i \nu t}$ erhalten wir einen selbstadjungierten Kern, der die klassische Fourier-Integraltransformation beschreibt. Auf Grund der totalen Symmetrie der Argumente t und ν fallen die beiden Orthogonalitätsbedingungen zu einer zusammen. Folglich ist

$$f(t) = \int_{-\infty}^{+\infty} \alpha(\nu) e^{2\pi i \nu t} d\nu, \quad (3.19)$$

$$\alpha(\nu) = \int_{-\infty}^{+\infty} f(\tau) \overline{e^{2\pi i \nu \tau}} d\tau = \int_{-\infty}^{+\infty} f(\tau) e^{-2\pi i \nu \tau} d\tau \quad (3.20)$$

die Fourier-Integraltransformation.

Beispiel 2 Eine andere Transformation ist die **Laplace-Transformation**:

$$\alpha(p) = \int_0^{+\infty} f(t) e^{-pt} dt \quad (3.21)$$

$$f(t) = \frac{1}{2\pi i} \int_{-\mu-i\infty}^{-\mu+i\infty} \alpha(p) e^{pt} dp. \quad (3.22)$$

Das diskrete Analogon zur Laplacetransformation ist die **z-Transformation**, siehe [53].

Beispiel 3 Ein weiteres Beispiel einer Integraltransformation ist die **Hilbert-Transformation**:

$$\beta(s) = V.P. \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{f(t)}{s-t} dt \quad (3.23)$$

$$f(t) = V.P. -\frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{\beta(s)}{t-s} ds. \quad (3.24)$$

Diese spielt in der Bildverarbeitung eine große Rolle bei den Quadratur-Filttern. Wie man sieht (siehe Abschn. 2.1.1), lässt sich diese mit der Faltung leicht ausdrücken:

$$\beta(s) = \frac{1}{\pi} \left(f(t) * \frac{1}{t} \right)(s) \Leftrightarrow f(t) = -\frac{1}{\pi} \left(\beta(s) * \frac{1}{s} \right)(t). \quad (3.25)$$

3.2 Analoge Fouriertransformation im endlichen Intervall (AFT)

Wir betrachten das Bildmodell $A1[0, X]$, demnach analoge Funktionen $f(x)$ im endlichen Intervall $[0, X]$. Als k -te Basisfunktion wählen wir

$$\varphi^{(k)}(x) = \frac{1}{\sqrt{X}} e^{2\pi i k \frac{x}{X}}, \quad k \in \mathbb{Z}. \quad (3.26)$$

Als Skalarprodukt benutzen wir

$$\langle f, g \rangle = \int_0^X f(\tau) \overline{g(\tau)} d\tau. \quad (3.27)$$

Man kann zeigen, dass diese Basisfunktionen mit dem Skalarprodukt ein vollständiges Orthonormalsystem bilden, daher können wir wie bei allgemeinen Basisentwicklungen nach einem Orthonormalsystem sofort aufschreiben:

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \alpha_k e^{+2\pi i k \frac{x}{X}} \leftrightarrow \alpha_k = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx. \quad (3.28)$$

Dies ist die übliche Darstellung der Fourierreihenentwicklung (AFT) in komplexer Schreibweise. Es sei betont: einer analogen Funktion $f(x)$ wird ein diskretes Spektrum $\alpha_k, k \in \mathbb{Z}$ zugeordnet:

$$f(x) \leftrightarrow (\alpha_{-\infty}, \dots, \alpha_{-1}, \alpha_0, \alpha_1, \dots, \alpha_{+\infty}). \quad (3.29)$$

Wir können also formal das Spektrum selbst nicht wieder transformieren, weil es keine analoge Funktion darstellt. Von Symmetrie kann also keine Rede sein, das Einzige, was symmetrisch ist, ist der Faktor $\frac{1}{\sqrt{X}}$. Oft werden in der Literatur diesbezüglich unsymmetrische Formen angegeben. Betrachten wir einmal die Basisfunktionen $\varphi^{(k)}(x) = e^{2\pi i k \frac{x}{X}}$, dann bilden diese bezüglich des Skalarproduktes $\langle f, g \rangle = \int_0^X f(x) \overline{g(x)} dx$ nur ein Orthogonalsystem, die Längen der Basisfunktionen sind nicht zu Eins normiert. Benutzen wir dagegen das Skalarprodukt $\langle f, g \rangle = \frac{1}{X} \int_0^X f(x) \overline{g(x)} dx$, dann bilden sie ein Orthonormalsystem. Der Faktor vor den Basisfunktionen muss also immer zu dem Faktor vor dem Skalarprodukt „passen“. Damit ergibt sich die unsymmetrische AFT:

$$f(x) = \sum_{k=-\infty}^{+\infty} \alpha_k e^{2\pi i k \frac{x}{X}} \leftrightarrow \alpha_k = \langle f, \varphi^{(k)} \rangle = \frac{1}{X} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx. \quad (3.30)$$

Benutzen wir andererseits die Basisfunktionen $\varphi^{(k)}(x) = \frac{1}{X} e^{2\pi i k \frac{x}{X}}$ mit dem Skalarprodukt $\langle f, g \rangle = X \int_0^X f(x) \overline{g(x)} dx$, dann bilden diese ein Orthonormalsystem. Damit ergibt sich eine andere unsymmetrische AFT:

$$f(x) = \frac{1}{X} \sum_{k=-\infty}^{+\infty} \alpha_k e^{2\pi i k \frac{x}{X}} \leftrightarrow \alpha_k = \langle f, \varphi^{(k)} \rangle = \int_0^X f(x) e^{-2\pi i k \frac{x}{X}}. \quad (3.31)$$

Im Folgenden benutzen wir immer die symmetrische FT mit dem „gewöhnlichen“ Skalarprodukt.

Wir bilden nun $\overline{\alpha_{-k}}$ und setzen voraus, dass f eine reelle Funktion ist:

$$\overline{\alpha_{-k}} = \frac{1}{\sqrt{X}} \int_0^X \overline{f(x)} e^{+2\pi i (-k) \frac{x}{X}} dx = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx = \alpha_k. \quad (3.32)$$

Das heißt, dass für reelle Funktionen $f \alpha_k = \overline{\alpha_{-k}}$ oder $\alpha_{-k} = \overline{\alpha_k}$ gilt. Mit diesem Wissen versuchen wir nun die reelle Fouriertransformation einer reellen Funktion in nicht komplexer Schreibweise herzuleiten. In der Reihenentwicklung spielen also für reelle Funktionen die Doppelterme

$$\gamma_k = \frac{1}{\sqrt{X}} (\alpha_k e^{2\pi i k \frac{x}{X}} + \overline{\alpha_k} e^{-2\pi i k \frac{x}{X}}) \quad (3.33)$$

die bestimmende Rolle. Mit $e^{i\varphi} = \cos \varphi + i \sin \varphi$ und $\alpha_k = a_k + ib_k$ ergibt sich:

$$\gamma_k = \frac{1}{\sqrt{X}} \left(2a_k \cos \left(2\pi k \frac{x}{X} \right) - 2b_k \sin \left(2\pi k \frac{x}{X} \right) \right). \quad (3.34)$$

Damit können wir die Fourierreihe aufschreiben:

$$f(x) = \frac{1}{\sqrt{X}} \alpha_0 + \sum_{k=1}^{\infty} \left(\frac{2}{\sqrt{X}} a_k \cos \left(2\pi k \frac{x}{X} \right) - \frac{2}{\sqrt{X}} b_k \sin \left(2\pi k \frac{x}{X} \right) \right). \quad (3.35)$$

Für die inverse FT gilt analog:

$$\begin{aligned} \alpha_k &= a_k + ib_k = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} \\ &= \frac{1}{\sqrt{X}} \int_0^X f(x) \left(\cos \left(2\pi k \frac{x}{X} \right) - i \sin \left(2\pi k \frac{x}{X} \right) \right). \end{aligned} \quad (3.36)$$

Für den Real- und Imaginärteil können wir nun getrennt ablesen:

$$\begin{aligned} \alpha_0 &= a_0, \quad a_k = \frac{1}{\sqrt{X}} \int_0^X f(x) \cos \left(2\pi k \frac{x}{X} \right) dx, \\ b_k &= -\frac{1}{\sqrt{X}} \int_0^X f(x) \sin \left(2\pi k \frac{x}{X} \right) dx \quad \forall k \neq 0. \end{aligned} \quad (3.37)$$

Durch eine letzte simple Substitution erhalten wir als Endresultat die Form:

$$f(x) = \frac{1}{\sqrt{X}} \left[\frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos \left(2\pi k \frac{x}{X} \right) + b_k \sin \left(2\pi k \frac{x}{X} \right) \right) \right] \quad (3.38)$$

mit

$$\begin{aligned} a_0 &= \frac{2}{\sqrt{X}} \int_0^X f(x) dx, \quad a_k = \frac{2}{\sqrt{X}} \int_0^X f(x) \cos \left(2\pi k \frac{x}{X} \right) dx, \\ b_k &= \frac{2}{\sqrt{X}} \int_0^X f(x) \sin \left(2\pi k \frac{x}{X} \right) dx \quad \forall k \neq 0. \end{aligned} \quad (3.39)$$

Wenn nicht ausdrücklich darauf hingewiesen wird, wählen wir auch bei reellen Funktionen immer die komplexe Schreibweise der FT. In der Physik ist es nicht üblich die Darstellung (3.38) zu benutzen, da diese die Addition einer sin- und cos-Funktion darstellt und somit etwas unübersichtlich ist. Für diese Zwecke ist es besser eine einzige sin- oder cos-Funktion mit einer Phasenverschiebung zu benutzen. Wir benutzen folglich ein *Sinusoid* oder eine *sinusoidale* Größe. Mit simplen Umrechnungsformeln erhalten wir dann

$$f(x) = \frac{1}{\sqrt{X}} \left[\frac{a_0}{2} + \sum_{k=1}^{\infty} A_k \sin \left(2\pi k \frac{x}{X} + \varphi_k \right) \right] \quad (3.40)$$

mit den Amplituden $A_k = \sqrt{a_k^2 + b_k^2}$ und den Phasen $\tan \varphi_k = \frac{a_k}{b_k}$. Ebenso kann man die cos-Funktion benutzen, da diese auch sinusoidal ist, dann ist

$$f(x) = \frac{1}{\sqrt{X}} \left[\frac{a_0}{2} + \sum_{k=1}^{\infty} A_k \cos \left(2\pi k \frac{x}{X} + \varphi_k \right) \right] \quad (3.41)$$

mit den Amplituden $A_k = \sqrt{a_k^2 + b_k^2}$ und den Phasen $\tan \varphi_k = -\frac{a_k}{b_k}$. Die FT für das Modell A2[X, Y] ist wieder simpel aufzuschreiben:

$$f(x, y) = \frac{1}{\sqrt{X}} \frac{1}{\sqrt{Y}} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \alpha_{k_1, k_2} e^{+2\pi i (k_1 \frac{x}{X} + k_2 \frac{y}{Y})} \quad (3.42)$$

$$\alpha_{k_1, k_2} = \frac{1}{\sqrt{X}} \frac{1}{\sqrt{Y}} \int_0^X \int_0^Y f(x, y) e^{-2\pi i (k_1 \frac{x}{X} + k_2 \frac{y}{Y})} dx. \quad (3.43)$$

3.3 Die endliche diskrete Fouriertransformation (DFT)

Gegeben sei eine diskrete Funktion $f_k, k = 0, 1, \dots, N-1$, die Funktionswerte können komplex oder reell sein. Wir stellen uns dabei gedanklich eine Abtastung einer analogen Funktion an N Abtaststellen vor. Die Abtaststellen nummerieren wir einfach neu durch mit den Indizes $0, 1, 2, \dots, N-1$. Im gleichen Koordinatensystem entspricht das einer Stauchung oder Streckung der Funktion bezüglich der x -Koordinaten. Als k -te diskrete Basisfunktion wollen wir

$$\varphi_n^{(k)} := \varphi^{(k)}(n) = \frac{1}{\sqrt{N}} e^{2\pi i k \frac{n}{N}} \quad (3.44)$$

verwenden. Mit dem Skalarprodukt

$$\langle f, g \rangle = \sum_{l=0}^{N-1} f_l \overline{g_l} \quad (3.45)$$

bilden diese ein Orthonormalsystem. Daher schreiben wir:

$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} \leftrightarrow \alpha_k = \langle f, \varphi^{(k)} \rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_j e^{-2\pi i k \frac{j}{N}}. \quad (3.46)$$

Wir wollen nun die DFT von einem anderen Standpunkt aus betrachten. Wir betrachten das Modell AFT und approximieren das Integral durch eine Quadraturformel, z. B. die einfache Rechteckregel:

$$\int_0^X g(x) dx \approx h \cdot \sum_{l=0}^{N-1} g\left(l \cdot \frac{X}{N}\right) = \frac{X}{N} \frac{1}{\sqrt{X}} \sum_{l=0}^{N-1} f(x_l) e^{-2\pi i k \frac{l \cdot X}{N} \cdot \frac{1}{X}} = C \cdot \alpha_k. \quad (3.47)$$

Bis auf einen Faktor C erhalten wir gerade die diskreten Fourierkoeffizienten. Wir sehen, dass N diskreten Funktionswerten f_l genau N komplexe Fourierkoeffizienten α_l zugeordnet sind:

$$(f_0, f_1, f_2, \dots, f_{N-1}) \leftrightarrow (\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{N-1}). \quad (3.48)$$

Dies bedeutet aber nicht, dass ein bestimmter Funktionswert einem bestimmten Fourierkoeffizienten zugeordnet ist, sondern nur, dass wieder eine diskrete Funktion mit N (komplexen) Funktionswerten entsteht. Dies hat zur Folge, dass man dann die Fourierkoeffizienten selbst mit der DFT transformieren kann. Die Verallgemeinerung der DFT auf Funktionen mit zwei unabhängigen Variablen (oder mehreren) ist wieder recht einfach:

$$f_{m,n} = \frac{1}{\sqrt{M}} \frac{1}{\sqrt{N}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \alpha_{k,l} e^{+2\pi i (m \frac{k}{M} + n \frac{l}{N})} \quad (3.49)$$

$$\alpha_{k,l} = \frac{1}{\sqrt{M}} \frac{1}{\sqrt{N}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_{m,n} e^{-2\pi i (k \frac{m}{M} + l \frac{n}{N})}. \quad (3.50)$$

Da bei der Transformation und der inversen Transformation der Faktor immer $\frac{1}{\sqrt{N}}$ ist, sprechen wir wieder von einer symmetrischen Form. Oft werden (wie bei der AFT) unsymmetrische Formen in der Literatur angegeben. Betrachten wir einmal die Basisfunktionen $\varphi_n^{(k)} := \varphi^{(k)}(n) = e^{2\pi i k \frac{n}{N}}$, dann bilden diese bezüglich des Skalarproduktes $\langle f, g \rangle = \sum_{l=0}^{N-1} f_l \overline{g_l}$ nur ein Orthogonalsystem, die Normen der Basisfunktionen sind nicht zu Eins normiert. Benutzen wir dagegen das Skalarprodukt $\langle f, g \rangle = \frac{1}{N} \sum_{l=0}^{N-1} f_l \overline{g_l}$, dann bilden sie ein Orthonormalsystem. Der Faktor vor den Basisfunktionen muss also immer zu dem Faktor vor dem Skalarprodukt „passen“. Damit ergibt sich eine unsymmetrische DFT:

$$f_n = \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} \leftrightarrow \alpha_k = \langle f, \varphi^{(k)} \rangle = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi i k \frac{j}{N}}. \quad (3.51)$$

Benutzen wir andererseits die Basisfunktionen $\varphi_n^{(k)} := \varphi^{(k)}(n) = \frac{1}{N} e^{2\pi i k \frac{n}{N}}$ mit dem Skalarprodukt $\langle f, g \rangle = N \sum_{l=0}^{N-1} f_l \overline{g_l}$, dann bilden diese ein Orthonormalsystem. Damit ergibt sich eine andere unsymmetrische DFT:

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{n}{N}} \leftrightarrow \alpha_k = \langle f, \varphi^{(k)} \rangle = \sum_{j=0}^{N-1} f_j e^{-2\pi i k \frac{j}{N}}. \quad (3.52)$$

Im Folgenden benutzen wir immer die symmetrische DFT (3.46) mit dem „normalen“ Skalarprodukt.

3.4 Die Fourier-Integraltransformation (IFT)

Die IFT wird häufig in der Physik, Elektrotechnik und Stochastik benutzt. In der Stochastik ist das Integralspektrum im Wesentlichen die sogenannte *charakteristische Funktion* einer Zufallsvariablen.

Wir betrachten das analoge Bildmodell $A1[-\infty, +\infty]$. Da die harmonischen Funktionen im Intervall $[0, X]$ ein Orthonormalsystem bilden und dabei X beliebig groß sein kann, dann dürfte es doch für das unendliche Intervall auch ohne Schwierigkeiten gehen. Hier liegt der Trugschluss, auf diesem Intervall bilden die harmonischen Funktionen kein Orthonormalsystem mehr, wir müssen uns etwas anderes einfallen lassen. Dazu gibt es zwei Ideen:

- a) Wir betrachten formal den Grenzübergang $X \rightarrow \infty$ und untersuchen was „passiert“.
- b) Wir versuchen ein übliches, abzählbar unendliches Basissystem auf eines mit „überabzählbar vielen Basisfunktionen“ zu erweitern.

3.4.1 Integraltransformation als Grenzwert

Wir betrachten die FT für das Bildmodell $A1[X]$ und beachten, dass $f(x)$ periodisch mit X ist:

$$\alpha_k = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx = \frac{1}{\sqrt{X}} \int_{-\frac{X}{2}}^{+\frac{X}{2}} f(x) e^{-2\pi i k \frac{x}{X}} dx. \quad (3.53)$$

Wir substituieren nun $v_k = \frac{k}{X}$ und bezeichnen mit $\alpha(v_k)$:

$$\alpha(v_k) = \sqrt{X} \cdot \alpha_k = \int_{-\frac{X}{2}}^{+\frac{X}{2}} f(x) e^{-2\pi i v_k x} dx. \quad (3.54)$$

Wenn wir nun $\Delta v_k := v_{k+1} - v_k = \frac{1}{X}$ setzen, dann geht die Fourierreihe

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \alpha_k e^{+2\pi i k \frac{x}{X}} \quad (3.55)$$

über in:

$$f(x) = \sum_{k=-\infty}^{\infty} \alpha(v_k) e^{+2\pi i v_k x} \cdot \Delta v_k. \quad (3.56)$$

Jetzt lassen wir $X \rightarrow \infty$ laufen, damit ist aber $\Delta v_k \rightarrow 0$ und wir erhalten

$$f(x) = \lim_{\Delta v_k \rightarrow 0} \left(\sum_{k=-\infty}^{\infty} \alpha(v_k) e^{+2\pi i v_k x} \cdot \Delta v_k \right) = \int_{-\infty}^{+\infty} \alpha(v) e^{+2\pi i v x} dv. \quad (3.57)$$

als Grenzwert das uneigentliche Riemann-Integral. Genauso ist dann:

$$\alpha(v) = \lim_{X \rightarrow \infty} \alpha(v_k) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i v x} dx. \quad (3.58)$$

Damit haben wir formal die Fourier-Integraltransformation (IFT) hergeleitet:

$$f(x) = \int_{-\infty}^{+\infty} \alpha(v) e^{+2\pi i v x} dv \leftrightarrow \alpha(v) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i v x} dx. \quad (3.59)$$

Man beachte: jetzt haben wir analoge Frequenzen v . Diese Form der IFT ist symmetrisch bezüglich der Vorfaktoren und auch symmetrisch in dem Sinne, dass eine analoge Funktion wieder in eine analoge Funktion überführt wird:

$$f(x) \leftrightarrow \alpha(v). \quad (3.60)$$

Man kann also die Fourier-Transformierte $\alpha(v)$ selbst wieder transformieren. Wie bei der AFT und der DFT gibt es auch hier bezüglich der Faktoren unsymmetrische Formen. In der Physik wird oft mit der Kreisfrequenz $\omega = 2\pi v$ anstatt der „normalen“ Frequenz v gearbeitet. Es ist dann:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \alpha\left(\frac{\omega}{2\pi}\right) e^{+i\omega x} d\omega. \quad (3.61)$$

Bezeichnen wir nun mit $\alpha^*(\omega) := \frac{1}{\sqrt{2\pi}} \alpha\left(\frac{\omega}{2\pi}\right)$ und lassen anschließend den * wieder weg, so erhalten wir die symmetrische IFT mit der Kreisfrequenz ω :

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \alpha(\omega) e^{+i\omega x} d\omega \leftrightarrow \alpha(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx. \quad (3.62)$$

Substituieren wir dagegen $\alpha^*(\omega) := \frac{1}{2\pi} \alpha(\frac{\omega}{2\pi})$, so erhalten wir eine unsymmetrische IFT mit der Kreisfrequenz ω :

$$f(x) = \int_{-\infty}^{+\infty} \alpha(\omega) e^{+i\omega x} d\omega \leftrightarrow \alpha(\omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx. \quad (3.63)$$

Substituieren wir schließlich $\alpha^*(\omega) := \alpha(\frac{\omega}{2\pi})$, so erhalten wir die zweite unsymmetrische Form der IFT:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \alpha(\omega) e^{+i\omega x} d\omega \leftrightarrow \alpha(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx. \quad (3.64)$$

Da bezüglich einer „überabzählbaren Basis“ ein Skalarprodukt mit Vorfaktoren verwendet werden muss (analog der Abschnitte AFT und DFT), sind die unsymmetrischen Formen immer etwas mit Vorsicht zu behandeln, wenn es um die genauen Vorfaktoren geht.

3.4.2 Integraltransformation mit stetigen Basen

Man kann die IFT auch einführen, indem wir eine überabzählbare Basis betrachten, siehe Abschn. 3.1. Wir benötigen also selbstadjungierte Kerne. Die Basisfunktion als Kern $\varphi(x, v)$ hatten wir interpretiert als die v -te Basisfunktion an der Stelle x . Man kann nun zeigen, dass

$$\varphi(x, v) = e^{+2\pi i vx}, \quad \varphi(x, \omega) = \frac{1}{\sqrt{2\pi}} e^{+i\omega x} \quad (3.65)$$

bezüglich des Skalarproduktes $\int_{-\infty}^{\infty} f(x) \overline{g(x)} dx$ selbstadjungierte Basiskerne darstellen, d. h. konkret:

$$\delta(t - \tau) = \int_{-\infty}^{\infty} \varphi(\eta, t) \overline{\varphi(\eta, \tau)} d\eta = \int_{-\infty}^{\infty} e^{+2\pi i \eta(t - \tau)} d\eta. \quad (3.66)$$

Obige Bedingung impliziert:

$$\int_{-\infty}^{\infty} e^{+2\pi i \eta x} d\eta = 0 \quad \forall x \neq 0. \quad (3.67)$$

Diese beiden Basiskerne führen auf die beiden symmetrischen IFT (3.59) und (3.62). Im Abschn. 3.1 hatten wir zwei Orthonormalitätsbedingungen für stetige Basen notiert, auf Grund der Symmetrie der Basiskerne fallen diese bei der IFT zu einer Bedingung zusammen. Wenn wir vor den Kernen den Faktor ändern, dann müssen wir wieder das Skalarprodukt so ändern, dass die Faktoren zueinander „passen“, siehe auch AFT und DFT. Dies führt dann zu den beiden unsymmetrischen IFT (3.63) und (3.64).

3.5 Die unendliche diskrete Fouriertransformation (UDFT)

Wir betrachten nun das Bildmodell $D1[-\infty, +\infty]$, d. h. wir betrachten eine diskrete Funktion mit dem Definitionsbereich von $-\infty$ bis $+\infty$ und wollen dafür eine unendliche diskrete FT herleiten. Dazu schreiben wir zunächst noch einmal die FT für das Modell $A1[X]$ auf:

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \alpha_k e^{+2\pi i k \frac{x}{X}} \leftrightarrow \alpha_k = \frac{1}{\sqrt{X}} \int_0^X f(x) e^{-2\pi i k \frac{x}{X}} dx. \quad (3.68)$$

Da die Funktion $f(x)$ analog ist, aber das Spektrum α_k diskret, vertauschen wir jetzt einfach formal die Rollen und beachten, dass X die Periode für $f(x)$ ist:

$$\alpha(\omega) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} g_k e^{+2\pi i k \frac{\omega}{X}} \leftrightarrow g_k = \frac{1}{\sqrt{X}} \int_{-\frac{X}{2}}^{\frac{X}{2}} \alpha(\omega) e^{-2\pi i k \frac{\omega}{X}} d\omega. \quad (3.69)$$

Da die Periode nicht vorgegeben ist, wählen wir $X = 2\pi$ und erhalten:

$$\alpha(\omega) = \frac{1}{\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} g_k e^{+ik\omega} \leftrightarrow g_k = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \alpha(\omega) e^{-ik\omega} d\omega. \quad (3.70)$$

Eigentlich wären wir schon fertig. Die Hin-und Rücktransformation besitzen aber nicht die Form der anderen FT, deshalb substituieren wir $\alpha_d(\omega) := \sqrt{2\pi} \cdot \alpha(-\omega)$ und erhalten schließlich die Endform der UDFT:

$$\alpha_d(\omega) = \sum_{k=-\infty}^{\infty} g_k e^{-ik\omega} \leftrightarrow g_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} \alpha_d(\omega) e^{+ik\omega} d\omega. \quad (3.71)$$

Diese Form der FT benötigen wir beim Beweis des Abtasttheorems für das Bildmodell $A1[-\infty, +\infty]$.

Bemerkung: Die UDFT ist auch die **z-Transformation** für $z = e^{i\omega}$, deshalb findet man oft die Bezeichnung $H(e^{i\omega})$ anstatt $\alpha_d(\omega)$. Es sei erinnert, die z-Transformation ist das diskrete Analogon zur Laplace-Transformation.

Es ist natürlich nun sehr leicht, die UDFT für 2D-Bilder aufzuschreiben:

$$\alpha_d(\omega_1, \omega_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} g_{k_1, k_2} e^{-ik_1 \omega_1 - ik_2 \omega_2} \quad (3.72)$$

$$g_{k_1, k_2} = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \alpha_d(\omega_1, \omega_2) e^{+ik_1 \omega_1 + ik_2 \omega_2} d\omega_1 d\omega_2. \quad (3.73)$$

Man beachte noch einmal: Das Spektrum einer diskreten Funktion für dieses Bildmodell ist eine analoge Funktion, diesbezüglich liegt also keine Symmetrie vor. Außerdem ist das Spektrum in der Form (3.71) **periodisch** mit der Periode 2π .

3.6 Fourier-Mellin-Transformation (FMT)

Eine der Fouriertransformation sehr ähnliche Transformation ist die *Mellin* und die *Fourier-Mellin* Transformation, benannt nach dem finnischen Mathematiker Hjalmar Mellin. Sie spielt eine Rolle im Zusammenhang mit der *log-polar*-Transformation. Da Polarkoordinaten verwendet werden, benutzen wir auch die Variable r für den Radius und θ für den Winkel. Die Mellin-Transformation der Funktion g lautet:

$$z(u) = \int_0^\infty g(r)r^{u-1}dr, \quad r \geq 0, u \text{ komplex.} \quad (3.74)$$

Da es Konvergenzprobleme des Integrals gibt, wird auch häufig u als rein imaginär angenommen:

$$z(w) = \int_0^\infty g(r)r^{iw-1}dr, \quad r \geq 0, w \text{ reell.} \quad (3.75)$$

Wir substituieren einmal in (3.75) $r = e^{-2\pi r'}$, dann ergibt sich:

$$z(w) = \int_0^\infty g'(r')e^{-2\pi iwr'}dr' \quad \text{mit } g'(r') = -2\pi g(e^{-2\pi r'}). \quad (3.76)$$

Damit sehen wir nun die klare Ähnlichkeit zur Fourier-Transformation: Die Mellin-Transformation ist die Fourier-Integraltransformation über einer exponentiellen Skala oder invers über einer logarithmischen Skala. Die Notation r soll an den Radius in Polarkoordinaten erinnern.

Nun wollen wir zur Fourier-Mellin-Transformation übergehen, indem wir die Fourier-Transformation bezüglich des Polarwinkels $0 \leq \theta \leq 2\pi$ betrachten. Dazu benutzen wir die unsymmetrische Fouriertransformation AFT aus Abschn. 3.2, (3.30):

$$\alpha_k = \frac{1}{X} \int_0^X f(x)e^{-2\pi ik\frac{x}{X}}dx \rightarrow \alpha_k = \frac{1}{2\pi} \int_0^{2\pi} f(\theta)e^{-ik\theta}d\theta. \quad (3.77)$$

Hängt nun $f(\theta)$ noch von dem Polarradius r ab, dann hängen die Fourierreffizienten auch von r ab:

$$\alpha_k(r) = \frac{1}{2\pi} \int_0^{2\pi} f(r, \theta)e^{-ik\theta}d\theta. \quad (3.78)$$

Nun wenden wir die Mellin-Transformation (3.75) auf die Funktion $\alpha_k(r)$ an, d. h. $g(r) = \alpha_k(r)$:

$$z(k, w) = \frac{1}{2\pi} \int_0^\infty \int_0^{2\pi} f(r, \theta) e^{-ik\theta} r^{iw-1} d\theta dr. \quad (3.79)$$

Die Rücktransformation ergibt sich dann zu:

$$f(r, \theta) = \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} z(k, w) e^{ik\theta} r^{-iw} dw. \quad (3.80)$$

Das Transformationspaar (3.79) und (3.80) nennt man Fourier-Mellin-Transformation. Die Rücktransformation (3.80) heißt auch *log polar circular harmonic decomposition of f*. Die FMT werden wir praktisch bei der signalorientierten Registrierung von Bildern einsetzen, siehe Abschn. 19.8.4.

3.7 Hilbert-Transformation (HIT)

Die Hilberttransformation ist eine Integraltransformation, die in der Bildverarbeitung eine große Rolle spielt. Wir definieren sie hier nur für reelle Funktionen f und nennen sie $\beta_f(s)$:

$$\beta_f(s) = \frac{1}{\pi} (\text{V.P.}) \int_{-\infty}^{\infty} \frac{f(t)}{s-t} dt \leftrightarrow f(t) = -\frac{1}{\pi} (\text{V.P.}) \int_{-\infty}^{\infty} \frac{\beta_f(s)}{t-s} ds. \quad (3.81)$$

(V.P.) bedeutet, dass die Integrale als Cauchyscher Hauptwert zu verstehen sind. Damit kann man allgemein $\beta = \text{HIT}(f)$ und $f = \text{HIT}^{-1}(\beta) = -\text{HIT}(\beta)$ schreiben. Weiterhin sieht man, dass die HIT sich als Faltung schreiben lässt:

$$\beta_f(s) = \frac{1}{\pi} f(s) * \frac{1}{s} \leftrightarrow f(t) = -\frac{1}{\pi} \beta(t) * \frac{1}{t}. \quad (3.82)$$

Da sich die Hilberttransformation als Faltung schreiben lässt, gilt trivialerweise:

$$\text{HIT}(f * g) = f * \text{HIT}(g) = \text{HIT}(f) * g. \quad (3.83)$$

Es sei nun $\alpha_f(\omega)$ das Spektrum von f und $\alpha_\beta(\omega)$ das Spektrum der Hilberttransformierten β , dann gilt folgende einfache Beziehung:

$$\alpha_\beta(\omega) = -i \cdot \text{sign}(\omega) \alpha_f(\omega). \quad (3.84)$$

Beide Spektren unterscheiden sich nur um eine Phasenverschiebung von $\pm \frac{\pi}{2}$. Man könnte demnach das Spektrum von f um $\pm \frac{\pi}{2}$ phasenverschieben, anschließend zurücktransformieren und hätte die Hilberttransformierte ausgerechnet. Für das Spektrum $\alpha_f(\omega) =$

$R(\omega) + iI(\omega)$ reeller Funktionen gilt $\alpha_f(\omega) = \overline{\alpha_f(-\omega)}$ und folglich $R(-\omega) = R(\omega)$ und $I(-\omega) = -I(\omega)$. Mit diesen Beziehungen können wir leicht

$$f(t) = \frac{1}{\pi} \int_0^\infty R(\omega) \cos \omega t d\omega - \frac{1}{\pi} \int_0^\infty I(\omega) \sin \omega t d\omega \quad (3.85)$$

ausrechnen. Wenn wir nun die Beziehung (3.84) nutzen, dann können wir ebenso

$$\beta_f(s) = \frac{1}{\pi} \int_0^\infty I(\omega) \cos \omega s d\omega + \frac{1}{\pi} \int_0^\infty R(\omega) \sin \omega s d\omega \quad (3.86)$$

ausrechnen. Dieses Grundprinzip der beiden Gleichungen (3.85) und (3.86) nennt man *trigonometrische Konjugation*. Die Hilberttransformation werden wir noch bei Energiefiltern anwenden, siehe Abschn. 6.5. Exemplarisch seien nun einige Beispiele von Funktionen und deren Hilberttransformierte angegeben:

$$\begin{aligned} f(t) = \text{const} &\Leftrightarrow \beta_f(s) = 0 \\ f(t) = \delta(t) &\Leftrightarrow \beta_f(s) = \frac{1}{\pi s} \\ f(t) = \text{rect}\left(\frac{t}{2t_0}\right) &\Leftrightarrow \beta_f(s) = -\frac{1}{\pi} \ln \left| \frac{s-t_0}{s+t_0} \right| \\ f(t) = \frac{1}{1+t^2} &\Leftrightarrow \beta_f(s) = \frac{s}{1+s^2} \\ f(t) = \text{sinc}(t) &\Leftrightarrow \beta_f(s) = \frac{1-\cos s}{s} \\ f(t) = \sin \omega t &\Leftrightarrow \beta_f(s) = -\cos \omega s \\ f(t) = \cos \omega t &\Leftrightarrow \beta_f(s) = \sin \omega s. \end{aligned} \quad (3.87)$$

In der Funktionentheorie heißt ein komplexes Signal $z(t)$ analytisch, wenn der Imaginärteil die Hilberttransformierte des Realteils ist:

$$z(t) = f(t) + i\beta_f(t). \quad (3.88)$$

Dies bedeutet:

$$\beta_f(t) = \text{HIT}(f(t)) \Leftrightarrow f(t) = -\text{HIT}(\beta_f(t)). \quad (3.89)$$

Der Realteil ist dann die negative Hilberttransformierte des Imaginärteils. Für analytische Funktionen gelten folgende Rechenregeln:

$$\beta_z(t) = \beta_f(t) - if(t), \quad (3.90)$$

$$\beta_{z_1}(t) \cdot z_2(t) = z_1 \cdot \beta_{z_2}(t), \quad (3.91)$$

$$z_1(t) \cdot z_2(t) = iz_1(t) \cdot \beta_{z_2}(t) = i\beta_{z_1}(t) \cdot z_2(t), \quad (3.92)$$

$$\beta_{z_1 \cdot z_2}(t) = -iz_1(t) \cdot z_2(t). \quad (3.93)$$

Das Folgende beschreibt das gewöhnliche komplexe Produkt zweier analytischer Funktionen:

$$z_1(t) = f_1(t) + i\beta_{f_1}(t), z_2(t) = f_2(t) + i\beta_{f_2}(t) \quad (3.94)$$

$$\rightarrow z_1(t)z_2(t) = [f_1(t)f_2(t) - \beta_{f_1}(t)\beta_{f_2}(t)] + i[f_1(t)\beta_{f_2}(t) + \beta_{f_1}(t)f_2(t)]. \quad (3.95)$$

Kombiniert man nun (3.94) mit der Regel (3.93) so folgt, dass das Produkt zweier analytischer Signale wieder ein analytisches Signal ist:

$$[f_1(t) \cdot \beta_{f_2}(t) + \beta_{f_1}(t) \cdot f_2(t)] = HIT[f_1(t) \cdot f_2(t) - \beta_{f_1}(t) \cdot \beta_{f_2}(t)]. \quad (3.96)$$

Grundlegende Eigenschaften der Fouriertransformation

In den folgenden Abschnitten sollen wesentliche Eigenschaften der Fouriertransformation nicht nur dargestellt werden, sondern wie sie insbesondere für die Bildverarbeitung nutzbringend verwendet werden können. Dabei werden generelle Eigenschaften für alle Typen der Fouriertransformation denjenigen Eigenschaften gegenübergestellt, die nur für spezielle Typen der Fouriertransformation gelten.

4.1 Bezeichnungen

Im Folgenden sollen „übliche“ Bezeichnungen, Termini usw. eingeführt werden.

- Alle Fourierkoeffizienten, z. B. α_k für die DFT, $\alpha(\nu)$ für die IFT, nennt man das Fourier-Frequenzspektrum oder einfach das **Spektrum** der Funktion f .
- Alle Beträge der Fourierkoeffizienten, z. B. $|\alpha_k|, |\alpha(\nu)|$ nennt man das **Amplitudenspektrum** der Funktion f .
- Alle Quadrate der Beträge der Fourierkoeffizienten, z. B. $|\alpha_k|^2, |\alpha(\nu)|^2$ nennt man das **Leistungsspektrum** oder Energiespektrum der Funktion f .
- Einen Bereich von ν_1 bis ν_2 nennt man *Bandbreite*. Manchmal wird dieser auch in *Oktaven* angegeben. Ist z. B. $0 < \nu_1 \leq \nu_2$, dann drückt O mit $\nu_2 = 2^O \nu_1$ die Bandbreite in Oktaven aus.
- Da alle Fourierkoeffizienten in der Regel komplex sind, besitzen diese komplexen Zahlen in der komplexen Ebene auch Polarkoordinaten, den Radius r und den Winkel φ . Diesen Winkel nennt man **Phase**. Daher bilden alle Längen r das Amplitudenspektrum, und die Menge aller Winkel das Phasenspektrum. Das Spektrum setzt sich also aus Amplituden- und Phasenspektrum zusammen.
- Um sich bewusst zu werden, dass die Basisfunktionen auch etwas mit Schwingungen zu tun haben, sollte man sich der Eulerschen Beziehung bedienen:

$$e^{i\varphi} = \cos \varphi + i \sin \varphi \rightarrow e^{-i\varphi} = \cos \varphi - i \sin \varphi. \quad (4.1)$$

Die inversen Beziehungen sind dann

$$\cos \varphi = \frac{e^{i\varphi} + e^{-i\varphi}}{2}, \quad \sin \varphi = \frac{e^{i\varphi} - e^{-i\varphi}}{2i}. \quad (4.2)$$

- Je nach Art der Manipulation von Fourierkoeffizienten spricht man von *Tiefpassfiltern*, *Bandpassfiltern* und *Hochpassfiltern*, wobei die „tiefen“, „mittleren“ und „hohen“ Frequenzen „durchgelassen“ werden. Ansonsten spricht man von *Sperren*.
- Verwendet man die Fourierkoeffizienten, so sprechen wir auch vom *Frequenzraum (frequency domain)*, im Unterschied dazu vom *Ortsraum (space domain)*. Beim Ortsraum verwendet man direkt das Signal oder Bild. In der klassischen Signaltheorie übernimmt häufig die Rolle des Ortsraumes der Zeitbereich.
- Die Fouriertransformierte der PSF (siehe Abschn. 2.3) nennt man *Transferfunktion (TF)* oder auch *Übertragungsfunktion (TF)*. Die auf den Gleichanteil normierte Transferfunktion nennt man *optische Transferfunktion (OTF)*. Der Betrag der OTF wird *Modulations-Transferfunktion (MTF)* genannt.
- Eine ideale PSF ist häufig der Einheitsimpuls und damit die MTF bzw. OTF eine konstante Funktion. Daher wird oft die OTF oder MTF als Kennlinie zur Beurteilung eines LSI-Systems herangezogen.

4.2 Nullter Fourierkoeffizient, Gleichanteil

Unter allen Fourierkoeffizienten nimmt der Nullte eine besondere Stellung ein. Für reelle Funktionen ist dieser Fourierkoeffizient immer reell. Wir schreiben ihn einmal für die DFT, AFT und IFT auf:

$$\alpha_0 = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l, \quad \alpha_0 = \frac{1}{\sqrt{X}} \int_0^X f(x) dx, \quad \alpha(0) = \int_{-\infty}^{+\infty} f(x) dx. \quad (4.3)$$

Wir sehen, dieser Koeffizient ist proportional zum Mittelwert der Funktion. In der Elektrotechnik wird dieser als *Gleichanteil* von f bzw. als *DC-Komponente (Direct Current)* bezeichnet. Oft wird von typischen Bandpassfiltern gefordert, dass ihre DC-Komponente Null sein soll. Das gefilterte Bild ist dann invariant gegenüber konstanten Grauwertadditionen, da die Fouriertransformierte einer konstanten Funktion den Einheitsimpuls ergibt.

4.3 Konvergenzverhalten, Gibbssches Phänomen

Im Folgenden betrachten wir Modell A1[X] und beschreiben das Konvergenzverhalten der Fourierreihe. Obwohl das Konvergenzverhalten schon lange bekannt war, konnte es erst 1966 von dem Mathematiker Carleson (Schweden) bewiesen werden. Erst 40 Jahre später,

2006, erhielt er für diese Leistung den Abel-Preis, den (quasi) Nobelpreis für Mathematik. Die Fields-Medaille ist auch ein (quasi) Nobelpreis für Mathematik, der Empfänger darf aber nicht älter als 40 Jahre sein. Damit kommt der Abel-Preis dem Nobelpreis etwas näher.

Da wir den Raum $L^2[X]$ und die vollständige Basis der harmonischen Funktionen betrachtet haben, muss die Konvergenz im quadratischen Mittel erfolgen. Aber was heißt dies nun genauer, wann liegt punktweise Konvergenz vor, wann gleichmäßige Konvergenz und wie schnell konvergieren die Teilsummen?

Lemma 4.1 *Es gelten folgende Aussagen zum Konvergenzverhalten:*

1. Wenn die Funktion im Intervall $[0, X]$ von beschränkter, totaler Variation ist (d. h. sie darf nicht „zu sehr schwingen“), dann liegt punktweise Konvergenz vor. Bei Unstetigkeitsstellen, diese dürfen nur Sprünge sein, konvergieren die Teilsummen gegen das Mittel aus rechtsseitigem und linksseitigem Grenzwert der Funktion in einem Punkt x , d. h. gegen $f(x) = \frac{1}{2}(f^-(x) + f^+(x))$.
2. Wenn die Funktion (mindestens einmal) differenzierbar ist, dann konvergieren die Teilsummen sogar gleichmäßig gegen f .
3. Je glatter die Funktion ist, d. h. je öfter sie differenzierbar ist, desto „schneller“ konvergieren die Teilsummen gegen f . Ist z. B. die Funktion f r -mal differenzierbar und die r -te Ableitung stetig, dann gilt:

$$\lim_{|k| \rightarrow \infty} |k|^r \alpha_k(f) = 0. \quad (4.4)$$

Interessant ist der Fall, wenn die Funktion Sprünge besitzt. Dann tritt eine „Konvergenzkuriosität“ ein, genannt **Gibbssches Phänomen** oder einfach **Ringing** genannt. Es treten an der Unstetigkeitsstelle sogenannte Über- und Unterschwinger auf. Das Kuriose daran ist, dass diese Über- und Unterschwinger nicht verschwinden, auch nicht für $k \rightarrow \infty$. Noch kurioser ist, dass die Höhe der Über- und Unterschwinger völlig unabhängig von k ist, wobei sich dabei ein relativer Fehler bezogen auf die tatsächliche Sprunghöhe von ca. 18 Prozent ergibt. Dieses kuriose Verhalten liegt daran, dass wir an der Sprungstelle nur punktweise Konvergenz haben und keine gleichmäßige Konvergenz. Dieses Ringing hat in der Bildverarbeitung große Bedeutung, da in Bildern häufig Kanten vorkommen, die als Sprünge von Funktionen zu deuten sind. In der Datenkompression, speziell in der mit Informationsverlust (z. B. JPEG, MPEG), möchte man eine Funktion, die in eine Reihe entwickelbar ist, durch eine endliche Teilsumme dieser Reihe approximieren. Benutzt man dabei Transformationen, die ähnlich der Fouriertransformation sind, muss man auf das Ringing achten. Besteht z. B. ein Bild aus einem schwarzen Buchstaben auf weißem Hintergrund, ist am Rande des Buchstabens ein Sprung endlicher Höhe vorhanden. Komprimiert man dieses Bild zu stark, tritt das Gibbssche Phänomen visuell in Form von **Ringing-Artefakten** auf, die optisch sehr störend wirken. Da eine endliche Teilsumme weiter nichts als ein **idealer Tiefpass** ist, sind als Filter oft ideale Tiefpässe nicht erwünscht, da sie diese Ringing-Artefakte erzeugen. Dies ist bei Kompressionen von Bildern usw. zu beach-

Abb. 4.1 a Originalbild mit „natürlichen“ Kanten (z. B. bei „FFT“) und „künstlichen“ Kanten (Sprung vom rechten Bildrand zum linken Bildrand.
 b Bei einem idealen Tiefpass des Bildes in a entstehen nun Wrap-around-Effekte und Ringing-Artefakte



ten. Wenn wir Bilder komprimieren wollen, dann sind Unstetigkeitsstellen sehr störend, nicht nur wegen der Ringing-Artefakte, sondern weil das Konvergenzverhalten schlecht ist, d. h. Bilder sich schlecht komprimieren lassen. Da Bilder mit der endlichen analogen oder diskreten Fouriertransformation immer periodisch fortgesetzt werden, können an den Bildrändern Sprünge entstehen. Diese Sprünge sind „künstlich“ durch die periodische Fortsetzung entstanden, und keine „natürliche“ Sprünge im Bild, wie bei Kanten. Die Effekte durch diese „künstlichen“ Sprünge nennt man *wrap-around-effects*. Diese sollten vermieden werden, z. B. durch gespiegelte Fortsetzung des Bildes oder Anwendung von Fensterfunktionen, die Bildränder aneinander „angleichen“. Ein Beispiel für diese Effekte ist in Abb. 4.1a dargestellt. Es ist schwarzer Text auf weißem Hintergrund zu sehen, also „echte“ Bildkanten. Weiterhin ist der rechte Bildteil schwarz, der linke weiß, so dass eine „künstliche“ Unstetigkeit bei periodischer Fortsetzung entsteht. Auf dieses Bild wurde nun ein idealer Tiefpass angewendet. Das Ergebnis ist in Abb. 4.1b zu sehen. Um den Text sind deutlich die Ringing-Artefakte aufgetreten. Weiterhin sind auch im linken Bildteil die Artefakte als wrap-around-Effekte in Erscheinung getreten. Aus den Konvergenzeigenschaften der Fourierreihen können wir für die Kompression folgendes ableiten. Bilder mit Unstetigkeitsstellen, steilen Kanten usw. lassen sich „schlecht“ komprimieren – je glatter das Bild ist, umso besser können wir komprimieren.

4.4 Linearität der Fouriertransformation

Die am einfachsten zu zeigende Eigenschaft der Fouriertransformation ist die Linearität, d. h. es gilt z. B. für die DFT

$$\alpha_k(\lambda f + \mu g) = \lambda \cdot \alpha_k(f) + \mu \cdot \alpha_k(g), \quad \lambda, \mu \in \mathbb{C}. \quad (4.5)$$

Dies liegt in der Linearität der verwendeten Summen und Integrale begründet.

4.5 Periodizität der Fourierkoeffizienten

Die Bildmodelle A1[X] und D1[N] setzen periodische Funktionen voraus, bzw. die Fouriertransformation erzwingt die Periodizität der Funktionen. Sind nun die Fourierkoeffizienten selbst periodisch oder nicht? Untersuchen wir als erstes einmal die DFT nach (3.46):

$$\alpha_l = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{-2\pi i \frac{kl}{N}}. \quad (4.6)$$

Dann gilt:

$$\begin{aligned} \alpha_{l+N} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{-2\pi i \frac{k(l+N)}{N}} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{-2\pi i \frac{kl}{N}} \cdot e^{-2\pi ik} \\ &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{-2\pi i \frac{kl}{N}} = \alpha_l. \end{aligned} \quad (4.7)$$

Auf Grund dessen, dass $e^{-2\pi ik} = 1, \forall k \in \mathbb{Z}$, folgt die Periodizität $\alpha_{l+N} = \alpha_l$ der diskreten Fourierkoeffizienten. Folglich müssen wir bei der Interpretation der diskreten Fourierkoeffizienten genau hinschauen, für welche Frequenzen sie eigentlich zuständig sind. Dazu ein

Beispiel Wir nehmen einmal an wir haben eine diskrete Funktion mit neun Funktionswerten. Dann haben wir auch bijektiv neun diskrete Fourierkoeffizienten:

$$(f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8) \leftrightarrow (\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8). \quad (4.8)$$

Auf Grund der Periodizität der neun diskreten Fourierkoeffizienten können wir diese auch so schreiben:

$$(\alpha_{-4}, \alpha_{-3}, \alpha_{-2}, \alpha_{-1}, \alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4). \quad (4.9)$$

Jetzt sehen wir deutlicher, welcher Koeffizient für welche Frequenz zuständig ist. Die Grenzfrequenz ist vier, da zu einer Frequenz zwei komplexe Fourierkoeffizienten gehören. Für die Visualisierung von Spektren ist also die zweite Variante besser, für das formale mathematische Rechnen die erste Variante. Wir sehen weiter, dass bei einer geraden Anzahl von Funktionswerten ein Symmetrieproblem auftritt. Nehmen wir z. B. acht Funktionswerte, dann gehen beide Beziehungen über in:

$$(f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7) \leftrightarrow (\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7) \quad (4.10)$$

bzw. die Interpretation mit der Periode acht:

$$(\alpha_{-3}, \alpha_{-2}, \alpha_{-1}, \alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4). \quad (4.11)$$

Da auf Grund der Periode acht $\alpha_4 = \alpha_{-4}$ gilt, können wir auch

$$(\alpha_{-4}, \alpha_{-3}, \alpha_{-2}, \alpha_{-1}, \alpha_0, \alpha_1, \alpha_2, \alpha_3) \quad (4.12)$$

schreiben. Auf jeden Fall ist für die vierte Frequenz nur ein komplexer Fourierkoeffizient zuständig. Auf Grund einer vernünftigen Interpretation der diskreten Fourierkoeffizienten wäre es für die inverse DFT eigentlich besser, folgendes zu schreiben:

$$f_k = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \alpha_l e^{+2\pi i \frac{kl}{N}} \leftrightarrow f_k = \frac{1}{\sqrt{N}} \sum_{l=-N/2}^{+N/2} \alpha_l e^{+2\pi i \frac{kl}{N}}, \quad N \text{ ungerade.} \quad (4.13)$$

Dann müssten wir aber noch für gerade N unterscheiden:

$$f_k = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \alpha_l e^{+2\pi i \frac{kl}{N}} \leftrightarrow f_k = \frac{1}{\sqrt{N}} \sum_{l=-N/2}^{+N/2-1} \alpha_l e^{+2\pi i \frac{kl}{N}}, \quad N \text{ gerade.} \quad (4.14)$$

Auf Grund dieser Unterscheidung zwischen ungerade und gerade ist es mathematisch auf Grund der Periodizität der diskreten Fourierkoeffizienten einfacher, die Summation von 0 bis $N - 1$ wie bisher laufen zu lassen.

Sind denn nun für analoge Funktionen (Modell A1[X]) die Fourierkoeffizienten auch periodisch? Wir schreiben analog:

$$\alpha_k(f) = \frac{1}{\sqrt{X}} \int_0^X f(t) e^{-2\pi i \frac{kt}{X}} dt \quad (4.15)$$

und nun formal analog (obwohl $k + X$ nicht interpretierbar ist)

$$\begin{aligned} \alpha_{k+X}(f) &= \frac{1}{\sqrt{X}} \int_0^X f(t) e^{-2\pi i \frac{(k+X)t}{X}} dt \\ &= \frac{1}{\sqrt{X}} \int_0^X f(t) e^{-2\pi i \frac{kt}{X}} \cdot e^{-2\pi it} dt \neq \alpha_k(f). \end{aligned} \quad (4.16)$$

Der Grund liegt darin, dass die Beziehung $e^{-2\pi it} = 1$ falsch ist, da t eine kontinuierliche Größe und nicht ganzzahlig ist. Für das Modell A1 $[-\infty, +\infty]$ ist es ohnehin Unsinn nach einer Periode zu suchen, da diese ja Unendlich wäre. Für diskrete Funktionen D1 $[-\infty, +\infty]$ sind die analogen Fourierkoeffizienten

$$\alpha_d(\omega) = \sum_{k=-\infty}^{+\infty} f_k e^{-ik\omega} \quad (4.17)$$

periodisch mit 2π wie man leicht sieht. Dieser Fall ist aber für uns nicht so interessant. Die DFT-Koeffizienten sind also (fast) die einzigen Koeffizienten, die selbst wieder periodisch sind.

4.6 Spektrum reeller Funktionen

Ist die zu transformierende Funktion f reell, so hatten wir die Eigenschaft $\alpha_k = \overline{\alpha_{-k}}$ bereits hergeleitet. Diese gilt für alle Formen der FT. Dies bedeutet für das Amplitudenspektrum eines reellen Bildes, dass dieses immer spiegelsymmetrisch zum Koordinatenursprung ist. Durch eine Rotation um 180 Grad wird das Amplitudenspektrum somit wieder in sich überführt. Für die DFT gibt es noch eine Besonderheit. Wenn N eine gerade Zahl ist, dann gibt es ja nur einen Koeffizienten $\alpha_{N/2}$, was ist dann mit dieser Eigenschaft? Nun, die DFT hat als einzige FT die Besonderheit, dass die Fourierkoeffizienten α_k selbst wieder periodisch mit N sind, damit gilt $\alpha_{N/2} = \alpha_{-N/2}$. Auf Grund unserer obigen Eigenschaft muss dann auch $\alpha_{N/2} = \overline{\alpha_{-N/2}}$ gelten, woraus folgt, dass der Koeffizient $\alpha_{N/2}$ bzw. auch $\alpha_{-N/2}$ rein reell sein muss. Demzufolge besitzen endliche, reelle und diskrete Funktionen außer dem reellen α_0 stets noch einen reellen Fourierkoeffizienten $\alpha_{N/2}$ bzw. $\alpha_{-N/2}$. Ist nun die reelle Funktion f noch zusätzlich eine gerade Funktion, d. h. $f_n = f_{-n}$, so gilt:

$$\begin{aligned}\alpha_k &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l e^{-2\pi i k \frac{l}{N}} = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_{-l} e^{-2\pi i k \frac{-l}{N}} \\ &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l \overline{e^{-2\pi i k \frac{l}{N}}} = \overline{\alpha_k}.\end{aligned}\quad (4.18)$$

Damit ist das Spektrum reeller, gerader Funktion rein reell. Ist die Funktion ungerade, d. h. $f_n = -f_{-n}$, dann ist das Spektrum rein imaginär, d. h. $\alpha_k = -\overline{\alpha_k}$. Diese Eigenschaften gelten wiederum für alle FT.

4.7 Parsevalsche Gleichung

Wir betrachten als Beispiel die DFT und bilden einmal unter Verwendung des orthonormalen Basissystems $\varphi^{(k)}(n)$ folgendes:

$$\begin{aligned}\langle f, f \rangle &= \sum_{k=0}^{N-1} |f_k|^2 = \left\langle \sum_{k=0}^{N-1} \alpha_k \varphi^{(k)}, \sum_{l=0}^{N-1} \alpha_l \varphi^{(l)} \right\rangle \\ &= \sum_{k,l} \alpha_k \cdot \overline{\alpha_l} \cdot \delta_{k-l} = \sum_{k=0}^{N-1} |\alpha_k|^2.\end{aligned}\quad (4.19)$$

Diese Gleichung wird als Parsevalsche Gleichung bezeichnet, oft auch als Energiegleichung und stellt eigentlich nur die Verallgemeinerung des gewöhnlichen Satzes des Pythagoras im rechtwinkligen Dreieck dar. Wir hatten diese für allgemeine, orthonormale Basissysteme bereits mit (3.8) kennengelernt. Diese Gleichung gilt für alle Orthonormalsysteme, demzufolge auch für alle Arten der FT. Im Grunde genommen wechseln wir von einem Orthonormalsystem zu einem anderen, was nichts anderes als eine Rotation darstellt. Bei

Rotationen bleiben die „Längen“ der Vektoren erhalten, sind invariant. Wir führen diese Invarianten (4.19) nochmals auf:

- AFT

$$\langle f, f \rangle := \int_0^X |f(x)|^2 dx = \sum_{k=-\infty}^{\infty} |\alpha_k|^2 = \langle \alpha, \alpha \rangle. \quad (4.20)$$

- DFT

$$\langle f, f \rangle := \sum_{k=0}^{N-1} |f_k|^2 = \sum_{k=0}^{N-1} |\alpha_k|^2 = \langle \alpha, \alpha \rangle. \quad (4.21)$$

- IFT

$$\langle f, f \rangle := \int_{-\infty}^{\infty} |f(x)|^2 dx = \int_{-\infty}^{\infty} |\alpha(v)|^2 dv = \langle \alpha, \alpha \rangle. \quad (4.22)$$

Wir sehen, dass die linke Seite nur noch von der Wahl des Skalarproduktes abhängt. Dies ist hilfreich bei der Bestimmung eventueller Faktoren, wenn man nicht das „gewöhnliche“ Skalarprodukt verwendet, insbesondere bei den unsymmetrischen FT. Die Parsevalsche Gleichung für die IFT (4.22) wird oft auch *Satz von Plancherel* genannt.

4.8 Faltungstheorem

Eine fundamentale Eigenschaft wird im Faltungstheorem formuliert. Es beschreibt den Zusammenhang von Faltung und Fouriertransformation. Viele wichtige Anwendungen der Fouriertransformation basieren auf dem Faltungstheorem. Wir wollen es für die DFT beweisen.

Satz 4.2 (Faltungstheorem) *Es seien zwei diskrete Funktionen f und g gegeben. Dann gilt:*

$$\alpha_k(f * g) = C \cdot \alpha_k(f) \cdot \alpha_k(g), \quad C \in \mathbb{R}, \quad (4.23)$$

$$\alpha_k(f \cdot g) = \frac{1}{C} \cdot (\alpha(f) * \alpha(g))_k, \quad C \in \mathbb{R}. \quad (4.24)$$

Beweis Die zweite Zeile ist dual zur ersten Zeile, wobei der Faktor C sich gerade reziprok verhält. Zunächst wollen wir zeigen, dass sich die zweite Zeile einfach aus der ersten ergibt und umgedreht. Wir benutzen dabei die Eigenschaften $\alpha_k(\alpha) = f_k^R$ und $\alpha_k(\alpha^R) = f_k$. Aus dem ersten Teil des Faltungstheorems folgt:

$$\alpha_k(\alpha(f) * \alpha(g)) = C \cdot f_k^R \cdot g_k^R = C \cdot (f \cdot g)_k^R. \quad (4.25)$$

Wir eliminieren die Spiegelung auf der rechten Seite:

$$\alpha_k^R(\alpha(f) * \alpha(g)) = C \cdot f_k \cdot g_k = C \cdot (f \cdot g)_k. \quad (4.26)$$

Jetzt bilden wir wieder auf der linken und rechten Seite das Spektrum:

$$\alpha_k(\alpha^R) = (\alpha(f) * \alpha(g))_k = C \cdot \alpha_k(f \cdot g). \quad (4.27)$$

Damit ist:

$$\alpha_k(f \cdot g) = \frac{1}{C} (\alpha(f) * \alpha(g))_k. \quad (4.28)$$

Nun beweisen wir den ersten Teil des Faltungstheorems. Es ist:

$$\begin{aligned} \alpha_k(f * g) &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \left(\sum_{m=0}^{N-1} f_m g_{l-m} \right) e^{-2\pi i \frac{k(l-m)}{N}} \\ &= \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m e^{-2\pi i \frac{km}{N}} \sum_{l=0}^{N-1} g_{l-m} e^{-2\pi i \frac{k(l-m)}{N}}. \end{aligned} \quad (4.29)$$

In die zweite Summe geht der Index m mit ein, so dass die beiden Summen scheinbar nicht entkoppelt sind. Wie man aber leicht sieht, hängt die zweite Summe überhaupt nicht vom Index m ab, eben auf Grund der Periodizität der e -Funktion und der Funktion g , sodass die beiden Summen doch entkoppelt sind und demzufolge gilt:

$$\begin{aligned} \alpha_k(f * g) &= \sqrt{N} \cdot \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m e^{-2\pi i \frac{km}{N}} \cdot \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} g_l e^{-2\pi i \frac{k(l-m)}{N}} \\ &= \sqrt{N} \alpha_k(f) \cdot \alpha_k(g). \end{aligned} \quad (4.30)$$

□

Damit haben wir das Faltungstheorem für die DFT bewiesen und gleichzeitig die Konstante $C = \sqrt{N}$ berechnet. Wie ermitteln wir aber nun generell den konkreten Wert des Faktors C ? Dazu gibt es einen kleinen „Trick“. Wir können stets schreiben:

$$\alpha_k(f) = \alpha_k(f * \delta) = C \cdot \alpha_k(f) \cdot \alpha_k(\delta) \rightarrow C = \frac{1}{\alpha_k(\delta)}. \quad (4.31)$$

Wir müssen folglich für die konkrete Fouriertransformation das Spektrum des Einheitsimpulses ausrechnen, dieses ist immer eine reelle Konstante. Der Kehrwert dieser Konstante ist der gesuchte Faktor beim Faltungstheorem.

4.9 Spektrum der Korrelationsfunktion

Wir können nun das Faltungstheorem auf alle Beziehungen anwenden bei denen die Faltung auftritt. So können wir die Korrelation auf die Faltung zurückführen:

$$f \circ g = f * \overline{g^R}. \quad (4.32)$$

Da $\alpha_k(\overline{g^R}) = \overline{\alpha_k(g)}$ ist, folgt:

$$\alpha_k(f \circ g) = C \cdot \alpha_k(f) \cdot \overline{\alpha_k(g)}. \quad (4.33)$$

Als Spezialfall können wir nun das Spektrum der *Autokorrelation* angeben:

$$\alpha_k(f \circ f) = C \cdot \alpha_k(f) \cdot \overline{\alpha_k(f)} = C \cdot |\alpha_k(f)|^2. \quad (4.34)$$

Dies bedeutet: das Spektrum der Autokorrelationsfunktion von f ist das Leistungsspektrum von f . In der Literatur, insbesondere im Zusammenhang mit stationären Prozessen, wird dieser einfache Zusammenhang als *Wiener-Chintchin-Theorem* bezeichnet. Manchmal wird auch (4.33) als Kreuzleistungsspektrum bezeichnet. Mit dem Faltungstheorem kann man nun viele Zusammenhänge recht simpel ausrechnen. Will man z. B. das Spektrum der Autokorrelation des Outputs eines LSI-Systems $y = h * x$ berechnen (Wiener-Lee-Beziehung, siehe (2.88)), so ist:

$$\begin{aligned} \alpha_k(y \circ y) &= C \cdot |\alpha_k(y)|^2 = \alpha_k((h * x) \circ (h * x)) = \alpha_k((h \circ h) * (x \circ x)) \\ &= C^3 \cdot |\alpha_k(h)|^2 \cdot |\alpha_k(x)|^2. \end{aligned} \quad (4.35)$$

Man kann also das Leistungsspektrum des Inputs eines LSI-Systems ausschließlich mit Hilfe des Leistungsspektrums der PSF h in das Leistungsspektrum des Outputs des LSI-Systems überführen.

4.10 Kohärenz

Oft interessiert die Korrelation zweier komplexer Spektren. Dazu bildet man das Quadrat des komplexen Korrelationskoeffizienten (siehe (19.88)) zweier Spektren und nennt dies Kohärenz (*coherence*). Sind z. B. zwei diskrete Spektren $\alpha_k(f)$ und $\alpha_l(g)$ gegeben, dann ist die Kohärenz gleich:

$$coherence = \rho_{f,g}^2 = \frac{|\sum_{i=0}^{N-1} \alpha_i(f) \cdot \overline{\alpha_i(g)}|^2}{\sum_{i=0}^{N-1} |\alpha_i(f)|^2 \sum_{i=0}^{N-1} |\alpha_i(g)|^2}. \quad (4.36)$$

4.11 Eigenwerte und Eigenfunktionen bezüglich der Faltung

Die Faltung hängt mit den Fourierkoeffizienten auch über ein Eigenwertproblem zusammen. Im Folgenden notieren wir diskrete Funktionen zum Teil als Spaltenvektoren. Um die Schreibweise zu vereinfachen, soll aus dem Zusammenhang heraus klar sein, ob es eine diskrete Funktion oder ein Spaltenvektor sein soll.

Lemma 4.3 (Eigenfunktionen bei der DFT) *Die diskreten Fourier-Basisfunktionen*

$$x^{(k)} = \frac{1}{\sqrt{N}} \left(1, e^{2\pi i \frac{k}{N}}, e^{2\pi i \frac{2k}{N}}, \dots, e^{2\pi i \frac{(N-1)k}{N}} \right)^T \quad (4.37)$$

mit den Fourierkoeffizienten $\alpha_n(x^{(k)}) = \delta_{k-n}$ sind Eigenfunktionen einer beliebigen diskreten Funktion f bezüglich der Faltung:

$$f * x^{(k)} = \sqrt{N} \cdot \alpha_k(f) \cdot x^{(k)}. \quad (4.38)$$

Wir sehen, nur die Eigenwerte hängen von der konkreten Funktion f ab, die Eigenfunktionen nicht.

Beweis Auf Grund des Faltungstheorems (siehe (4.23))

$$\alpha_n(h) = \sqrt{N} \alpha_n(f) \alpha_n(g) \quad (4.39)$$

für die Faltungsgleichung $h = f * g$ und der trivialen Eigenschaft $\alpha_n(x^{(k)}) = \delta_{n-k}$ können wir schreiben:

$$\alpha_n(f * x^{(k)}) = \sqrt{N} \cdot \alpha_n(f) \cdot \alpha_n(x^{(k)}) = \sqrt{N} \cdot \alpha_n(f) \cdot \delta_{n-k} = \sqrt{N} \cdot \alpha_k(f) \cdot \delta_{n-k}. \quad (4.40)$$

Jetzt können wir die linke Seite und die rechte Seite zurücktransformieren und erhalten das gewünschte Ergebnis (4.38).

In Abschn. 2.2 haben wir gezeigt, dass jeder diskreten Funktion f genau eine Zirkularmatrix C_f zuweisbar ist, damit gilt:

$$f * x^{(k)} = C_f \cdot x^{(k)}. \quad (4.41)$$

Dabei ist C_f die der Funktion f zugeordnete Zirkularmatrix. Damit sind die Fourierkoeffizienten (bis auf einen Faktor) die Eigenwerte der Zirkularmatrix C_f und die Fourier-Basis-Vektoren $x^{(k)}$ die Eigenvektoren, wobei diese völlig unabhängig von der Funktion f sind. \square

Eigenwertprobleme von Matrizen finden wir bei den mannigfachsten Anwendungen, oft stellt die Matrix des Eigenwertproblems eine Kovarianzmatrix dar. Eine Kovarianzmatrix ist symmetrisch und positiv semidefinit. Könnte es nun zwischen Zirkularmatrizen

und Kovarianzmatrizen einen Zusammenhang geben und damit zu unserem Faltungs-Eigenwertproblem? Wenn die Kovarianzen nicht vom Ort abhängen, dann entsteht tatsächlich eine Zirkularmatrix. Wir bilden folglich von der Funktion f unsere bekannte Autokorrelation $f \circ f$ und betrachten die Zirkularmatrix $\mathbf{C}_{f \circ f}$ dieser Autokorrelationsfunktion. Wenn die Kovarianzen nicht vom Ort abhängen, dann ist unsere Zirkularmatrix $\mathbf{C}_{f \circ f}$ identisch (bzw. eine Schätzung) der Kovarianzmatrix des stochastischen Feldes \mathbf{F} (wir betrachten die Funktionswerte von f als Zufallsvariable und schreiben deshalb ein großes F). Wann hängen aber die Kovarianzen nicht vom Ort ab? In der Stochastik nennt man diese Felder **stationär** im erweiterten Sinne. Aus (4.38) folgt dann:

$$\mathbf{C}_{f \circ f} \cdot \mathbf{x}^{(k)} = N \cdot |\alpha_k(f)|^2 \cdot \mathbf{x}^{(k)}. \quad (4.42)$$

Da für stationäre Felder $\mathbf{C}_{f \circ f}$ sogar eine Kovarianzmatrix ist, haben wir einen Fall, für den wir die Lösung des Eigenwertproblems kennen. Bei der PCA (siehe Abschn. 19.10.1) ist dies genau die Aufgabe, für stationäre Felder haben wir durch (4.42) die Lösung für die PCA gefunden. Interessant ist dann, dass alle Eigenvektoren überhaupt nicht vom konkreten Bild f abhängen, die Reihenfolge bezüglich der Größe der Eigenwerte kann sich jedoch ändern.

4.12 Normabschätzungen und Kondition

Zum besseren Verständnis dieses Abschnittes sollte man erst Abschn. 2.2.2 lesen. Zur Beurteilung der Stabilität von inversen Faltungen benötigen wir oft Normabschätzungen der Faltung $f * g$. Wir entwickeln jetzt g nach den Basisfunktionen (Vektoren $x^{(k)}$):

$$g = \sum_{j=0}^{N-1} \alpha_j(g) \cdot x^{(j)}. \quad (4.43)$$

Damit erhalten wir für die Euklidische Norm:

$$\begin{aligned} \|f * g\| &= \left\| \sum_{j=0}^{N-1} \alpha_j(g) (f * x^{(j)}) \right\| = \left\| \sum_{j=0}^{N-1} \alpha_j(g) \sqrt{N} \alpha_j(f) x^{(j)} \right\| \\ &\leq \sqrt{N} \max_l |\alpha_l(f)| \cdot \left\| \sum_{j=0}^{N-1} \alpha_j(g) x^{(j)} \right\| = \sqrt{N} \max_l |\alpha_l(f)| \cdot \|g\|. \end{aligned} \quad (4.44)$$

Das ist gleichzeitig eine optimale Normabschätzung, da für Funktionen $g = x^{(j)}$ die Beziehung $\max_l |\alpha_l(x^{(l)})| = |\alpha_j(x^{(j)})|$ gilt und damit die Ungleichung zur Gleichung wird:

$$\|f * x^{(j)}\| = \sqrt{N} \max_l |\alpha_l(f)| \cdot \|x^{(j)}\|. \quad (4.45)$$

Diese Normabschätzung benötigen wir wieder bei der inversen Faltung. Weiterhin folgt

$$\|a\| = \|a * \delta\| \leq \sqrt{N} \max_j |\alpha_j(a)|, \quad \|a^{-1}\| = \|a^{-1} * \delta\| \leq \sqrt{N} \max_j |\alpha_j(a^{-1})| \quad (4.46)$$

$$\alpha_k(\delta) = \alpha_k(a * a^{-1}) = \frac{1}{\sqrt{N}} = \sqrt{N} \alpha_k(a) \alpha_k(a^{-1}) \rightarrow \alpha_k(a^{-1}) = \frac{1}{N} \frac{1}{\alpha_k(a)}. \quad (4.47)$$

Daraus folgt für die Konditionszahl $K(a) = \|a\|_s \cdot \|a^{-1}\|_s$ (siehe (2.67)) mit der Spektral-norm $\|\cdot\|_s$ (siehe Abschn. 2.2.2) die Beziehung

$$K(a) = \frac{\max_j |\alpha_j(a)|}{\min_k |\alpha_k(a)|}. \quad (4.48)$$

Gegeben sei eine Faltungsgleichung $g = h * f$, wobei g und h gegeben sind und f gesucht ist. Wir nehmen an, die Faltungsinverse h^{-1} existiere, dann ist die Lösung $f = h^{-1} * g$. Wie in Abschn. 2.2.2 ausgeführt, bestimmt die Konditionszahl die Stabilität des Systems. Je kleiner $K(a)$ ist, desto stabiler lässt sich die Faltung invertieren. Weiterhin ist ablesbar, dass die „Schwankungsbreite“ des Amplitudenspektrums entscheidend ist. Minimal ist $K(a)$ sicher dann, wenn das Amplitudenspektrum konstant ist, dann liegen maximal stabile Systeme vor. Dies können wir simulieren. Wenn wir folglich eine PSF konstruieren, indem wir auf dem Einheitskreis in der komplexen Ebene Punkte zufällig erzeugen, d. h. alle haben den Betrag Eins, aber die Phase wird zufällig gewählt (z. B. ein weiß gemachtes Bild), dann ist diese PSF bezüglich der inversen Faltung maximal stabil. Natürlich gilt dies auch für den Einheitsimpuls, er hat sogar ein konstantes Spektrum.

4.13 Faltungs-Iterationen

Wir betrachten die Faltungs-Fixpunktgleichung $f = h * f + g$, dann können wir zeigen:

Satz 4.4 *Es sei h eine diskrete Funktion der Periode N . Es konvergiert die Jacobi-Faltungsiteration*

$$f^{(m+1)} = h * f^{(m)} + g, \quad m = 0, 1, \dots \quad (4.49)$$

für eine beliebige Startfunktion $f^{(0)}$ immer mit

$$\lim_{m \rightarrow \infty} f^{(m)} = f = (\delta - h)^{-1} * g \quad (4.50)$$

und das inverse Element $(\delta - h)^{-1}$ existiert genau dann, wenn

$$\max_k |\alpha_k(h)| < \frac{1}{\sqrt{N}} \quad (4.51)$$

für die Fourierkoeffizienten $\alpha_k(h)$ von h erfüllt ist.

Beweis In der Numerik der linearen Algebra gibt es einen Konvergenzsatz bezüglich der Jacobi-Iteration bei linearen Gleichungssystemen: Als notwendiges und hinreichendes Konvergenzkriterium müssen die Beträge aller Eigenwerte der Iterationsmatrix kleiner Eins sein. Da die Eigenwerte der Zirkularmatrix C_h bis auf einen Faktor aber genau die diskreten Fourerkoeffizienten sind, ergibt sich sofort obiges Konvergenzkriterium. \square

Wenn wir von der Faltungsgleichung $g = h * f$ ausgehen, diese in eine Fixpunktgleichung umformen

$$f = f + c \cdot (g - h * f) \Leftrightarrow f = (\delta - c \cdot h) * f + cg \quad (4.52)$$

erhalten wir als Bedingung:

$$\max_l |1 - \sqrt{N}c \cdot \alpha_l(h)| < 1. \quad (4.53)$$

4.14 Reelle diskrete Fouriertransformation – Hartleytransformation

Für reelle Funktionen f hatten wir die Eigenschaft $\alpha_k(f) = \overline{\alpha}_{-k}(f)$ gezeigt, d. h. die Hälfte aller diskreten Fourerkoeffizienten ist redundant. Das Spektrum als komplexe Funktion ist folglich eine hermitesche Funktion (hermitian, selbstadjungiert). Selbstverständlich ist auch umgekehrt das Spektrum einer hermiteschen Funktion rein reell. Bei praktischen Anwendungen brauchen wir also nur die Hälfte aller Koeffizienten zu speichern. Da dies nicht schön zu verwalten und unsymmetrisch ist, gibt es auch noch andere Ideen. Wir bilden derart eine reelle Funktion $z_k = G(\alpha_k)$, so dass aus den z_k die Koeffizienten α_k wieder ausgerechnet werden können. Ein schönes einfaches Beispiel ist die Summe oder Differenz von Realteil und Imaginärteil. Betrachten wir einmal die Differenz als Beispiel, also

$$\alpha_k = a_k + ib_k, \quad z_k = G(\alpha_k) = a_k - b_k. \quad (4.54)$$

Für reelle f gilt dann $z_{-k} = a_k + b_k$, folglich haben wir zwei Gleichungen mit zwei Unbekannten und können die a_k und b_k aus den z_k und z_{-k} wieder ausrechnen. Speziell mit diesem G haben wir eine totale Symmetrie:

$$\beta_k = G(\alpha_k) \Leftrightarrow f_k = G(\alpha_k(\beta)). \quad (4.55)$$

Wir müssen aber noch zeigen, daß sich die inverse Transformation ebenso verhält. Dazu drücken wir als erstes die β_k durch die komplexen α_k aus, es ist zunächst:

$$\alpha_k = a_k + ib_k \quad (4.56)$$

$$\overline{\alpha}_k = a_k - ib_k \quad (4.57)$$

und damit

$$a_k = \frac{\alpha_k + \bar{\alpha}_k}{2}, \quad b_k = \frac{\alpha_k - \bar{\alpha}_k}{2i}. \quad (4.58)$$

Damit wird β_k zu

$$\beta_k = a_k - b_k = \frac{\alpha_k + \bar{\alpha}_k}{2} - \frac{\alpha_k - \bar{\alpha}_k}{2i} = \frac{1}{2}\alpha_k(1+i) + \frac{1}{2}\bar{\alpha}_k(1-i). \quad (4.59)$$

Nun bilden wir davon das Spektrum und nutzen für reelle f die Regeln $\alpha_l(\bar{\alpha}) = f_l$ und $\alpha_l(\alpha) = f_l^R$

$$\begin{aligned} \alpha_l(\beta) &= \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \left(\frac{1}{2}\alpha_m(1+i) + \frac{1}{2}\bar{\alpha}_m(1-i) \right) e^{-2\pi il \frac{m}{N}} \\ &= \frac{1}{2\sqrt{N}} \sum_{m=0}^{N-1} \alpha_m e^{-2\pi il \frac{m}{N}} + \frac{i}{2\sqrt{N}} \sum_{m=0}^{N-1} \alpha_m e^{-2\pi il \frac{m}{N}} \\ &\quad + \frac{1}{2\sqrt{N}} \sum_{m=0}^{N-1} \bar{\alpha}_m e^{-2\pi il \frac{m}{N}} - \frac{i}{2\sqrt{N}} \sum_{m=0}^{N-1} \bar{\alpha}_m e^{-2\pi il \frac{m}{N}} \\ &= \frac{1}{2}f_l^R + \frac{i}{2}f_l^R + \frac{1}{2}f_l - \frac{i}{2}f_l. \end{aligned} \quad (4.60)$$

Damit wird aber

$$G(\alpha_l(\beta)) = \frac{1}{2}f_l^R - \frac{1}{2}f_l^R + \frac{1}{2}f_l + \frac{1}{2}f_l = f_l. \quad (4.61)$$

Folglich haben wir die Rücktransformation gezeigt. Diese symmetrische, reelle Fouriertransformation nennt man *Hartley-Transformation*. Da wir bei der Transformation die Differenz aus Real- und Imaginärteil bilden, können wir die Transformation auch direkt in der Form

$$\beta_k = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m \left(\cos 2\pi k \frac{m}{N} + \sin 2\pi k \frac{m}{N} \right) \quad (4.62)$$

$$f_k = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} \beta_m \left(\cos 2\pi k \frac{m}{N} + \sin 2\pi k \frac{m}{N} \right) \quad (4.63)$$

schreiben. Diese Basisfunktionen der Hartley-Transformation werden auch *cas(z)*-Funktionen genannt, also

$$\text{cas}(z) := \cos(z) + \sin(z) = \sqrt{2} \cos\left(z - \frac{\pi}{4}\right). \quad (4.64)$$

Schreiben wir die diskreten Basisfunktionen als orthonormiertes System von Spaltenvektoren in eine Matrix H und fassen wieder die diskreten Funktionen als Spaltenvektoren auf, so ergibt sich

$$\beta = H \cdot f \leftrightarrow f = H \cdot \beta. \quad (4.65)$$

Die reelle Matrix H ist also zu sich selbst invers $H^{-1} = H$.

4.15 Separierbarkeit der mehrdimensionalen Fouriertransformation

Wir betrachten nun die zweidimensionale DFT:

$$\begin{aligned} \alpha_{k,l} &= \frac{1}{\sqrt{M}} \frac{1}{\sqrt{N}} \sum_m \sum_n f_{m,n} e^{-2\pi i (k \frac{m}{M} + l \frac{n}{N})} \\ &= \frac{1}{\sqrt{M}} \sum_m \left(e^{-2\pi i k \frac{m}{M}} \frac{1}{\sqrt{N}} \sum_n f_{m,n} e^{-2\pi i l \frac{n}{N}} \right). \end{aligned} \quad (4.66)$$

Das heißt, zuerst transformieren wir die Spalten des Bildes, und anschließend die Zeilen des transformierten Bildes. Damit braucht man eine zweidimensionale DFT nicht zu implementieren. Es reicht, zweimal eine eindimensionale DFT anzuwenden.

4.16 Implementierung der inversen Fouriertransformation

Wenn man die DFT implementiert, braucht man die Rücktransformation nicht extra zu implementieren. Da das Spektrum α_k selbst wieder eine diskrete, komplexe Funktion ist, können wir auch das Spektrum selbst wieder mit der DFT transformieren:

$$\alpha_l(\alpha) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{-2\pi i \frac{lk}{N}} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{+2\pi i \frac{(-l)k}{N}} = f_{-l} = f_l^R. \quad (4.67)$$

Das Spektrum des Spektrums ergibt folglich die gespiegelte Originalfunktion. Wir brauchen also nur die Transformierte zu entspiegeln. Genauso sehen wir auch:

$$\alpha_l(\alpha^R) = f_l, \quad (4.68)$$

d. h. wir können auch vorher das Spektrum spiegeln. Weiterhin gilt:

$$\overline{\alpha_l(\alpha)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{+2\pi i \frac{lk}{N}} = f_l, \quad (4.69)$$

sodass wir das Spektrum nur zu konjugieren brauchen, dann zu transformieren und wieder konjugieren. Das letzte konjugieren fällt bei reellwertigen Funktionen f allerdings weg.

Die gleichen Beziehungen gelten auch für die Integraltransformation. Das Modell A1[X] fällt diesbezüglich völlig aus der Rolle, denn die Originalfunktion $f(x)$ ist eine analoge Funktion, während das Spektrum diskret ist, folglich können wir nicht das Spektrum transformieren. Bei diesem Modell muss echt zwischen Hin- und Rücktransformation unterschieden werden.

4.17 Verschiebungstheorem

Wir betrachten einmal „reine Verschiebungen“ $S_{t_0}g(t) = g(t - t_0)$ und berechnen das Integralspektrum:

$$\begin{aligned}\alpha_{S_{t_0}g}(\omega) &= \int_{-\infty}^{\infty} g(t - t_0) e^{-i\omega t} dt = \int_{-\infty}^{\infty} g(\tau) e^{-i\omega(\tau+t_0)} d\tau \\ &= e^{-i\omega t_0} \int_{-\infty}^{\infty} g(\tau) e^{-i\omega\tau} d\tau = \alpha_g(\omega) \cdot e^{-i\omega t_0}.\end{aligned}\quad (4.70)$$

Verschiebungen im Ortsraum bewirken Modulationen im Frequenzraum. Nun betrachten wir zusätzlich zur Verschiebung im Ortsraum noch eine Modulation im Ortsraum $g_{\omega_0, t_0}(t) = g(t - t_0) e^{i\omega_0 t}$ und erhalten:

$$\begin{aligned}\alpha_{g_{\omega_0, t_0}}(\omega) &= \int_{-\infty}^{\infty} g(t - t_0) e^{i\omega_0 t} e^{-i\omega t} dt = \int_{-\infty}^{\infty} g(t - t_0) e^{-i(\omega - \omega_0)t} dt \\ &= \alpha_g(\omega - \omega_0) e^{-i(\omega - \omega_0)t_0}.\end{aligned}\quad (4.71)$$

Modulationen im Ortsraum bewirken somit Verschiebungen des Spektrums. Wenn wir z. B. reelle Funktionen betrachten, dann ist es sinnvoll, dass auch die Modulation reell ist. Dann können wir notieren:

$$\begin{aligned}g_{\omega_0}(t) &= g(t) \cos \omega_0 t = g(t) \left(\frac{e^{i\omega_0 t} + e^{-i\omega_0 t}}{2} \right) \\ \leftrightarrow \alpha_{g_{\omega_0}}(\omega) &= \frac{\alpha_g(\omega - \omega_0) + \alpha_g(\omega + \omega_0)}{2}.\end{aligned}\quad (4.72)$$

Bisher haben wir Verschiebungen und Modulationen mit der IFT beschrieben, dies gilt entsprechend natürlich auch für alle anderen FT. Benutzen wir aber die DFT, dann sind eigentlich nur ganzzahlige Verschiebungen n zugelassen:

$$\begin{aligned}\alpha_k(S_n f) &= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_{l-n} e^{-2\pi i l \frac{k}{N}} = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m e^{-2\pi i (m+n) \frac{k}{N}} \\ &= \alpha_k(f) e^{-2\pi i n \frac{k}{N}}.\end{aligned}\quad (4.73)$$

Weiterhin kann man das Verschiebungstheorem lediglich als Spezialfall des Faltungstheorems (4.23) auffassen, denn es gilt:

$$\begin{aligned} S_n f = S_n(f * \delta) &= f * S_n \delta \Leftrightarrow \alpha_k(S_n f) = \sqrt{N} \alpha_k(f) \cdot \alpha_k(S_n \delta) \\ &= \alpha_k(f) \cdot \left(\sqrt{N} \alpha_k(S_n \delta) \right). \end{aligned} \quad (4.74)$$

Dann muss man aber das Spektrum des verschobenen Einheitsimpulses ausrechnen.

4.18 Fouriertransformierte spezieller Funktionen

- **Einheitsimpuls**

Egal welches Bildmodell wir wählen, der Einheitsimpuls ist wohl die wichtigste Funktion. Das Spektrum lässt sich leicht ausrechnen. Für die DFT, AFT und IFT ist

$$\alpha_k(\delta) = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \delta_l e^{-2\pi i k \frac{l}{N}} = \frac{1}{\sqrt{N}}, \quad \alpha_k(\delta) = \frac{1}{\sqrt{X}}, \quad \alpha_\delta(v) = 1. \quad (4.75)$$

In (4.75) ist das Spektrum des Einheitsimpulses zu sehen, aber es sind drei verschiedene Funktionen, die den Einheitsimpuls beschreiben. Die erste ist der endliche diskrete Einheitsimpuls, die zweite ist die Diracfunktion, allerdings mit periodischer Wiederholung (Periode X), sozusagen ein Dirac-Kamm. Die dritte Funktion ist die eigentliche Dirac-Funktion. Das Spektrum des Einheitsimpulses ist folglich immer reell und konstant. Welche Konstante benutzt wird, lässt sich direkt ausrechnen oder wir benutzen das Faltungstheorem:

$$\alpha_k(f) = \alpha_k(f * \delta) = C \cdot \alpha_k(f) \cdot \alpha_k(\delta) \rightarrow \alpha_k(\delta) = \frac{1}{C}. \quad (4.76)$$

Die Konstante beim Faltungstheorem bestimmt folglich die Konstante des Einheitsimpulses und umgedreht. Damit können wir nun auch den Einheitsimpuls darstellen:

$$\text{DFT (Bildmodell D1}[N]\text{)}: \quad \delta_l = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i k \frac{l}{N}}, \quad (4.77)$$

$$\text{AFT (Bildmodell A1}[X]\text{)}: \quad \delta_X(x) = \frac{1}{X} \sum_{k=-\infty}^{\infty} e^{2\pi i k \frac{x}{X}}, \quad (4.78)$$

$$\text{IFT (Bildmodell A1}[-\infty, \infty]\text{)}: \quad \delta(x) = \int_{-\infty}^{\infty} e^{2\pi i v x} dv. \quad (4.79)$$

Interessant ist, dass diese Summen/Reihen/Integrale für alle $l \neq 0$ bzw $x \neq 0$ verschwinden müssen. In (4.77) kann man dies leicht durch die endliche geometrische Summe

zeigen:

$$\sum_{k=0}^{N-1} e^{2\pi i k \frac{1}{N}} = \sum_{k=0}^{N-1} \left(e^{2\pi i \frac{1}{N}} \right)^k = \frac{\left(e^{2\pi i \frac{1}{N}} \right)^N - 1}{e^{2\pi i \frac{1}{N}} - 1} = 0. \quad (4.80)$$

In (4.78) haben wir eine Darstellung für den Dirac-Kamm gefunden, welcher aber im mathematischen Sinne „merkwürdig“ ist. Auf der linken Seite steht der Dirac-Kamm als eine Distribution, auf der rechten Seite dagegen eine „normale“ Funktionen-Reihe, die fast nirgends im üblichen Sinne konvergiert. Man kann diese Darstellung aber „retten“, wenn wir die e -Funktionen auf der rechten Seite auch als Distributionen auffassen und die Konvergenz auch im Sinne von Distributionen interpretieren. In diesem Sinne ist diese Darstellung auch Grenzwert des Dirichletschen Integralkernes, siehe auch (7.2). Einen Dirac-Kamm können wir aber auch mit der eigentlichen Dirac-Funktion durch Verschieben und Addition der Dirac-Funktion erzeugen ($\delta_X(x) = \sum_n \delta(x - nX)$), so dass wir eine interessante Beziehung erhalten:

$$\delta_X(x) = \sum_{n=-\infty}^{\infty} \delta(x - nX) = \frac{1}{X} \sum_{k=-\infty}^{\infty} e^{2\pi i k \frac{x}{X}}. \quad (4.81)$$

Wir setzen nun einmal in (4.81) die Intervalllänge zu $X = 1$ und erhalten:

$$\sum_{k=-\infty}^{\infty} \delta(x - k) = \sum_{l=-\infty}^{\infty} e^{2\pi i l x}. \quad (4.82)$$

Diese Beziehung wollen wir nun benutzen:

$$\begin{aligned} \sum_{n=-\infty}^{\infty} f(n) &= \sum_{n=-\infty}^{\infty} \left(\int_{-\infty}^{\infty} \alpha(v) e^{2\pi i nv} dv \right) = \int_{-\infty}^{\infty} \alpha(v) \left[\sum_{k=-\infty}^{\infty} e^{2\pi i kv} \right] dv \\ &= \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} \alpha(v) \delta(k - v) = \sum_{k=-\infty}^{\infty} \alpha(k). \end{aligned} \quad (4.83)$$

Damit erhalten wir mit Hilfe des Dirac-Kammes die Beziehung:

$$\sum_{n=-\infty}^{\infty} f(n) = \sum_{k=-\infty}^{\infty} \alpha(k), \quad (4.84)$$

welche man *Poissonsche Formel* nennt. Nun wollen wir vom Dirac-Kamm das Integralspektrum nach Modell A1 $[-\infty, \infty]$ ausrechnen und dabei die Beziehung (4.81) nutzen. Folglich brauchen wir nur von jedem Summanden $\frac{1}{X} e^{2\pi i k \frac{x}{X}}$ das Integralspektrum berechnen und erhalten unter Benutzung von (4.79):

$$\alpha_{\delta_X}(v) = \frac{1}{X} \sum_{k=-\infty}^{\infty} \delta\left(v - \frac{k}{X}\right). \quad (4.85)$$

Wir sehen: das Integralspektrum eines Dirac-Kammes ist wieder ein Dirac-Kamm, aber auf einem „inversen Gitter“, d. h. hat der Dirac-Kamm eine grobe Auflösung, dann hat der Kamm im Spektrum eine feine Auflösung und umgekehrt. Die Anwendung des Dirac-Kammes auf eine Funktion $f(x)$ wird auch als Abtastung von f mit der idealen Abtastfunktion δ_X bezeichnet:

$$f_a(x) = f(x) \cdot \delta_X(x) = \sum_k f(x) \delta(x - kX). \quad (4.86)$$

Obwohl dies eine Abtastung darstellt, ist diese Funktion immer noch eine analoge, wenn auch mathematisch eine merkwürdige Funktion, nämlich die Multiplikation der Funktion mit dem Dirac-Kamm, der selbst aber eine Distribution darstellt. Wir wollen nun das Integralspektrum dieser Abtastfunktion ausrechnen und benutzen dabei wieder den Verschiebungsooperator $S_u f$ und beachten, dass das Spektrum des Produktes zweier Funktionen die Faltung der beiden Spektren ist:

$$\begin{aligned} \alpha_{f_a}(v) &= \frac{1}{X} \sum_k \alpha_f(v) * S_{\frac{k}{X}} \delta(v) = \frac{1}{X} \sum_k S_{\frac{k}{X}} (\alpha_f(v) * \delta(v)) \\ &= \frac{1}{X} \sum_k \alpha_f \left(v - \frac{k}{X} \right). \end{aligned} \quad (4.87)$$

Das Spektrum der Abtastfunktion ergibt sich durch Addition von Verschiebungen des analogen Integralspektrums der abzutastenden Funktion $f(x)$. Wenn wir nun im Gegensatz dazu die diskrete Funktion $f_d(n) = f(n \cdot X)$ betrachten und deren Spektrum laut dem Modell D1 $[-\infty, \infty]$ berechnen (siehe Abschnitt Abtasttheoreme), so sehen wir, dass dieses Spektrum identisch ist mit dem Integralspektrum (4.87). Das Spektrum bez. der UDFT ist nach (3.71) periodisch mit 2π im Argument. Im Abschnitt Abtasttheoreme haben wir dieses für die diskrete, abgetaste Funktion mit $\alpha_d(\omega T)$ hergeleitet, daher ist diese periodisch in v mit $\frac{1}{T}$. Auch das Integralspektrum (4.87) ist tatsächlich periodisch in v mit $\frac{1}{T}$ ($T = X$), so dass beide Spektren vollständig identisch sind. Deshalb ist es gerechtfertigt, die merkwürdige Funktion (4.86) als analoge Abtastung zu bezeichnen. In der Literatur wird oft obige Vorgehensweise zum direkten Beweis des Abtasttheorems für das unendliche Bildmodell benutzt. Die gleiche Abtastfunktion $f_a(x)$ hatten wir im Abschn. 2.1.1 im Zusammenhang mit der „gemischten“ Faltung kennengelernt.

- **Konstante Funktion**

Wir wollen nun das Spektrum einer konstanten Funktion berechnen, z. B. $f_l = \frac{1}{\sqrt{N}}$:

$$\alpha_k(f) = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} \frac{1}{\sqrt{N}} e^{-2\pi i k \frac{l}{N}} = \overline{\delta_k} = \delta_k. \quad (4.88)$$

Das Spektrum des Einheitsimpulses ist folglich eine konstante Funktion und das Spektrum einer konstanten Funktion ist (bis auf einen Faktor) der Einheitsimpuls.

• Kosinusfunktion

Als „triviale“ Funktion wollen wir nun die Kosinus- oder Sinusfunktion betrachten, da hier die Abhängigkeit vom Bildmodell sehr schön zu sehen ist. Wir betrachten $f(x) = \cos(tx)$, $t \in \mathbb{R}$, wobei t eine beliebige reelle Zahl ist. Zunächst betrachten wir das Bildmodell A1 $[-\infty, +\infty]$, also die Integraltransformation und berechnen das Spektrum. Es ist:

$$\begin{aligned}\alpha_{\cos(tx)}(\omega) &= \int_{-\infty}^{\infty} \left(\frac{e^{itx} + e^{-itx}}{2} \right) e^{-i\omega x} dx = \frac{1}{2} \int_{-\infty}^{\infty} e^{ix(t-\omega)} dx + \frac{1}{2} \int_{-\infty}^{\infty} e^{-ix(\omega+t)} dx \\ &= C \cdot (\delta(\omega - t) + \delta(\omega + t)).\end{aligned}\quad (4.89)$$

Für die Sinusfunktion erhalten wir einen ähnlichen, aber rein imaginären Ausdruck. Um ein ähnliches Ergebnis im Modell A1 $[X]$ zu erhalten, sind einige wesentliche Dinge zu beachten:

- Das Spektrum ist unendlich ausgedehnt, aber diskret. Daher ist der Einheitsimpuls im Spektrum zwar unendlich ausgedehnt, aber diskret. Folglich können wir nicht t kontinuierlich annehmen, sondern ganzzahlig $t = l$, $l \in \mathbb{Z}$.
- Die Wahl des Intervales X ist nun entscheidend. Wollen wir tatsächlich die periodische Funktion $\cos(lx)$ beschreiben, dann müssen wir für X die Periode $\frac{2\pi}{l}$ wählen.
- Unter diesen Voraussetzungen erhalten wir (bis auf einen Faktor C) die gleiche Darstellung:

$$\alpha_k(\cos lx) = C \cdot (\delta_{k-l} + \delta_{k+l}). \quad (4.90)$$

• Rechteckfunktion

Wir betrachten eine gerade Rechteckfunktion der Periode X :

$$f(x) = \begin{cases} I_{\min}, & \text{wenn } \frac{-X}{2} \leq x < \frac{-X}{4} \\ I_{\max}, & \text{wenn } \frac{-X}{4} \leq x \leq \frac{+X}{4} \\ I_{\min}, & \text{wenn } \frac{+X}{4} < x \leq \frac{+X}{2} \end{cases} \quad (4.91)$$

und verwenden deshalb die AFT mit dem Modell A1 $[X]$. Da die Funktion reell und gerade ist, verschwinden alle Sinus-Terme und es ergibt sich:

$$f(x) = \frac{1}{2}(I_{\max} + I_{\min}) + \frac{2}{\pi}(I_{\max} - I_{\min}) \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \cos \left[2\pi(2k-1) \frac{x}{X} \right]. \quad (4.92)$$

Möchte man das Spektrum einer nicht symmetrischen Rechteckfunktion ausrechnen, so brauchen wir nur eine Verschiebung d in (4.92) berücksichtigen:

$$f(x-d) = \frac{1}{2}(I_{\max} + I_{\min}) + \frac{2}{\pi}(I_{\max} - I_{\min}) \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \cos \left[2\pi(2k-1) \frac{(x-d)}{X} \right]. \quad (4.93)$$

Diese periodische Rechteckfunktion wird *square wave* genannt und oft mit der Verschiebung $d = \frac{X}{4}$ und $I_{\min} = -1, I_{\max} = +1$ angegeben, dann wird (4.93) zu

$$f\left(x - \frac{X}{4}\right) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{1}{2k-1} \sin\left[\frac{2\pi(2k-1)x}{X}\right]. \quad (4.94)$$

Eng verwandt mit der Rechteckfunktion ist die umgedrehte Frage: Das Spektrum sei ein idealer Tiefpass, also eine spezielle Rechteckfunktion, wie lautet dann die Funktion im Ortsraum? Folglich muss diese Funktion so ähnlich aussehen wie das Spektrum in (4.92) und wird Dirichletscher Integralkern $D_n(x)$ genannt, siehe (7.2).

- **Gaußfunktion**

Für praktische Anwendungen hat die Gaußfunktion eine große Bedeutung. Als Dichte der Normalverteilung geschrieben ergibt sich:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \leftrightarrow \alpha_f(\omega) = e^{-\frac{\sigma^2\omega^2}{2}}. \quad (4.95)$$

Die Fouriertransformierte ist folglich wieder eine Gaußfunktion, allerdings verschwindet der Faktor und die Rolle von σ vertauscht sich. Dabei haben wir die IFT (3.64) benutzt. Manchmal ist es hilfreich, die Gaußfunktion in der Form:

$$f(x) = e^{-ax^2} \leftrightarrow \alpha_f(\omega) = \sqrt{\frac{\pi}{a}} e^{-\frac{\omega^2}{4a}}. \quad (4.96)$$

zu verwenden. Benutzt man nicht die Kreisfrequenz, dann wird aus Symmetriegründen häufig das Transformationspaar:

$$f(x) = e^{-\pi bx^2} \leftrightarrow \alpha_f(v) = \frac{1}{\sqrt{b}} e^{-\pi \frac{v^2}{b}} \quad (4.97)$$

angegeben.

- **Betragsfunktion**

Als Beispiel einer stetigen, aber nicht überall differenzierbaren Funktion wollen wir die Betragsfunktion $f(x) = |x|$ betrachten. Dabei ist das Bildmodell ganz wichtig, weil sich die resultierenden Funktionen grundlegend unterscheiden. Wir wählen A1[2π] und beachten, dass $|x|$ eine gerade Funktion ist (wir verschieben gedanklich $|x|$ um π). Dann ergibt sich:

$$f(x) = |x| = \sum_{k=-\infty}^{\infty} \frac{(-1)^k - 1}{\pi k^2} e^{+ikx} = \frac{\pi}{2} - \frac{4}{\pi} \sum_{n=0}^{\infty} \frac{\cos((2n+1)x)}{(2n+1)^2}. \quad (4.98)$$

4.19 Spektrum der Ableitungen

Wenn wir eine Funktion ableiten – wie ändert sich dann die Fouriertransformierte der Ableitung? Wir wählen die IFT mit dem Modell A1 $[-\infty, +\infty]$

$$\alpha_f(v) = \int_{-\infty}^{\infty} f(\tau) e^{-2\pi i v \tau} d\tau, \quad f(x) = \int_{-\infty}^{\infty} \alpha_f(v) e^{+2\pi i x v} dv. \quad (4.99)$$

Wir setzen im Folgenden voraus, dass $f(x)$ entsprechend oft differenzierbar ist und dass man unter dem Integral differenzieren darf. Folglich können wir die Ableitung einfach berechnen zu

$$f'(x) = \int_{-\infty}^{\infty} 2\pi i v \alpha(v) e^{+2\pi i x v} dv. \quad (4.100)$$

Aus dieser Darstellung können wir sofort das Spektrum der Ableitung ablesen:

$$\alpha_{f'}(v) = 2\pi i v \cdot \alpha_f(v) \rightarrow \alpha_f(v) = \frac{1}{2\pi i v} \alpha_{f'}(v). \quad (4.101)$$

Damit gilt für die n -te Ableitung sofort:

$$\alpha_{f^{(n)}}(v) = (2\pi i v)^n \alpha_f(v). \quad (4.102)$$

Hohe Frequenzen werden bei den Ableitungen mehr gewichtet als niedrige Frequenzen. Bilden wir nun das Spektrum des Laplace-Operators

$$\alpha_{\Delta f}(u, v) = \alpha_{f_{xx} + f_{yy}}(u, v) = -(2\pi)^2(u^2 + v^2) \alpha_f(u, v). \quad (4.103)$$

Im Spektralbereich heißt also eine Laplace-Filterung eine Multiplikation eines Fourierkoeffizienten mit dem Quadrat des Abstandes der Frequenzen zum Koordinatenursprung.

Betrachten wir weiterhin die Heavisidesche Sprungfunktion $\Theta(x)$ und wir berechnen deren Spektrum:

$$\alpha_{\Theta}(v) = \int_0^{\infty} e^{-2\pi i v x} dx = \frac{1}{2} \delta(v) + \frac{1}{2\pi i v}. \quad (4.104)$$

Das Integral existiert eigentlich gar nicht: das Ergebnis muss als verallgemeinerte Funktion aufgefasst werden. Dann fällt uns der Zusammenhang zur Ableitung (4.101) für $v \neq 0$ auf. Da das Spektrum des Einheitsimpulses Eins ist, ist dies nun das Spektrum des (unbestimmt) integrierten Einheitsimpulses. Damit ist die Dirac-Funktion die Ableitung der Heavisideschen Sprungfunktion.

Die Ableitungs-Eigenschaften kann man nun nutzen um z. B. Differentialgleichungen zu lösen. Man führt dabei die Differentialgleichung auf eine algebraische Gleichung zurück, löst diese und transformiert die Lösung zurück. Dies funktioniert aber nur, wenn für alle in die Differentialgleichung eingehenden Funktionen einschließlich der gesuchten Lösung die Spektren bezüglich der IFT auch existieren. Da dies oft nicht der Fall ist, eignet sich z. B. die Laplace-Transformation besser zum Lösen von Differentialgleichungen. Dazu ein

Beispiel Gegeben sei die Differentialgleichung

$$-u''(x) + u(x) = e^{-|x|}. \quad (4.105)$$

Wir müssen voraussetzen, dass die Spektren von $u(x)$ und $e^{-|x|}$ existieren. Es ist

$$\alpha_{e^{-|x|}}(\nu) = \frac{2}{1 + 4\pi^2\nu^2}. \quad (4.106)$$

Nun wollen wir eine Lösung der Differentialgleichung berechnen. Dazu benutzen wir unsere Ableitungsregeln:

$$-(2\pi i\nu)^2 \alpha_u(\nu) + \alpha_u(\nu) = \frac{2}{1 + 4\pi^2\nu^2}. \quad (4.107)$$

Wir klammern $\alpha_u(\nu)$ aus und lösen danach auf:

$$\alpha_u(\nu) = \frac{2}{(1 + 4\pi^2\nu^2)^2}. \quad (4.108)$$

Zufällig ist dies das Produkt des Spektrums von $e^{-|x|}$ mit sich selbst, daher können wir zur Rücktransformation das Faltungstheorem anwenden und erhalten eine *Autofaltung*

$$u(x) = \frac{1}{2} e^{-|x|} * e^{-|x|} \quad (4.109)$$

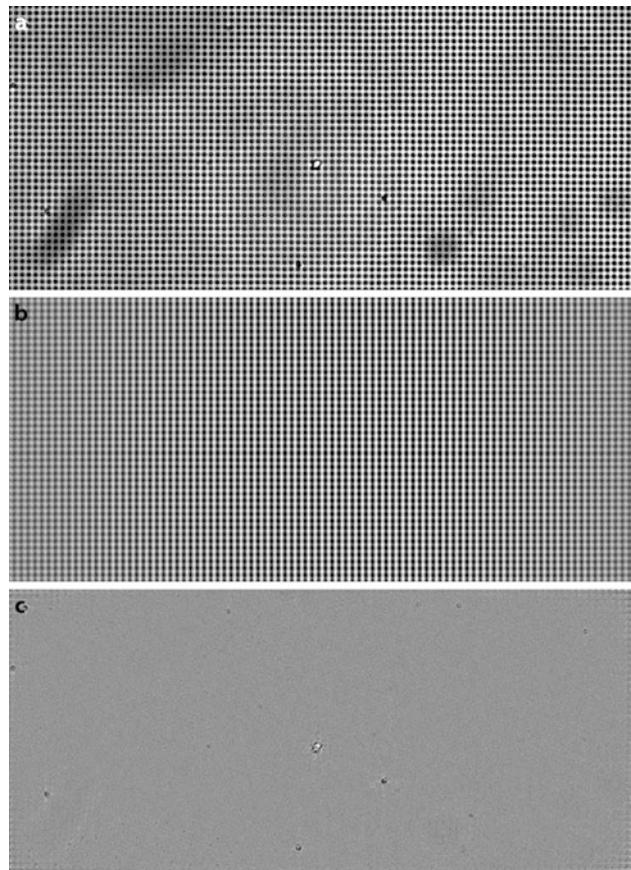
als eine spezielle Lösung dieser Differentialgleichung.

4.20 Periodische Bilder

In periodischen Bildern werden bestimmte Fourierkoeffizienten besonders groß sein, nämlich diejenigen, die gerade die periodischen Verläufe beschreiben. Oft gibt es aber kein periodisches Bild in Reinkultur, sondern es können mehrere periodische Strukturen vorhanden sein. Es gibt nun zwei Möglichkeiten:

- a) Wir selektieren eine periodische Struktur, oder
- b) Wir entfernen jegliche periodische Struktur aus dem Bild.

Abb. 4.2 a Originalbild: Sieb oder Textur mit Fehl- oder Störstellen. b Self-Filtering, c Whitening des Bildes aus a



Das Entfernen jeglicher periodischer Struktur nennen wir **whitening** des Bildes:

$$\beta_k = \frac{\alpha_k}{|\alpha_k|}, \quad |\alpha_k| \neq 0, \quad k = 0, \dots, N - 1. \quad (4.110)$$

Das *whitening* wird manchmals auch als *phase-only transformation* bezeichnet. Das Amplitudenspektrum $|\beta_k|$ des weiß-gemachten Bildes ist identisch Eins, demzufolge auch dessen Leistungsspektrum. Das Leistungsspektrum ist aber gleichzeitig das Spektrum der Autokorrelationsfunktion. Damit ist das Spektrum der Autokorrelationsfunktion reell und konstant, womit die Autokorrelationsfunktion selbst der Einheitsimpuls ist. Dies bedeutet, bei Verschiebung des Bildes ist die Korrelation stets Null. Sollten in praktischen Anwendungen Fourierkoeffizienten Null sein, so dass wir nicht teilen können, dann bleiben diese einfach Null.

Wenn wir nun statt durch die Beträge der Fourierkoeffizienten zu teilen, das Gegenteil machen, nämlich mit diesen multiplizieren, so tritt auch der gegenteilige Effekt ein, die periodischen Strukturen werden verstärkt, wobei sich die dominanteste periodische Struktur

durchsetzen wird:

$$\beta_k = \alpha_k \cdot |\alpha_k| \cdot \sqrt{k^2}, \quad k = -N/2, \dots, -1, +1, \dots, N/2. \quad (4.111)$$

Da die hohen Frequenzen durch die Multiplikationen zu sehr gedämpft werden, bilden wir einen Korrekturfaktor, damit die verbleibende periodische Struktur nicht zu unscharf wird.

Beide Filter sind eindeutig nichtlineare Filter, da die Faktoren vom Bild selbst abhängen. Daher werden solche Filter oft auch als ***self-filtering*** bezeichnet. ***Whitening*** ist demzufolge auch ***self-filtering***, hier bleiben wir aber beim Begriff ***whitening***. ***Whitening*** bzw. ***self-filtering*** können beide auch praktisch genutzt werden. Betrachten wir dazu eine periodische Textur mit Störstellen, siehe Abb. 4.2a. Dann sollen diese Störstellen, falls vorhanden, detektiert werden. Für solch einen Fall ist in Abb. 4.2b, c die Wirkung von ***self-filtering*** und ***whitening*** zu sehen.

4.21 Fouriertransformation für komplexe Funktionen

4.21.1 Blockmatching

In der Regel sind Bilder oder andere Signale reelle Funktionen. Man kann aber auch sinnvolle Anwendungen für komplexe Funktionen finden, siehe z. B. Abschn. 4.9. Wenn wir z. B. beim klassischen Blockmatching in der Bildverarbeitung zwei Blöcke, d. h. zwei Bilder gleichzeitig bearbeiten bzw. transformieren müssen, dann können wir aus Effizienzgründen einen kleinen Trick benutzen. Das erste Bild f_1 wird zum Realteil und das zweite Bild f_2 zum Imaginärteil einer komplexen Funktion $f = f_1 + i f_2$. Wir berechnen nun die DFT von f und aus diesem Spektrum wieder die Spektren von f_1 und f_2 , es ist also nur eine DFT nötig. Es gilt:

$$z_k = \alpha_k(f) = \alpha_k(f_1 + i f_2) = \alpha_k(f_1) + i \alpha_k(f_2) \quad (4.112)$$

$$\bar{z}_k = \overline{\alpha_k(f)} = \overline{\alpha_k(f_1) - i \alpha_k(f_2)} = \alpha_{-k}(f_1) - i \alpha_{-k}(f_2). \quad (4.113)$$

Damit ist:

$$\bar{z}_{-k} = \alpha_k(f_1) - i \alpha_k(f_2)$$

und wir können berechnen:

$$\alpha_k(f_1) = \frac{\bar{z}_{-k} + z_k}{2}, \quad \alpha_k(f_2) = \frac{i}{2} (\bar{z}_{-k} - z_k). \quad (4.114)$$

Damit haben wir also die Spektren der beiden reellen Bilder f_1 und f_2 wieder aus dem Spektrum von f berechnet.

4.21.2 Trigonometrische Interpolation

Im Folgenden betrachten wir die geschlossenen Konturen von Objekten. Da diese Randkurven von Objekten sind, wählen wir eine parametrische Beschreibung, z. B. $(x(t), y(t))$, $t \in [0, T]$, wobei t ein beliebiger Parameter ist. Ist der Rand diskret, dann können wir die Konturpunkte als Paare (x_k, y_k) , $k = 0, 1, \dots, N - 1$ aufzählen. Nun interpretieren wir beide Darstellungen in der komplexen Ebene, d. h. jeder Konturpunkt ist dann eine komplexe Zahl. Wir können also schreiben:

$$z(t) = x(t) + iy(t), \quad t \in [0, T] \quad \text{oder} \quad z_k = x_k + iy_k, \quad k = 0, \dots, N - 1. \quad (4.115)$$

Durch diesen „Trick“ haben wir nun eine analoge oder diskrete Funktion mit nur einer unabhängigen Variablen erhalten, allerdings sind die Funktionswerte komplexe Zahlen. Wir können auf diese Funktionen die eindimensionale analoge oder diskrete Fouriertransformation anwenden. Bisher haben wir immer die Periodizität als lästiges Übel angesehen. In dem Falle von Konturen sind die Funktionen jedoch tatsächlich periodisch. Nehmen wir einmal die obige, diskrete Darstellung der Kontur. Die Funktionswerte sind komplexe Zahlen, wobei diese die Koordinaten der Konturpunkte repräsentieren. Das Argument k , $k = 0, \dots, N - 1$ ist eine Nummerierung der Konturpunkte, folglich eine Aufzählung. Wo wir die Aufzählung beginnen, also den Startpunkt wählen, ist zunächst völlig willkürlich, da die Funktionen tatsächlich periodisch sind. Es handelt sich dann lediglich um eine zyklische Verschiebung der komplexen Funktion. Wenn wir nun durch die Konturpunkte ein trigonometrisches Interpolationspolynom legen, brauchen wir nur die Darstellung (5.3) benutzen:

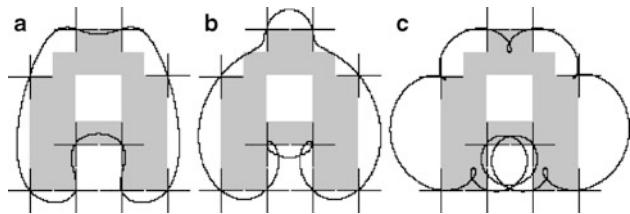
$$z_{k_0}(t) = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1} \alpha_k(z) e^{2\pi i k \frac{t}{T}}, \quad t \in [0, T]. \quad (4.116)$$

Welches k_0 wählen wir nun? Da wir im Gegensatz zum Abtasttheorem keine weiteren Informationen zur Verfügung haben, wählen wir das Polynom, welches uns am „vernünftigsten“ erscheint. Große k_0 bewerten damit hohe Frequenzen und die Polynome werden stark variieren, siehe Abb. 4.3. Also nehmen wir nur die niedrigsten Frequenzen mit $k_0 = -N/2$, folglich:

$$z_{k_0}(t) = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{+\frac{N}{2}} \alpha_k(z) e^{2\pi i k \frac{t}{T}} \quad (4.117)$$

und damit das glatteste Polynom, siehe Abb. 4.3a. Wenn wir z. B. die Polynome (4.116) mit M Punkten diskretisieren, natürlich mit mehr Punkten als die Kontur selbst Punkte hat,

Abb. 4.3 Trigonometrische Interpolation mit $N = 10$ Stützstellen und **a** $k_0 = -5$, **b** $k_0 = -2$, **c** $k_0 = 0$ bezüglich (4.116)



folglich $M \gg N$, so setzen wir $t_l = l \cdot \frac{T}{M}$, $l = 0, \dots, M-1$ in (4.116) ein und erhalten:

$$\begin{aligned} z_{k_0}(t_l) &= \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k(z) e^{2\pi i k \frac{l \cdot \frac{T}{M}}{T}} \\ &= \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k(z) e^{2\pi i k \frac{l}{M}}, \quad l = 0, \dots, M-1. \end{aligned} \quad (4.118)$$

Die komplexen Zahlen $z_{k_0}(t_l)$ interpretieren wir wieder als Konturpunkte und können so die Kontur mit neuen interpolierten Punkten darstellen. Auf diese Weise wurden auch Abb. 4.3a–c erstellt.

4.21.3 Trigonometrische Approximation

Im Folgenden gehen wir vom trigonometrischen Interpolationspolynom der diskreten Konturpunkte aus, und zwar mit $k_0 = -N/2$, demzufolge:

$$z(t) = \frac{1}{\sqrt{N}} \sum_{k=-\frac{N}{2}}^{+\frac{N}{2}} \alpha_k(z) e^{2\pi i k \frac{t}{T}}, \quad t \in [0, T]. \quad (4.119)$$

Wenn wir nun sukzessive hohe Frequenzen weglassen, erhalten wir kein Interpolationspolynom mehr, sondern die Kontur wird nur noch durch ein Polynom approximiert. Im Grunde genommen ist dies weiter nichts als ein Tiefpass des Interpolationspolynoms, folglich:

$$z_P(t) = \frac{1}{\sqrt{N}} \sum_{k=-P}^{+P} \alpha_k(z) e^{2\pi i k \frac{t}{T}}, \quad 0 < P < \frac{N}{2}. \quad (4.120)$$

Je kleiner P wird, umso „größer“ werden die Konturen approximiert, siehe Abb. 4.4. In Abb. 4.5 sind die Tiefpässe einer realen Kontur zu sehen.

Abb. 4.4 Approximation der Kontur mit den ersten a 3, b 10, c 50 Frequenzen – idealer Tiefpass

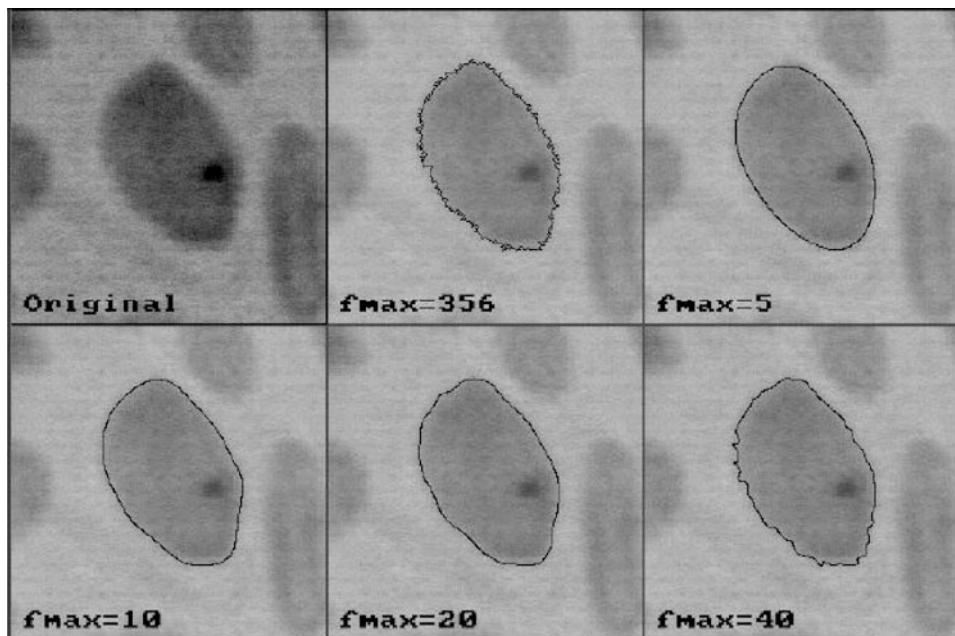
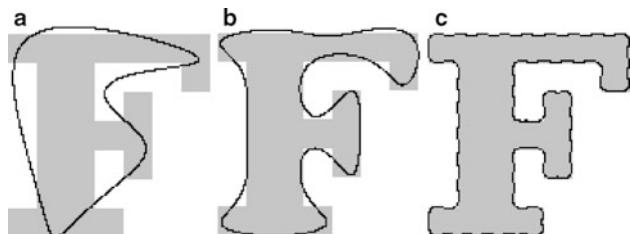


Abb. 4.5 Trigonometrische Approximation – Ideale Tiefpässe einer realen Kontur

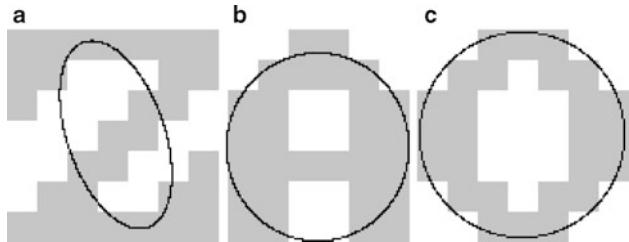
4.21.4 Trigonometrisches Ellipsenfitting

Eine interessante Frage ist nun: wie sieht die „gröbste“ Approximation aus? Es kommt sicher $P = 0$ nicht in Frage, also bleibt nur $P = 1$ übrig:

$$z_1(t) = \frac{1}{\sqrt{N}} \sum_{k=-1}^{+1} \alpha_k(z) e^{2\pi i k \frac{t}{T}} = \frac{1}{\sqrt{N}} \left[\alpha_0 + \alpha_{-1} e^{-2\pi i \frac{t}{T}} + \alpha_1 e^{2\pi i \frac{t}{T}} \right]. \quad (4.121)$$

Diese komplexe Schreibweise einer Funktion zerlegen wir wieder in Realteil und Imaginärteil unter Benutzung der Eulerschen Formel (4.1) und erhalten die Parameterdarstellung

Abb. 4.6 Trigonometrisches Ellipsenfitting und Grad der Elliptizität mit **a** $e_{\text{form}} = 0,656$, **b** $e_{\text{form}} = 0,076$, **c** $e_{\text{form}} = 0,006$ nach (4.124)



einer geschlossenen Kurve:

$$x^{[\text{Ellipse}]}(t) = \frac{1}{\sqrt{N}} \left[\operatorname{Re}(\alpha_0) + \operatorname{Re}(\alpha_1 + \alpha_{-1}) \cdot \cos \frac{2\pi t}{T} + \operatorname{Im}(\alpha_{-1} - \alpha_1) \cdot \sin \frac{2\pi t}{T} \right], \quad (4.122)$$

$$y^{[\text{Ellipse}]}(t) = \frac{1}{\sqrt{N}} \left[\operatorname{Im}(\alpha_0) + \operatorname{Re}(\alpha_1 - \alpha_{-1}) \cdot \sin \frac{2\pi t}{T} + \operatorname{Im}(\alpha_{-1} + \alpha_1) \cdot \cos \frac{2\pi t}{T} \right]. \quad (4.123)$$

Das ist eine bekannte parametrische Form der Darstellung einer Ellipse. Damit ist das größte Polynom $z_1(t)$ stets eine Ellipse und kann zum Ellipsenfitting von Konturen benutzt werden, siehe Abb. 4.6. Man benötigt dazu eigentlich nur zwei Fourierkoeffizienten und braucht somit nicht die vollständige DFT der Konturpunkte auszuführen. Dieses Ellipsenfitting funktioniert aber nur gut, wenn die vollständige Kontur gegeben ist – ein Kontursegment reicht dazu nicht aus. Ist die vollständige Kontur gegeben, dann ist diese Methode eine sehr elegante und effektive Methode. Zusätzlich kann man sofort ein Maß e_{form} für die Elliptizität von Konturen aufstellen. Dazu nutzen wir das Leistungsspektrum bis auf den Nullten Koeffizienten. Wir summieren (bzw. integrieren) über das Leistungsspektrum und normieren bezüglich der beiden ersten Fourierkoeffizienten:

$$e_{\text{form}} = \frac{\left(\sum_{k \notin \{0, -1, 1\}} |\alpha_k(z)|^2 \right)}{(|\alpha_{-1}(z)|^2 + |\alpha_1(z)|^2)}. \quad (4.124)$$

Je größer folglich e_{form} ist, desto unähnlicher ist das Objekt einer Ellipse. Diese Maße sind in Abb. 4.6 eingetragen und entsprechen auch unserer Vorstellung vom Grade der Elliptizität.

Nun wollen wir noch kurz die Kovarianz dieses Ellipsenfittings bezüglich affinen Transformationen zeigen. Dazu benutzen wir die komplexe Schreibweise $z' = az + b\bar{z} + c$ einer affinen Transformation, siehe (4.199). Wir interpretieren eine Kontur als ein Polygon, d. h. als eine geordnete Menge von Punkten z_0, z_1, \dots, z_{N-1} . Wenn nun eine affine Transformation auf diese Punkte angewendet wird, erhalten wir wieder transformierte Punkte in der entsprechenden geordneten Folge $z'_0, z'_1, \dots, z'_{N-1}$: dies ist im Folgenden unsere Annahme. Affin kovariantes Fitting von Ellipsen heißt, dass die beiden gefitteten Ellipsen durch die gleiche affine Transformation ineinander überführbar sind. Das Fitting beider Ellipsen be-

deutet dann:

$$\begin{aligned} z_1(t) &= \frac{1}{\sqrt{N}} \left[\alpha_0 + \alpha_{-1} e^{-2\pi i \frac{t}{N}} + \alpha_1 e^{2\pi i \frac{t}{N}} \right], \\ z'_1(t) &= \frac{1}{\sqrt{N}} \left[\alpha'_0 + \alpha'_{-1} e^{-2\pi i \frac{t}{N}} + \alpha'_1 e^{2\pi i \frac{t}{N}} \right], \end{aligned} \quad (4.125)$$

wobei die α'_k die diskreten Fourierkoeffizienten der affin transformierten Punkte z'_l sind. Wir müssen nun zeigen, dass die affine Transformation von $z_1(t)$ genau $z_1(t)'$ ergibt. Dazu benutzen wir obige komplexe Schreibweise einer affinen Transformation, setzen in diese Beziehung für z die Ellipse $z_1(t)$ ein und erhalten:

$$\begin{aligned} z'(t) &= \frac{1}{\sqrt{N}} \left[\left(a\alpha_0 + b\bar{\alpha}_0 + \sqrt{N}c \right) + \left((a\alpha_{-1} + b\bar{\alpha}_1) e^{-2\pi i \frac{t}{N}} \right) \right] \\ &\quad + \frac{1}{\sqrt{N}} \left[\left((a\alpha_1 + b\bar{\alpha}_{-1}) e^{2\pi i \frac{t}{N}} \right) \right]. \end{aligned} \quad (4.126)$$

Andererseits gilt:

$$\begin{aligned} \alpha'_k &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} z'_n e^{-2\pi i k \frac{n}{N}} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} (az_n + b\bar{z}_n + c) e^{-2\pi i k \frac{n}{N}} \\ &= \frac{1}{\sqrt{N}} a \sum_{n=0}^{N-1} z_n e^{-2\pi i k \frac{n}{N}} + \frac{1}{\sqrt{N}} b \sum_{n=0}^{N-1} \bar{z}_n e^{-2\pi i k \frac{n}{N}} + \frac{1}{\sqrt{N}} c \sum_{n=0}^{N-1} e^{-2\pi i k \frac{n}{N}}. \end{aligned} \quad (4.127)$$

Daraus folgt nun

$$\alpha'_0 = a\alpha_0 + b\bar{\alpha}_0 + \sqrt{N}c, \quad \alpha'_1 = a\alpha_1 + b\bar{\alpha}_{-1}, \quad \alpha'_{-1} = a\alpha_{-1} + b\bar{\alpha}_1. \quad (4.128)$$

Genau dies wollten wir für das affine Fitting zeigen. Bei praktischen Aufgaben, obwohl affine Transformationen vorliegen, sind die Punkte allerdings nicht affine Abbilder der Punkte der anderen Kontur. Die Anzahl der Punkte ist nicht die gleiche, im Gitter entstehen Diskretisierungsfehler usw. Damit unsere obigen Annahmen bezüglich der affinen Invarianz des Ellipsenfittings stimmen, muss man die Kontur affin **reparametrisieren**. Legt man dagegen auf die affine Invarianz keinen Wert, kann man das Ellipsenfitting auch ohne Reparametrisierung anwenden.

4.21.5 Normalisierung von Punktmengen

Eine Kontur ist eine geordnete Menge von Punkten, damit eine Funktion mit komplexen Funktionswerten. Von dieser Funktion können wir die diskreten Fourierkoeffizienten bilden und mit Hilfe der sogenannten Normalisierung Invarianten der Kontur berechnen. Bevor wir die Normalisierung bei den Konturen zeigen, soll diese kurz bei ungeordneten Punktmengen demonstriert werden, obwohl auf diese die DFT nicht angewendet werden kann.

Definition 4.5 *Ungeordnete Punktmengen sind Punktmengen bei denen die Aufzählung der Punkte keine Rolle spielt.*

Geordnete Punktmengen unterliegen einer Ordnung bezüglich der Aufzählung der Punkte, z. B. die Eckpunkte von Polygonen. Konturen sind spezielle Polygone. Wir betrachten eine Menge von Punkten in der Ebene:

$$(x_k, y_k), k = 0, 1, \dots, N-1 \leftrightarrow z_k = x_k + iy_k, k = 0, 1, \dots, N-1. \quad (4.129)$$

Aus dem Zusammenhang soll immer hervorgehen, ob diese geordnet ist oder nicht. Wenn wir z. B. eine diskrete Kontur vorliegen haben, dann nummerieren wir einfach die Konturpunkte im mathematisch positiven Umlaufsinn durch, beginnend bei irgendeinem Konturpunkt oder bei einem Polygon mit einem Polygon-Eckpunkt. Weiter nehmen wir im Folgenden an, dass die durch eine geometrische Transformation entstehende Kontur oder Punktmenge auch tatsächlich so vorliegt, nämlich als Transformation dieser Punkte und dass wir genau beim transformierten Startpunkt die Aufzählung auch wieder beginnen. Später werden wir zeigen, wie diese praktisch nicht einhaltbaren Voraussetzungen behandelt werden. Insbesondere bei den Fourierdeskriptoren werden wir die Normalisierungsmethode als Methode zur Herleitung von Invarianten benutzen, daher werden wir kurz diese Methode erläutern.

Wie betrachten zwei Punktmengen P und P' , wobei P' aus P z. B. durch eine Ähnlichkeitstransformation hervorgehe. Für alle $z_n \in P$ und $z'_n \in P'$ gilt dann:

$$z'_n = (re^{i\varphi})z_n + d = cz_n + d, \quad z_n = x_n + iy_n. \quad (4.130)$$

Die komplexe Zahl $d = d_x + id_y$ beschreibt die Translation und die komplexe Zahl $c = c_x + ic_y$ beschreibt mit r die Skalierung und mit φ den Rotationswinkel. In der komplexen Schreibweise haben wir nur zwei Parameter c und d , in der rellen Schreibweise sind es vier. Wir wählen nun von den N Punkten willkürlich zwei aus, z. B. z_0 und z_1 . Diese transformieren wir nun so, dass sich bestimmte komplexe Zielpunkte ergeben, z. B. wählen wir:

$$0 = cz_0 + d, \quad 1 = cz_1 + d. \quad (4.131)$$

Nun haben wir zwei Gleichungen mit zwei Unbekannten und lösen diese:

$$c = \frac{1}{z_1 - z_0}, \quad d = -\frac{z_0}{z_1 - z_0}. \quad (4.132)$$

Mit dieser konkreten Transformation transformieren wir nun alle anderen Punkte:

$$z'_n = \frac{z_n}{z_1 - z_0} - \frac{z_0}{z_1 - z_0} = \frac{z_n - z_0}{z_1 - z_0}, \quad n \neq 0, 1. \quad (4.133)$$

Diese z'_n beziehen sich nun auf die Standardlage der beiden Punkte z_0 und z_1 – wir haben sie diesbezüglich normalisiert. Daher bilden die beiden Punkte z_0 und z_1 eine normalisierte

Basis aller Punkte bezüglich der Ähnlichkeitstransformationen. Drei Punkte würden eine Basis einer affinen Transformation, und vier Punkte die Basis einer projektiven Transformation bilden. Unsere zwei ausgewählten Punkte müssen wir aber in jeder Punktmenge kennen, es müssen natürlich immer die beiden gleichen Punkte sein. Die praktisch einfachste Lösung sind Marker: wir markieren irgendwie zwei typische Punkte, die wir immer wiederfinden. In der Astronomie wäre dies denkbar, da man auf einer Sternenkarte immer zwei typische, bekannte Sterne als Basis benutzen kann. Dieses Verfahren hat auch noch den Vorteil, dass die Punktmengen nicht die gleiche Anzahl von Punkten besitzen müssen, da man einfach für jeden vorhandenen Punkt zwei Invarianten, also normalisierte Koordinaten berechnet. Unsere komplexen normalisierten Koordinaten schreiben wir nochmals in reellen Koordinaten auf:

$$\begin{aligned}x'_n &= \frac{(x_n - x_0)(x_1 - x_0) + (y_n - y_0)(y_1 - y_0)}{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \\y'_n &= \frac{(y_n - y_0)(x_1 - x_0) - (x_n - x_0)(y_1 - y_0)}{(x_2 - x_1)^2 + (y_2 - y_1)^2}.\end{aligned}\quad (4.134)$$

Diese Normalisierung können wir aber dann nicht verwenden, wenn wir die beiden Punkte nicht auffinden. Im *worst case* bilden wir dann $\frac{N(N-1)}{2}$ Basen und überprüfen, ob wenigstens eine brauchbare darunter ist. Bei Polygonen ist dies allerdings einfacher, wenn wir als Basis zwei aufeinanderfolgende Punkte im mathematisch positiven Sinne nehmen, dann hätten wir nur N Basen im *worst case*. Bei affinen und projektiven Basen erhöht sich dies entsprechend.

Wollten wir nun Punktmengen ohne Referenzen durch Invarianten beschreiben, dann müssen wir permutationsinvariante Merkmale erzeugen. Die einfachsten permutationsinvarianten Merkmale $c_{k,l}$ sind Summen von Funktionen der Punkte, z. B.:

$$c_{k,l} = \sum_{j=0}^{N-1} z_j^k \cdot (\overline{z}_j)^l \quad \left(m_{k,l} = \sum_{j=0}^{N-1} x_j^k \cdot y_j^l \right). \quad (4.135)$$

Diese $c_{k,l}$ werden diskrete, komplexe Momente genannt, während die $m_{k,l}$ die üblichen diskreten, reellen Momente sind, siehe Abschn. 19.1. Zur Normalisierung wählen wir folgende Momente aus:

$$\begin{aligned}0 &= \frac{1}{N} c'_{1,0} = \frac{1}{N} \sum_{n=0}^{N-1} z'_n = \frac{1}{N} \sum_{n=0}^{N-1} (cz_n + d) \\&= c \cdot \frac{1}{N} \sum_{n=0}^{N-1} z_n + d\end{aligned}\quad (4.136)$$

$$\begin{aligned}1 &= \frac{1}{N} c'_{2,0} = \frac{1}{N} \sum_{n=0}^{N-1} z'^2_n = \frac{1}{N} \sum_{n=0}^{N-1} (c^2 z_n^2 + 2cdz_n + d^2) \\&= c^2 \frac{1}{N} \sum_{n=0}^{N-1} z_n^2 + 2cd \frac{1}{N} \sum_{n=0}^{N-1} z_n + d^2.\end{aligned}\quad (4.137)$$

Wir haben wieder zwei Gleichungen mit zwei Unbekannten, es ergibt sich:

$$c = \frac{\pm 1}{\sqrt{\frac{1}{N} \sum_{n=0}^{N-1} z_n^2 - (\frac{1}{N} \sum_{n=0}^{N-1} z_n)^2}}, \quad d = \frac{\mp \frac{1}{N} \sum_{n=0}^{N-1} z_n}{\sqrt{\frac{1}{N} \sum_{n=0}^{N-1} z_n^2 - (\frac{1}{N} \sum_{n=0}^{N-1} z_n)^2}} \quad (4.138)$$

$$\rightarrow z'_n = \pm \frac{z_n - \frac{1}{N} \sum_{n=0}^{N-1} z_n}{\sqrt{\frac{1}{N} \sum_{n=0}^{N-1} z_n^2 - (\frac{1}{N} \sum_{n=0}^{N-1} z_n)^2}}. \quad (4.139)$$

Die Vorzeichen sind durch 180 Grad Rotationen bedingt. Setzen wir nun diese normierten z'_n in weitere diskrete, komplexe Momente $c'_{k,l}$ ein, dann sind diese Werte komplexe Invarianten einer gegebenen Punktmenge in Bezug auf Ähnlichkeitstransformationen.

4.21.6 Fourierdeskriptoren und Translationen

Für Fourierdeskriptoren werden ausschließlich geordnete Punktmengen verwendet. Eine Translation in der komplexen Ebene lässt sich durch

$$z_n^{[T]} = z_n + c, \quad n = 0, \dots, N-1, \quad c \in \mathbb{C} \quad (4.140)$$

beschreiben. Dabei ist c eine konstante komplexe Zahl, die die Translation beschreibt. Darauf wenden wir nun die DFT mit den uns bekannten Regeln an:

$$\alpha_k(z^{[T]}) = \alpha_k(z) + \alpha_k(c) = \alpha_k(z) + c\sqrt{N}\delta_k \quad \forall k. \quad (4.141)$$

Damit gilt $\alpha_k(z^{[T]}) = \alpha_k(z)$, $\forall k \neq 0$, d. h. alle außer dem Nullten Fourierkoeffizienten sind invariant gegenüber einer Translation. Der Nullte Fourierkoeffizient beschreibt die „Lage“ des Objektes, da er als Gleichanteil oder DC-Komponente (siehe (4.3)) proportional zum Schwerpunkt des Objektes ist.

4.21.7 Fourierdeskriptoren und isotrope Skalierungen

Eine isotrope Skalierung in der komplexen Ebene wird genauso einfach wie die Translation durch

$$z_n^{[S]} = s \cdot z_n, \quad n = 0, \dots, N-1, \quad s \in \mathbb{R}, \quad s > 0 \quad (4.142)$$

beschrieben, wobei s eine reelle Konstante ist. Dann gilt:

$$\alpha_k(z^{[S]}) = s \cdot \alpha_k(z) \quad \forall k. \quad (4.143)$$

Um den Skalierungsparameter s zu eliminieren, wenden wir die Normierungsmethode an. Wir wählen einen stabilen Koeffizienten α_q aus, z. B. $q = 1$ oder $q = -1$ und normieren diesen auf den Betrag $s_q > 0$, z. B. $s_q = 1$, d. h. wir berechnen s aus:

$$|\alpha_q(z^{[S]})| = s \cdot |\alpha_q(z)| = s_q = \text{const} > 0 \rightarrow s = \frac{s_q}{|\alpha_q|}. \quad (4.144)$$

Dieses berechnete s setzen wir nun in die Transformation (4.143) ein und erhalten:

$$\alpha_k(z^{[S]}) = s \cdot \alpha_k(z) = \frac{s_q}{|\alpha_q|} \alpha_k \quad (4.145)$$

als skalierungsinvariante Fourierdeskriptoren. Wenn wir den Nullten Koeffizienten ausschließen, dann liefert (4.145) translationsinvariante und skalierungsinvariante Fourierdeskriptoren.

4.21.8 Fourierdeskriptoren für Rotation und Startpunktwahl

Als nächstes betrachten wir „reine“ Rotationen mit dem Rotationswinkel β :

$$z_n^{[R]} = e^{i\beta} \cdot z_n, \quad n = 0, \dots, N-1 \rightarrow \alpha_k(z^{[R]}) = e^{i\beta} \cdot \alpha_k(z), \quad k = 0, \dots, N-1. \quad (4.146)$$

Da nur die Phase beeinflusst wird, gilt $|\alpha_k(z^{[R]})| = |\alpha_k(z)|$, d. h. das Amplitudenspektrum ist rotationsinvariant. Wollen wir diesen Informationsverlust durch Elimination der Phase vermeiden, normieren wir wieder. Wir können den Fourierrekoeffizienten α_q aus der Skalierungsnormierung benutzen, da wir diesen jetzt bezüglich der Phase normieren wollen. Es bezeichne $\varphi(z)$ die Phase von z , dann normieren wir:

$$\varphi(\alpha_q(z^{[R]})) = \varphi(\alpha_q(z)) + \beta + 2\pi l = 0, \quad l \in \mathbb{Z}. \quad (4.147)$$

Für $l = 0$ wird nach β umgestellt, in die Transformation (4.146) eingesetzt, und wir erhalten

$$\alpha_k(z^{[R]}) = e^{-i\varphi(\alpha_q)} \cdot \alpha_k(z) \quad \forall k \quad (4.148)$$

rotationsinvariante Fourierdeskriptoren.

Die Voraussetzung, dass wir immer den richtigen Startpunkt finden, ist praktisch jedoch völlig unmöglich. Wir werden die Aufzählung stets mit einem beliebigen Startpunkt beginnen. Dies ist eine Startpunkt-Verschiebung um unbekannte d Eckpunkte der Kontur oder des Polygons:

$$z_n^{[SH]} = (S_d z)_n = z_{n-d}. \quad (4.149)$$

Mit dem Verschiebungstheorem (siehe Abschn. 4.17) und der Normierung der Phase eines Koeffizienten α_p zu Null, ergibt sich:

$$\alpha_k(z^{[SH]}) = \alpha_k(z)e^{-2\pi ik\frac{d}{N}} \rightarrow \varphi(\alpha_p(z^{[SH]})) = \varphi(\alpha_p(z)) - 2\pi p \frac{d}{N} + 2\pi l = 0. \quad (4.150)$$

Wenn wir (4.150) nach der Verschiebung d auflösen und alle Fourierdeskriptoren damit transformieren, haben wir startpunktinvariante Deskriptoren. Da wir aber auch die Phase des q -ten Koeffizienten mittransformieren, wird die Phasennormierung bezüglich der Rotation zerstört. Diese beiden Phasennormierungen scheinen sich zu widersprechen. Dieses Problem kann man lösen, indem wir beide Phasen gleichzeitig normieren, d. h. aber, wir müssen ein Gleichungssystem mit zwei Gleichungen und zwei Unbekannten eindeutig lösen. Vielleicht gelingt uns dies aber leichter durch Hintereinanderausführung beider Transformationen. Zu diesem Zwecke betrachten wir gleich Ähnlichkeitstransformationen, da die Skalierung und Translation keine Schwierigkeiten bereiten. Wir lassen also den Nullten Koeffizienten weg und betrachten die Koeffizienten nach einer Skalierung und Rotation:

$$\alpha_k^{[S,R]} = \frac{s_q}{|\alpha_q|} e^{-i\varphi(\alpha_q)} \alpha_k. \quad (4.151)$$

Wenn wir diese $\alpha_k^{[S,R]}$ nun startpunkt-verschieben, transformiert sich auf der rechten Seite $\varphi(\alpha_q)$ und α_k bezüglich des Verschiebungstheorems, $|\alpha_q|$ bleibt invariant, daher gilt:

$$\alpha_k^{[SH,S,R]} = \alpha_k^{[S,R]} e^{-2\pi i(k-q)\frac{d}{N}}. \quad (4.152)$$

Wenn wir nun für einen Index p mit $p \neq q$ die Phase zu Null normieren und d ausrechnen, bleibt die Normierung für den Index q erhalten, da obige Formel für alle d gilt. Die Phasennormierung ergibt:

$$\varphi(\alpha_p^{[S,R]}) - 2\pi(p-q)\frac{d}{N} + 2\pi l = 0. \quad (4.153)$$

Wir lösen nach d auf und setzen in die Transformation (4.152) ein:

$$\alpha_k^{[SH,S,R]} = \alpha_k^{[S,R]} e^{-i(q-k)\left[\frac{\varphi(\alpha_p^{[S,R]})}{p-q} + 2\pi \frac{l}{p-q}\right]}. \quad (4.154)$$

Diese sind aber nur eindeutig, wenn der Ausdruck $\frac{l}{p-q}$ ganz ist. Für $p = q+1$ ist dies sicher der Fall. Damit sind

$$\alpha_k^{[SH,S,R]} = \alpha_k^{[S,R]} e^{-i(q-k)\varphi(\alpha_{q+1}^{[S,R]})}, \quad k \neq 0 \quad (4.155)$$

Abb. 4.7 Eine Kontur mit gleichem Amplitudenspektrum
a wie die Außenkontur des Buchstabens A, **b** wie die Kontur des Buchstabens F, **c** wie die Kontur des Buchstabens Z



Fourierdeskriptoren, die gegen Ähnlichkeitstransformationen und Startpunktverschiebungen invariant sind.

Da das viel einfacher zu berechnende Amplitudenspektrum invariant gegenüber Rotationen und Startpunktverschiebungen ist, wird dieses häufig zur Klassifikation benutzt. Da wir aber gegenüber dem eigentlichen Spektrum einen Informationsverlust in Kauf nehmen, entsteht die Frage wie groß dieser Verlust ist. Damit haben wir wieder die generelle Frage, ob in der Amplitude oder in der Phase die meiste Information steckt. Dazu machen wir ein Experiment. Von den Konturen in Abb. 4.7 wurden die Fourierdeskriptoren berechnet. Dann wurde jeweils das Amplitudenspektrum berechnet und zusätzlich zum Amplitudenspektrum die Phasen zufällig gewählt. Anschließend wurde zurücktransformiert und eine Kontur erhalten, die die gleichen Invarianten wie die Originalkontur besitzt. Alle diese Objekte können wir folglich mit dem Amplitudenspektrum als Merkmal nicht mehr unterscheiden. In Abb. 4.7 sind Konturen zu sehen, die das gleiche Amplitudenspektrum wie die Konturen der „Buchstaben“ besitzen. Der Informationsverlust ist beträchtlich. Damit lassen sich gleiche Buchstaben zwischen verschiedenen Fonts nicht mehr unterscheiden.

Neben den Ähnlichkeitstransformationen kann man auch Fourierdeskriptoren für affine Transformationen angeben. Dazu muss man bezüglich der Normierung ein lineares Gleichungssystem explizit analytisch lösen. Da diese affin invarianten Fourierdeskriptoren in der Praxis schwierig einsetzbar sind, sei hier lediglich auf die Literatur [86] verwiesen.

4.21.9 Fourierdeskriptoren in der Praxis

Die praktische Nutzung von Fourierdeskriptoren ist mit Schwierigkeiten verbunden. Wir nehmen dazu an, aus dem praktischen Problem sei eine geometrische Transformation T bekannt. Weiterhin sei von einem Referenzobjekt O die diskrete Kontur:

$$z_l = x_l + i y_l, \quad l = 0, \dots, N - 1 \quad (4.156)$$

bekannt. Von einem unbekannten Objekt O' berechnen wir ebenfalls die diskrete Kontur:

$$z'_k = x'_k + i y'_k, \quad k = 0, \dots, M - 1. \quad (4.157)$$

Es zeigen sich im Wesentlichen zwei Schwierigkeiten:

- Es ist in der Regel $M \neq N$, d. h. die Anzahl der Konturpunkte stimmt nicht überein.
- Wir nehmen einmal an, es sei zufällig $M = N$. Dann muss bis auf eine unbekannte Startpunktverschiebung d die Kovarianz $z'_{j-d} = Tz_j$, $j = 0, \dots, N - 1$ gelten. Auf Grund der Diskretisierungsfehler wird dies nicht der Fall sein. Ein extremes Beispiel dafür ist ein Rechteck O in einer horizontalen Lage. Das Rechteck O' sei in 45-Grad-Lage detektiert. Bereits in diesem Fall stimmt die diagonale Rasterung mit der horizontalen überhaupt nicht mehr überein.

Bevor wir folglich die Fourierdeskriptoren anwenden, müssen wir beide Probleme lösen bzw. näherungsweise lösen. Dazu können wir die Kontur wie folgt reparametrisieren:

- Zunächst erstellen wir aus der diskreten Kontur z_i , $i = 0, \dots, N - 1$ eine analoge Kontur $z(t)$, $t \in [0, T]$ durch Interpolation oder Approximation. Dies geschieht natürlich auch für das Objekt O' mit der Bezeichnung $z'(t')$, $t' \in [0, T']$. Im einfachsten Falle interpretieren wir die Kontur als Polygon, was allerdings keine robuste Approximation darstellt. Dies ist bisher nur eine analoge Parametrisierung der Kontur, aber noch nicht die gewünschte Reparametrisierung. Wir nehmen jetzt an, es sei $z'(0) = Tz(0)$, d. h. wir kennen genau die Startpunkte.
- Wir wählen einen geeigneten Parameter s anstatt t für $z(t)$ und s' anstatt t' für $z'(t')$. Es muss dann eine monotone Funktion $s' = h(s)$ geben, so dass die Kovarianz $z'(s') = Tz(s)$ gilt. Die Funktion h muss also im Einklang mit der Transformation T stehen. Dazu ein

Beispiel Als Beispiel betrachten wir Ähnlichkeitstransformationen für T und als Parameter für die Reparametrisierung die Bogenlänge s . Die Translation ist bei den Ähnlichkeitstransformationen völlig uninteressant. Wir nutzen die bekannte Beziehung (19.17):

$$(ds')^2 = (dx')^2 + (dy')^2. \quad (4.158)$$

Für affine Transformationen (siehe Abschn. 1.4.3) gilt dann:

$$(ds')^2 = (dx')^2 + (dy')^2 = (a_{11}dx + a_{12}dy)^2 + (a_{21}dx + a_{22}dy)^2. \quad (4.159)$$

Für Ähnlichkeitstransformationen mit der Skalierung a und dem Winkel φ gilt:

$$a_{11} = a \cos \varphi, \quad a_{12} = -a \sin \varphi, \quad a_{21} = a \sin \varphi, \quad a_{22} = a \cos \varphi, \quad (4.160)$$

demzufolge ist:

$$(ds')^2 = a^2((dx)^2 + (dy)^2) = a^2(ds)^2. \quad (4.161)$$

Daraus folgt die Funktion $s' = h(s)$ zu:

$$ds' = a \cdot ds, \quad a > 0 \rightarrow s' = a \cdot s = h(s), \quad a > 0. \quad (4.162)$$

Damit ist gezeigt, dass die Parametrisierung mit der Bogenlänge s bezüglich Ähnlichkeitstransformationen eine geeignete Reparametrisierung darstellt. Für affine Transformationen ist die Reparametrisierung viel schwieriger, dazu müssen wir eine Flächenparametrisierung durchführen, siehe [86]. Für die Bogenlänge s und Ähnlichkeitstransformationen gehen wir in der Praxis folgendermaßen vor:

- Die analoge Kontur $z(t)$ reparametrisieren wir bezüglich der Bogenlänge und anschließend tasten wir diese Kontur mit einer äquidistanten Schrittweite bezüglich der Bogenlänge s ab, so dass wir stets N diskrete Konturpunkte z_k^s , $k = 0, \dots, N - 1$ erhalten.
- Von diesen Konturpunkten berechnen wir die auch startpunktinvarianten Fourierdeksiptoren (4.155).

Wenn wir bei der Abtastung immer einen „geeigneten“ Startpunkt zufällig treffen, dann ist alles in Ordnung. Oft liegen wir aber „zwischen richtigen“ Abtastpunkten. Dieses Problem können wir nur näherungsweise lösen, indem wir N genügend groß wählen, so dass dieser Fehler hinreichend klein wird.

4.22 Cepstrum

Das Wort Cepstrum wurde invers aus Spec und dem Wort trum gebildet und wurde bereits 1963 von Bogert, Healy und Tukey [5] eingeführt. Das Cepstrum $c(f)$ von f ist definiert als:

$$c(f) := \alpha^{-1} [\log \alpha(f)]. \quad (4.163)$$

Wir bilden folglich das Spektrum, logarithmieren dies und transformieren zurück. Ohne die Logarithmierung würde einfach wieder das Original f entstehen. In dieser Form wird es auch als komplexes Cepstrum bezeichnet, da der Logarithmus von komplexen Zahlen zu bilden ist. Oft wird auch:

$$c(f) := |\alpha^{-1} [\log |\alpha(f)|]| \quad (4.164)$$

als Cepstrum bezeichnet. Bezieht man sich auf das Leistungsspektrum, so spricht man auch oft vom „power cepstrum“ pc:

$$pc(f) := |\alpha^{-1} [\log |\alpha(f)|^2]|^2. \quad (4.165)$$

Manche Autoren schreiben auch:

$$c(f) := |\alpha [\log |\alpha(f)|]|. \quad (4.166)$$

Da dies nur die gespiegelte Version ergibt, ist dies jedoch völlig uninteressant. Das Cepstrum wurde ursprünglich benutzt um in seismologischen, verrauschten und überlagerter Echosignalen die Zeitverzögerung auszurechnen. Für die Bildverarbeitung bedeutet dies: wenn man statt einem Bild und dessen Verschiebung nur das überlagerte Bild mit dessen Verschiebung hat, kann man trotzdem die Verschiebung noch ausrechnen. Generell dient der Logarithmus beim Cepstrum dazu, Multiplikationen im Frequenzbereich (z. B. eine Faltung im Ortsbereich ergibt Multiplikationen im Frequenzbereich) auf additive Überlagerungen zurückzuführen, um so bestimmte Komponenten trennen zu können. Die Cepstrum-Methode wird vorwiegend dann angewendet, wenn aus zwei additiv überlagerten Bildern, wobei ein Bild die Verschiebung des anderen darstellt, die Verschiebung ausgerechnet werden soll, und zwar ausschließlich und nur aus dem additiv überlagerten Bild, siehe dazu Abschn. 19.8.4.

4.23 Modifizierte diskrete Fouriertransformationen (MDFT)

Die grundlegende Variante der DFT (3.46) kann man so modifizieren, dass man eine ganze Klasse von modifizierten DFT's erhält. Im Abschn. 5.4.2 haben wir eine solche benutzt und für die Diskrete Cosinus-Transformation (DCT) werden wir auch eine solche benötigen. Die Idee ist, die anloge Funktion zuerst zu verschieben und dann diskret abzutasten, oder die analoge Funktion zuerst zu modulieren und dann diskret abzutasten. Wir schreiben erst einmal die allgemeine Form auf und nennen diese modifizierte DFT folglich MDFT. Als allgemeine Basisfunktionen dienen

$$\varphi^{(k),u,v} = \frac{1}{\sqrt{N}} e^{2\pi i \frac{(n+u)(k+v)}{N}}, \quad (4.167)$$

die ein Orthonormalsystem bilden. Daher erhalten wir das Transformationspaar:

$$f_n^{u,v} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k^{u,v} e^{+2\pi i \frac{(n+u)(k+v)}{N}} \leftrightarrow \alpha_n^{u,v} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k^{u,v} e^{-2\pi i \frac{(k+u)(n+v)}{N}}. \quad (4.168)$$

In (4.168) haben wir zwei Indizes u, v eingeführt. Wenn diese ganzzahlig sind, dann ändert sich auf Grund der Periodizität der DFT nicht viel – dies ist keine neue Transformation. Sind dagegen u, v nicht ganz, so haben wir neue Eigenschaften, z. B. geht die Periodizität verloren. Man kann dies leicht überprüfen, da die MDFT auf die DFT zurückführbar ist:

$$\begin{aligned} \alpha_n^{u,v} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k^{u,v} e^{-2\pi i \frac{(k+u)(n+v)}{N}} \\ &= e^{-2\pi i u \frac{n}{N}} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left[f_k^{u,v} e^{-2\pi i \frac{v(k+u)}{N}} \right] e^{-2\pi i k \frac{n}{N}}. \end{aligned} \quad (4.169)$$

Die Bezeichnung $f_k^{u,v}$ für die diskreten Funktionswerte soll nur andeuten, dass mit der analogen Funktion vor der Abtastung bezüglich u, v etwas passiert ist. Für uns sind dies abstrakt nur die vorgegebenen, diskreten Funktionswerte. Für $u = 0, v = 0$ erhalten wir die klassische DFT. Nun wollen wir die Parameter u, v deuten, d. h. was ist vor der Abtastung der analogen Funktion passiert? Wir wissen aus dem Abschn. 5.1, dass die DFT-Koeffizienten α_k weiter nichts als die analogen Koeffizienten β_k einer bandbegrenzten Funktion $f(x)$ sind. Betrachten wir zunächst einen Parameter λ und schreiben eine analoge, bandbegrenzte Funktion auf:

$$f(x + \lambda) = \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{2\pi i k \frac{(x+\lambda)}{X}}. \quad (4.170)$$

Der Parameter λ ist hier nur eine Verschiebung der analogen Funktion $f(x)$. Wir tasten nun die analoge, verschobene Funktion $f(x + \lambda)$ äquidistant ab mit $x_l = l \cdot \frac{X}{N}$ und ersetzen die absolute Verschiebung λ durch die relative Verschiebung u bezüglich der Schrittweite $\frac{X}{N}$ mit $\lambda = u \cdot \frac{X}{N}$:

$$f_l^{u,0} = f\left(l \frac{X}{N} + u \frac{X}{N}\right) = \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{2\pi i k \frac{(l+u)}{N}}. \quad (4.171)$$

Wir sehen, die β_k übernehmen bis auf einen Faktor die Rolle der $\alpha_k^{u,v}$. Damit ist der Parameter u deutbar als relative Verschiebung der analogen Funktion.

Nun wollen wir den Parameter v untersuchen, dazu betrachten wir:

$$e^{-2\pi i v \frac{x}{X}} f(x) = \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{2\pi i k \frac{x}{X}} \rightarrow f(x) = \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{2\pi i \frac{x(k+v)}{X}}. \quad (4.172)$$

Im linken Teil steht folglich nicht die bandbegrenzte Fourierdarstellung von $f(x)$, sondern von $e^{-2\pi i v \frac{x}{X}} f(x)$, also der modulierten Funktion. Wir tasten wie immer mit $x_l = l \frac{X}{N}$ ab und erhalten:

$$e^{-2\pi i v \frac{l}{N}} f(x_l) = \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{2\pi i k \frac{l}{N}} \rightarrow f(x_l) = \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{2\pi i \frac{l(k+v)}{N}}. \quad (4.173)$$

Wir sehen wieder: die β_k übernehmen bis auf einen Faktor die Rolle der $\alpha_k^{u,v}$. Damit ist der Parameter v deutbar als Modulation der Funktion $f(x)$. Da der Modulation eigentlich eine Verschiebung im Frequenzraum entspricht, sagt man auch oft, v entspreche einer Verschiebung im Spektrum, so wie u einer Verschiebung im Ortsraum entspricht. Da die β_k aber selbst diskret sind, kann man eigentlich nur ganzzahlige Verschiebungen betrachten. Daher ist es mathematisch sauberer, nur von Modulation der Funktion $f(x)$ in einem Intervall zu sprechen.

4.24 Diskrete Kosinus Transformation (DCT)

Im JPEG-Standard ist eine reelle diskrete Fouriertransformation enthalten – die diskrete Kosinustransformation, welche erstmals in [1] erwähnt wurde. Wir wollen diese nun herleiten. Da diese Transformation für die Datenkompression entworfen wurde, gilt es einige Besonderheiten zu beachten:

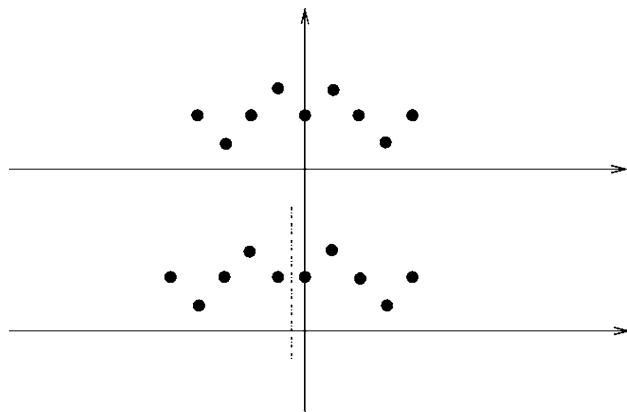
- Sie soll der DFT ähnlich sein, d. h. ihre komprimierenden Eigenschaften besitzen.
- Sie soll von vornherein eine reelle Transformation sein, um Speicher zu sparen.
- Die *wrap-around*-Effekte sollen von vornherein vermieden werden. Durch die periodische Fortsetzung eines Bildes haben wir am Bildrand oft Unstetigkeitsstellen. Diese führen bei einem idealen Tiefpass zu Ringing-Artefakten. Außerdem liegt ein ähnlicher Effekt wie bei der Konvergenz von Fourreihen vor: Je glatter eine Funktion ist, umso weniger Terme der Reihe benötigt man für gute Approximationen, d. h. umso besser ist die Kompressionseigenschaft. Daher soll diese Unstetigkeit am Bildrand vorn vornher ein vermieden werden. Unstetigkeiten an „echten“ Kanten können wir dagegen nicht vermeiden.
- Die Transformation soll sich bezüglich Software und Hardware effektiv implementieren lassen.

Idee Wir setzen eine im Intervall $[0, N - 1]$ definierte, diskrete Funktion gespiegelt fort, dann hat sie die doppelte Intervalllänge $2N$. Anschließend setzen wir sie wie immer periodisch fort. Dadurch hat sie am Bildrand keine Unstetigkeiten mehr, sie wäre bezüglich der Intervallmitte symmetrisch und daher die DFT rein reell. Wenn wir nun durch einen Trick die Intervalllänge $2N$ wieder auf die ursprüngliche Größe N reduzieren könnten, dann hätten wir alle Forderungen erfüllt.

Bis auf die Implementierung wollen wir diese Schritte nun einzeln untersuchen. Gegeben sei also eine diskrete Funktion $f_k, k = 0, \dots, N - 1$. Diese wollen wir nun gespiegelt fortsetzen, so dass ein Symmetriezentrum entsteht. In Abb. 4.8 sind 5 diskrete Funktionswerte zu sehen, diese beginnen am Koordinatenursprung. Im oberen Teil setzen wir gespiegelt nach links fort. Da bei $x = 0$ das Symmetriezentrum liegt, dürfen wir nur 4 Funktionswerte spiegeln. Wollen wir aber alle 5 Werte spiegeln, so entsteht die untere Abbildung. Das Symmetriezentrum liegt jetzt bei $x = -\frac{1}{2}$. Man sieht an diesem Beispiel, es gibt mehrere Möglichkeiten der gespiegelten Fortsetzung, deshalb gibt es auch in der Literatur mehrere DCT-Varianten. Wir entscheiden uns nun für die verbreiteste Form (in der Abb. 4.8 der untere Fall), obwohl das Symmetriezentrum nicht bei Null liegt. Daher müssen wir gedanklich die analoge Funktion um $\frac{1}{2}$ nach „links“ verschieben und dann an $2N$ äquidistanten Stützstellen abtasten. Wir benötigen folglich die MDFT für $v = 0$ (da wir nichts modulieren) und $u = +\frac{1}{2}$, da wir nach „links“ verschieben. Für den oberen Fall aus Abb. 4.8 würde die übliche DFT ausreichen. Die diskrete, gespiegelte Funktion bezeichnen wir jetzt mit \tilde{f} :

$$\tilde{f}_k = \begin{cases} f_k, & k = 0, \dots, N - 1 \\ f_{2N-1-k}, & k = N, \dots, 2N - 1 \end{cases} \quad (4.174)$$

Abb. 4.8 Zwei Möglichkeiten von gespiegelten Fortsetzungen einer diskreten Funktion mit 5 Funktionswerten



bzw. die aufgezählten Funktionswerte pro Periode:

$$\tilde{f} = (f_0, f_1, \dots, f_{N-1}, f_{N-1}, f_{N-2}, \dots, f_0). \quad (4.175)$$

Von diesen Funktionswerten bilden wir nun die MDFT für $u = \frac{1}{2}$, $v = 0$:

$$\alpha_n^{\frac{1}{2},0}(\tilde{f}) = \frac{1}{\sqrt{2N}} \sum_{k=0}^{2N-1} \tilde{f}_k e^{-2\pi i \frac{(k+\frac{1}{2})n}{2N}} = \frac{1}{\sqrt{2N}} \sum_{k=0}^{2N-1} \tilde{f}_k e^{-\pi i \frac{(2k+1)n}{2N}}. \quad (4.176)$$

Als erstes schauen wir uns die Summanden an. Wir betrachten den l -ten Summanden und den $2N - 1 - l$ -ten Summanden, die immer paarweise auftreten müssen:

$$f_l e^{-\pi i \frac{(2l+1)n}{2N}}, f_{l-1} e^{-\pi i \frac{(2(2N-1-l)+1)n}{2N}} = f_l e^{-\pi i \frac{-(+2l+1)n}{2N}}. \quad (4.177)$$

Diese Paare treten folglich immer konjugiert komplex auf, so dass die Sinus-Terme verschwinden und nur noch die Kosinus-Terme übrigbleiben (Eigenschaft einer geraden Funktion). Die Kosinus-Terme treten dann allerdings doppelt auf, so dass sich folgende Summe ergibt:

$$\alpha_n^{\frac{1}{2},0}(\tilde{f}) = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} f_k \cos \pi \frac{(2k+1)n}{2N}, \quad n = 1, \dots, 2N-1. \quad (4.178)$$

Den Ausdruck für $\alpha_0^{\frac{1}{2},0}(\tilde{f})$ halbieren wir einfach, dies ist zwar fasch, wenn wir dies aber bei der Rücktransformation beachten, dann stimmt wieder das Gesamtsystem:

$$\alpha_0^{\frac{1}{2},0}(\tilde{f}) = \frac{1}{\sqrt{2N}} \sum_{k=0}^{N-1} f_k. \quad (4.179)$$

Die Transformation ist uns nun mit N Funktionswerten f_k gelungen. Wir haben aber immer noch $2N$ -MDFT-Koeffizienten zu berechnen. Vielleicht besitzen diese aber interessan-

te Eigenschaften und lassen sich auf die Hälfte reduzieren? Und tatsächlich, folgende Eigenschaften lassen sich einfach verifizieren:

$$\alpha_n^{\frac{1}{2},0}(\tilde{f}) = -\alpha_{2N-n}^{\frac{1}{2},0}(\tilde{f}), \quad \alpha_N^{\frac{1}{2},0}(\tilde{f}) = 0, \quad \alpha_{n+2lN}^{\frac{1}{2},0}(\tilde{f}) = (-1)^l \alpha_n^{\frac{1}{2},0}(\tilde{f}). \quad (4.180)$$

Die erste Bedingung ist eine Symmetrieeigenschaft, die dritte Eigenschaft ist eine Art Periode. Diese Eigenschaften nutzen wir nun für die Rücktransformation aus, so dass wir nur die ersten N Koeffizienten benötigen. Da $\alpha_0^{\frac{1}{2},0}(\tilde{f})$ auf Grund der Symmetrie nur einmal auftritt, verdoppeln wir ihn trotzdem, da wir ihn bei der Hintransformation bewusst halbiert hatten. Daher erhalten wir

$$\tilde{f}_n = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} \alpha_k^{\frac{1}{2},0}(\tilde{f}) \cos \pi \frac{(2n+1)k}{2N}, \quad n = 0, \dots, 2N-1. \quad (4.181)$$

Da wir nur die ersten N Funktionswerte f_k und die ersten N MDFT-Koeffizienten benötigen, benennen wir diese Größen einfach um und erhalten allgemein die eigentliche DCT:

$$DCT_n(f) = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} f_k \cos \pi \frac{(2k+1)n}{2N}, \quad n = 1, \dots, N-1 \quad (4.182)$$

$$DCT_0(f) = \frac{1}{\sqrt{2N}} \sum_{k=0}^{N-1} f_k, \quad (4.183)$$

$$f_n = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} DCT_k(f) \cos \pi \frac{(2n+1)k}{2N}, \quad n = 0, \dots, N-1. \quad (4.184)$$

Da der Koeffizient $DCT_0(f)$ etwas aus der Rolle fällt, bevorzugen manche Autoren auch die Schreibweise:

$$DCT_0(f) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k, \\ DCT_n(f) = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} f_k \cos \pi \frac{(2k+1)n}{2N}, \quad n = 1, \dots, N-1. \quad (4.185)$$

$$f_n = \frac{1}{\sqrt{N}} DCT_0(f) + \frac{2}{\sqrt{2N}} \sum_{k=1}^{N-1} DCT_k(f) \cos \pi \frac{(2n+1)k}{2N}, \quad n = 0, \dots, N-1. \quad (4.186)$$

Die DCT für zweidimensionale Bilder schreiben wir ebenfalls auf. Man muss dabei besonders den Nullten Koeffizienten beachten:

$$DCT_{m,n}(f) = \frac{2}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f_{k,l} \cos\left(\pi \frac{(2k+1)m}{2M}\right) \cos\left(\pi \frac{(2l+1)n}{2N}\right), \quad (4.187)$$

$$DCT_{0,n}(f) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f_{k,l} \cos\left(\pi \frac{(2l+1)n}{2N}\right), \quad n \geq 1 \quad (4.188)$$

$$DCT_{m,0}(f) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f_{k,l} \cos\left(\pi \frac{(2k+1)m}{2M}\right), \quad m \geq 1 \quad (4.189)$$

$$DCT_{0,0}(f) = \frac{1}{2\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f_{k,l} \quad (4.190)$$

$$f_{m,n} = \frac{2}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} DCT_{k,l}(f) \cos\left(\pi \frac{(2m+1)k}{2M}\right) \cos\left(\pi \frac{(2n+1)l}{2N}\right). \quad (4.191)$$

Eine rein reelle Transformation würden wir auch sofort erhalten, wenn wir die Funktion an einem Symmetriezentrum ungerade fortsetzen. Dadurch bleiben nur die Sinus-Terme übrig und diese nennen wir dann diskrete Sinus-Transformation. Dadurch erzeugen wir aber am Rande künstlich Unstetigkeitsstellen, daher lassen sich diese Transformationen für Datenkompressionen nicht so gut anwenden wie die Kosinus-Transformationen.

4.25 Unschärferelation

Als Auflösung Δf eines Signals $f(t)$ im Punkt t_0 wollen wir die in der Wahrscheinlichkeitsrechnung bekannte Standardabweichung benutzen. Da ein beliebiges Signal $f(t)$ aber keine Dichte darstellt, beziehen wir uns auf die Energie $|f(t)|^2$ und normieren, indem wir durch das Integral über die Energie teilen. Es sei nun wieder $\alpha_f(\omega)$ das Spektrum von f , wobei wir den Kern $e^{-i\omega t}$ benutzen. Dann ist die Auflösung im Ortsraum:

$$(\Delta f)^2 = \frac{\int_{-\infty}^{\infty} (t - t_0)^2 |f(t)|^2 dt}{\int_{-\infty}^{\infty} |f(t)|^2 dt} \quad (4.192)$$

und entsprechend im Frequenzraum:

$$(\Delta \alpha)^2 = \frac{\int_{-\infty}^{\infty} (\omega - \omega_0)^2 |\alpha_f(\omega)|^2 d\omega}{\int_{-\infty}^{\infty} |\alpha_f(\omega)|^2 d\omega}. \quad (4.193)$$

Damit kann man die folgende „Umschärferelation der Nachrichtentechnik“ beweisen:

$$\Delta f \cdot \Delta \alpha \geq \frac{1}{2}. \quad (4.194)$$

Die modulierten Gaußfunktionen:

$$G_{\omega_0, t_0}(t) = e^{-\frac{(t-t_0)^2}{2\sigma^2}} \cdot e^{i\omega_0 t} \quad (4.195)$$

deren Spektrum sich leicht ausrechnen lässt:

$$\alpha_G(\omega) = \sqrt{2\pi}\sigma e^{-\frac{(\omega-\omega_0)^2\sigma^2}{2}} \cdot e^{-i(\omega-\omega_0)t_0}, \quad (4.196)$$

heißen **Gabor-Funktionen** (Gabor 1946) und besitzen die minimale Unschärfe, d. h. sie erfüllen die Unschärfe (4.194) als Gleichung. Die Gaborfunktionen (4.195) besitzen die Auflösungen:

$$\Delta f = \frac{\sigma}{\sqrt{2}}, \Delta \alpha = \frac{1}{\sqrt{2}\sigma}. \quad (4.197)$$

Die Gaborfunktionen besitzen im Ortsraum als auch im Frequenzraum die gleichen Auflösungen, wenn $\sigma = 1$ gilt. Die Unschärfe (4.194) hat zur Folge, dass man nicht gleichzeitig eine hohe Auflösung im Ortsraum als auch im Frequenzraum erreichen kann. Die Funktionen, die aber dennoch einen besten Kompromiss bilden, sind die Gabor-Funktionen.

4.26 Affine Transformationen

Gegeben seien zwei Bilder f und f' , wobei $f' = T f$ durch eine geometrische Transformation T aus f hervorgeht. Oft ist T eine affine Transformation:

$$\mathbf{x}' = \mathbf{A} \cdot \mathbf{x} + \mathbf{b} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} a_{10} \\ a_{20} \end{pmatrix}. \quad (4.198)$$

Oft (insbesondere bei Fourierdeskriptoren von Konturen) ist es günstig, affine Transformation in komplexer Schreibweise zu nutzen:

$$z' = a \cdot z + b \cdot \bar{z} + c \quad (4.199)$$

mit

$$\begin{aligned} a &= \frac{1}{2}(a_{11} + a_{22}) + \frac{i}{2}(a_{21} - a_{12}), \\ b &= \frac{1}{2}(a_{11} - a_{22}) + \frac{i}{2}(a_{21} + a_{12}), \\ c &= a_{10} + i a_{20}. \end{aligned} \quad (4.200)$$

Weiter soll $f'(\mathbf{x}') = f(\mathbf{x})$ gelten. Die Frage ist wie bereits beim Verschiebungstheorem, wie sich das Spektrum von f transformiert. Für die Untersuchungen müssen wir nun

tatsächlich 2D-Funktionen und die 2D-Fourier-Transformation benutzen, da es z. B. ein-dimensionale Rotationen nicht gibt. Weiterhin können wir zur Untersuchung auch die DFT nicht nutzen, da es für den diskreten Fall keine Rotationen gibt. Es bleibt uns nichts weiter übrig als die 2D-Fourier-Integraltransformation (IFT) anzuwenden. Wir schreiben diese gleich in vektorieller Notation auf:

$$\begin{aligned}\alpha_{f'}(\omega) &= \iint f'(\mathbf{x}') e^{-i\omega^T \mathbf{x}'} d\mathbf{x}' = \iint f(\mathbf{x}) e^{-i\omega^T (\mathbf{A}\mathbf{x} + \mathbf{b})} \cdot |\det(\mathbf{A})| d\mathbf{x} \\ &= |\det(\mathbf{A})| \cdot e^{-i\omega^T \mathbf{b}} \cdot \alpha_f(\mathbf{A}^T \omega).\end{aligned}\quad (4.201)$$

Wenn wir genau die inverse Beziehung haben wollen, dann setzen wir einfach $\mathbf{A}^T \omega = \beta$, damit ist $\omega = \mathbf{A}^{T^{-1}} \beta$. Wir lösen nach $\alpha_f(\beta)$ auf und ersetzen β wieder formal durch ω und erhalten somit:

$$\alpha_f(\omega) = \frac{1}{|\det(\mathbf{A})|} \alpha_{f'}(\mathbf{A}^{T^{-1}} \omega) \cdot e^{+i\omega^T \mathbf{A}^{-1} \mathbf{b}}.\quad (4.202)$$

Nun betrachten wir zunächst 4.202 für speziellen Transformationen, z. B. reine Rotationen $\mathbf{x}' = \mathbf{R}\mathbf{x}$ mit der Rotationsmatrix \mathbf{R} und $\mathbf{b} = \mathbf{0}$. Dann geht (4.202) über in:

$$\alpha_f(\omega) = \alpha_{f'}(\mathbf{R}^{T^{-1}} \omega) = \alpha_{f'}(\mathbf{R} \cdot \omega).\quad (4.203)$$

Wir sehen: falls das Bild rotiert, rotiert auch das Spektrum in gleichem Orientierungssinn mit. Betrachten wir nun eine reine isotrope Skalierung mit der Diagonalmatrix \mathbf{D} und $\mathbf{b} = \mathbf{0}$. Dann geht (4.202) über in:

$$\alpha_f(\omega) = \alpha_{f'}(\mathbf{D}^{T^{-1}} \omega) \cdot \frac{1}{|\det(\mathbf{D})|} = \alpha_{f'}(\mathbf{D}^{-1} \cdot \omega) \cdot \frac{1}{|\det(\mathbf{D})|}.\quad (4.204)$$

Wir sehen: bis auf einen Faktor verhält sich das Spektrum gegenläufig, d. h. wird das Bild f vergrößert, verkleinert sich entsprechend das Spektrum und umgekehrt. In der Praxis treten aber alle diese Transformationen in der Regel zusammen mit einer Verschiebung auf, daher bilden wir das Amplitudenspektrum:

$$|\alpha_f(\omega)| = \frac{1}{|\det(\mathbf{A})|} |\alpha_{f'}(\mathbf{A}^{T^{-1}} \omega)|.\quad (4.205)$$

Wir sehen in (4.205), dass das Amplitudenspektrum **invariant** gegenüber Verschiebungen ist, aber noch abhängig von den übrigen Parametern der affinen Transformation ist. Wollen wir folglich die Verschiebungen gleichzeitig berücksichtigen, müssen wir in den Beziehungen (4.203) und (4.204) zu den Beträgen übergehen.

4.27 Lokales Energiemodell

4.27.1 Lokale Energie

Häufig werden in der Bildverarbeitung Begriffe aus der Physik entlehnt, obwohl der physikalische Aspekt, der hinter dem Begriff steckt, im Bild überhaupt nicht gegeben ist. So werden z. B. ständig Energiebegriffe verwendet.

Im Folgenden betrachten wir nur reelle Funktionen und benutzen die AFT mit der Darstellung (3.41), wobei wir $a_0 = 0$ annehmen:

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos\left(2\pi k \frac{x}{X} + \varphi_k\right). \quad (4.206)$$

Die Hilbertrtansformierte $h(x)$ von $f(x)$ ist (siehe Abschn. 3.7):

$$h(x) = \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \sin\left(2\pi k \frac{x}{X} + \varphi_k\right). \quad (4.207)$$

Ein Energievektor $\mathbf{e}(x)$ ist dann durch:

$$\mathbf{e}(x) = \frac{1}{2} [f(x), h(x)]^T \quad (4.208)$$

definiert. Eine lokale, skalare Energiefunktion $E(x)$ ist dann der Betrag dieses Vektors:

$$\begin{aligned} E^2(x) &= f^2(x) + h^2(x) \\ &= \left[\frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos\left(2\pi k \frac{x}{X} + \varphi_k\right) \right]^2 + \left[\frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \sin\left(2\pi k \frac{x}{X} + \varphi_k\right) \right]^2. \end{aligned} \quad (4.209)$$

Es sei nun Θ_e der Richtungswinkel des Energievektors $\mathbf{e}(x)$. Wenn wir das Skalarprodukt von $\mathbf{e}(x)$ mit einem Einheitsvektor in diese Richtung bilden, erhalten wir den Betrag von $\mathbf{e}(x)$:

$$\begin{aligned} E(x) &= \left[\frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos\left(2\pi k \frac{x}{X} + \varphi_k\right), \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \sin\left(2\pi k \frac{x}{X} + \varphi_k\right) \right] \cdot [\cos \Theta_e, \sin \Theta_e]^T \\ &= \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos\left(2\pi k \frac{x}{X} + \varphi_k\right) \cdot \cos \Theta_e + \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \sin\left(2\pi k \frac{x}{X} + \varphi_k\right) \cdot \sin \Theta_e \\ &= \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos\left(2\pi k \frac{x}{X} + \varphi_k - \Theta_e\right). \end{aligned} \quad (4.210)$$

Der Winkel ist somit gleich:

$$\tan \Theta_e = \frac{\sum_{k=1}^{\infty} A_k \sin(2\pi k \frac{x}{X} + \varphi_k)}{\sum_{k=1}^{\infty} A_k \cos(2\pi k \frac{x}{X} + \varphi_k)}. \quad (4.211)$$

Die Beziehung (4.210) kann man nun durch die Extremalaufgabe:

$$E(x) = \max_{\Theta} \left(p(x, \theta) = \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos \left(2\pi k \frac{x}{X} + \varphi_k - \Theta \right) \right) \quad (4.212)$$

ausdrücken, wobei die Lösung für Θ die Größe Θ_e aus (4.210) ist. Wie in (3.94) und (3.96) betrachten wir streng analytische Funktionen $z_f(t) = f(t) + i\beta_f(t)$, $z_g(t) = g(t) + i\beta_g(t)$ mit $E^2(f) = E^2(z_f) = f^2 + \beta_f^2$ und $E^2(g) = E^2(z_g) = g^2 + \beta_g^2$. Wir identifizieren immer den Realteil als das Bild, welches wir eigentlich betrachten. Nun gilt:

$$E(z_f(t) \cdot z_g(t)) = E(f) \cdot E(g). \quad (4.213)$$

Dies wollen wir zeigen:

$$\begin{aligned} E^2(z_f(t) \cdot z_g(t)) &= [fg - \beta_f \beta_g]^2 + [f\beta_g + g\beta_f]^2 \\ &= f^2 g^2 - 2fg\beta_f \beta_g + \beta_f^2 \beta_g^2 + f^2 \beta_g^2 + 2fg\beta_f \beta_g + g^2 \beta_f^2 \\ &= (f^2 + \beta_f^2)(g^2 + \beta_g^2) = E^2(f) \cdot E^2(g). \end{aligned} \quad (4.214)$$

Daher gilt für die komplexe Division analog:

$$E(z_f/z_g) = \frac{E(f) \cdot E(g)}{E^2(g)}. \quad (4.215)$$

4.27.2 Phasenkongruenz

Nun wollen wir die Phasenkongruenz (*phase congruency*) einführen. Dazu benötigen wir zunächst ein gewichtetes Maß für die Varianz der Phasen:

$$r(x, \Theta) = \frac{\sum_{k=1}^{\infty} A_k (2\pi k \frac{x}{X} + \varphi_k - \Theta)^2}{\pi^2 \sum_{k=1}^{\infty} A_k}. \quad (4.216)$$

Damit dieses für Winkel überhaupt funktionieren kann, müssen wir die Phasen $2\pi k \frac{x}{X} + \varphi_k - \Theta \bmod(2\pi)$ berechnen und in das Intervall $(-\pi, +\pi]$ verschieben. Die gewichtete Phasenvarianz r liegt nun im Intervall $[0, 1]$. Maximale *Phasenkongruenz* bedeutet nun minimale Phasenvarianz. Bei gegebenem, festen x ist nun Θ derart gesucht, dass die Phasenvarianz r minimal wird. Wir wissen aus der Stochastik, dass der Erwartungswert die Varianz minimiert, folglich ist das gewichtete Mittel:

$$\Theta_r = \frac{\sum_{k=1}^{\infty} A_k (2\pi k \frac{x}{X} + \varphi_k)}{\sum_{k=1}^{\infty} A_k} \quad (4.217)$$

die Größe, die die Phasenvarianz minimiert und damit die Phasenkongruenz maximiert. Eine vorzeichenbehafteter mittlerer Wert ist natürlich bei Winkelberechnungen etwas problematisch. Gibt es nun einen Zusammenhang zwischen Θ_r und Θ_e ? Da Θ_e die Funktion $p(x, \Theta)$ maximiert (4.212), muss für kleine Phasen $2\pi k \frac{x}{X} + \varphi_k - \Theta_e \approx 0$ gelten:

$$\begin{aligned} E(x) &= \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \cos \left(2\pi k \frac{x}{X} + \varphi_k - \Theta_e \right) \\ \rightarrow \frac{\partial E(x)}{\partial \Theta_e} &= \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \sin \left(2\pi k \frac{x}{X} + \varphi_k - \Theta_e \right) \\ &\approx \frac{1}{\sqrt{X}} \sum_{k=1}^{\infty} A_k \left(2\pi k \frac{x}{X} + \varphi_k - \Theta_e \right) = 0. \end{aligned} \quad (4.218)$$

Daraus folgt

$$\Theta_e \approx \frac{\sum_{k=1}^{\infty} A_k (2\pi k \frac{x}{X} + \varphi_k)}{\sum_{k=1}^{\infty} A_k} = \Theta_r. \quad (4.219)$$

Damit können wir für kleine Phasen $2\pi k \frac{x}{X} + \varphi_k - \Theta_e \approx 0$ einen direkten Zusammenhang zwischen der lokalen Energie und der Phasenkongruenz (bzw. Phasenvarianz) herstellen:

$$\begin{aligned} \frac{1}{\pi^2} - \frac{1}{2} r(x, \Theta_r) &= \frac{\sum_{k=1}^{\infty} A_k - \frac{1}{2} \sum_{k=1}^{\infty} A_k (2\pi k \frac{x}{X} + \varphi_k - \Theta_r)^2}{\pi^2 \sum_{k=1}^{\infty} A_k} \\ &= \frac{\sum_{k=1}^{\infty} A_k (1 - \frac{1}{2} (2\pi k \frac{x}{X} + \varphi_k - \Theta_r)^2)}{\pi^2 \sum_{k=1}^{\infty} A_k} \\ &\approx \frac{\sum_{k=1}^{\infty} A_k (\cos(2\pi k \frac{x}{X} + \varphi_k - \Theta_e))}{\pi^2 \sum_{k=1}^{\infty} A_k} = \frac{E(x)}{\pi^2 \sum_{k=1}^{\infty} A_k}. \end{aligned} \quad (4.220)$$

Damit ist:

$$E(x) = \sum_{k=1}^{\infty} A_k \left(1 - \frac{\pi^2}{2} r(x, \Theta_r) \right). \quad (4.221)$$

In der Literatur macht man manchmal einen „Trick“. Obige Beziehung für kleine Phasen wird benutzt um gleich statt der Phasenvarianz die Phasenkongruenz $PC(x)$ zu definieren:

$$PC(x) = \max_{\Theta} \left(\frac{\sum_{k=1}^{\infty} A_k (\cos(2\pi k \frac{x}{X} + \varphi_k - \Theta))}{\sum_{k=1}^{\infty} A_k} \right). \quad (4.222)$$

Diese ist dann selbst normiert mit $0 \leq PC(x) \leq 1$ und es gilt die Beziehung $E(x) = c \cdot PC(x)$, d. h. $E(x)$ und $PC(x)$ unterscheiden sich generell nur durch einen Faktor. Wenn die Phasen aber nicht klein sind, dann stimmt diese Beziehung insofern nicht, dass die Phasenkongruenz nicht das richtige Maß mehr ist, da diese dann zu sehr von der Phasenvarianz abweicht.

4.27.3 Funktionen mit hoher Phasenkongruenz

Wir betrachten noch einmal die Fourierdarstellung der Rechteckfunktion (4.94):

$$f\left(x - \frac{X}{4}\right) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{1}{2k-1} \sin\left[\frac{2\pi(2k-1)x}{X}\right]. \quad (4.223)$$

Man sieht, es ist $r(0, 0) = 0$ und damit liegt bei $x = 0$ maximale Phasenkongruenz vor. Bei $x = 0$ hat die Rechteckfunktion (4.94) eine ideale Stufenkante. Daher deuten Stellen mit großer Phasenkongruenz auf Kanten im Bild hin.

Nehmen wir weiterhin einmal die Darstellung des Dirichletschen Integralkernes (7.2):

$$D_n(x) = \sum_{k=-n}^n 1 \cdot e^{+2\pi i k \frac{x}{X}} = 1 + 2 \cdot \sum_{k=1}^n \cos\left(2\pi k \frac{x}{X}\right). \quad (4.224)$$

Bei $x = 0$ liegt wieder maximale Phasenkongruenz vor, beschrieben werden durch die Summen allerdings Linien einer bestimmten Dicke. Daher besitzt die „Mitte“ einer Linie auch hohe Phasenkongruenz.

In den folgenden Abschnitten soll das Abtasttheorem hergeleitet und interpretiert werden. Eine verbale Einführung dazu gab es im Abschn. 1.1. Je nach Bildmodell gibt es ein Abtasttheorem für Funktionen im endlichen Intervall oder im unendlichen Intervall. Die Herleitung unterscheidet sich nur durch die Art der verwendeten Mathematik, üblicherweise wird dies fast ausschließlich im unendlichen Intervall getan, dabei braucht man aber uneigentliche Integrale und Reihen. Viel einfacher ist die Mathematik im endlichen Intervall. Das mathematische Ergebnis kann demnach nicht formal gleich sein, die Interpretation ist aber dieselbe.

5.1 Abtasttheorem im endlichen Intervall

Nun soll das Abtasttheorem für eine analoge Funktion im Intervall $[0, X]$ hergeleitet werden. Dazu ist es erforderlich zu erklären, was ein trigonometrisches Interpolationspolynom ist. Gegeben sei im Intervall $[0, X]$ eine analoge Funktion $f_a(x)$. Wir betrachten an n äquidistanten Stützstellen

$$x_n = n \frac{X}{N}, \quad n = 0, 1, \dots, N - 1 \quad (5.1)$$

die Funktionswerte $f_a(x_n)$, $n = 0, 1, \dots, N - 1$ und bezeichnen diese mit f_n , $n = 0, 1, \dots, N - 1$. Von dieser diskreten Funktion f berechnen wir die diskreten Fourierkoeffizienten:

$$\alpha_k = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l e^{-2\pi i k \frac{l}{N}} = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} f_l e^{-2\pi i k \frac{l}{N}}. \quad (5.2)$$

Dann ist die von einem ganzzahligen Index k_0 und den diskreten Fourierkoeffizienten α_k abhängige Funktion:

$$f_{k_0}(x) = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{x}{X}} \quad (5.3)$$

ein trigonometrisches Interpolationspolynom. Dies ist recht einfach zu zeigen. Wir bilden:

$$\begin{aligned} f_{k_0}(x_l) &= \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{x_l}{X}} = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{l \frac{X}{N}}{X}} \\ &= \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{l}{N}}. \end{aligned} \quad (5.4)$$

Auf der rechten Seite steht gerade die inverse, diskrete Fouriertransformation. Da die diskreten Fourierkoeffizienten selbst mit N periodisch sind, gilt nun:

$$f_{k_0}(x_l) = \frac{1}{\sqrt{N}} \sum_{k=k_0}^{N-1+k_0} \alpha_k e^{2\pi i k \frac{l}{N}} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{2\pi i k \frac{l}{N}} = f_l = f_a(x_l). \quad (5.5)$$

Damit haben wir die Interpolationseigenschaft für beliebige k_0 gezeigt. Im Gegensatz zu $f_{k_0}(x_l) = f_a(x_l)$ hängt aber das Polynom $f_{k_0}(x)$ vom Index k_0 ab, d. h. für verschiedene k_0 erhalten wir auch verschiedene Interpolationspolynome.

Es sei nun eine analoge Funktion $f_a(x)$ im Intervall $[0, X]$ gegeben. Die AFT lautet dann

$$f_a(x) = \frac{1}{\sqrt{X}} \sum_{k=-\infty}^{\infty} \beta_k e^{+2\pi i k \frac{x}{X}} \leftrightarrow \beta_k = \frac{1}{\sqrt{X}} \int_0^X f_a(x) e^{-2\pi i k \frac{x}{X}} dx. \quad (5.6)$$

Wir tasten nun die analoge Funktion $f_a(x)$ an den äquidistanten Stützstellen $x_n = \frac{n}{N} X$ ab und erhalten damit die diskrete Funktion $f_d(x_n)$ mit

$$f_a(x_n) = f_d(x_n), \quad n = 0, 1, \dots, N-1. \quad (5.7)$$

Satz 5.1 (Abtasttheorem) *Ist die analoge Funktion bandbegrenzt im folgenden Sinne:*

$$f_a(x) = \frac{1}{\sqrt{X}} \sum_{k=k_0}^{N-1+k_0} \beta_k e^{+2\pi i k \frac{x}{X}}, \quad (5.8)$$

dann kann die analoge Funktion $f_a(x)$ vollständig aus der diskreten Funktion $f_d(x_n)$ rekonstruiert werden und ist gleich dem trigonometrischen Interpolationspolynom $f_{k_0}(x)$.

Beweis Den Beweis führen wir o. B. d. A. für $k_0 = 0$. Zweifelsohne ist

$$z(x) = f_0(x) - f_a(x) = 0, \quad x = x_l, \quad l = 0, 1, \dots, N-1. \quad (5.9)$$

Die Funktion $z(x)$ besitzt also gerade an den Abtaststellen Nullstellen. Wir wollen aber zeigen, dass $z(x)$ identisch Null ist. Wir setzen nun ein:

$$z(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \alpha_k e^{+2\pi i k \frac{x}{N}} - \frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \beta_k e^{+2\pi i k \frac{x}{N}} = 0, \quad x = x_l, \quad l = 0, 1, \dots, N-1. \quad (5.10)$$

Wichtig ist: Die α_k sind die diskreten Fourierkoeffizienten aus der DFT und die β_k sind die diskreten Fourierkoeffizienten der AFT im Intervall. Wir fassen sie Summen zusammen:

$$z(x) = \sum_{k=0}^{N-1} \left(\frac{\alpha_k}{\sqrt{N}} - \frac{\beta_k}{\sqrt{X}} \right) e^{+2\pi i k \frac{x}{N}} = 0, \quad x = x_l, \quad l = 0, 1, \dots, N-1. \quad (5.11)$$

Kürzen wir die Koeffizienten mit γ_k ab, so ist

$$\begin{aligned} z(x) &= \sum_{k=0}^{N-1} \gamma_k e^{+2\pi i k \frac{x}{N}} = 0, \quad x = x_l, \quad l = 0, 1, \dots, N-1 \\ \Leftrightarrow z(x_l) &= z_l = \sum_{k=0}^{N-1} \gamma_k e^{+2\pi i k \frac{x_l}{N}} = 0 \end{aligned} \quad (5.12)$$

ein lineares homogenes Gleichungssystem in γ_k . Damit $z(x)$ identisch Null ist, müssen wir zeigen, dass die triviale Lösung $\gamma_k = 0$ die einzige Lösung ist. Zunächst vergessen wir die FT und zeigen dies nur mit Mitteln der linearen Algebra. Dazu betrachten wir die Koeffizientenmatrix dieses linearen Gleichungssystems, die wir mit \mathbf{A} bezeichnen wollen:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i \frac{1}{N}} & e^{2\pi i \frac{2}{N}} & \cdots & e^{2\pi i \frac{N-1}{N}} \\ 1 & e^{2\pi i \frac{2}{N}} & e^{2\pi i \frac{4}{N}} & \cdots & e^{2\pi i \frac{2(N-1)}{N}} \\ 1 & e^{2\pi i \frac{3}{N}} & e^{2\pi i \frac{8}{N}} & \cdots & e^{2\pi i \frac{3(N-1)}{N}} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & e^{2\pi i \frac{N-1}{N}} & e^{2\pi i \frac{2(N-1)}{N}} & \cdots & e^{2\pi i \frac{(N-1)(N-1)}{N}} \end{pmatrix}. \quad (5.13)$$

Diese Matrix hat die Form der Vandermonde-Matrix:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \varphi_0 & \varphi_1 & \varphi_2 & \cdots & \varphi_{N-1} \\ \varphi_0^2 & \varphi_1^2 & \varphi_2^2 & \cdots & \varphi_{N-1}^2 \\ \varphi_0^3 & \varphi_1^3 & \varphi_2^3 & \cdots & \varphi_{N-1}^3 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \varphi_0^{N-1} & \varphi_1^{N-1} & \varphi_2^{N-1} & \cdots & \varphi_{N-1}^{N-1} \end{pmatrix}. \quad (5.14)$$

Dabei ist $\varphi_l = e^{2\pi i \frac{l}{N}}$. Es gilt $\det(\mathbf{A}) \neq 0$ genau dann wenn $\varphi_l \neq \varphi_j$ für $i \neq j$ gilt. Dies ist in der Tat erfüllt, sodass das Gleichungssystem nur die triviale Lösung haben kann. Damit folgt aber mit $z(x) \equiv 0$ die Identität $f_0(x) \equiv f_a(x)$, was zu zeigen war. \square

Den Beweis mit der Vandermonde-Matrix haben wir deshalb hier gebracht, weil wir dieses Prinzip noch im Abschn. 5.4.4 benötigen.

Nun schauen wir uns die γ_k in (5.12) genauer an. Die γ_k sind bis auf einen Faktor identisch mit den diskreten Fourierkoeffizienten, daher muss gelten:

$$\gamma_k = C \cdot \sum_{l=0}^{N-1} f_l e^{-2\pi i \frac{kl}{N}}. \quad (5.15)$$

Setzen wir nun $f_l = 0, l = 0, \dots, N-1$ ein, so folgt als einzige Lösung $\gamma_k = 0, k = 0, \dots, N-1$. Nun wollen wir noch eine Herleitung betrachten, bei der eine variable Anzahl von Abtastpunkten erlaubt ist. Bisher haben wir nicht beachtet, dass die Vandermondesche Matrix sogar eine Orthogonalmatrix ist. Dazu betrachten wir nun M Abtastpunkte mit $M \geq N$. Diese setzen wir in die bandbegrenzte Fourierreihe ein:

$$\begin{aligned} f_a(x_l) &= \frac{1}{\sqrt{X}} \sum_{k=k_0}^{N-1} \beta_k e^{2\pi i k \frac{l}{M}} \\ &\rightarrow \sum_{k=k_0}^{N-1} \beta_k e^{2\pi i k \frac{l}{M}} = f_a(x_l) \cdot \sqrt{X}, l = 0, 1, \dots, M-1. \end{aligned} \quad (5.16)$$

Nun versuchen wir dieses lineare Gleichungssystem in den Unbekannten β_k direkt zu lösen. Dazu sehen wir uns wieder die Koeffizientenmatrix mit $k_0 = 0$ an:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i \frac{1}{M}} & e^{2\pi i \frac{2}{M}} & \cdots & e^{2\pi i \frac{N-1}{M}} \\ 1 & e^{2\pi i \frac{2}{M}} & e^{2\pi i \frac{4}{M}} & \cdots & e^{2\pi i \frac{2 \cdot (N-1)}{M}} \\ 1 & e^{2\pi i \frac{3}{M}} & e^{2\pi i \frac{8}{M}} & \cdots & e^{2\pi i \frac{3 \cdot (N-1)}{M}} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & e^{2\pi i \frac{M-1}{M}} & e^{2\pi i \frac{2 \cdot (M-1)}{M}} & \cdots & e^{2\pi i \frac{(M-1) \cdot (N-1)}{M}} \end{pmatrix}. \quad (5.17)$$

Wenn wir uns an die Basisfunktionen der DFT mit M Stützstellen erinnern, dann hatten wir vereinbart: die k -te diskrete Basisfunktion an der Stelle n war $\varphi_n^k = \frac{1}{\sqrt{M}} e^{2\pi i k \frac{n}{M}}$. In der Matrix (5.17) stehen genau als Spaltenvektoren die ersten N Basisfunktionen der insgesamt M Basisfunktionen. Diese Matrix ist also spaltenorthonormal, allerdings bis auf den Faktor $\frac{1}{\sqrt{M}}$. Es sei nun $\boldsymbol{\beta}$ der Spaltenvektor aller β_k und \mathbf{f}_a der Spaltenvektor aller $f_a(x_l)$. Dann lässt sich das Gleichungssystem (5.16) mit $k_0 = 0$ auch so formulieren:

$$\frac{1}{\sqrt{M}} \mathbf{A} \cdot \boldsymbol{\beta} = \mathbf{f}_a \cdot \sqrt{X} \cdot \frac{1}{\sqrt{M}}. \quad (5.18)$$

Da \mathbf{A} (mit dem Faktor) spaltenorthonormal ist, können wir auflösen:

$$\boldsymbol{\beta} = \frac{1}{\sqrt{M}} \overline{\mathbf{A}}^T f_a \cdot \sqrt{X} \cdot \frac{1}{\sqrt{M}}. \quad (5.19)$$

Nun schreiben wir wieder die k -te Komponente von $\boldsymbol{\beta}$ auf:

$$\beta_k = \frac{\sqrt{X}}{\sqrt{M}} \cdot \frac{1}{\sqrt{M}} \sum_{l=0}^{M-1} f_a(x_l) e^{-2\pi i k \frac{l}{M}}, \quad k = 0, 1, \dots, N-1. \quad (5.20)$$

Auf der rechten Seite kommen als Faktoren die diskreten Fourierkoeffizienten vor. Folglich ist:

$$\beta_k = \frac{\sqrt{X}}{\sqrt{M}} \cdot \alpha_k. \quad (5.21)$$

Wir sehen, das ist das gleiche Ergebnis $\gamma_k = 0$ wie im Beweis mit den Beziehungen (5.16) und (5.12). Diese Beziehung gilt natürlich nicht nur für $k_0 = 0$, sondern für beliebiges k_0 . Man muss dabei nur an die Periodizität der α_k denken:

$$\beta_k = \frac{\sqrt{X}}{\sqrt{M}} \cdot \alpha_{k+m \cdot M}, \quad k = k_0, \dots, k_0 + N - 1, \quad m \in \mathbb{Z}. \quad (5.22)$$

Bei dieser Herleitung hatten wir $M \geq N$ vorausgesetzt.

Was passiert eigentlich, wenn eine Unterabtastung mit $M < N$ vorliegt? Dann ist trivialerweise das Gleichungssystem nicht mehr eindeutig lösbar und damit die bandbegrenzte Funktion nicht mehr eindeutig rekonstruierbar. Aber vielleicht ist doch noch etwas an Information zu retten? Wenn $M < N$ gilt, dann bilden die ersten M Spalten der Matrix \mathbf{A} aus (5.17) ein Orthonormalsystem (mit dem Faktor $\frac{1}{\sqrt{M}}$). Auf Grund der Periode M (da $M < N$ ist M Periode) wiederholt sich dann dieses Orthonormalsystem, allerdings davon nur die ersten $N - M$ Spalten, da wir nur N Spalten der Matrix \mathbf{A} haben. Wenn wir folglich wie bei der bisherigen Herleitung die linke und rechte Seite des Gleichungssystems mit der konjugiert komplexen, transponierten Matrix von \mathbf{A} multiplizieren, entsteht folgende Gaußsche Normalenmatrix:

$$\frac{1}{\sqrt{M}} \overline{\mathbf{A}}^T \cdot \frac{1}{\sqrt{M}} \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ & & & \vdots & & & & \vdots & & \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ & & & \vdots & & & & \vdots & & \end{pmatrix}. \quad (5.23)$$

Man sieht, in $2M - N$ Zeilen stehen genau jeweils eine 1, und in $2(N - M)$ Zeilen stehen jeweils genau zwei solche 1. Die Zeilen, in denen zwei Einsen stehen, lassen sich folglich nicht nach den entsprechenden zwei Unbekannten auflösen, dagegen lassen sich die Variablen, die den Zeilen mit genau einer Eins entsprechen, wie vorher exakt eliminieren. Bei obiger Matrix haben wir $M < N$ angenommen, dies reicht aber noch nicht, genaugenommen haben wir die Ungleichung $\frac{N}{2} < M < N$ angenommen. Ist nämlich $M \leq \frac{N}{2}$, dann entstehen in allen Zeilen weitere Einsen, da die Periode M ist, folglich können wir keine einzige Unbekannte mehr eindeutig eliminieren. Ist dagegen $\frac{N}{2} < M < N$, dann können wir trotz Unterabtastung noch $2M - N$ Unbekannte β_k eliminieren. Alle Ausführungen bezo gen sich bisher auf N komplexe Fourierkoeffizienten und M Stützstellen, d. h. auch für komplexwertige Funktionen. Wie ist dies aber nun speziell bei reellen Funktionen mit einer Bandbegrenzung bezüglich der tiefen Frequenzen? Dazu nehmen wir die Bandbegrenzung einer reellen Funktion in der folgenden Form an:

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-K}^{+K} \beta_k e^{+2\pi i k \frac{x}{X}}. \quad (5.24)$$

Für reelle Funktionen gilt dann $\beta_k = \bar{\beta}_{-k}$. Für die bandbegrenzte Funktion $f(x)$ gilt dann $N = 2K + 1$ und die Grenzfrequenz ist K . Wir benötigen also zur eindeutigen Rekonstruktion mindestens $2K + 1$ Abtastpunkte. Setzen wir $N = 2K + 1$ in die Ungleichung $\frac{N}{2} < M < N$ ein, so folgt die Ungleichung $K < M \leq 2K$. Grob gesprochen heißt dies: Wir müssen mehr Abtastpunkte als die Grenzfrequenz zur Verfügung haben, um wenigstens einige Frequenzen im Signal wiederherstellen zu können. Ist die Grenzfrequenz höher als die Anzahl der Abtastpunkte, dann ist gar nichts mehr rekonstruierbar. Welche Frequenzen sind es nun konkret, die dann noch rekonstruierbar sind? Dazu betrachten wir die Matrix (5.17), allerdings mit $k_0 = -K$. Dann sehen wir, dass die Zeilen mit den beiden Einsen den hohen Frequenzen und die Zeilen mit nur einer Eins den tiefen Frequenzen entsprechen. Wenn sich M und N um $m = N - M$ unterscheiden, dann sind die oberen m Frequenzen nicht mehr rekonstruierbar, aber die tiefen $K - m$ Frequenzen sind eindeutig rekonstruierbar. Dieses ist für Registierungsaufgaben sehr nützlich. Dazu ein

Beispiel Wir nehmen an, wir haben zwei unterabgetastete Bilder mit Aliasing-Effekten und die Unterabtastung bezüglich der Grenzfrequenz entspricht obigen Bedingungen $K < M \leq 2K$. Weiter nehmen wir an, die analogen, noch nicht abgetasteten Bilder sind mit K bandbegrenzt und unterscheiden sich durch eine Translation, die wir aber nicht kennen. Wir kennen nur die beiden unterabgetasteten, durch Aliasing verfälschten Bilder. Wir wollen nun ausschließlich aus den unterabgetasteten Bildern die Translation mit Subpixelgenauigkeit bestimmen, folglich eine Registrierung vornehmen, siehe auch Abschn. 19.8.4. Auf Grund der Aliasingeffekte scheint dies unmöglich zu sein. Vorausgesetzt, unsere Annahmen stimmen tatsächlich, filtern wir beide Bilder mit idealen Tiefpassen derart, dass die hohen Frequenzen, die durch das Aliasing verfälscht wurden, eliminiert werden. Durch unsere obigen Annahmen sind die tiefen Frequenzen noch richtig. Im Ortsraum wäre das eine Faltung. Wie wir wissen, ist die Faltung verschiebungsinvariant. Folglich können wir die tiefpassgefilterten Bilder registrieren und erhalten die exakte Subpixel-Translation. Dies

geht natürlich nur, wenn das Aliasing nicht zu drastisch ist, d. h. nur wenn die Ungleichung $K < M \leq 2K$ tatsächlich erfüllt ist. Da wir in der Praxis die Grenzfrequenz K selten kennen, wissen wir auch nicht, wie unser idealer Tiefpass zu wählen ist. Nun könnte man auf die Idee kommen und schlussfolgern, dass man nur die erste Frequenz oder nur einige tiefe Frequenzen übrig lässt. Das hat aber den Nachteil, dass dann kaum Information übrig ist und man daraus nicht unbedingt die Translation bestimmen kann. Man wird folglich empirisch die Größe des Tiefpasses wählen müssen.

Zusammenfassung

- Wir nehmen an, dass das reelle analoge Signal im Sinne (5.24) bandbegrenzt ist. Mit anderen Worten: wir kennen a priori das Frequenzband bestehend aus genau $N = 2K + 1$ komplexen Fourierkoeffizienten. Nun tasten wir das Signal an M Stellen äquidistant ab. Für die Anzahl der Abtaststellen M gelte die Beziehung $M \geq N$ ($M > 2K$). Dann können wir allein aus der Kenntnis der Abtastwerte und den daraus mit der DFT berechneten diskreten Fourierkoeffizienten die Fourierkoeffizienten des analogen Signals berechnen und damit das analoge Signal vollständig rekonstruieren. Das rekonstruierte Signal ist dann identisch mit dem trigonometrischen Interpolationspolynom.
- Haben wir dagegen weniger Abtaststellen M als die Anzahl N der Fourierkoeffizienten des analogen Signals, d. h. weniger als die doppelte Grenzfrequenz plus eins, dann liegt eine Unterabtastung vor. Es kann Aliasing auftreten und das Signal ist nicht mehr eindeutig rekonstruierbar. Ist allerdings $\frac{N}{2} < M < N$ ($K < M \leq 2K$), so ist der Übergang „fließend“, es tritt nicht gleich eine „Katastrophe“ auf, es sind in diesem Falle immer noch einzelne Frequenzen vollständig rekonstruierbar.
- Ist $M < \frac{N}{2}$ ($M \leq K$), dann ist aus den Abtastwerten nichts mehr eindeutig vom Original bestimmbar.
- Treten bei Translations-Registrierungsaufgaben Probleme bezüglich einer Unterabtastung auf, dann sollte man die Bilder mit einem idealen Tiefpassfilter transformieren. Die Größe des idealen Tiefpassfilters muss leider empirisch festgelegt werden. Die gefilterten Bilder kann man nun einer Translations-Registrierung unterziehen.
- Ideale Tiefpassfilter werden in der Bildverarbeitung selten verwendet. Sie erzeugen z. B. die Ringing-Artefakte, die in der Datenkompression überhaupt nicht erwünscht sind. Jetzt haben wir mit der Translations-Registrierung ein schönes Beispiel, wo sie sogar erforderlich sein können.

5.2 Abtasttheorem für das unendliche Bildmodell

Nun betrachten wir analoge Funktionen $f_a(t)$ im Intervall $-\infty < t < +\infty$ und verwenden die Fourierintegraltransformation:

$$\alpha_a(\omega) = \int_{-\infty}^{\infty} f_a(t) e^{-i\omega t} dt, \quad f_a(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \alpha_a(\omega) e^{+i\omega t} d\omega. \quad (5.25)$$

Nun benötigen wir die dazu passende diskrete FT, nicht die DFT, sondern die UDFT:

$$\alpha_d(\omega) = \sum_{n=-\infty}^{\infty} f_d(n)e^{-i\omega n}, \quad f_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \alpha_d(\omega)e^{+i\omega n} d\omega. \quad (5.26)$$

Man könnte auch das Integralspektrum verwenden, dann müssen wir aber die ideale Abtastfunktion auf die Funktion anwenden, siehe Abschn. 2.39 und 4.18. Wir wollen nun die analoge Funktion an äquidistanten Stützstellen mit der Schrittweite T abtasten und untersuchen, ob wir aus dem Spektrum dieser diskreten Funktion das Spektrum der analogen Funktion berechnen können, damit könnten wir trivialerweise auch die analoge Funktion allein aus den Abtastwerten vollständig rekonstruieren. Dies ist die gleiche Idee wie beim Abtasttheorem im endlichen Intervall, wir benötigen eine entsprechende Beziehung wie (5.22). Wir bilden also die Abtastpunkte $n \cdot T, n \in \mathbb{Z}$ mit $f_d(n) = f_a(nT)$. Damit ist:

$$\begin{aligned} f_d(n) &= f_a(nT) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \alpha_a(\omega)e^{i\omega nT} d\omega \\ &= \sum_{l=-\infty}^{\infty} \frac{1}{2\pi} \int_{\frac{(2l-1)\pi}{T}}^{\frac{(2l+1)\pi}{T}} \alpha_a(\omega)e^{i\omega nT} d\omega. \end{aligned} \quad (5.27)$$

Wir summieren also über Integrale mit den Integrationslängen von $\frac{2\pi}{T}$. Wir substituieren $\omega = v + \frac{2\pi l}{T}$ und bilden:

$$\begin{aligned} \int_{\frac{(2l-1)\pi}{T}}^{\frac{(2l+1)\pi}{T}} \alpha_a(\omega)e^{i\omega nT} d\omega &= \int_{\frac{-\pi}{T}}^{+\frac{\pi}{T}} \alpha_a\left(v + \frac{2\pi l}{T}\right) e^{ivnT} e^{i2\pi ln} dv \\ &= \int_{\frac{-\pi}{T}}^{+\frac{\pi}{T}} \alpha_a\left(v + \frac{2\pi l}{T}\right) e^{ivnT} dv. \end{aligned} \quad (5.28)$$

Daraus folgt:

$$f_d(n) = f_a(nT) = \frac{1}{2\pi} \int_{-\frac{\pi}{T}}^{+\frac{\pi}{T}} \sum_{l=-\infty}^{\infty} \alpha_a\left(v + \frac{2\pi l}{T}\right) e^{ivnT} dv. \quad (5.29)$$

Wir substituieren nochmals $v = \frac{\omega}{T}$ und erhalten:

$$f_d(n) = f_a(nT) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[\frac{1}{T} \sum_{l=-\infty}^{\infty} \alpha_a\left(\frac{\omega}{T} + \frac{2\pi l}{T}\right) \right] e^{i\omega n} d\omega. \quad (5.30)$$

Wenn wir dies mit obigem Spektrum für $f_d(n)$ vergleichen, können wir sofort ablesen:

$$\alpha_d(\omega) = \frac{1}{T} \sum_{l=-\infty}^{\infty} \alpha_a \left(\frac{\omega}{T} + \frac{2\pi l}{T} \right), \quad (5.31)$$

oder wir schreiben dies in der Form:

$$\alpha_d(\omega T) = \frac{1}{T} \sum_{l=-\infty}^{\infty} \alpha_a \left(\omega + \frac{2\pi l}{T} \right). \quad (5.32)$$

Wir haben also eine Beziehung der Form $\alpha_d(\omega) = function(\alpha_a(\omega))$ gefunden. Eigentlich interessiert uns aber die inverse Beziehung entsprechend (5.22), da wir das analoge Spektrum aus dem diskreten rekonstruieren wollen. Das heißt aber, wir müssen untersuchen, wann die Beziehung (5.32) invertierbar ist. Das diskrete Spektrum ist eine additive Überlagerung von Translationen des analogen Spektrums. Diese Beziehung ist so nicht ohne weiteres nach α_a auflösbar. Wir müssen irgendwie garantieren, dass in der Summe der Translationen des analogen Spektrums das eigentliche originale Spektrum noch erkennbar ist. Dies würde gelingen, wenn die additiven Überlagerungen sich nicht gegenseitig „stören“. Dazu machen wir eine erste Annahme. Das analoge Spektrum α_a sei bandbegrenzt in folgendem Sinne:

$$\alpha_a(\omega) = 0 \quad \text{für } |\omega| \geq |\omega_0|. \quad (5.33)$$

Nun müssen wir das Abtastintervall T so wählen, dass sich die verschobenen und additiv überlagerten Spektren nicht gegenseitig „stören“, d. h.:

$$\left| \frac{2\pi(l+1)}{T} - \frac{2\pi l}{T} \right| \geq 2|\omega_0| \rightarrow \frac{\pi}{T} \geq |\omega_0| \rightarrow T \leq \frac{\pi}{|\omega_0|}. \quad (5.34)$$

Folglich können wir nun die Beziehung (5.32) für einen bestimmten Frequenzbereich auch so schreiben:

$$\alpha_d(\omega T) = \frac{1}{T} \alpha_a(\omega) \quad \text{für } |\omega| \leq |\omega_0| \leq \frac{\pi}{T}. \quad (5.35)$$

Jetzt können wir die Beziehung (5.35) nach α_a auflösen:

$$\alpha_a(\omega) = \begin{cases} T \cdot \alpha_d(\omega T) & \text{für } |\omega| \leq |\omega_0| \leq \frac{\pi}{T} \\ 0 & \text{sonst.} \end{cases} \quad (5.36)$$

Nun können wir den Ausdruck für α_a aus (5.36) in die IFT einsetzen:

$$\begin{aligned}
 f_a(x) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \alpha_a(\omega) e^{+i\omega x} d\omega = \frac{1}{2\pi} \int_{-\omega_0}^{+\omega_0} T \alpha_d(\omega T) e^{+i\omega x} d\omega \\
 &= \frac{1}{2\pi} \int_{-\omega_0}^{+\omega_0} T \left[\sum_{n=-\infty}^{+\infty} f_d(n) e^{-i\omega T n} \right] e^{+i\omega x} d\omega \\
 &= \frac{T}{2\pi} \sum_{n=-\infty}^{+\infty} f_d(n) \int_{-\omega_0}^{+\omega_0} e^{i\omega(x-nT)} d\omega \\
 &= \frac{T}{2\pi} \sum_{n=-\infty}^{+\infty} f_d(n) \left[\frac{1}{i(x-nT)} e^{+i\omega(x-nT)} \right]_{-\omega_0}^{+\omega_0} \\
 &= \frac{T}{\pi} \sum_{n=-\infty}^{+\infty} f_d(n) \frac{\omega_0}{2i(x-nT)\omega_0} [e^{+i\omega_0((x-nT))} - e^{-i\omega_0((x-nT))}] \\
 &= \frac{\omega_0 T}{\pi} \sum_{n=-\infty}^{+\infty} f_d(n) \frac{\sin(\omega_0(x-nT))}{\omega_0(x-nT)} \\
 &= \frac{\omega_0 T}{\pi} \sum_{n=-\infty}^{+\infty} f_d(n) \cdot \text{sinc}(\omega_0(x-nT)). \tag{5.37}
 \end{aligned}$$

Die Originalfunktion $f_a(x)$ lässt sich somit mit der sinc-Funktion als Interpolationspolynom allein aus den diskreten Werten wieder rekonstruieren. Die Formel (5.37) erinnert wieder an die „gemischte“ Faltung und tritt hier im Zusammenhang mit der Interpolation auf. Mit der Abkürzung $\text{sinc}_{\omega_0}(x) = \text{sinc}(\omega_0 x)$ gilt dann:

$$\begin{aligned}
 f_a(x) &= \frac{\omega_0 T}{\pi} \sum_{n=-\infty}^{+\infty} f_a(nT) \cdot \text{sinc}(\omega_0(x-nT)) \\
 &= \frac{\omega_0 T}{\pi} \sum_{n=-\infty}^{+\infty} f_a(nT) \cdot \text{sinc}_{\omega_0}(x-nT) \\
 &= \frac{\omega_0 T}{\pi} \int_{-\infty}^{\infty} \text{sinc}_{\omega_0}(x-\xi) \sum_{n=-\infty}^{+\infty} f_a(\xi) \delta(\xi-nT) d\xi. \tag{5.38}
 \end{aligned}$$

Es tritt wieder die Abtastung $f_\delta(x) = \sum_n f_a(x) \delta(x-nT)$ mit dem Dirac-Kamm auf. Folglich ist die Rekonstruktion darstellbar durch eine Faltung der sinc-Funktion mit dieser Abtastungsfunktion:

$$f_a(x) = \frac{\omega_0 T}{\pi} \cdot f_\delta(x) * \text{sinc}_{\omega_0}(x). \tag{5.39}$$

Bemerkung 1 Benutzen wir statt der Kreisfrequenz die „normalen“ Frequenzen, also $|\omega_0| = 2\pi|\nu_0| \leq \frac{\pi}{T}$, so erhalten wir $T \leq \frac{1}{2|\nu_0|}$. Dies entspricht der „üblichen“ Formulierung des Abtasttheorems, wenn wir eine zum Koordinatenursprung symmetrische Bandbegrenzung haben.

Bemerkung 2 Wir haben die Bandbegrenzung symmetrisch zum Koordinatursprung gewählt. Allgemein können wir die Bandbegrenzung auf ein beliebiges Intervall beziehen, z. B. $\frac{2\pi}{T} \geq \overline{\omega_0} - \underline{\omega_0}$. Daraus folgt $T \leq \frac{2\pi}{\overline{\omega_0} - \underline{\omega_0}}$ und mit $\omega = 2\pi\nu$ die Beziehung $T \leq \frac{1}{\overline{\nu_0} - \underline{\nu_0}}$.

Bemerkung 3 Das Frequenzband bezieht sich immer auf die komplexen Fourierkoeffizienten. Haben wir z. B. ein reelles Signal, dann gibt es gar keine einseitige Bandbegrenzung, da zu einem Fourierkoeffizienten $\alpha(\nu) \neq 0$ auch immer $\alpha(-\nu) \neq 0$ ist. Dann lautet mit $0 \in [\underline{\nu_0}, \overline{\nu_0}]$ die Abtastbedingung $T \leq \frac{1}{2(\overline{\nu_0} - \underline{\nu_0})}$.

Bemerkung 4 Welche Funktionen können denn von vornherein keine Bandbegrenzung haben? Dies sind z. B. Funktionen mit Sprüngen oder „Knickstellen“.

5.3 Abtasttheorem für 2D-Signale

Nun wollen wir das Abtasttheorem für 2D-Signale betrachten. Eine naheliegende Frage ist nun: Was soll denn da der grundlegende Unterschied zur äquidistanten Abtastung von 1D-Signalen sein? Die Antwort ist einfach: Außer der äquidistanten Abtastgeometrie im Rechteckgitter gibt es noch andere reguläre Gittergeometrien, man könnte die Abtastpunkte also auch noch anders verteilen als im Rechteckgitter. Es gibt in der Ebene aber nur drei homogene Gitter: Das Rechteckgitter, das Dreiecksgitter und das Hexagonalgitter. Zunächst formulieren wir einmal die Bandbegrenzung völlig unabhängig vom Gitter:

$$\alpha_a(\omega_1, \omega_2) = 0 \quad \text{für } |\omega_1| \geq |\omega_{10}|, |\omega_2| \geq |\omega_{20}|, \quad (5.40)$$

diese rechteckige Bandbegrenzung ist konsistent mit dem Rechteckgitter und führt zur Abtastbedingung:

$$T_1 \leq \frac{\pi}{\omega_{0x}}, \quad T_2 \leq \frac{\pi}{\omega_{0y}}.$$

Die Bandbegrenzung (5.40) als Rechteck ist aus praktischen Gründen völlig unrealistisch, warum sollte diagonal eine andere Bandbegrenzung sein als horizontal oder vertikal? Das würde bedeuten, die Bandbegrenzung hängt von der Rotation einer Kamera ab. Logischer ist deshalb eine Bandbegrenzung in Form eines Kreises:

$$\alpha_a(\omega_1, \omega_2) = 0 \quad \text{für } \omega_1^2 + \omega_2^2 \geq r^2, \quad (5.41)$$

wobei der Radius r des Kreises die eigentliche Bandbegrenzung beschreibt. Wenn man nun den Kreis in ein Quadrat einbettet (Quadrat mit den Seitenlängen r), dann gelten wieder die Abtastbedingungen (5.40) für das Quadratgitter. Man kann nun zeigen (siehe [21]), dass zu der Bandbegrenzung mit einem Kreis das Hexagonalgitter am besten konsistent ist und zwar in dem Sinne, dass man beim Hexagonalgitter 13,4 Prozent weniger Gitterpunkte benötigt als beim Quadratgitter und zwar bei gleicher Auflösung. Dies hätte durchaus praktische Konsequenzen, da man auf einem Chip 13,4 Prozent weniger Sensorelemente anbringen müsste ohne dass sich die Auflösung ändern würde.

5.4 Anwendungen des Abtasttheorems

5.4.1 Verkleinern von Bildern (Shrinking)

Eine unmittelbare Anwendung des Abtasttheorems finden wir beim Verkleinern von digitalen Bildern. Wir nehmen einmal ein eindimensionales Bild der Größe N an, dann haben wir genau N DFT-Koeffizienten. Eine simple Methode das Bild zu verkleinern ist: Wir lassen jedes zweite Pixel einfach weg, dann hat das Bild die Dimension $N/2$. Sind nun die hohen Frequenzen, also die oberen $N/2$ DFT-Koeffizienten zufällig Null, dann ist diese Art der Verkleinerung in Ordnung, ansonsten liegt eindeutig eine Unterabtastung vor. So mit könnten feinste Strukturen verschwinden oder Aliasing-Effekte auftreten. Um diese zu vermeiden müssten wir vorher das Bild mit einem Tiefpass filtern. Hier würde sich ein idealer Tiefpaß anbieten, wir setzen also die oberen $N/2$ -DFT-Koeffizienten Null, transformieren zurück und nehmen nur jedes zweite Pixel. Ideale Tiefpässe können aber diese Ringing-Artefakte erzeugen, folglich nehmen wir igendeinen Tiefpass im Frequenzraum oder im Ortsraum. Dadurch wird das Bild etwas unschärfer, dies ist aber der Preis, den man zahlen muss, um die Unterabtastung zu vermeiden.

5.4.2 Vergrößern von Bildern (Zooming, Superresolution)

Das einfachste Verfahren ist sicher die **Pixelreplikation**, also das Verdoppeln der Pixel. Etwas eleganter sind sicher Interpolationsverfahren, das Bild wird dadurch größer, aber die eigentliche Auflösung des Bildes wird sich dadurch nicht ändern. Es kommen keine neuen Informationen hinzu. Gibt es eine Möglichkeit, die Auflösung wirklich zu erhöhen? Das Schlagwort dazu heißt **Superresolution**, siehe z. B. [54].

Zur Vereinfachung der Schreibweise nehmen wir wieder eindimensionale Funktionen und demonstrieren das Grundprinzip. Von einem Bild machen wir mehrere Aufnahmen mit je N Pixeln pro Bild. Aus diesen Bildern der Auflösung N wollen wir ein Bild der Auflösung $2N$ berechnen, welches tatsächlich mehr Information trägt. Eine Grundvoraussetzung ist natürlich, dass die Bilder mit N Pixeln Unterabtastungen sind, ansonsten kann man nicht die Auflösung echt erhöhen. Zur Bandbegrenzung nehmen wir das Modell $A1[X]$. Wir nehmen aus Symmetriegründen an, es seien bezüglich eines idealen Tiefpasses die ersten $M = 2N - 1$ komplexen Fourierkoeffizienten ungleich Null:

$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-(N-1)}^{N-1} \beta_k e^{+2\pi i k \frac{x}{X}}. \quad (5.42)$$

Bezüglich (5.24) ist $K = N - 1$. Nun verschieben wir die Funktion um u und erhalten

$$f(x - u) = \frac{1}{\sqrt{X}} \sum_{k=-(N-1)}^{N-1} \beta_k e^{+2\pi i k \frac{x}{X}} \cdot e^{-2\pi i k \frac{u}{X}}. \quad (5.43)$$

Nun wollen wir beide Funktionen mit N Stützstellen abtasten und versuchen, aus diesem linearen Gleichungssystem die Fourierkoeffizienten β_k auszurechnen. Anschließend brauchen wir nur die Funktion $f(x)$ in (5.42) mit $2N$ Stützstellen äquidistant abtasten und die Superresolution ist gelungen. Wir tasten also jetzt beide Funktionen mit N Stützstellen an den äquidistanten Punkten $x = x_l = l \cdot \frac{X}{N}$ ab:

$$f(x_l) = \frac{1}{\sqrt{X}} \sum_{k=-(N-1)}^{N-1} \beta_k e^{2\pi i k \frac{l}{N}}, \quad l = 0, \dots, N-1. \quad (5.44)$$

Die Verschiebung u drücken wir durch $u = \lambda \frac{X}{N}$ aus und erhalten:

$$f(x_l - u) = \frac{1}{\sqrt{X}} \sum_{k=-(N-1)}^{N-1} \beta_k e^{2\pi i k \frac{l}{N}} \cdot e^{-2\pi i k \frac{\lambda}{N}} = \frac{1}{\sqrt{X}} \sum_{k=-(N-1)}^{N-1} \beta_k e^{2\pi i k \frac{(l-\lambda)}{N}}. \quad (5.45)$$

Das lineare Gleichungssystem (5.44) hat $2N - 1$ Unbekannte und N Gleichungen, ist daher nicht lösbar. Deshalb benötigen wir weitere Gleichungen, nämlich die der verschobenen Funktion (5.45). Mit beiden zusammen haben wir nun $2N - 1$ Unbekannte β_k und $2N$ Gleichungen, so dass formal eine eindeutige Lösung möglich wäre. Die Gleichungen (5.45) dürfen aber nicht linear abhängig von den Gleichungen (5.44) sein. Man sieht leicht, lineare Abhängigkeiten entstehen sofort, wenn die Verschiebung λ ein ganzzahlig Vielfaches der Abtast-Intervallbreite $\frac{X}{N}$ ist. Daher müssen wir fordern: λ darf nicht ganzzahlig sein, d. h. $\lambda \in \mathbb{Z}$. Dies hat sofort praktische Konsequenzen. Die Verschiebung muß ja bekannt sein, d. h. sie muss mit Bild-Registriermethoden bestimmt werden und zwar mit **Subpixelgenauigkeit**. Im Abschn. 4.23 hatten wir aus Verschiebungen der Funktion oder des Spektrums modifizierte DFT angegeben. Wir sehen, die Funktionen $e^{2\pi i k \frac{(l-\lambda)}{N}}$ aus (5.45) sind bis auf den Faktor $\frac{1}{\sqrt{N}}$ genau diese Basisfunktionen einer modifizierten DFT und bilden ein Orthonormalsystem. Genauso sind die Funktionen $e^{2\pi i k \frac{l}{N}}$ aus (5.44) bis auf den Faktor $\frac{1}{\sqrt{N}}$ Basisfunktionen der eigentlichen DFT und bilden auch ein Orthonormalsystem. Während die Basisfunktionen $e^{2\pi i k \frac{l}{N}}$ periodisch mit N sind, sind die Basisfunktionen $e^{2\pi i k \frac{(l-\lambda)}{N}}$ nicht mehr periodisch mit N . Nun könnten wir diesen Abschnitt beenden mit dem Hinweis, das lineare Gleichungssystem muss mit numerischen Methoden gelöst werden und dann sind wir fertig. Wenn wir z. B. Bilder der Dimension (1000×1000) haben, also 1.000.000 Pixel, dann wäre ein lineares Gleichungssystem der Dimension $(2.000.000, 2.000.000)$ zu lösen, was scheinbar unmöglich ist. Folglich untersuchen wir, ob das Gleichungssystem eine spezielle Struktur besitzt und sich etwas leichter lösen lässt. Dazu schreiben wir jetzt das

Gleichungssystem in Matrix-Schreibweise auf. Die k -te Basifunktion $\varphi^k(l) = e^{2\pi i k \frac{l}{N}}$ denken wir uns jetzt als Spaltenvektor φ^k mit N Komponenten, genauso $\varphi^{k,(\lambda)}(l) = e^{2\pi i k \frac{(l-\lambda)}{N}}$. Dann lautet das lineare Gleichungssystem in der Form $\mathbf{Ax} = \mathbf{b}$:

$$\frac{1}{\sqrt{N}} \begin{pmatrix} \varphi^{-(N-1)} & \dots & \varphi^0 & \dots & \varphi^{N-1} \\ \varphi^{-(N-1),(\lambda)} & \dots & \varphi^{0,(\lambda)} & \dots & \varphi^{N-1,(\lambda)} \end{pmatrix} \cdot \begin{pmatrix} \beta_{-(N-1)} \\ \vdots \\ \beta_0 \\ \vdots \\ \beta_{N-1} \end{pmatrix} = \frac{1}{\sqrt{N}} \cdot \sqrt{X} \cdot \begin{pmatrix} f \\ f_\lambda \end{pmatrix} \quad (5.46)$$

mit $f = (f(x_l))$, $f_\lambda = (f(x_l - u))$ als Spaltenvektoren. Wir multiplizieren die linke und rechte Seite mit der Matrix $\bar{\mathbf{A}}^T$ (der Faktor $\frac{1}{\sqrt{N}}$ gehört mit zur Matrix \mathbf{A}) und erhalten die folgenden Gaußschen Normalengleichungen $\bar{\mathbf{A}}^T \mathbf{Ax} = \bar{\mathbf{A}}^T \mathbf{b}$:

$$\left(\begin{array}{ccccccccc} 2 & 0 & \cdots & 0 & 0 & (1 + e^{-2\pi i \lambda}) & 0 & \cdots & 0 \\ 0 & 2 & \cdots & 0 & 0 & 0 & (1 + e^{-2\pi i \lambda}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2 & 0 & 0 & 0 & \cdots & (1 + e^{-2\pi i \lambda}) \\ 0 & 0 & \cdots & 0 & 2 & 0 & 0 & \cdots & 0 \\ (1 + e^{+2\pi i \lambda}) & 0 & \cdots & 0 & 0 & 2 & 0 & \cdots & 0 \\ 0 & (1 + e^{+2\pi i \lambda}) & \cdots & 0 & 0 & 0 & 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (1 + e^{+2\pi i \lambda}) & 0 & 0 & 0 & \cdots & 2 \end{array} \right) \cdot \begin{pmatrix} \beta_{-(N-1)} \\ \vdots \\ \beta_0 \\ \vdots \\ \beta_{N-1} \end{pmatrix} = \sqrt{X} \frac{1}{N} \begin{pmatrix} \sum_{l=0}^{N-1} f(x_l) e^{+2\pi i (N-1) \frac{l}{N}} + \sum_{l=0}^{N-1} f(x_l - u) e^{+2\pi i (N-1) \frac{(l-\lambda)}{N}} \\ \vdots \\ \sum_{l=0}^{N-1} f(x_l) + \sum_{l=0}^{N-1} f(x_l - u) \\ \vdots \\ \sum_{l=0}^{N-1} f(x_l) e^{-2\pi i (N-1) \frac{l}{N}} + \sum_{l=0}^{N-1} f(x_l - u) e^{-2\pi i (N-1) \frac{(l-\lambda)}{N}} \end{pmatrix}. \quad (5.47)$$

Wir sehen, die Unbekannten β_k treten als Summen von Paaren auf. Wir schreiben dies einmal auf. Mit den Abkürzungen

$$g_k = \frac{\sqrt{X}}{N} \left(\sum_{l=0}^{N-1} f(x_l) e^{-2\pi i k \frac{l}{N}} + \sum_{l=0}^{N-1} f(x_l - u) e^{-2\pi i k \frac{(l-\lambda)}{N}} \right), \quad (5.48)$$

$$g_{k+N} = \frac{\sqrt{X}}{N} \left(\sum_{l=0}^{N-1} f(x_l) e^{-2\pi i k \frac{l}{N}} + e^{+2\pi i \lambda} \sum_{l=0}^{N-1} f(x_l - u) e^{-2\pi i k \frac{(l-\lambda)}{N}} \right) \quad (5.49)$$

erhalten wir die Gleichungssysteme für $k = -(N-1), \dots, -1$:

$$\begin{aligned} 2\beta_k + (1 + e^{-2\pi i \lambda})\beta_{k+N} &= g_k \\ (1 + e^{2\pi i \lambda})\beta_k + 2\beta_{k+N} &= g_{k+N}. \end{aligned} \quad (5.50)$$

Diese lösen wir auf:

$$\beta_k = \frac{\frac{1}{2}g_{k+N}(1 + e^{-2\pi i \lambda}) - g_k}{\cos 2\pi \lambda - 1}, \quad \beta_{k+N} = \frac{\frac{1}{2}g_k(1 + e^{+2\pi i \lambda}) - g_{k+N}}{\cos 2\pi \lambda - 1}. \quad (5.51)$$

Eigentlich benötigen wir nur die β_k , da für reelle Funktionen $\beta_{-k} = \overline{\beta_k}$ gelten muss. Die entscheidenden Größen sind also die g_k und g_{k+N} für die Berechnung der β_k , deshalb drücken wir diese jetzt direkt durch α_k der DFT aus:

$$g_k = \sqrt{\frac{X}{N}} \left(\alpha_k(f) + e^{2\pi i k \frac{1}{N}} \cdot \alpha_k(f_\lambda) \right) \quad (5.52)$$

$$g_{k+N} = \sqrt{\frac{X}{N}} \left(\alpha_k(f) + e^{2\pi i \lambda} \cdot e^{2\pi i k \frac{1}{N}} \cdot \alpha_k(f_\lambda) \right). \quad (5.53)$$

Zu beachten ist, dass in obigen Formeln der Index k nur von $-(N-1)$ bis -1 läuft. Die Größe β_0 ist ein Sonderfall und lässt sich trivialerweise aufschreiben:

$$\beta_0 = \frac{\sqrt{X}}{2N} \left(\sum_{l=0}^{N-1} f(x_l) + \sum_{l=0}^{N-1} f(x_l - u) \right) = \frac{1}{2} \sqrt{\frac{X}{N}} (\alpha_0(f) + \alpha_0(f_\lambda)). \quad (5.54)$$

Die β_k für $k = 1, \dots, N-1$ ergeben sich aus $\beta_k = \overline{\beta_{-k}}$. Da wir nun aus diesen diskreten Werten die β_k berechnet haben, kennen wir nun auch die analoge Funktion $f(x)$ entsprechend (5.42). Diese tasten wir nun an $2N$ Stellen $x_l = l \frac{X}{2N}$ ab und erhalten somit ein Bild mit echt höherer Auflösung. In den obigen Formeln haben wir die Kenntnis der Verschiebung λ vorausgesetzt, wobei λ nicht ganzzahlig sein darf. Es muss also eine Verschiebung im Subpixelbereich bestimmt werden. Dies ist eigentlich ein Widerspruch: Wir haben zwei Bilder, die unterabgetastet sind und sollen aus denen eine Subpixelverschiebung bestimmen, die sich aber auf eine höhere Bandbegrenzung bezieht. Es fehlt uns eigentlich die entsprechende Information.

In der Abb. 5.1 ist eine bandbegrenzte Funktion mit 39 komplexen Fourierkoeffizienten zu sehen, d. h. die reelle Bandbegrenzung liegt bei 19. Wir tasten diese Funktion nun mit 20 Stützstellen ab (Unterabtastung), allerdings zweimal, wobei die zweite Abtastung gegenüber der ersten genau um 0,5 Pixel verschoben ist. Allein aus diesen zwei diskreten „Bildern“ ist nun exakt die Subpixel-Translation 0,5 zu ermitteln. In der Abb. 5.2 sind die beiden diskreten „Bilder“ aus Abb. 5.1 nochmals dargestellt, wobei eine lineare Interpolation verwendet wurde. Diese Kurven sollen also durch eine 0,5 Pixel-Translation zur „Deckung“ gebracht werden. Man sieht aus dieser Abbildung, dies scheint unmöglich zu sein. Und dennoch wäre bei diesem extremen Beispiel die Subpixelverschiebung bestimmbar. Im Abschn. 5.1 hatten wir die Bedingung $M > 2K$ angegeben, um die analoge Funktion eindeutig zu rekonstruieren. Dann gab es aber noch eine zweite Ungleichung $K < M \leq 2K$, in diesem Falle sind noch die tiefen Frequenzen rekonstruierbar, und diese enthalten die exakte Subpixelverschiebung. In der Abb. 5.1 ist $M = 20$ und $K = 19$, daher

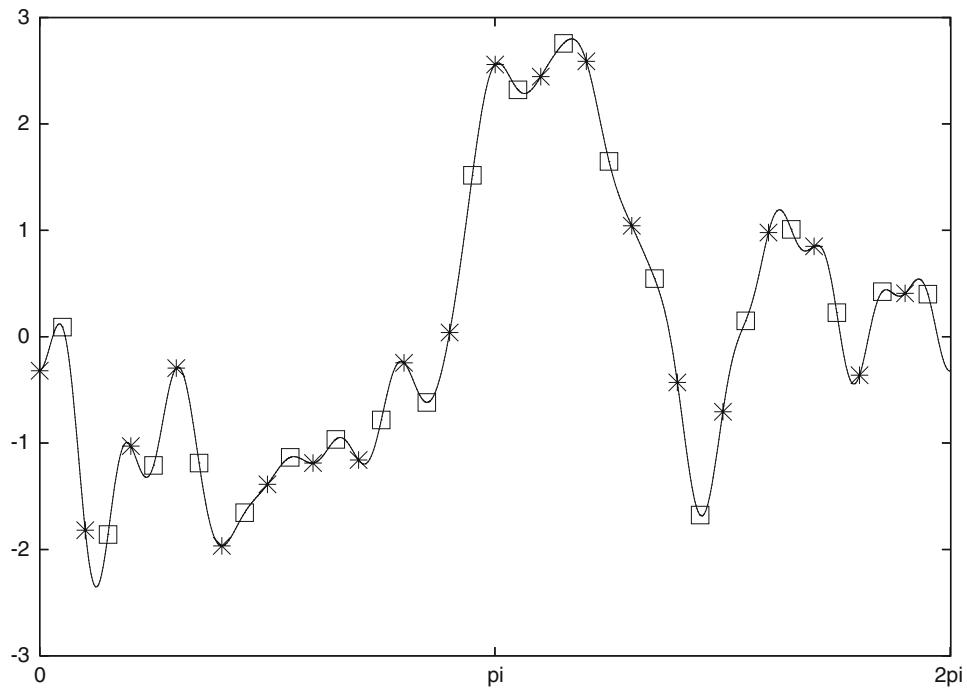


Abb. 5.1 Bandbegrenzte Funktion mit zwei zueinander verschobenen äquidistanten Abtastungen

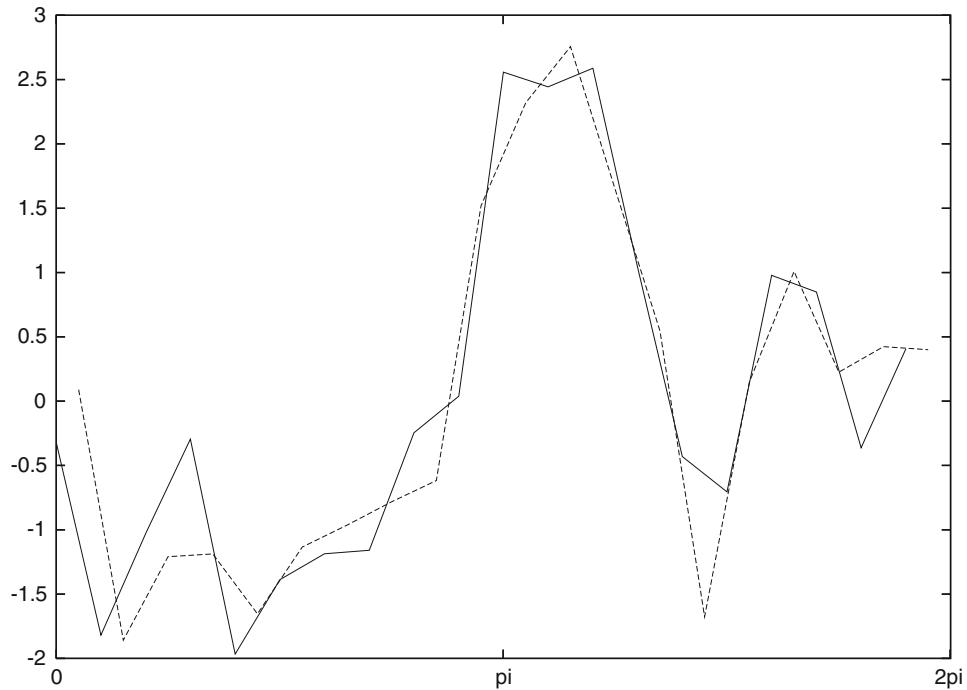


Abb. 5.2 Beide Abtastungen aus Abb. 5.1 wurden linear interpoliert

kann nur noch die 1. Frequenz allein aus den unterabgetasteten Werten berechnet werden. Aus den 1. Frequenzen für beide Bilder kann man nun die Subpixelverschiebung von 0,5 exakt berechnen. Dieses Beispiel ist also so gewählt worden, dass die Verschiebung gerade noch bestimmbar ist, das ist sicher verblüffend. Wählt man weniger als 20 Abtastpunkte, dann kann man tatsächlich keine Subpixel-Registrierung mehr durchführen. Ähnliche Aussagen findet man z. B. in [78]. Auf Grund unserer Annahme, bei N Pixeln liege eine Bandbegrenzung mit $2N - 1$ komplexen Fourierkoeffizienten vor, benötigen wir also mindestens zwei Bilder um die höhere Auflösung auszurechnen. Haben wir mehr als zwei Bilder, z. B. $p > 2$, dann erweitern diese lediglich das lineare Gleichungssystem (5.46), welches im Sinne der kleinsten Quadrate wieder gelöst wird. Es ändert sich im Wesentlichen nichts an der Lösung, diese wird nur numerisch stabiler. Wir wollen diese kurz aufschreiben. Die g_k und g_{k+N} werden zu (formal sei $\lambda_0 = 0, u_0 = 0$):

$$g_k = \frac{\sqrt{N}}{N} \left(\sum_{m=0}^{p-1} \sum_{l=0}^{N-1} f(x_l - u_m) e^{-2\pi i k \frac{(l-\lambda_m)}{N}} \right), \quad (5.55)$$

$$g_{k+N} = \frac{\sqrt{N}}{N} \left(\sum_{m=0}^{p-1} e^{2\pi i \lambda_m} \sum_{l=0}^{N-1} f(x_l - u_m) e^{-2\pi i k \frac{(l-\lambda_m)}{N}} \right). \quad (5.56)$$

Als Lösungen für die β_{k+N} ergibt sich dann:

$$\beta_{k+N} = \frac{\frac{1}{p} \left(\sum_{m=0}^{p-1} e^{+2\pi i \lambda_m} \right) \cdot g_k - g_{k+N}}{\frac{1}{p} \left(p + 2 \sum_{m=1}^{p-1} \cos 2\pi \lambda_m + 2 \sum_{k,l=1, k \neq l}^{p-1} \cos 2\pi (\lambda_k - \lambda_l) \right)}. \quad (5.57)$$

Man könnte allerdings bei p Bildern die Grenzfrequenz erhöhen zu $p \cdot N - 1$ und immer noch eine eindeutige Lösung ausrechnen. Daran sieht man, ohne wirkliches A-priori-Wissen über die Bandbegrenzung kann man so ziemlich alles ausrechnen, man muss nur die geeigneten Annahmen treffen.

5.4.3 Superresolution und Bildrestaurierung

Im vorigen Abschnitt haben wir die „reine“ mathematische *Superresolution* betrachtet. Bei realen Kameras haben wir eigentlich noch zusätzlich die Probleme der Bildrestaurierung zu beachten, also die Faltung und additives Rauschen. Die Bildrestaurierung für sich ist schon ein schwieriges Problem, nun kommt noch zusätzlich die Superresolution hinzu. Daher vernachlässigen wir erst einmal das Rauschen und betrachten nur die Faltung. Dann besteht das mathematische Modell in folgendem:

Das bandbegrenzte, analoge Signal $f(x)$ wird mit einer PSF $h(x)$ gefaltet und anschließend mit N Pixeln abgetastet. Dabei haben wir wieder $p - 1$ Subpixel-Translationen des Signals $f(x)$, die auch gefaltet werden und anschließend je mit N Pixeln abgetastet vorliegen. Die PSF $h(x)$ und damit die Fourierkoeffizienten γ_k der PSF $h(x)$ seien bekannt.



Abb. 5.3 Originalbild mit feinen Strukturen. Das Bild wurde freundlicherweise von Dr. Wolfgang Ortmann zur Verfügung gestellt

Wenn wir wieder das Signal im Intervall $[0, X]$ bandbegrenzt ansetzen:

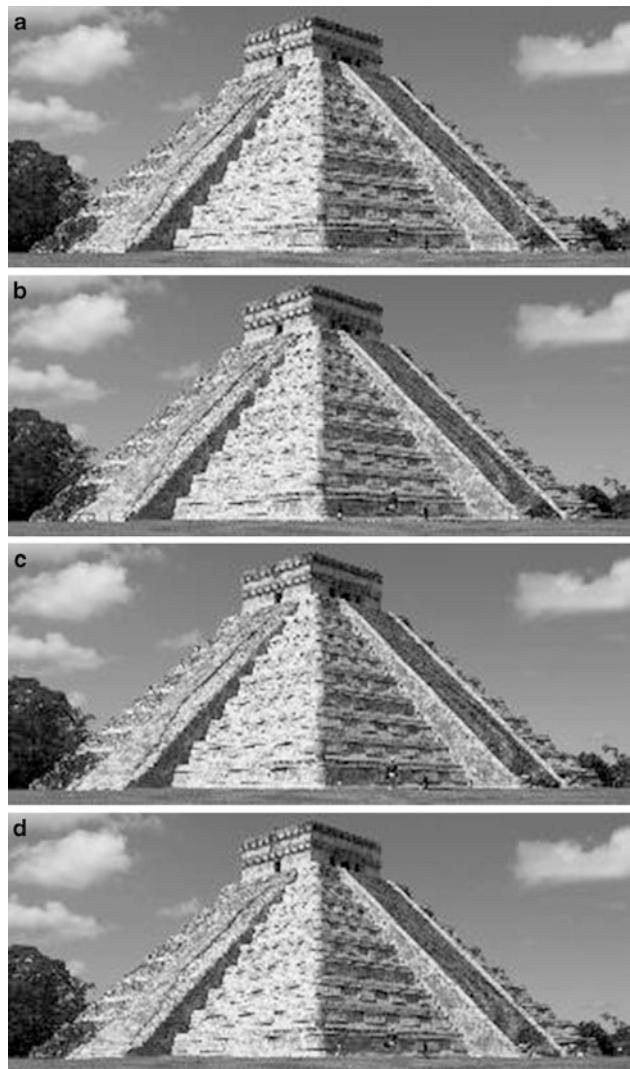
$$f(x) = \frac{1}{\sqrt{X}} \sum_{k=-(N-1)}^{N-1} \beta_k e^{+2\pi i k \frac{x}{X}}, \quad (5.58)$$

dann erhalten wir laut Faltungstheorem für f und die translatierten Funktionen $f(x - u_m)$

$$g(x - u_m) = (h * f)(x - u_m) = \sum_{k=-(N-1)}^{N-1} \gamma_k \cdot \beta_k e^{+ik(x-u_m)}, \quad m = 0, \dots, p-1. \quad (5.59)$$

Wenn wir einmal annehmen, es gelte $\gamma_k \neq 0$, $k = -(N-1), \dots, 0, \dots, (N-1)$, dann haben wir nur eine simple Faltungsinvolution zu beachten, d. h. der Faktor \sqrt{X} ist in g_k und g_{k+N} zu streichen und die Lösungen für β_k bzw. β_{k+N} sind durch γ_k bzw. γ_{k+N} zu teilen. Diese Formel besagt aber auch (was wir aber schon längst wissen), die Faltung ist translationsinvariant, d. h. die gefalteten Funktionen g unterscheiden sich exakt durch die gleichen Verschiebungen wie die originalen Funktionen. Wir nehmen einmal an, die Anzahl N der Pixel sei ungerade, und weiter nehmen wir an, dass $h(x)$ ein Tiefpass derart darstellt, so dass die gefaltete Funktion g eine Grenzfrequenz von $N/2$ (ganzzahlige Division) besitzt. Dann können wir die analoge, gefaltete Funktion g vollständig aus den N Abtastwerten rekonstruieren (trigonometrisches Interpolationspolynom). Aus diesen rekonstruierten Interpolationspolynomen können wir exakt die Subpixelverschiebungen u_m berechnen. Wir könnten folglich unter diesen gemachten Annahmen tatsächlich aus den Bildern mit N Pixeln die exakte Subpixelverschiebung berechnen. Das Problem besteht nun aber darin, dass wir die β_k , für die $\gamma_k = 0$ gilt, nicht mehr rekonstruieren können. Es liegt der Fall der Bildrestaurierung vor, bei dem das Original nicht mehr eindeutig bestimmbar ist.

Abb. 5.4 a Das Bild in Abb. 5.3 wurde unterabgetastet mit dem Faktor $\frac{1}{4}$ in x - und y -Richtung, d.h. es wurde jedes vierte Pixel in x - und y -Richtung abgespeichert. b–d Das Bild in Abb. 5.3 wurde unterabgetastet mit dem Faktor $\frac{1}{4}$, wobei eine Subpixelverschiebung $\frac{1}{4}$ (b), $\frac{2}{4}$ (c) bzw. $\frac{3}{4}$ (d) in x - und y -Richtung gegenüber dem Bild in a vorliegt



In der Abb. 5.3 ist ein Bild einer Pyramide zu sehen. Auf Grund der „feinen Treppenstrukturen“ wird man bei einer Unterabtastung sicher Aliasingeffekte erhalten. Dazu wurde ein Experiment gemacht. Das Bild in Abb. 5.3 wurde in 16 Bildern unterabgetastet, wobei die abgetasteten Bilder tatsächlich eine Subpixelverschiebung um ein Vielfaches von $\frac{1}{4}$ gegeneinander besitzen. In Abb. 5.4 sind 4 der 16 unterabgetasteten Bilder zu sehen. In den unterabgetasteten Bildern kann man deutlich das Aliasing erkennen. Nun wurden mit Registrieralgorithmen die Subpixelverschiebungen der unterabgetasteten Bilder berechnet und durch Superresolution ein höher aufgelöstes Bild berechnet. Dieses Bild stimmt bis auf „numerische Effekte“ mit dem Originalbild 5.3 überein.

5.4.4 Bemerkungen zum Abtasttheorem

Liegt dem Abtasttheorem ein allgemeines Prinzip zugrunde? Natürlich ist dies so und jeder könnte sein eigenes Abtasttheorem formulieren. Wir nehmen einmal an, eine gegebene Funktion f über dem Intervall $[0, X]$ sei in eine gewöhnliche Potenzreihe entwickelbar:

$$f(x) = \sum_{k=0}^{\infty} a_k x^k, \quad (5.60)$$

z. B. die Taylorreihe im Entwicklungspunkt $x = 0$. Dann nennen wir die Funktion **polynombegrenzt** (anstatt bandbegrenzt), wenn die Reihe zur endlichen Summe entartet:

$$f(x) = \sum_{k=0}^{N-1} a_k x^k, \quad (5.61)$$

d. h. die Funktion ist selbst ein Polynom $N - 1$ -ten Grades. Wenn wir nun die Funktion an N Stellen abtasten $x_l, l = 0, \dots, N - 1$, dann können wir allein aus der Kenntnis der Abtastwerte die Funktion wieder eindeutig rekonstruieren, wir brauchen wieder nur das entsprechende lineare Gleichungssystem aufzuschreiben:

$$\begin{aligned} a_0 + a_1 x_0 &+ a_2 x_0^2 + \cdots + a_{N-1} x_0^{N-1} = f(x_0) \\ a_0 + a_1 x_1 &+ a_2 x_1^2 + \cdots + a_{N-1} x_1^{N-1} = f(x_1) \\ &\vdots \\ a_0 + a_1 x_{N-1} &+ a_2 x_{N-1}^2 + \cdots + a_{N-1} x_{N-1}^{N-1} = f(x_{N-1}). \end{aligned} \quad (5.62)$$

Die Koeffizientenmatrix \mathbf{A} des linearen Gleichungssystems (5.62):

$$\mathbf{A} = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{N-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{N-1} \\ & & & \vdots & \\ 1 & x_{N-1} & x_{N-1}^2 & \cdots & x_{N-1}^{N-1} \end{pmatrix} \quad (5.63)$$

heißt *Vandermonde-Matrix*, deren Determinante ist gleich:

$$\det(\mathbf{A}) = \prod_{N-1 \geq k > j \geq 0} (x_k - x_j). \quad (5.64)$$

Damit ist das Gleichungssystem (5.62) genau dann eindeutig lösbar, wenn die x_j paarweise verschieden sind. Die x_j müssen nicht äquidistant gewählt werden, Hauptsache sie sind paarweise verschieden. Es fällt sofort auf und das kann kein Zufall sein: die Vandermonde-Matrix hatten wir schon beim Beweis des Abtasttheorems im Intervall kennengelernt, siehe (5.13) und (5.14). Das Gleichungssystem (5.62) können wir nun nur numerisch nach den

Koeffizienten a_k auflösen. Die Lösungen a_k hängen von den Funktioneswerten $f(x_k)$ und den Stützstellen x_k ab. Diese Abhängigkeit bzw. Rechenvorschrift nennen wir einfach DPT (Diskrete Polynom Transformation). Da wir diese DPT nicht explizit aufschreiben können, sondern nur numerisch realisieren können, versucht man oft die Basispolynome so zu modifizieren, dass das Gleichungssystem explizit auflösbar wird, z. B. die *Lagrange Interpolation*. Mit der Lagrange-Interpolation macht man einen Darstellungstrick: man verlagert Wissen in die Basisfunktionen, nämlich die Stützstellen x_i selbst. Die i -te Basisfunktion $l_i(x)$ wird zu:

$$l_i(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_{N-1})}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_{N-1})}. \quad (5.65)$$

Damit wird das (bandbegrenzte) Polynom (5.61) zu:

$$f(x) = \sum_{k=0}^{N-1} b_k l_k(x). \quad (5.66)$$

Da für die Basisfunktionen die Beziehung $l_i(x_j) = \delta_{i-j}$ gilt, ist die Koeffizientenmatrix \mathbf{A} in (5.62) nicht nur eine Orthogonalmatrix, sie ist sogar die Einheitsmatrix. Daher können wir sofort die Lösung $b_i = f(x_i)$ aufschreiben. Diese Lösung würden wir jetzt DLT (Diskrete Lagrange Transformation) nennen, allerdings ist diese trivial, daher unbrauchbar. Das liegt daran, dass wir die Basisfunktionen nicht allgemein genug gewählt haben, sie hängen direkt von den Stützstellen ab. Allerdings könnten wir jetzt mit (5.66) die DPT berechnen, wir brauchen nur die $l_i(x)$ auszumultiplizieren, alle Koeffizienten vor einer Potenz von x zusammenfassen und hätten die Darstellung für die a_i gefunden und somit die DPT. Diese Ausdrücke werden aber recht unübersichtlich sein.

Sinnvoller scheint es zu sein, solche Basissysteme zu wählen, bei denen dann die Matrix \mathbf{A} eine Orthonormalmatrix ist, d. h. die diskretisierten Basisfunktionen sind orthogonale Vektoren (auf die Länge Eins kann anschließend noch normiert werden). Die Fouriertransformation im endlichen Intervall besitzt nun eine sehr schöne Eigenschaft: Die analogen Basisfunktionen bilden ein Orthonormalsystem. Wenn wir nun die Stützstellen äquidistant wählen, dann sind (überraschenderweise) die diskretisierten Basisfunktionen, die jetzt Vektoren darstellen, auch orthogonal. Die Spaltenvektoren der Matrix \mathbf{A} sind jetzt orthogonal (siehe (5.17)). Wir können das Gleichungssystem ganz leicht lösen, die Lösung nennen wir DFT. Da wir die DFT im Komplexen formulieren, ist die Matrix \mathbf{A} auch komplex. Daher formulieren wir einmal die Bandbegrenzung im Reellen und im Intervall $[0, 2\pi]$ (d. h. $X = 2\pi$) mit der Grenzfrequenz N :

$$f(x) = \frac{1}{\sqrt{2\pi}} \left(\frac{a_0}{2} + \sum_{k=1}^N a_k \cos kx + \sum_{k=1}^N b_k \sin kx \right). \quad (5.67)$$

Laut Abtasttheorem benötigen wir nun im Intervall $[0, 2\pi]$ mindestens $M = 2N + 1$ Abtastpunkte, um das trigonometrische Interpolationspolynom berechnen zu können. Bezeichnen wir die Abtastpunkte mit x_0, x_1, \dots, x_{M-1} und setzen $a_0 = 0$, so erhalten wir aus (5.67)

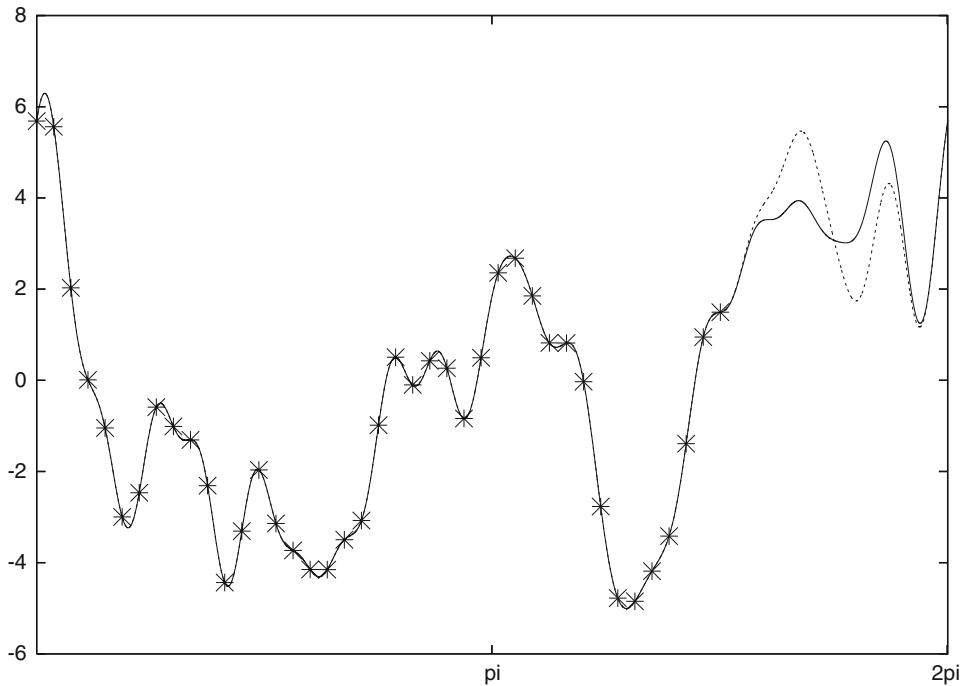


Abb. 5.5 $M \geq 2N + 1$ äquidistante Stützstellen füllen nicht das ganze Intervall aus

das folgende lineare Gleichungssystem:

$$\begin{pmatrix} \cos x_0 & \dots & \cos Nx_0 & \sin x_0 & \dots & \sin Nx_0 \\ \cos x_1 & \dots & \cos Nx_1 & \sin x_1 & \dots & \sin Nx_1 \\ \vdots & & & & \vdots & \\ \cos x_{M-1} & \dots & \cos Nx_{M-1} & \sin x_{M-1} & \dots & \sin Nx_{M-1} \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ \vdots \\ a_N \\ b_1 \\ \vdots \\ b_N \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{M-1}) \end{pmatrix} \quad (5.68)$$

Wenn wir nun das Gleichungssystem (5.68) lösen, dann erhalten wir die allgemeinste Form der DFT einer reellen Funktion $f(x)$. Insbesondere brauchen die Abtastpunkte nicht äquidistant sein, sie können beliebig gewählt werden, es muss nur das Gleichungssystem lösbar sein. Wählt man z. B. die Abtastpunkte so, dass größere Abtastlücken entstehen (siehe Abb. 5.5, im rechten Teil des Intervales liegen keine Abtastpunkte), dann kann das Gleichungssystem schlecht konditioniert werden und damit wird die Lösung instabil. Dies drückt sich darin aus, dass dann das Interpolationspolynom in der Abtastlücke von der Funktion $f(x)$ abweichen kann. In der Abb. 5.6 sind die Instabilitäten bei nichtäquidistanter Abtastung und größeren Lücken zu sehen. Stabil wird die Lösung, wenn die Abtastpunkte „gleichmäßig“ über das Intervall $[0, X]$ verteilt werden und keine größere Ab-

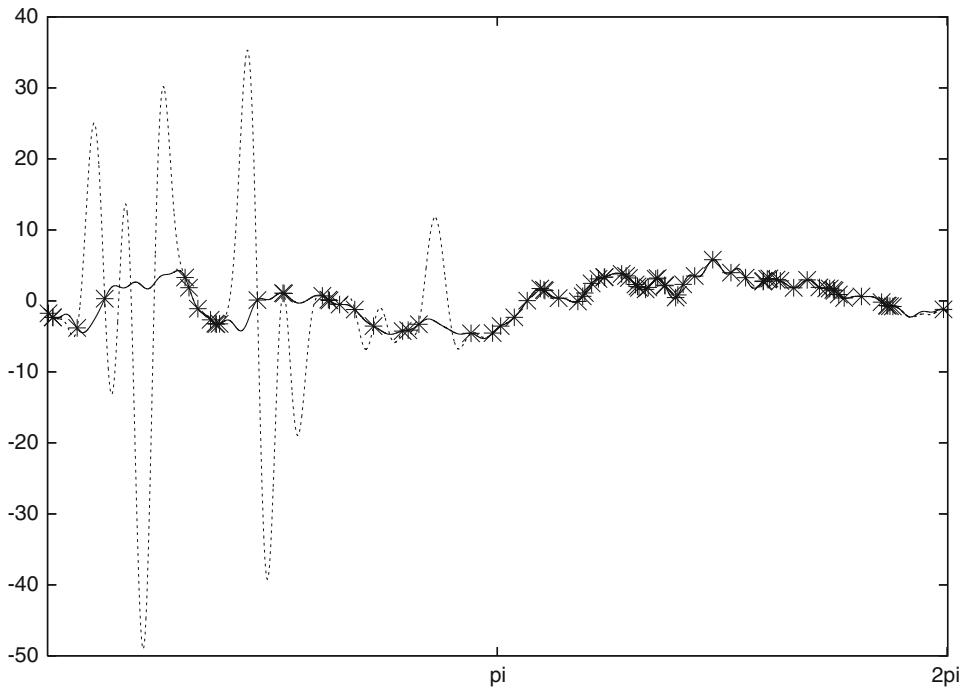


Abb. 5.6 $M \geq 2N + 1$ nichtäquidistante Stützstellen mit größeren Lücken

tastlücke entsteht. Der stabilste Fall ist der, wenn man äquidistante Abtastpunkte über das ganze Intervall wählt. Die Lösung des linearen Gleichungssystems entspricht dann der klassischen DFT.

5.4.5 Pyramiden

Burt und Adelson führten bereits 1981–1983 das Konzept der Gauß- und Laplacepyramide ein. Rosenfeld begründete 1984 das Gebiet der *Multiresolutional Image Processing*. Das Grundprinzip der Gaußpyramide ist sehr simpel. Jede Schicht der Gaußpyramide entsteht durch Gaußfilterung der darunterliegenden Schicht und „Weglassen“ jedes zweiten Pixels. Die Laplacepyramide entsteht einfach durch Differenzbildung zweier Schichten der Gaußpyramide. Man muss aber darauf achten, dass beide Schichten der Gaußpyramide die gleiche Anzahl von Pixeln besitzen. Dies kann z. B. durch Interpolation der oberen Schicht erfolgen. Damit sind mit der Gauß- und der Laplacepyramide alle Schichten wieder vollständig rekonstruierbar. Was haben nun diese Pyramiden mit dem Abtasttheorem zu tun? Das Gaußfilter ist ein LSI-Filter, und zwar ein Tiefpassfilter in Abhängigkeit vom gewählten σ . Wir nehmen einmal an, man könnte σ so wählen, dass ein idealer Tiefpass derart



Abb. 5.7 Lenna. **a** Gaußpyramide, **b** Laplacepyramide

entsteht, dass genau immer die „untere“ Hälfte aller Frequenzen passieren dürfen. Die erste Schicht der Gaußpyramide enthält dann die ersten $n/2$ Frequenzen. Nach dem Abtasttheorem ist dann die erste Schicht bandbegrenzt mit der Grenzfrequenz $n/2$, folglich brauchen wir nur die Hälfte aller Abtastwerte und können die Zwischenwerte durch trigonometrische Interpolation eindeutig rekonstruieren. Die Laplacepyramide speichert die Differenz zwischen vollständig interpolierter erster Schicht und der untersten Schicht, sie würde damit genau das obere Frequenzband repräsentieren. Natürlich kann man kein σ wählen, so dass dieser ideale Tiefpass entsteht, aber näherungsweise kann man dies realisieren. Das „Geniale“ an diesen Pyramiden ist aber, dass wir einen Gaußfilter beschränkter Ortsausdehnung nutzen und beliebig „schmale“ Tiefpässe erzeugen können, allerdings erst in den höheren Schichten der Pyramiden. Für die Gaußfilter werden oft Binomialfilter implementiert. Ein Nachteil gegenüber den Wavelets ist z. B., dass wir pro Schicht 50% aller Pixel zusätzlich abspeichern müssen. Die Gauß- und Laplacepyramiden werden oft in der Datenkompression und dem sogenannten *Mosaicing* eingesetzt, d. h. man setzt viele kleine Bilder zu einem großen Bild zusammen, wobei die „Nahtstellen“ besondere Probleme bereiten (siehe auch [51]).

6.1 Der Leck-Effekt (Leakage effect)

In der Signalanalyse und Nachrichtentechnik wird theoretisch immer eine Funktion $f(t)$ bezüglich des Modells A1 $[-\infty, +\infty]$ betrachtet. Dies entspricht aber nie der Realität. In der Realität haben wir nur ein endliches Intervall zur Verfügung, außerhalb dieses Intervales setzen wir die Funktion zu Null und erhalten damit eine Funktion $f'(t)$. Es soll aber vom Spektrum der Funktion $f'(t)$ auf das Spektrum von $f(t)$ geschlossen werden. Ist dies überhaupt möglich? Die Funktion $f'(t)$ ergibt sich als Multiplikation von $f(t)$ mit der Rechteckfunktion $\text{rect}(a \cdot t)$. Nach dem Faltungstheorem gilt dann:

$$\alpha_{f'(t)}(v) = \alpha_{f(t) \cdot \text{rect}(a \cdot t)}(v) = C \cdot \alpha_{f(t)}(v) * \alpha_{\text{rect}(a \cdot t)}(v). \quad (6.1)$$

Das Spektrum von $f(t)$ wird also durch eine Faltung mit dem Spektrum der **Fensterfunktion** „verschmiert“. Nun kann man statt der Rechteckfunktion eine andere Fensterfunktion $g(t)$ wählen, so dass diese „Verschmierung“ irgendeinem Zielergebnis unterworfen wird. Dazu betrachtet man das Leistungsspektrum $|\alpha_g(t)|^2$ der Fensterfunktion $g(t)$. Diese Leistungsspektren besitzen einen zentralen Peak und viele Nebenmaxima, die man als „Seitenkeulen“ (*side lobes*) bezeichnet. Die „Verschmierung“ ist sicher minimal, wenn man diese Seitenkeulen möglichst klein hält und gleichzeitig der zentrale Peak möglichst schnell abfällt. Dazu wird oft die Maßeinheit Dezibel benutzt:

$$\text{dB} = 10 \cdot \log_{10} x. \quad (6.2)$$

Es bedarf in der konkreten Anwendung immer einer Erklärung, was x für ein Verhältnis darstellt. Nun charakterisiert man z. B. die Fensterfunktionen durch das Verhältnis der Höhe des ersten (größten) Nebenmaximums zur Höhe des zentralen Peaks und gibt diese Größe für die meisten Fensterfunktionen in (dB) an. Diese Größe sollte also möglichst groß sein. So ist sie z. B. für das Rechteckfenster gleich $-13,2$ dB. Das Minus bedeutet nur, dass man

eine Dämpfung meint und keine Verstärkung. Als eine weitere Größe dient die Bandbreite. Man definiert aber in diesem Zusammenhang eine spezielle Bandbreite, z. B. die berühmte 3 dB Bandbreite $-3 \approx 10 \cdot \log_{10}(1/2)$, d. h. wann der zentrale Peak im Leistungsspektrum auf die Hälfte abgefallen ist. Diese Größe sollte möglichst klein sein. Auf diese Größen hin sind viele Fensterfunktionen untersucht worden, siehe z. B. [12]. Die bekanntesten Fensterfunktionen $g(t)$ sind :

- Rechteck-Fenster
- Dreiecks-Fenster (Fejer-Fenster)
- Kosinus-Fenster
- Hanning-Fenster
- Hamming-Fenster
- Triplett-Fenster
- Gauß-Fenster
- Kaiser-Bessel-Fenster
- Blackman-Harris-Fenster.

Bezüglich der beiden oben aufgeführten charakteristischen Größen des Leistungsspektrums der Fenster-Funktionen ist das Rechteck-Fenster das „schlechteste“ Fenster, und das Kaiser-Bessel-Fenster ist das „beste“ Fenster.

Alles bisher ist aber nur eine theoretische Erklärung des Leck-Effektes für das Modell A1 $[-\infty, +\infty]$. In einem endlichen Intervall kann man praktisch nur das Modell A1 $[X]$ verwenden, „ganz praktisch“ sogar nur die DFT mit dem Modell D1 $[N]$ mit einer periodischen Wiederholung der Funktion. Mit der DFT kann man sogar in Spezialfällen das Spektrum von $f(t)$ richtig wiedergeben. Ist $f(t)$ selbst periodisch und man wählt als Intervall $[0, X]$ genau die Periodenlänge bzw. ein ganzzahlig Vielfaches davon und ist weiterhin $f(t)$ bandbegrenzt, dann gibt die DFT (falls man das Abtasttheorem beachtet) das Spektrum sogar richtig wieder, ansonsten wird es durch den Leck-Effekt verfälscht.

6.2 Das komplexe Spektrogramm

Wie durch den Leck-Effekt schon angedeutet, möchte man Frequenzen einem bestimmten Ort zuordnen, dies geschieht in der klassischen Signaltheorie mit der *Short Time Fourier Transform* oder *Windowed Fourier Transform*. In der Bildverarbeitung wird dies komplexes Spektrogramm (*complex spectrogram*) genannt:

$$\alpha_{f \cdot S_u g}(u, \omega) = \int_B f(t)g(t-u)e^{-i\omega t} dt. \quad (6.3)$$

Die Funktion f wird folglich mit einer verschobenen Fensterfunktion g multipliziert und davon wird das Spektrum berechnet. Nehmen wir einmal an, die Fensterfunktion g sei eine

gerade Funktion, dann erhalten wir die Faltung:

$$\alpha_{f \cdot S_u g}(u, \omega) = \int_B f(t) e^{-i\omega t} g(u - t) dt = f(u) e^{-i\omega u} * g(u). \quad (6.4)$$

Für das folgende normieren wir die Fensterfunktion zu:

$$\|g\|^2 = \int_B g(t) \overline{g(t)} dt = \int_B |g(t)|^2 dt = 1. \quad (6.5)$$

Für das Spektrogramm erhalten wir nun die Parsevalsche Gleichung zu:

$$\int_B |f(t)|^2 dt = \frac{1}{2\pi} \int_B \int_B |\alpha_{f \cdot S_u g}(u, \omega)|^2 du d\omega. \quad (6.6)$$

Denn mit

$$\int_B |f(t)|^2 |g(t-u)|^2 dt = \frac{1}{2\pi} \int_B |\alpha_{f \cdot S_u g}(u, \omega)|^2 d\omega, \quad (6.7)$$

und der Normierung der Fensterfunktion gilt:

$$\int_B \int_B |f(t)|^2 |g(t-u)|^2 dt du = \int_B |f(t)|^2 \int_B |g(t-u)|^2 dt du = \int_B |f(t)|^2 dt. \quad (6.8)$$

Das inverse Spektrogramm wird zu:

$$f(t) = \frac{1}{2\pi} \int_B \int_B \alpha_{f \cdot S_u g}(u, \omega) \overline{g(t-u)} e^{i\omega t} d\omega du, \quad (6.9)$$

denn es ist:

$$f(t)g(t-u) = \frac{1}{2\pi} \int_B \alpha_{f \cdot S_u g}(u, \omega) e^{i\omega t} d\omega, \quad (6.10)$$

somit also:

$$f(t)|g(t-u)|^2 = \frac{1}{2\pi} \int_B \alpha_{f \cdot S_u g}(u, \omega) \overline{g(t-u)} e^{i\omega t} d\omega, \quad (6.11)$$

und daher:

$$\int_B f(t) |g(t-u)|^2 du = f(t) = \frac{1}{2\pi} \int_B \int_B \alpha_{f \cdot S_u g}(u, \omega) \overline{g(t-u)} e^{i\omega t} d\omega du. \quad (6.12)$$

Formal mathematisch haben wir folglich eine lokale Fouriertransformation eingeführt mit Hin- und Rücktransformation. Der entscheidende Nachteil ist aber, dass sich die Anzahl der Variablen erhöht. Bei zweidimensionalen Bildern ist dies noch extremer: das komplexe Spektrogramm hängt dann statt von zwei sogar von vier Variablen ab.

Die lokale Fouriertransformation hatten wir bisher so formuliert: man verschiebe die Fensterfunktion g um u Einheiten nach „rechts“, multipliziere sie dann mit der Bildfunktion f und anschließend wird dieses Produkt Fourier-transformiert. Was ist wenn wir umgedreht vorgehen: Man verschiebe die Bildfunktion f um $-u$ Einheiten nach „links“, multipliziere sie dann mit der Fensterfunktion und dieses Produkt wird Fourier-transformiert. Inhaltlich ändert sich nicht viel, die Funktionen im Ortsraum sind nur gegeneinander verschoben, das Spektrum ändert sich nur bezüglich des Verschiebungstheorems. Der Vorteil ist eine normalisierte Lage der Fensterfunktion:

$$\alpha_{g \cdot S_{-u} f}(u, \omega) = \int_B g(t)f(t+u)e^{-i\omega t} dt = \int_B g(-\tau)f(u-\tau)e^{+i\omega\tau} d\tau. \quad (6.13)$$

Wenn wir annehmen, dass die Fensterfunktion eine gerade Funktion ist (z. B. Gaußfunktion, Rechteckfunktion), dann gilt

$$\begin{aligned} \alpha_{g \cdot S_{-u} f}(u, \omega) &= \int_B g(-\tau)f(u-\tau)e^{+i\omega\tau} d\tau = \int_B g(\tau)e^{+i\omega\tau} f(u-\tau)d\tau \\ &= (g(u) \cdot e^{+i\omega u}) * f(u). \end{aligned} \quad (6.14)$$

Damit lässt sich jeder Fourierkoeffizient als Faltung einer modulierten Fensterfunktion mit der Bildfunktion darstellen. Wenn g die Gaußfunktion darstellt, dann liegt für eine Frequenz ω_0 die Faltung von f mit dem Gaborfilter *Gabor* (siehe Abschn. 6.4) vor. Das Spektrum von G ist dann wiederum:

$$\alpha_G(\omega) = e^{-\frac{(\omega-\omega_0)^2 a^2}{2}}, \quad (6.15)$$

d. h. ein typischer Bandpassfilter mit dem Zentrum ω_0 . Wir wollen nun einmal die lokale Fouriertransformation für den diskreten Fall aufschreiben. Die diskrete Fensterfunktion g sei gerade, wir verschieben f um $-n$ Positionen nach „links“, multiplizieren mit der Fensterfunktion und berechnen den k -ten diskreten Fourierkoeffizienten:

$$\begin{aligned} \alpha_k(g \cdot S_{-n} f) &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} g_j f_{j+n} e^{-2\pi i k \frac{j}{N}} = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} g_l f_{n-l} e^{+2\pi i k \frac{l}{N}} \\ &= ((g \cdot \varphi^{(k)}) * f)_n. \end{aligned} \quad (6.16)$$

Dabei ist $\varphi^{(k)}$ die k -te diskrete Basisfunktion.

6.3 Wigner-Ville-Orts-Frequenz-Darstellung

Wigner (1932) und später Ville (1948) haben diese Transformation im Zusammenhang mit der Quantenmechanik eingeführt. Für die Bildverarbeitung ist sie eine typische Orts-Frequenz-Darstellung, wobei die Funktion selbst als ihr eigenes Fenster fungiert. Sie ist definiert zu:

$$W_f(u, \omega) = \int_{-\infty}^{\infty} f(u + \frac{\tau}{2}) \overline{f\left(u - \frac{\tau}{2}\right)} e^{-i\omega\tau} d\tau. \quad (6.17)$$

Sie ist damit die Fouriertransformation der Funktion $f_2(u, x) = f(u + \frac{x}{2}) \overline{f(u - \frac{x}{2})}$. Da $f_2(x)$ hermitesch ist, ist $W_f(u, \omega)$ rein reell. Was bedeutet diese Funktion $f_2(u, x)$? Dazu betrachten wir die Autokorrelation:

$$\begin{aligned} (f \circ f)(x) &= \int_{-\infty}^{\infty} f(u) \overline{f(u-x)} du = \int_{-\infty}^{\infty} f\left(u + \frac{x}{2}\right) \overline{f\left(u - \frac{x}{2}\right)} du \\ &= \int_{-\infty}^{\infty} f_2(u, x) du. \end{aligned} \quad (6.18)$$

Die Autokorrelation kann somit als örtliche Mittelung dieser Kovarianzfunktion $f_2(u, x)$ aufgefasst werden. Die Wigner-Ville Transformation beschreibt damit das Spektrum dieser noch ortsabhängigen Kovarianzfunktion $f_2(u, x)$. Die inverse Transformation ist:

$$f(x) = \frac{C}{2\pi} \int_{-\infty}^{\infty} W_f\left(\frac{x}{2}, \omega\right) e^{i\omega x} d\omega, \quad C = \frac{|f(0)|}{\overline{f(0)}}. \quad (6.19)$$

Die inverse Transformation ist folglich nur bis auf einen Faktor bestimmt. Die Transformation ergibt sich auch aus der Symmetrierelation:

$$W_f(u, \omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \alpha\left(\omega + \frac{\gamma}{2}\right) \overline{\alpha\left(\omega - \frac{\gamma}{2}\right)} e^{-iyu} dy. \quad (6.20)$$

Da das Spektrum der Autokorrelation das Leistungsspektrum ist, ergeben sich zwangsläufig die Beziehungen:

$$\int_{-\infty}^{\infty} W_f(u, \omega) du = |\alpha(\omega)|^2, \quad \int_{-\infty}^{\infty} W_f(u, \omega) d\omega = 2\pi |f(u)|^2. \quad (6.21)$$

Die Transformation hat auch ungünstige Eigenschaften, z. B. ist die Transformation einer Summe von Funktionen nicht die Summe der Transformationen, da gemischte Terme infolge der Nichtlinearität von $f(t)$ auftreten.

6.4 Gabor-Filter

Gaborfunktionen hatten wir schon im Abschn. 4.25 im Zusammenhang mit der Unschärferelation kennengelernt. Gaborfunktionen sind Gaborfilter, die eine spezielle Orts-Frequenz-Darstellung realisieren. Die Fensterfunktion Gabor(x, y) ist eine modulierte Gauß-Funktion, siehe Abschn. 6.2. Dies wollen wir nun für 2D-Bilder aufschreiben. Die allgemeinste Form einer 2D Gauß-Funktion erzeugt Isolinien, die Ellipsen darstellen. Folglich stellen wir uns einen Kreis vor, dessen Mittelpunkt irgendwo liegt. Wir legen den Koordinatenursprung in den Kreismittelpunkt, führen eine anisotrope Skalierung durch (dadurch wird der Kreis zur Ellipse in achsenparalleler Lage) und zum Schluß rotieren wir noch diese Ellipse. Wir benötigen also fünf Parameter, einmal den Kreismittelpunkt (x_c, y_c), die anisotrope Skalierung mit a, b und die Rotation mit φ für die Ellipse und schließlich noch einen Faktor K für die Amplitudenskalierung:

$$g(x, y) = K \cdot e^{(-\pi(a^2[(x-x_c)\cos\varphi+(y-y_c)\sin\varphi]^2+b^2[-(x-x_c)\sin\varphi+(y-y_c)\cos\varphi]^2))} \quad (6.22)$$

Der komplexe Modulationsterm (siehe auch Abschn. 4.25) ist (wir benutzen jetzt nicht die Kreisfrequenz):

$$\begin{aligned} h(x, y)_c &= e^{+2\pi i(u_0(x-x_0)+v_0(y-y_0))} = e^{+2\pi i(u_0x+v_0y-u_0x_0-v_0y_0)} = e^{i(2\pi(u_0x+v_0y)+P)} \\ &= \cos[2\pi(u_0x+v_0y)+P] + i \sin[2\pi(u_0x+v_0y)+P]. \end{aligned} \quad (6.23)$$

Wenn wir von reellen Funktionen ausgehen, dann treten die komplexen Fourierkoeffizienten konjugiert komplex auf, dann müssten wir die Modulation reell

$$\begin{aligned} h(x, y)_r &= \frac{1}{2} (e^{+i(2\pi(u_0x+v_0y)+P)} + e^{-i(2\pi(u_0x+v_0y)+P)}) \\ &= \cos[2\pi(u_0x+v_0y)+P] \end{aligned} \quad (6.24)$$

ansetzen. Damit hat ein Gaborfilter neun freie Parameter, allerdings sind nur fünf bedeutsam. Nun wollen wir die Fouriertransformierte der vollständigen Gaborfunktion (Gaborfilter) aufschreiben und benutzen (3.59):

$$Gabor(x, y) = g(x, y) \cdot h(x, y)_c. \quad (6.25)$$

Zur Fouriertransformation der Gauß-Funktion benutzen wir das Transformationspaar (4.97). Wir brauchen nun die Transformierte nicht direkt ausrechnen, wir nutzen einige Rechenregeln. Wenn wir rotieren, dann rotiert das Spektrum in gleicher Weise, wir notieren folglich die Analogie:

$$\begin{aligned} x' &= (x - x_c) \cos \varphi + (y - y_c) \sin \varphi & u' &= (u - u_0) \cos \varphi + (v - v_0) \sin \varphi \\ y' &= -(x - x_c) \sin \varphi + (y - y_c) \cos \varphi & v' &= -(u - u_0) \sin \varphi + (v - v_0) \cos \varphi. \end{aligned} \quad (6.26)$$

Eine Modulation im Ortsraum bedeutet eine Verschiebung im Frequenzraum und umgekehrt, daher ist die Transformierte von (6.25):

$$Gabor(x, y) \leftrightarrow \alpha_{\text{Gabor}}(u, v) = \frac{K}{ab} e^{-\pi\left(\frac{(u-u_0)^2}{a^2} + \frac{(v-v_0)^2}{b^2}\right)} \cdot e^{+i(-2\pi(x_c(u-u_0)+y_c(v-v_0))+P)}, \quad (6.27)$$

welche wiederum einen typischen Bandpassfilter darstellt. Gaborfilter werden in vielen Anwendungsgebieten eingesetzt, z. B. bei biometrischen Anwendungen (*fingerprint*), bei Texturen, Defektstellen-Detektion in Mustern und vieles mehr. Die Einstellung der wesentlichen Parameter hängt von der konkreten Aufgabe ab, oft wird eine Filterbank von Gaborfiltern benutzt, d. h. eine Menge von Gaborfiltern mit verschiedenen Parametern.

6.5 Quadratur-Filter (Energie-Filter)

Energiefilter wurden im Zusammenhang mit der Hilbert-Transformation und der Phasenkongruenz besprochen. Oft möchte man diese aber nicht global auf das ganze Bild anwenden, sondern auch frequenz- oder ortsabhängig benutzen. Als Alternative zum komplexen Spektrogramm bietet sich eine Bandpassfilterung an. Dazu stellen wir Forderungen an die Filterfunktion $f_e(x)$:

- Die Energiefilterung sollte invariant gegenüber Grauwertverschiebungen sein (Grauwertadditionen, Helligkeitsveränderungen). Daher muss die DC-Komponente des Bandpasses gleich Null sein.
- Die Phase der Funktion sollte durch den Bandpass nicht verändert werden, daher muss das Spektrum rein reell sein oder die Funktion $f_e(x)$ muss eine gerade Funktion sein.
- Die Hilberttransformierte einer geraden Funktion ist trivialerweise eine ungerade Funktion $f_o(x) = HIT(f_e(x))$.
- Zur Quadratur benötigen wir eigentlich die Hilberttransformierte $HIT(f_e * f)$. Da aber nach (3.83) $HIT(f_e * f) = HIT(f_e) * f = f_o * f$ gilt, können wir die Energie nach:

$$E^2(f_e * f) = (f_e * f)^2 + (f_o * f)^2 \quad (6.28)$$

berechnen.

Gabor Filter Oft werden Gabor-Filter als Quadraturfilter angegeben. Dazu wollen wir eine Gaborfunktion in der Form:

$$G_{\omega_0,0}(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} e^{i\omega_0 t} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \cos(\omega_0 t) + i \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \sin(\omega_0 t) \quad (6.29)$$

annehmen. Der Realteil $G_+(t)$ ist tatsächlich eine gerade Funktion und der Imaginärteil $G_-(t)$ eine ungerade Funktion. Nun müssen wir noch untersuchen, ob G selbst analytisch ist und die DC-Komponente des Spektrums Null ist. Wie wir schon wissen, ist das Spek-

trum dieser komplexen Gaborfunktion $\alpha_G(\omega) = e^{-\frac{(\omega-\omega_0)^2 \sigma^2}{2}}$. Mit dieser Beziehung und der Eulerschen Formel sind dann die Spektren von Realteil und Imaginärteil gleich:

$$\alpha_{G_+} = \frac{1}{2} \left(e^{-\frac{(\omega-\omega_0)^2 \sigma^2}{2}} + e^{-\frac{(\omega+\omega_0)^2 \sigma^2}{2}} \right), \quad (6.30)$$

$$\alpha_{G_-} = \frac{1}{2i} \left(e^{-\frac{(\omega-\omega_0)^2 \sigma^2}{2}} - e^{-\frac{(\omega+\omega_0)^2 \sigma^2}{2}} \right). \quad (6.31)$$

Gilt nun die Beziehung $\alpha_{G_-} = (-i) \cdot \text{sign}(\omega) \cdot \alpha_{G_+}$? Dies gilt mathematisch exakt auf keinen Fall, wenn aber die sich im positiven Teil befindlich Gauß-Funktion und die sich im negativen Teil befindliche Gauß-Funktion nur wenig überlappen, dann gilt diese Beziehung näherungsweise schon. Die DC-Komponenten sind auch nicht Null, aber bei geringer Überlappung sind diese auch näherungsweise Null. Daher gilt:

Wenn das Frequenzband schmalbandig ist und dessen Zentrum ω_0 „genügend weit weg vom Koordinatenursprung“ ist, dann sind Gaborfilter tatsächlich näherungsweise Quadraturfilter.

Für zweidimensionale Bilder kann man sich die Quadratur-Gabor-Filter entsprechend aus Abschn. 6.4 selbst zusammenstellen und parametrisieren. In der Literatur werden Gabor-Filter oft als Quadratur Filter bezeichnet. Der Vorteil der Gaborfilter ist zweifelsohne die optimale Auflösung im Ortsraum als auch im Frequenzraum entsprechend der Unschärferelation. Weiterhin kann man für die Filter sowohl im Ortsraum als auch im Frequenzraum analytische Ausdrücke angeben (nicht zu verwechseln mit analytischen Funktionen). Die Gaborfilter haben allerdings drei Nachteile:

- Sie sind genaugenommen keine Quadraturfilter. Nur unter oben genannten Voraussetzungen sind sie näherungsweise Quadraturfilter.
- Die Menge aller Gaborfilter bilden kein Orthonormalsystem, sie enthalten damit Redundanzen.
- Die Gaborfilter haben keinen kompakten Support.

Aus diesem Grunde gibt es in der Literatur Versuche im Frequenzraum andere geeignete Bandpassfilter als Quadraturfilter zu entwickeln. Je nach Zielkriterium ergeben sich dann andere Vor- und Nachteile. So gibt es z. B.

- log-Gabor Filter
- S-Gabor Filter (Stretched Gabor Filter)
- Gaussche Ableitungsfilter
- DoG Filter
- Cauchy Filter
- Deriche Filter
- Sarkar and Boyer Filter.

Näheres dazu findet man in [6].

7.1 Tiefpassfilter

Vom Gibbsschen Phänomen kennen wir schon den Begriff „idealer Tiefpass“. Dort haben wir bemerkt, dass beim idealen Tiefpass „Ringing-Artefakte“ in der Nähe von Kanten entstehen. Daher ist ein idealer Tiefpass im Sinne der Datenkompression überhaupt nicht ideal. Idealer Tiefpass heißt nun, wir multiplizieren die Fourierkoeffizienten eines Bildes mit der Rechteckfunktion. Wir schreiben dies einmal für das Modell $A1[X]$ auf:

$$\alpha_k = \begin{cases} 1 & |k| \leq n \\ 0 & \text{sonst.} \end{cases} \quad (7.1)$$

Durch die Multiplikation im Frequenzraum haben wir im Ortsraum eine Faltung des Bildes $f(x)$ mit der Funktion $h(x)$, deren Fouriertransformierte die Rechteckfunktion darstellt. Daher transformieren wir nun die Rechteckfunktion zurück:

$$\begin{aligned} \sqrt{X} \cdot h(x) &= D_n(x) = \sum_{k=-n}^n 1 \cdot e^{+2\pi i k \frac{x}{X}} = 1 + 2 \cdot \sum_{k=1}^n \cos\left(2\pi k \frac{x}{X}\right) \\ &= \begin{cases} \frac{\sin((2n+1)\pi \frac{x}{X})}{\sin(\pi \frac{x}{X})} & x \neq l \cdot X \\ 2n+1 & x = l \cdot X. \end{cases} \end{aligned} \quad (7.2)$$

Die Funktion $D_n(x)$ wird als **Dirichletscher Integralkern** bezeichnet und ist das Analogon für die sinc-Funktion beim Modell $A1[-\infty, +\infty]$. Für diese Funktion gilt $\lim_{n \rightarrow \infty} D_n(x) = \delta(x)$, womit wir eine weitere Darstellung der δ -Funktion haben, allerdings mit der Periode X , d. h. es entsteht ein Dirac-Kamm. Weiterhin ist dieser Grenzwert im Sinne von Distributionen aufzufassen, siehe auch (4.78). Die Filterung einer Funktion

$f(x)$ mit einem idealen Tiefpass bedeutet mathematisch die Approximation der Funktion $f(x)$ durch eine Partialsumme $f_n(x)$ der Fourierreihe von $f(x)$ mit n Summanden. Folglich können wir diese Partialsummen durch eine Faltung von $D_n(x)$ mit $f(x)$ darstellen:

$$f_n(x) = \frac{1}{\sqrt{X}} D_n(x) * f(x) = \frac{1}{\sqrt{X}} \int_0^X D_n(x - \tau) f(\tau) d\tau. \quad (7.3)$$

Damit kann man das Konvergenzverhalten von Fourierreihen $f_n(x)$ mit dem Dirichlet-schen Integralkern untersuchen. Es gibt auch Anwendungen, bei denen ein idealer Tiefpass sogar erwünscht ist, siehe Abschn. 5.1.

Wählen wir die Manipulation im Frequenzraum etwas „weicher“, z. B. eine Dreiecks-funktion, dann erhalten wir einen „nichtidealen“ Tiefpass:

$$\alpha_k = \begin{cases} 1 - \frac{|k|}{n} & |k| \leq n \\ 0 & \text{sonst,} \end{cases} \quad (7.4)$$

dann ergibt sich für $h(x)$:

$$\begin{aligned} \sqrt{X} \cdot h(x) &= \sum_{k=-n}^n \left(1 - \frac{|k|}{n}\right) \cdot e^{+2\pi i k \frac{x}{X}} = 1 + 2 \cdot \sum_{k=1}^n \left(1 - \frac{k}{n}\right) \cos\left(2\pi k \frac{x}{X}\right) \\ &= \begin{cases} \frac{1}{n+1} \frac{\sin^2((n+1)\pi \frac{x}{X})}{\sin^2(\pi \frac{x}{X})} & x \neq l \cdot X \\ n+1 & x = l \cdot X. \end{cases} \end{aligned} \quad (7.5)$$

Beispiele für weitere Tiefpassfilter sind das Mittelwertfilter (Rechteckfunktion) und das Gaußfilter.

7.2 Unscharfe Maskierung (Unsharp Masking)

Dieser Begriff schafft eigentlich mehr Verwirrung. Er hat sich aus der analogen Photogra- phie bis heute erhalten. Man wollte im Fotolabor ein aufgenommenes Bild „nachschärfen“:

- Der Negativ-Film wurde mit einer Milchglasscheibe bedeckt und damit der Film be- lichtet, so dass ein leicht unscharfes Positiv entstand.
- Dieses unscharfe Positiv legte man über das originale Negativ. Diese beiden zusammen bilden nun die Unschärfe-Maske. Mit dieser Unschärfe-Maske wurde nun ein Film kur- zeitig vorbelichtet.
- Die Vorbelichtung wurde abgebrochen und weiterhin mit dem original Negativ weiter belichtet bis das Bild fertig ist.

Wenn wir dies nun „übersetzen“, so finden wir:

Durch die Milchglasscheibe erzeugen wir vom Original $f_{i,j}$ ein „verschmiertes Bild“ $h * f$, dieses wird vom Original abgezogen, womit wir die Maske $f_{maske} = f - h * f$ erhalten. Diese Maske betont besonders die Kanten im Bild, daher wird diese Maske nun zum Original addiert, wobei man die Maske mit einem Wichtungsfaktor c versehen muss, um die Stärke der Kantenschärfung zu gewichten:

$$f' = f + c \cdot (f - h * f) = (1 + c)f - c \cdot h * f = ((1 + c)\delta - c \cdot h) * f. \quad (7.6)$$

Für $c > 1$ nennt man dies auch *highboost filtering*. Die Faltungsmaske im Ortsraum $g = (1 + c)\delta - c \cdot h$ kann man sich beliebig konstruieren durch einen Tiefpass-Filter h und einen Faktor c . Wählen wir z. B. für h einen 3×3 -Mittelwertfilter und $c = 9$, dann erhalten wir im Ortsraum die Faltungsmaske

$$g = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}. \quad (7.7)$$

Spektral heißt *unsharp masking*, zum Spektrum des Bildes f addieren wir das Hochpassgefilterte Spektrum des Bildes. Das Spektrum von $g = (1 + c)\delta - c \cdot h$ für eindimensionale, diskrete Funktionen lautet:

$$\alpha_k(g) = (1 + c)\alpha_k(\delta) - c \cdot \alpha_k(h) = \frac{(1 + c)}{\sqrt{N}} - c \cdot \alpha_k(h). \quad (7.8)$$

Wir sehen, dies ist kein typischer „Pass“-Filter, sondern die hohen Frequenzen werden verstärkt, aber die tiefen Frequenzen je nach Wahl von c nicht ausgeblendet. Man nennt daher solche Filter oft *high-frequency-emphasis filter*. Man kann diese noch verallgemeinern, indem man zwei Gewichte einführt:

$$\alpha_k(g) = \frac{(c_1 + c_2)}{\sqrt{N}} - c_2 \cdot \alpha_k(h). \quad (7.9)$$

Bemerkung Moderne Bildbearbeitungsprogramme verfügen alle über eine Schärfungsfunktion mit *unsharp masking*. Oft muss man noch eine Schwelle einstellen. Durch den additiven Hochpass werden auch Rauschen und andere feine Störungen verstärkt. Durch die Schwelle unterdrückt man dann lokal diese Verstärkungen. Dann ist dieses Filter aber nicht mehr das klassische LSI-Filter, sondern ein nichtlineares Filter.

7.3 Pruning-Filter

Das Pruning-Filter ist ein typischer Versuch, das Signal-Rausch-Verhältnis im Frequenzraum zu verbessern. Es ist ein klassisches nichtlineares Filter. Bezuglich des Modells $g = h * f + N$ bezeichne $\alpha_k(g)$ das Spektrum von g und $\alpha_k(N)$ das Spektrum des Rauschens, welches auch ein Zufallsfeld darstellt. Es sei nun (von irgendwoher) der Erwartungswert $E\{|\alpha_k(N)|^2\}$ bekannt (das mittlere Leistungsspektrum). Nun manipulieren wir nur das Anplitudenspektrum von g , die Phase belassen wir wie sie ist. Wir ziehen nun vom Leistungsspektrum von g das mittlere Leistungsspektrum des Rauschens ab:

$$|\alpha_k(g)^p| = \begin{cases} \sqrt{|\alpha_k(g)|^2 - E\{|\alpha_k(N)|^2\}}, & |\alpha_k(g)|^2 - E\{|\alpha_k(N)|^2\} > 0 \\ 0, & \text{sonst.} \end{cases} \quad (7.10)$$

Für die Phase φ gilt $\varphi(\alpha_k(g)^p) = \varphi(\alpha_k(g))$. Anschließend wird das Spektrum $\alpha_k(g)^p$ zurücktransformiert.

7.4 Homorphe Filter

Wir betrachten ein klassisches Beleuchtungsmodell:

$$f(x, y) = i(x, y) \cdot r(x, y), \quad (7.11)$$

wobei $i(x, y)$ die Beleuchtung und $r(x, y)$ die Reflexion ist. Wenn man annimmt, dass tiefe Frequenzen für die Beleuchtung zuständig sind und hohe Frequenzen für die Reflexion, dann möchte man entsprechend diesem Modell die hohen Frequenzen verstärken, um z. B. den Kontrast anzuheben. Dies geht aber entsprechend diesem Modell nicht, da Beleuchtung und Reflexion durch eine Multiplikation verbunden sind. Wenn dies eine Addition wäre, dann könnte man diese trennen. Folglich logarithmieren wir beide Seiten, erhalten somit die Summe der Logarithmen: $\log f = \log i + \log r$. Das logarithmierte Bild $\log f$ transformieren wir mit der DFT, wenden auf das Spektrum einen Hochpassfilter an, transformieren zurück und „entlogarithmieren“. Das gefilterte Bild wird dann im Allgemeinen kontrastreicher sein.

7.5 DoB-Filter

Im Folgenden sollen einige Beispiele für typische Bandpaßfilter angegeben werden. Wenn einem Tiefpass ein Hochpass folgt, entsteht ein Bandpass-Filter. Typisch sind Glättungsfilter (z. B. Gauß-Filter) gefolgt von Ableitungsfilter (z. B. Laplace-Filter), so entsteht z. B. der bekannte Kantendetektions-Filter, der LoG-Filter oder auch *Mexican hat* genannt, siehe auch [29]. Diesen kann man näherungsweise auch als Differenz von zwei Gauß-Filters darstellen, wobei die Varianzen in einem bestimmten Verhältnis stehen müssen,

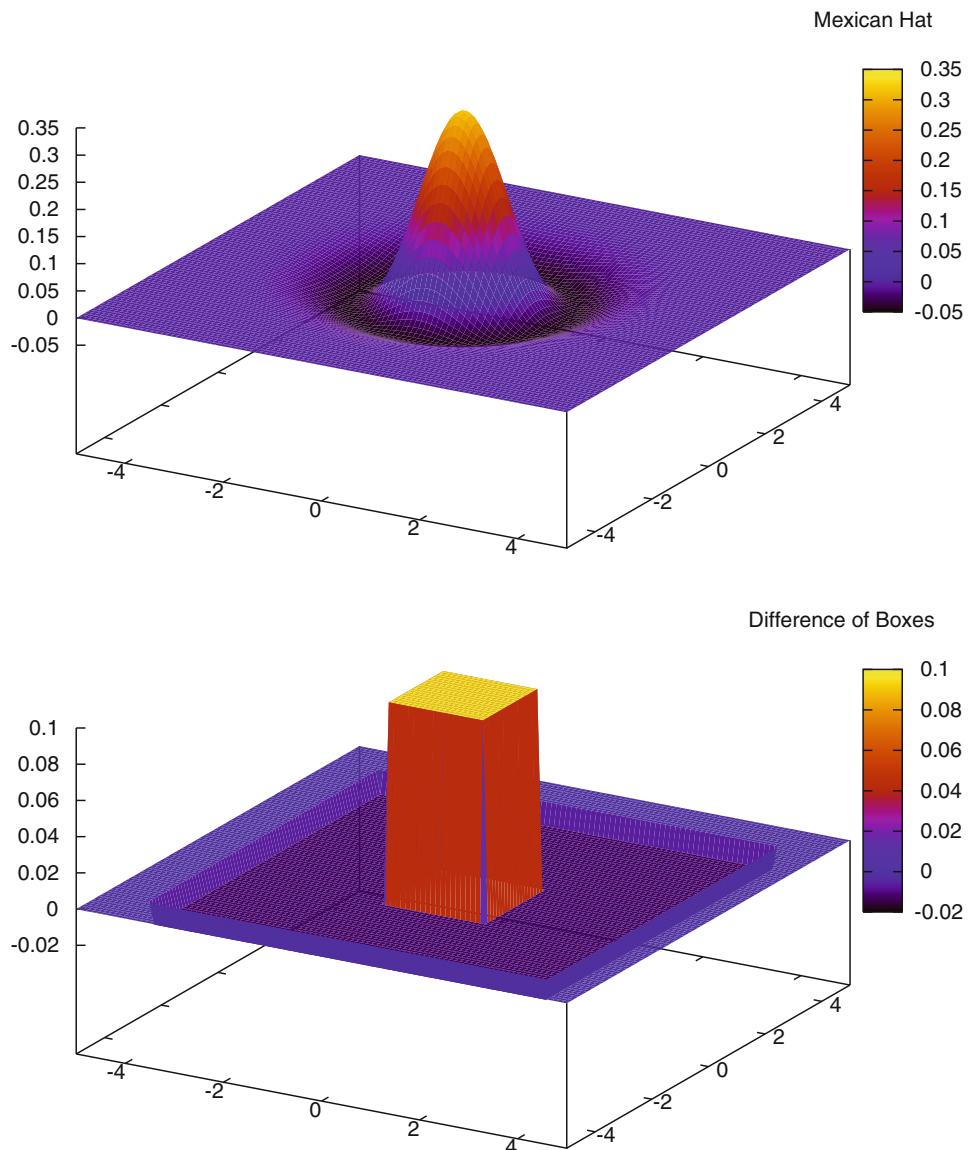


Abb. 7.1 Vergleich Mexican-hat- und DoB-Filter

siehe [29]. Rosenfeld, siehe [60], hat bereits 1971 vorgeschlagen, den DoG-Filter wiederum durch einen DoB-Filter (**Difference of Boxes**) zu approximieren, weil dieser eine nahezu konstante Zeitkomplexität bezüglich der Filtergröße besitzt und somit sehr effektiv zu implementieren ist, siehe [59]. Die Implementierung hängt eng mit der Nutzung von *integral images* zusammen, siehe Abschn. 19.9.6. Die DoB-Filter sind demnach typische



Abb. 7.2 Beispiele für die Anwendung der DoB-Filter bei der Texterkennung (Ausweise und Kfz-Kennzeichen)

Bandpass-Filter und hängen von zwei Parametern m und M ab:

$$\begin{aligned} \text{DoB}_{m,M}(g) &= \frac{1}{m} \sum_{i=1}^m g_i - \frac{1}{M} \sum_{l=1}^M g_l \\ &= \frac{M}{M-n} \left(\frac{1}{m} \sum_{i=1}^m g_i - \frac{1}{M-m} \sum_{l=m+1}^M g_l \right). \end{aligned} \quad (7.12)$$

Für $M = 7$ und $m = 3$ ist in (8.19) eine 2D-Maske angegeben. Die DoB-Filter kann man effektiv einsetzen zur Vorverarbeitung von OCR-Dokumenten, z. B. Nummernschild-Erkennung, siehe Abb. 7.2. Die Parameter m und M richten sich nach der Schriftgröße, der Breite der Linien und den Abständen zwischen den Symbolen. Deshalb ist es auch sinnvoll, mehrere DoB-Filter zu verwenden und das beste Ergebnis zu verwenden. Wie in Abb. 7.2 zu sehen ist, wurde der negative Output der Filter auf Null gesetzt, nur der positive Output wurde verwendet. Selbst der 1D-DoB-Filter (7.12) kann sinnvoll eingesetzt werden. Wir nehmen einmal an, in einem Bild von Blutgefäßen soll die Breite der Gefäße an verschiedenen Stellen gemessen werden. Dazu detektiert man das Gefäß und bestimmt an der interessierenden Stelle die Querrichtung des Gefäßes. Nun tastet man das Gefäß entlang dieser Querrichtung mittels bilinearer Interpolation ab und erhält eine 1D-Querschnittsfunktion. An Hand dieser 1D-Querschnittsfunktion ist nun die Breite des Gefäßes zu bestimmen. In der Abb. 7.3 ist im oberen Teil eine solche 1D-Querschnittsfunktion zu sehen. Das lokale Maximum in der Mitte beschreibt das Gefäß. Nun ist die Ausdehnung dieses Extremums zu bestimmen. Die Kurve ist nicht nur verrauscht, sondern der stetige Abfall der Kurve deutet auch auf Shading hin, so dass die Breitenmessung schwierig zu sein scheint. Nun wenden wir auf diese Funktion ein 1D-DoB-Filter an. Die Größe des Filters sollte minimal 2 bis 3-mal größer als die Gefäßbreite

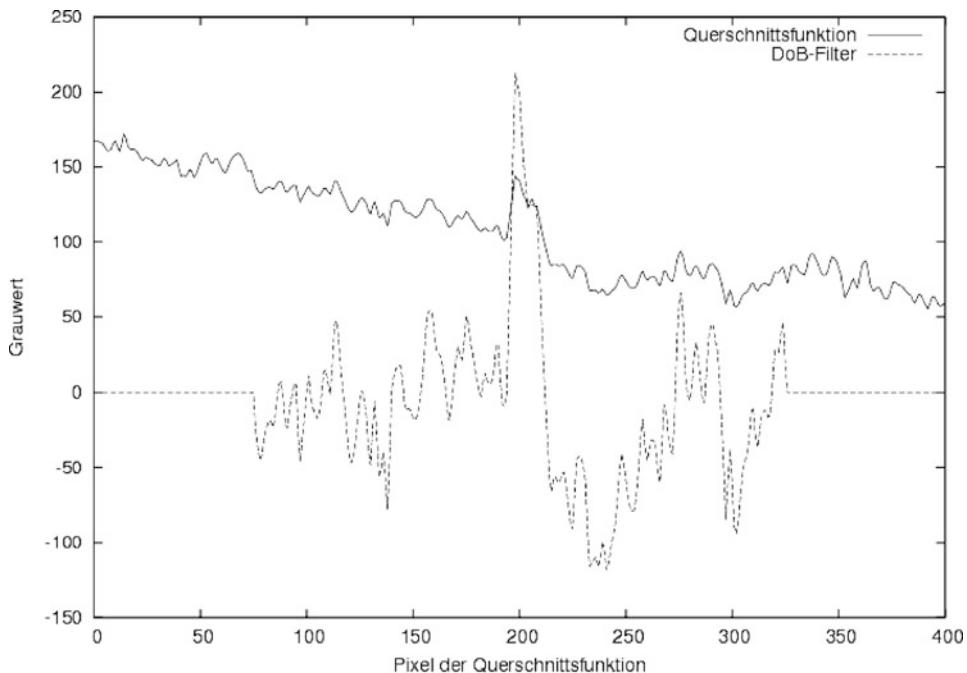


Abb. 7.3 Senkrecht zu einem Gefäß in einem medizinischen Bild ist der Grauwertverlauf als Querschnittsfunktion zu sehen. Diese Querschnittsfunktion wurde mit einem 1D-DoB-Filter transformiert und ergibt die zweite Funktion. Die Breite des Gefäßes kann nun leicht durch Vermessung des zentralen Peaks ermittelt werden

sein, dies muss man durch A-priori-Wissen garantieren. In der Abb. 7.3 ist nun im unteren Teil das Ergebnis eines 1D-DoB-Filters zu sehen, wobei der Randbereich entsprechend der Größe des Filters einfach auf Null gesetzt wurde. Da wir nun einen Punkt auf dem Gefäß kennen, können wir diesen als Startpunkt eines 1D-Region Growing-Verfahrens nutzen und im *Output* des DoB-Filters rechts und links von diesem Startpunkt die Nulldurchgänge suchen.

7.6 Scharfe Bilder (Fokussierung)

Im Folgenden soll das klassische Problem des automatischen Scharfstellens einer Kamera betrachtet werden. Als gegebenen betrachten wir eine Serie von Bildern mit verschiedenem Fokus und wir wollen das schärfste Bild ermitteln. Wir stellen uns eine solche Serie von Bildern vor, beginnend mit einer großen Unschärfe, dann werden die Bilder immer schärf-fer bis sie wieder unscharf werden. In Abb. 7.4 sind drei Bilder aus einer Bildserie einer idealen Schwarz-Weiß-Kante zu sehen, wobei das mittlere Bild das schärfste Bild ist. Zur Berechnung des schärfsten Bildes benötigen wir ein Unglattheitsmaß. Dafür kommen Ab-

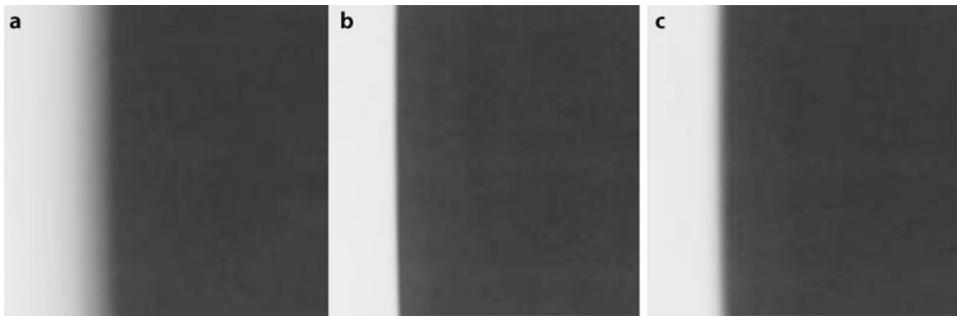


Abb. 7.4 a Erstes Bild (unscharf), b „Mittleres“ Bild (scharf), c Letztes Bild (unscharf) einer Kante aus einer Serie von verschiedenen fokussierten Bildern

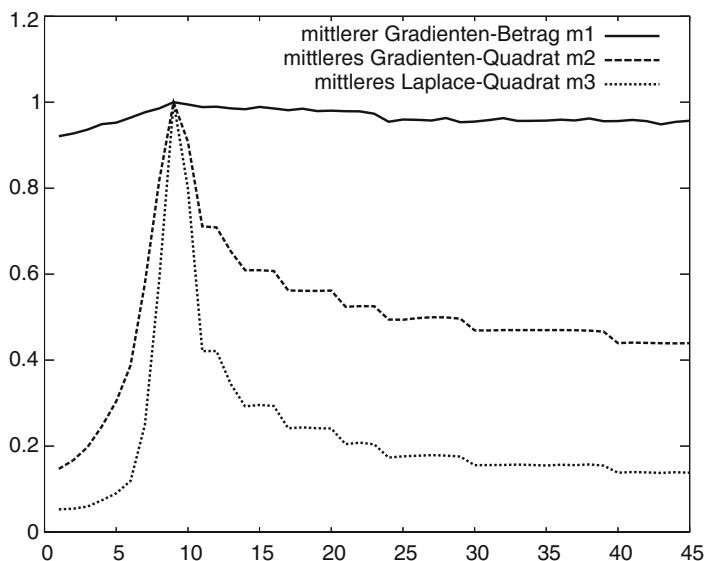


Abb. 7.5 Vergleich der Maße m_1, m_2, m_3

leitungen in Frage. Da es bei unserem Beispiel genügt, nur horizontale Zeilen des Bildes zu betrachten, beschränken wir uns auf eindimensionale Funktionen $f(x)$. Als Maß könnte man z. B. die L1-Norm $m_1 = \int_a^b |f'(x)|dx$ verwenden und das Bild berechnen, bei dem m_1 maximal ist. Haben wir wie im Beispiel eine ideale Stufenkante aufgenommen, so ist theoretisch die erste Ableitung immer nichtnegativ und wir können den Betrag weglassen, d. h. es gilt $m_1 = \int_a^b |f'(x)|dx = \int_a^b f'(x)dx = f(b) - f(a)$. Damit hängt das Maß m_1 überhaupt nicht mehr von der Steilheit der Kante ab. Wir wollen aber doch gerade das Bild mit der steilsten Kante, da dies als schärfstes Bild zu interpretieren wäre. Um dies zu vermeiden, entscheiden wir uns für die L2-Norm der ersten Ableitung $m_2 = \int_a^b |f'(x)|^2 dx$ oder für die L2-Norm der 2. Ableitung $m_3 = \int_a^b |f''(x)|^2 dx$. In der Abb. 7.5 sind für 45 Bil-

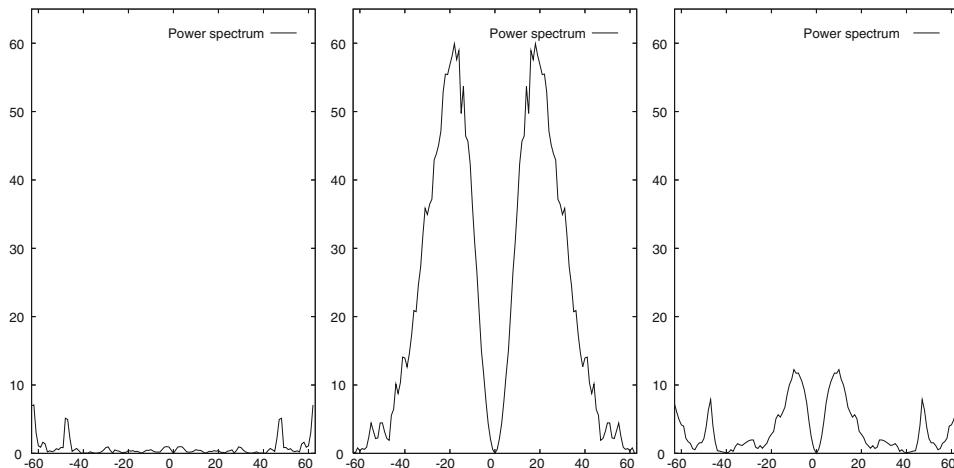


Abb. 7.6 Leistungsspektren: unscharf, scharf, unscharf

der (in Abb. 7.4 sind drei davon zu sehen) die Maße $m1$, $m2$ und $m3$ als Kurve für diese Bilder berechnet. Die deutlichste Ausprägung des uns interessierenden Maximums hat die Kurve mit dem Maß $m3$. Die Kurve des Maßes $m1$ ist praktisch unbrauchbar. Bei anderen Bildern, wenn viele kleine Strukturen enthalten sind und nicht nur eine Kante wie im Beispiel, dann ist auch das Maß $m1$ durchaus brauchbar, auch wenn das Maximum des Maßes $m2$ viel deutlicher ausgeprägt ist. Aus vielen praktischen Tests ist ersichtlich, daß das Maß $m3$ deutlich überlegen ist. Da gibt es sicher einen Zusammenhang zu den DOB-Filters, die ja auch grob auf den 2. Ableitungen beruhen. Bei Bildern nehmen wir für die zweite Ableitung den Laplace-Filter. Was hat dies nun alles mit Spektren zu tun? Nun Ableitungsfilter sind Hochpaßfilter. Vom gefilterten Bild berechnen wir die L2-Norm, die nach der Parsevalschen Gleichung der Summe bzw. dem Integral des Leistungsspektrums entspricht. Wir suchen also von den hochpaßgefilterten Bildern dasjenige, welches das „energiereichste“ Leistungsspektrum besitzt. Die Leistungsspektren der drei Bilder bezüglich Abb. 7.4a–c sind in der Abb. 7.6 zu sehen. Dabei wurden die Spektren eindimensional berechnet, d. h. die Bilder wurden nur zweilenweise bearbeitet und von allen Zeilen das Mittel genommen. Man sieht wie sich das Leistungsspektrum des scharfen Bildes deutlich von den Leistungsspektren der unscharfen Bilder abhebt. Nun folgen

Anwendungen

- Ein übliche Methode ist das automatische Scharfstellen bei einer handelsüblichen Kamera.
- Da die Schärfe von der Brennweite und der Blende abhängt, kann man die Schärfe bzw. Unschärfe lokal zur Tiefenberechnung nutzen, siehe Abschn. 16.6.8. Dazu muss lokal aus einer Serie von Aufnahmen das schärfste Bild berechnet werden.

- Ein Bild mit einer großen Tiefenschärfe erreicht man durch eine sehr kleine Blende. Der Nachteil ist, dass nicht genügend Licht eindringt. Daher benutzt man eine große Blende und macht eine Serie von Aufnahmen mit verschiedenen Fokussierungen. Durch die große Blende können sehr kurze Belichtungszeiten benutzt werden, so dass ein „Wackeln“ der Kamera hoffentlich keine Rolle spielt. Nun wird das eigentliche Bild aus allen Bilder zusammengesetzt, indem man pro Pixel den Grauton des schärfsten Bildes benutzt. Durch die Fokussierung entstehen auch geometrische Veränderungen, so dass man noch das „richtige“ Pixel finden muss. Durch die Übergänge von scharfen zu unscharfen Bildsegmenten entstehen Kanten, die optisch stören. Diese muss man noch mit entsprechenden Filtertechniken glätten.

Diese Technik des Zusammensetzens nennt man *focus stacking*.

- Das *focus stacking* ist z. B. quasi-Standard in der Konfokalen Laser Raster Mikroskopie. Die Bilder im *z-Stack* sind zum Teil scharf und zum Teil unscharf. Da aber keine geometrischen Transformationen vorliegen, kann man relativ einfach das *focus stacking* anwenden.

Im folgenden Kapitel werden wir uns mit dem Filtern direkt im Ortsraum beschäftigen und typische lineare aber auch nichtlineare Filter kennenlernen. Weiterhin werden wir die Vorteile von Richtungsfilters aufzeigen.

8.1 Lineare und verschiebungsinvariante Filter

8.1.1 Berechnung der Filterkoeffizienten

LSI-Filter werden durch Filtermasken repräsentiert, die ortsunabhängig und bildunabhängig sein müssen, siehe Abschn. 2.3. In der einschlägigen Literatur werden oft die Masken für bestimmte Aufgaben angegeben, wie z. B. die Masken für Mittelwertfilter (Boxfilter), Gaußfilter, Laplacefilter, Prewittfilter, Sobelfilter, Kirschfilter, Mexican-Hat-Filter. Wir wollen hier einen anderen Weg gehen und angeben, wie man diese Masken herleiten kann. Ableitungsfilter entsprechen formal den in der Numerischen Mathematik verwendeten Differenzenoperatoren, sodass wir das Vorgehen aus der Numerischen Mathematik übernehmen können. Dort ist es üblich, an die diskreten Funktionswerte ein Interpolationspolynom (oder alternativ ein Approximationspolynom) anzupassen, um dann die Koeffizienten des Polynomes zu benutzen. Da bei einem realen Bild die Grauwerte immer verrauscht sind, entscheiden wir uns für ein Approximationspolynom anstatt eines Interpolationspolynoms. Dazu bestimmen wir von einem Polynom L -ten Grades $g(x, y)$, das wir an der Pixelposition (x_0, y_0) entwickeln, die Koeffizienten in Abhängigkeit von den Grauwerten aus einer bestimmten Pixelumgebung U des Aufpunktes (x_0, y_0) :

$$g(x, y) = \sum_{n=0}^L \sum_{m=0}^n a_{n-m,m} (x - x_0)^{n-m} (y - y_0)^m. \quad (8.1)$$

Wenn die gewählte Umgebung U zum Grad L des Polynomes „passt“, dann können wir mit einem Schätzer (LSE-Methode, LAD-Methode) die Koeffizienten $a_{p,q}$ aus den Grauwerten $g_{k,l}$ der Pixelumgebung berechnen. Umgebung, Grad des Polynomes und Gewichte für die einzelnen Pixel hängen von der Aufgabe ab, die das Filter zu lösen hat. Ist das Ziel ein Glättungsfilter zu entwerfen, dann ist die Mittelwerterhaltung des Bildes eine sinnvolle Restriktion. Durch die Mittelwerterhaltung wird das Bild nach der Filterung nicht heller oder dunkler.

Da die Modellgleichung (8.1) linear in den Unbekannten $a_{p,q}$ ist, hängen die Unbekannten auch tatsächlich nur linear von den Grauwerten der Umgebung U ab (falls wir die LSE-Methode benutzen, siehe Abschn. 21.2.1). Dazu sollen nun nachfolgend einige Beispiele angegeben werden. Zur einfachen Schreibweise legen wir den Koordinatenursprung in den Aufpunkt (x_0, y_0) .

- Wir passen ein Polynom ersten Grades ($L = 1$) – also eine Ebene an:

$$S = \sum_{(k,l) \in U} (a_{0,0} + a_{1,0}k + a_{0,1}l - g_{k,l})^2 \rightarrow \text{Minimum.} \quad (8.2)$$

Zur Lösung der LSE-Aufgabe (siehe Abschn. 21.2.1) gehen wir formal vor: zunächst schreiben wir die Messgleichungen auf:

$$a_{0,0} + a_{1,0}k + a_{0,1}l = g_{k,l}, \quad (k, l) \in U. \quad (8.3)$$

Optional könnten die Messgleichungen für jedes Pixel individuell gewichtet werden. Wir nehmen aber der Einfachheit halber eine uniforme Gewichtung an. Nun multiplizieren wir beide Seiten mit der transponierten Koeffizientenmatrix und erhalten die Gaußschen Normalengleichungen:

$$\begin{pmatrix} \sum_{k,l \in U} 1 & \sum_{k,l \in U} k & \sum_{k,l \in U} l \\ \sum_{k,l \in U} k & \sum_{k,l \in U} k^2 & \sum_{k,l \in U} kl \\ \sum_{k,l \in U} l & \sum_{k,l \in U} kl & \sum_{k,l \in U} l^2 \end{pmatrix} \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{0,1} \end{pmatrix} = \begin{pmatrix} \sum_{k,l \in U} g_{k,l} \\ \sum_{k,l \in U} kg_{k,l} \\ \sum_{k,l \in U} lg_{k,l} \end{pmatrix}. \quad (8.4)$$

Die Normalenmatrix ist wie immer die Kovarianzmatrix oder die Matrix der zweiten Momente. Wenn wir diese Normalengleichungen nach den Unbekannten auflösen wollen, müssen wir mit der inversen Matrix multiplizieren und erhalten die Lösung. Entweder wir rechnen die inverse Matrix direkt aus oder wir invertieren sie mit numerischen Methoden. Im vorliegenden Beispiel ist es einfacher, die Umgebung quadratisch und symmetrisch zu wählen. Dadurch verschwinden alle zweiten Momente außerhalb der Hauptdiagonalen und wir können trivialerweise nach den Unbekannten auflösen:

$$a_{0,0} = \frac{1}{\sum_{k,l \in U} 1} \sum_{k,l \in U} g_{k,l}, \quad a_{1,0} = \frac{\sum_{k,l \in U} kg_{k,l}}{\sum_{k,l \in U} k^2}, \quad a_{0,1} = \frac{\sum_{k,l \in U} lg_{k,l}}{\sum_{k,l \in U} l^2}. \quad (8.5)$$

Die Größe $a_{0,0}$ dient zur Berechnung des Funktionswertes im Aufpunkt, $a_{1,0}$ zur Berechnung der partiellen Ableitung nach x und $a_{0,1}$ zur Berechnung der partiellen Ableitung nach y . Nun können wir obige Lösungen durch Filtermasken beschreiben. Damit ergibt sich beispielsweise für eine (5×5) -Umgebung:

$$a_{0,0} \leftarrow \frac{1}{25} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad a_{1,0} \leftarrow \frac{1}{50} \begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix} \quad (8.6)$$

$$a_{0,1} \leftarrow \frac{1}{50} \begin{pmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}. \quad (8.7)$$

Um beispielsweise die Sobel-Masken abzuleiten, braucht man nur eine (3×3) -Umgebung zu wählen und die mittleren Pixel höher zu wichten. Um Masken für die Ableitungen zweiter Ordnung abzuleiten, benötigen wir ein Polynom mindestens zweiten Grades.

- Wir passen ein Polynom zweiten Grades ($L = 2$) an:

$$S = \sum_{(k,l) \in U} (a_{0,0} + a_{1,0}k + a_{0,1}l + a_{2,0}k^2 + a_{1,1}k \cdot l + a_{0,2}l^2 - g_{k,l})^2 \rightarrow \text{Minimum.} \quad (8.8)$$

Wenn wir die Gaußschen Normalengleichungen aufstellen, sind nicht alle Elemente außerhalb der Hauptdiagonalen Null. Unmittelbar können wir nach

$$\begin{aligned} a_{1,1} &= \frac{1}{\sum_{k,l \in U} k^2 l^2} \sum_{k,l \in U} k l g_{k,l}, \\ a_{1,0} &= \frac{\sum_{k,l \in U} k g_{k,l}}{\sum_{k,l \in U} k^2}, \quad a_{0,1} = \frac{\sum_{k,l \in U} l g_{k,l}}{\sum_{k,l \in U} l^2} \end{aligned} \quad (8.9)$$

auflösen. Nach den anderen Koeffizienten aufgelöst, ergibt:

$$\begin{pmatrix} a_{0,0} \\ a_{2,0} \\ a_{0,2} \end{pmatrix} = \begin{pmatrix} \sum_{k,l \in U} 1 & \sum_{k,l \in U} k^2 & \sum_{k,l \in U} l^2 \\ \sum_{k,l \in U} k^2 & \sum_{k,l \in U} k^4 & \sum_{k,l \in U} k^2 l^2 \\ \sum_{k,l \in U} l^2 & \sum_{k,l \in U} k^2 l^2 & \sum_{k,l \in U} l^4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \sum_{k,l \in U} g_{k,l} \\ \sum_{k,l \in U} k^2 g_{k,l} \\ \sum_{k,l \in U} l^2 g_{k,l} \end{pmatrix}. \quad (8.10)$$

Wählen wir für U eine (3×3) -Umgebung, dann ist:

$$\begin{pmatrix} a_{0,0} \\ a_{2,0} \\ a_{0,2} \end{pmatrix} = \frac{1}{18} \begin{pmatrix} 10 & -6 & -6 \\ -6 & 9 & 0 \\ -6 & 0 & 9 \end{pmatrix} \cdot \begin{pmatrix} \sum_{k,l \in U} g_{k,l} \\ \sum_{k,l \in U} k^2 g_{k,l} \\ \sum_{k,l \in U} l^2 g_{k,l} \end{pmatrix}. \quad (8.11)$$

Daraus können wir wiederum die Filtermasken ablesen:

$$a_{0,0} \leftarrow \frac{1}{9} \begin{pmatrix} -1 & 2 & -1 \\ 2 & 5 & 2 \\ -1 & 2 & -1 \end{pmatrix}, \quad a_{2,0} \leftarrow \frac{1}{6} \begin{pmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{pmatrix}, \quad a_{0,2} \leftarrow \frac{1}{6} \begin{pmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{pmatrix}. \quad (8.12)$$

Wir sehen, dass $a_{0,0}$ als Approximation für den Funktionswert am Aufpunkt nicht unbedingt durch das Mittelwertfilter realisiert werden muss. Weiterhin können wir nun

$$\Delta_{xx} + \Delta_{yy} \leftarrow 2a_{2,0} + 2a_{0,2} \leftarrow \frac{1}{3} \begin{pmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{pmatrix} \quad (8.13)$$

als Maske für den Laplace-Operator verwenden.

- Den Entwurf der Masken nach dieser Approximationsmethode können wir leicht programmtechnisch automatisieren. Nach Vorgabe der Umgebung, des Grades des Polynomes und der Gewichte der einzelnen Pixel können wir die Gaußschen Normalengleichungen aufstellen, die Matrix invertieren und anschließend die Filtermasken berechnen.

Eine weitere Möglichkeit, um LSI-Filter zu entwerfen, ist die Diskretisierung (Abtastung) analoger Funktionen. Ein typisches Beispiel ist die Gaußfunktion, wobei wir noch den Parameter σ (Standardabweichung) festlegen müssen. Weiterhin ist dabei das Abtasttheorem zu beachten, sodass die Filtermaske möglichst keine Unterabtastung realisiert. Man kann auch LSI-Filter mit den Methoden der Stochastik entwerfen. So kann die Binomialverteilung als Approximation der Normalverteilung dienen. Daher können wir Binomialfilter als Näherung für Gaußfilter benutzen. Die Binomialverteilung

$$P(X = k) = \binom{n}{k} p^k q^{n-k} \quad (8.14)$$

benutzen wir mit $p = \frac{1}{2}$. So ergibt sich das 1D-Filter mit $n = 2$ zu $\text{binom}_3 = \frac{1}{4}(1, 2, 1)$. Die 2D-Binomialfilter erzeugen wir mit einer 2D-Faltung der 1D-Filter, siehe auch (2.8) und (2.9):

$$\text{binom}_{3,3} = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} * \frac{1}{4} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}, \quad (8.15)$$

oder in Vektorschreibweise als dydisches Produkt:

$$\text{binom}_{3,3} = \frac{1}{4} \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \cdot \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}. \quad (8.16)$$

Weiterhin können wir LSI-Filter durch Hintereinanderausführung zweier bekannter LSI-Filter entwerfen. Dazu diene folgendes

Beispiel Der Mexican-Hat basiert darauf, dass man zuerst das Rauschen eines Bildes dämpft und anschließend mit dem Laplacefilter Kantenpunkte extrahiert. Diese beiden Filter kann man natürlich durch die Faltung oder Korrelation der Masken zu einem einzigen Filter zusammenfassen, siehe auch Abschn. 2.1.2. Wir wählen zur Rauschdämpfung das (3×3) -Binomialfilter und anschließend ein (3×3) -Laplacefilter und erhalten:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 4 & 3 & 1 \\ 3 & 0 & -6 & 0 & 3 \\ 4 & -6 & -20 & -6 & 4 \\ 3 & 0 & -6 & 0 & 3 \\ 1 & 3 & 4 & 3 & 1 \end{pmatrix}, \quad (8.17)$$

wobei wir eine (5×5) -Maske für den Mexican-Hat erhalten haben.

8.1.2 Implementierung von LSI-Filtern

Um 2D-LSI-Filter im Ortsraum effizient zu implementieren, bedarf es einiger Überlegungen. Die Grundidee ist oft, eine 2D-Filterung auf zwei 1D-Filterungen zurückzuführen. Dazu muss das Filter separierbar sein, d. h. ein Filter h muss $h_{i,j} = x_i \cdot y_j, \forall i, j$ zerlegbar sein, siehe auch (2.8) und (2.9). Dies kann man mit der Singulärwertzerlegung (SVD) feststellen, siehe Abschn. 22.3.2. Weiterhin kann man mit einem Approximationssatz den „Grad“ der mehrfachen Separierbarkeit bestimmen und ausnutzen, siehe ebenfalls Abschn. 22.3.2. Natürlich weiß man von gewissen Standardfiltern, ob sie separierbar sind oder nicht. So sind beispielsweise das klassische Mittelwertfilter, das klassische Gaußfilter und das Binomialfilter separierbar und lassen sich damit effektiv implementieren. Das Mittelwertfilter hat außer der Separierbarkeit noch eine günstige Zusatzeigenschaft:

- Implementiert wird infolge der Separierbarkeit nur ein 1D-Mittelwertfilter.
- Bis auf einen Normierungsfaktor werden die Grauwerte aus einer 1D-Umgebung summiert, wir speichern die Summe in einer Summationsvariablen. Wenn wir nun das 1D-Filter um eine Position verschieben, muss lediglich ein Grauwert aus der Summationsvariablen „ausgetragen“ und ein neuer Grauwert „eingetragen“ werden. Diese Ein- und Austragung hängt nicht von der Größe des Filters ab. Daher kann man das Mittelwertfilter mit konstanter Komplexität in Abhängigkeit von der Filtergröße implementieren.

Die effektive Implementierung des Mittelwertfilters kann man bei anderen Aufgaben geschickt ausnutzen, was folgende Beispiele veranschaulichen.

Beispiel 1 Es soll effektiv der Strukturtensor für alle Pixel eines Bildes

$$\begin{pmatrix} \sum_{x,y \in U} f_x^2 & \sum_{x,y \in U} f_x \cdot f_y \\ \sum_{x,y \in U} f_x \cdot f_y & \sum_{x,y \in U} f_y^2 \end{pmatrix} \quad (8.18)$$

implementiert werden, siehe Abschn. 17.5.2. Dazu bilden wir von dem Bild f mittels LSI-Filter die Ableitungsbilder f_x und f_y und daraus die drei Bilder f_x^2 , $f_x \cdot f_y$ und f_y^2 . Auf die drei Bilder wenden wir nun ein Mittelwertfilter entsprechend der Umgebungsgröße U an und erhalten je Pixel den Strukturtensor.

Beispiel 2 Zur Approximation des LoG-Filters nutzt man das DoB-Filter, siehe Abschn. 7.5. Die Filtermaske hängt von zwei Größen M und m ab, siehe (7.12). Für $M = 7$, $m = 3$ sieht die Maske folgendermaßen aus:

$$DoB \rightarrow \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & +1 & +1 & +1 & -1 & -1 \\ -1 & -1 & +1 & +1 & +1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}. \quad (8.19)$$

Dieser Maske sieht man nicht direkt an, dass man das Filter sehr effektiv implementieren kann. Entsprechend (7.12) ist diese Maske als Differenz zweier Mittelwertfilter darstellbar und daher mit konstanter Komplexität in Abhängigkeit von der Filtergröße implementierbar.

Beispiel 3 Wir nehmen einmal an, ein Binärbild sei gegeben. Die Aufgabe bestehe darin, ein Quadrat pixelweise über das Bild „zu schieben“ und jeweils die Anzahl der schwarzen Punkte im Quadrat zu zählen. Dies realisiert (bis auf einen Skalierungsfaktor) genau das Mittelwertfilter. Solche „Filter“ werden z. B. in der fraktalen Bildanalyse benötigt, z. B. bei der Berechnung der fraktalen Kästchendimension und der *lacunarity*, siehe z. B. (12.2).

8.2 Nichtlineare Filter

Für lineare Filter (lineare Operatoren L) muss

$$L(\lambda f + \mu g) = \lambda Lf + \mu Lg \quad \forall \lambda, \mu \text{ und Bilder } f, g \quad (8.20)$$

gelten. Für LSI-Filter gilt zusätzlich zu (8.20) noch die Verschiebungsinvarianz. Es gibt kaum praktisch relevante lineare Filter, die nicht verschiebungsinvariant sind. Dagegen gibt es viele praktisch relevante Filter, die nicht die Bedingung (8.20) erfüllen. Diese bezeichnen wir als nichtlineare Filter.

8.2.1 Funktionen von LSI-Filtern

Im Folgenden betrachten wir nichtlineare Funktionen von LSI-Filtern, insbesondere Funktionen von Ableitungen, als einfachste Form nichtlinearer Filter.

- **Gradientenapplikation**

Die einfachste nichtlineare Funktion ist der Betrag des Gradienten, der als Maß für die Stärke eines Kantenpunktes dient. Für den Gradienten ∇f gilt:

$$\text{grad } f = \nabla f = (f_x \quad f_y)^T, \quad |\nabla f| = \sqrt{f_x^2 + f_y^2}, \quad |\nabla f| \rightarrow \sqrt{a_{1,0}^2 + a_{0,1}^2}. \quad (8.21)$$

Damit ist der Betrag des Gradienten durch ein nichtlineares Filter zu berechnen, wobei die Größen $a_{1,0}$ und $a_{0,1}$ durch LSI-Filter entsprechend Abschn. 8.1.2 zu realisieren sind.

- **Gradientenrichtung**

Zusätzlich zur Gradientenstärke ist es natürlich auch möglich die Richtung des Gradienten zu erhalten:

$$d = \arctan\left(\frac{f_y}{f_x}\right). \quad (8.22)$$

Diese wird bei vielen Anwendungen quantisiert und in eine endliche Anzahl von möglichen Richtungen eingeteilt. Zu beachten ist hier, dass bei einer geringen Gradientenstärke, die Berechnung der Richtung nicht stabil durchgeführt werden kann.

- **Richtungsableitungen**

Kann man die Berechnung der Gradientenstärke auf Farbbilder bzw. mehrkanalige Bilder verallgemeinern? Für skalare Funktionen $f(x, y)$ (Grauwertfunktionen) gibt es die Richtungsableitung:

$$\frac{\partial f}{\partial \mathbf{l}} = (\nabla f)^T \cdot \begin{pmatrix} \cos \alpha \\ \cos \beta \end{pmatrix}. \quad (8.23)$$

Dabei ist \mathbf{l} die Richtung, die durch den Vektor der Richtungscosinus bestimmt ist. Eine Kante verläuft senkrecht zu der Richtung, für welche die Richtungsableitung maximal ist. Auf Grund dessen, dass die Richtungsableitung das Skalarprodukt von Gradient und Richtung ist, wird das Maximum für die Richtung angenommen, in die der Gradient zeigt. Daher können wir sofort den Betrag des Gradienten als Stärke eines Kantenpunktes interpretieren. Es sei nun $\mathbf{g}(x, y)$ eine Vektorfunktion. Wir benutzen hier eine Vektorfunktion für Farbbilder, daher hat der Vektor genau drei Komponenten. Die Richtungsableitung lautet dann:

$$\frac{\partial \mathbf{g}}{\partial \mathbf{l}} = \frac{\partial \mathbf{g}}{\partial x} \cos \alpha + \frac{\partial \mathbf{g}}{\partial y} \cos \beta = \begin{pmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_1}{\partial y} \\ \frac{\partial g_2}{\partial x} & \frac{\partial g_2}{\partial y} \\ \frac{\partial g_3}{\partial x} & \frac{\partial g_3}{\partial y} \end{pmatrix} \begin{pmatrix} \cos \alpha \\ \cos \beta \end{pmatrix} = \mathbf{J} \cdot \mathbf{l}. \quad (8.24)$$

Die Matrix \mathbf{J} heißt *Jakobimatrix*. Die Richtungsableitung ist jetzt selbst ein Vektor, dessen Betrag als Stärke der Änderung von \mathbf{g} in der Richtung \mathbf{l} zu werten ist. Wir suchen die Richtung \mathbf{l} , in der der Betrag von $\mathbf{J}\mathbf{l}$ maximal ist (wir erhalten dasselbe Ergebnis, wenn wir den Betrag quadrieren). Dies führt auf die Optimierungsaufgabe:

$$S(\mathbf{l}) = (\mathbf{J}\mathbf{l})^T \mathbf{J}\mathbf{l} = \mathbf{l}^T \mathbf{J}^T \mathbf{J}\mathbf{l} = \mathbf{l}^T \mathbf{M}\mathbf{l} \rightarrow \text{Maximum mit } \mathbf{l}^T \mathbf{l} = 1. \quad (8.25)$$

Wir bilden die Lagrange-Funktion $L = \mathbf{l}^T \mathbf{M}\mathbf{l} - \lambda(\mathbf{l}^T \mathbf{l} - 1)$ und leiten diese ab $\nabla L = 2\mathbf{M}\mathbf{l} - 2\lambda\mathbf{l} = 0$. Damit erhalten wir das Eigenwertproblem:

$$\mathbf{M}\mathbf{l} = \lambda\mathbf{l}, \quad (8.26)$$

wobei für den größten Eigenwert λ_{\max} und den zugehörigen, normierten Eigenvektor \mathbf{l}_{\max} gilt

$$S(\mathbf{l}_{\max}) = \mathbf{l}_{\max}^T \mathbf{M} \mathbf{l}_{\max} = \lambda_{\max} \mathbf{l}_{\max}^T \mathbf{l}_{\max} = \lambda_{\max}. \quad (8.27)$$

Damit ist λ_{\max} der Wert der größten Richtungsableitung und \mathbf{l}_{\max} der Vektor, der in die Richtung der größten Richtungsableitung zeigt. Für jedes Pixel ist demzufolge die Matrix \mathbf{M} mittels der LSI-Ableitungsfilter zu berechnen und anschließend das Eigenwertproblem (8.26) zu lösen.

- **Mittelung von Orientierungen**

Orientierungen bzw. Richtungen spielen eine große Rolle. Eine lokale Orientierung kann z. B. durch den Gradienten beschrieben werden. Oft sind Orientierungen auch nur Winkel. Wenn zum Winkel noch eine Gewicht dazukommt, dann liegt ein Vektor vor. Beim Winkel müssen wir unterscheiden: Sollen Winkel bis 360 Grad oder nur bis 180 Grad betrachtet werden. Da Winkel zyklisch sind, ist eine Mittelung von Winkeln problematisch. Daher muss zunächst genau definiert werden, was eine Orientierung für eine konkrete Aufgabe bedeutet. Dazu einige Beispiele:

- Gegeben seien Winkel bis 360 Grad oder bis 180 Grad und diese Winkel sollen gemittelt werden. Da die zyklische Natur der Winkel Probleme bereitet, rechnen wir die Winkel in Einheitsvektoren um, mitteln die Einheitsvektoren und berechnen den Winkel des gemittelten Einheitsvektors.
- Oft sind als Orientierungen Winkel und Gewichte gegeben. Nun rechnen wir die Vektoren aus, die die Richtung des Winkels und die Länge des Gewichtes haben. Wir mitteln nun diese Vektoren und sind fertig.
- Wenn nun Winkel und Gewichte gegeben sind, aber die Winkel nur bis 180 Grad zu betrachten sind, können wir nicht einfach die Vektoren addieren. Ein typisches Beispiel sind Gradienten, deren Mittelung im Fenster kann sogar den Nullvektor ergeben. Wir müssen nun durch einen Trick die Periode auf 180 Grad halbieren. Dies geschieht simpel, indem wir den Winkel eines jeden Vektors verdoppeln, diese Vektoren mitteln und zum Schluss den Winkel des gemittelten Vektors wieder halbieren. Dabei kann man sogar noch die Gewichte, folglich die Längen der Vektoren

ändern. Die Verdoppelung des Winkels kann man numerisch recht einfach durchführen: Man fasst den Vektor als komplexe Zahl auf und quadriert sie. Dann mittelt man die komplexen Zahlen und „zieht“ die Wurzel daraus. Durch das Quadrieren haben wir nicht nur den Winkel verdoppelt, sondern auch das Gewicht. Wenn man dies vermeiden möchte, zieht man nach jeder Quadrierung aus dem Betrag der komplexen Zahl die Wurzel.

- Man kann auch mitteln, indem man ein Optimierungsproblem löst. Dies wird z. B. durch den Strukturtensor realisiert, siehe Abschn. 17.5.2. Dabei wird der Winkel auch nur bis 180 Grad betrachtet und ein Max- sowie ein Min-Problem gelöst. Dadurch erhalten wir zwei Hauptrichtungen mit zwei Bewertungen und haben somit mehr Information als bei einer „üblichen“ Mittelung.

- **Krümmungsfilter**

- Es gibt auch nichtlineare Filter, die Ableitungen höherer als erster Ordnung verwenden, z. B. Krümmungsfilter. Es sind nicht Krümmungsberechnungen von konkreten Linien oder Konturen gemeint, sondern es wird wie üblich ein vollständiges Grauwertbild „gefiltert“. Dazu schneiden wir die Grauwertfunktion $f(x, y)$ mit einer Konstanten c . Wir betrachten folglich die durch $f(x, y) = c$ entstandenen *Isolnien* und untersuchen deren Krümmungen. Die Krümmung einer Kurve in einem Kurvenpunkt ist die Ableitung des Tangentenwinkels nach der Bogenlänge, folglich $K = \frac{d\alpha}{ds}$. Damit ist mit $ds = \sqrt{dx^2 + dy^2}$:

$$\frac{d\alpha}{ds} = \frac{d\alpha}{dy'} \frac{dy'}{ds} = \frac{1}{1+y'^2} \frac{y'' dx}{\sqrt{dx^2 + dy^2}} = \frac{y''}{(1+y'^2)^{\frac{3}{2}}}. \quad (8.28)$$

Da wir aber $y = y(x)$ (und damit y', y'') nicht explizit gegeben haben, müssen wir diese aus der impliziten Gleichung $f(x, y) = c$ ermitteln. Daher ist:

$$f_x + f_y y' = 0 \rightarrow y' = -\frac{f_x}{f_y} \quad (8.29)$$

$$f_{xx} + f_{xy} y' + f_{yx} y' + f_{yy} y' y' + f_y y'' = 0 \quad (8.30)$$

$$\rightarrow y'' = \frac{-f_{xx} + 2f_{xy} \frac{f_x}{f_y} - f_{yy} \frac{f_x^2}{f_y^2}}{f_y}. \quad (8.31)$$

Diese Ableitungen setzen wir in (8.28) ein und erhalten:

$$K = \frac{-f_{xx} f_y^2 + 2f_{xy} f_x f_y - f_{yy} f_x^2}{(f_x^2 + f_y^2)^{\frac{3}{2}}}. \quad (8.32)$$

Alle diese eingehenden Ableitungen ersetzen wir wieder durch die LSI-Filter. Formal ist somit $f_x = a_{1,0}$, $f_y = a_{0,1}$, $f_{xx} = 2a_{2,0}$, $f_{xy} = a_{1,1}$, $f_{yy} = 2a_{0,2}$ zu setzen. Weiterhin

interessiert uns eigentlich nur die Krümmung von typischen Kanten im Bild. Daher multiplizieren wir K noch mit dem Betrag des Gradienten $\sqrt{f_x^2 + f_y^2}$ und erhalten:

$$K_{\text{curvature}} = 2 \frac{-a_{2,0}a_{0,1}^2 + a_{1,1}a_{1,0}a_{0,1} - a_{0,2}a_{1,0}^2}{a_{1,0}^2 + a_{0,1}^2}. \quad (8.33)$$

Für typische 2D-Extrempunkte, wie lokale Minima, lokale Maxima und lokale Sattelpunkte, existiert die Krümmung nicht.

- Wenn man besonders extreme Krümmungspunkte detektieren möchte, dann kann man diese auch Eckpunkte nennen. Diese kann man auch mit Ableitungen erster Ordnung finden. Dazu benutzen wir wieder einmal den Strukturtensor, siehe Abschn. 17.5.2. Die beiden Eigenwerte sind die Bewertungen der beiden gefundenen Hauptrichtungen. Wenn wir zwei wesentliche Hauptrichtungen gefunden haben, dann deutet das auf eine Ecke hin. Daher interpretieren wir Punkte als Eckpunkte, wenn der kleinere Eigenwert „genügend groß“ ist. Dieser Algorithmus wird auch Shi-Tomasi-Detektor oder Kanade-Tomasi-Detektor genannt. Ecken werden häufig als *Interest Points* betrachtet und als Referenzpunkte in der Stereobildanalyse benutzt, siehe Abschn. 16.4.

8.2.2 Richtungsfilter

Als Richtungsfilter bezeichnen wir Filter, die pro Pixel in Abhängigkeit von einer „lokalen Strukturrichtung“ ausgewählt werden. Diese Richtung kann die Richtung einer Linie oder Kante sein. Zur Berechnung der Linien- oder Kantenrichtung benutzen wir den Strukturtensor, siehe Abschn. 17.5.2 und (8.18). Die Umgebung zur Berechnung des Strukturtensors darf nicht zu klein, aber auch nicht zu groß gewählt werden. Ist sie zu klein (man benutzt nur den Gradienten im Aufpunkt), dann streut die Richtung von Pixel zu Pixel sehr stark. Ist sie zu groß, spiegelt sie nicht die typische lokale Richtungsstruktur wieder. Als Linien- oder Kantenrichtung benutzen wir den Eigenvektor von (8.18) zum kleinsten Eigenwert. Praktisch berechnen wir aber den Eigenvektor zum größten Eigenwert und verwenden den Vektor, der auf diesen senkrecht steht. Wenn man nun die Richtung bestimmt hat, wählt man je nach Aufgabenstellung einen Filter aus. In Abb. 8.2 ist als zweites Bild die Filtermaske einer Richtungsglättung zu sehen. Das innere Rechteck, dessen Orientierung vorher mit dem Strukturtensor berechnet wurde, stellt ein Mittelwertfilter dar. Natürlich könnte man diesen Mittelwert-Richtungsfilter durch einen Gaußfilter ersetzen. Es ist offensichtlich, dass diese gerichteten Glättungsfilter in engem Zusammenhang zu den Diffusionsfiltern stehen, siehe Weickert [88]. In Abb. 8.2 ist als erstes Bild die Maske eines Richtungs-DoB-Filters zu sehen, der zur Liniendetektion benutzt wird. Dieser stellt die Verallgemeinerung der DoB-Filter bezüglich einer Richtung dar, siehe Abschn. 7.5 und (8.19). Weiterhin ist in Abb. 8.2 als drittes Bild die Maske eines Richtungs-Kantenfilters zu sehen, wobei die Ähnlichkeit zum bekannten Hückel-Kantenfilter nicht zu übersehen ist.

Abb. 8.1 Eingabebild für die Vibrissenerkennung als Beispielanwendung für Richtungsfilter



In Abb. 8.1 ist eine Ratte mit Vibrissen als Linienstrukturen zu sehen, wobei diese Linien sehr „zerklüftet“ sind.

In den Bildern der Abb. 8.3 sind die Wirkungen dieser nichtlinearen Richtungsfilter zu sehen. In der Abb. 5.3 ist eine Pyramidenbild zu sehen, welches in der Abb. 8.4 einer Richtungsglättung zur „künstlerischen Verbesserung“ unterzogen wurde.

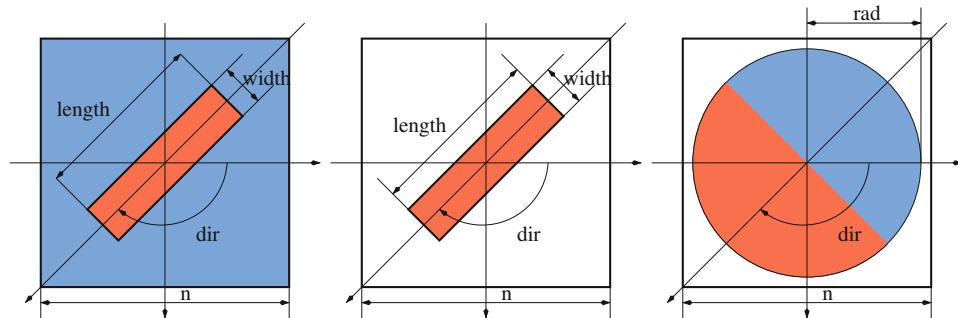


Abb. 8.2 Darstellung der Filter: Richtungs-DoB, Richtungs-Smear, Richtungs-Edge



Abb. 8.3 Anwendung der Richtungsfilter auf das Eingabebild in Abb. 8.1: Richtungs-DoB, Richtungs-Smear, Richtungs-Edge



Abb. 8.4 Richtungs-Smear angewendet auf das Bild in Abb. 5.3

8.2.3 Relaxationsfilter

Ganz allgemein kann man jedes Filter als vektorielle Funktion \mathbf{F} des Bildes f ansehen, so dass wieder ein neues Bild g entsteht, folglich: $g = \mathbf{F}(f)$. Bei gegebenem \mathbf{F} kann man auch einen Fixpunkt (Fixbild) h mit $h = \mathbf{F}(h)$ berechnen. Dies geschieht durch eine Folge von Iterationen, wobei das Bild f als Startpunkt (Startbild) dient. Natürlich muss man die Existenz von h voraussetzen und die Konvergenz der Iteration annehmen. Eine einfache vektorielle Funktion \mathbf{F} ist die Folgende. Dazu wählen wir für jedes Pixel eine Umgebung U , wobei f_{\min} der minimale Grauwert und f_{\max} der maximale Grauwert aus der Umgebung ist (zur Umgebung eines Pixels gehört immer auch das Pixel selbst). Dann ist \mathbf{F} :

$$g_{i,j} = \begin{cases} f_{\min}, & \text{falls } f_{i,j} - f_{\min} < f_{\max} - f_{i,j} \\ f_{i,j}, & \text{falls } f_{i,j} - f_{\min} = f_{\max} - f_{i,j} \\ f_{\max}, & \text{falls } f_{i,j} - f_{\min} > f_{\max} - f_{i,j}. \end{cases} \quad (8.34)$$

Nach einer bestimmten Anzahl von Relaxationsschritten bricht man die Iteration ab. Dieses *Relaxationsfilter* dient im Wesentlichen zur Kontrastverbesserung von Bildern.

8.2.4 Morphologische Filter

Erosion und Dilation Die mathematische Morphologie ist eng mit der Theorie der Mengen verbunden. Oft werden in diesem Zusammenhang Hadwiger (1957), Matheron (1975) und Serra (1982) genannt. Aus der Morphologie von Punktmengen sind zwei wesentliche Operationen bekannt: die *Erosion* und *Dilation*. Dazu betrachten wir zwei Punktmengen

A und B in der Ebene. Die *Minkowski-Addition* \oplus (*Dilation*) ist folgendermaßen definiert:

$$A \oplus B = \{c \mid c = a + b, a \in A, b \in B\} \leftrightarrow A \oplus B = \{p \mid B_p \cap A \neq \emptyset\} \quad (8.35)$$

$$\leftrightarrow A \oplus B = \bigcup_{p \in A} B_p. \quad (8.36)$$

Dabei wird B Strukturelement genannt. Weiterhin bedeutet die Schreibweise M_p , dass die Menge M um p translatiert wird. Wenn das Strukturelement B ein Kreis mit dem Mittelpunkt im Ursprung ist, dann bedeutet dies, dass dieser Kreis „außen“ auf A abgerollt wird. Die Kurve des Mittelpunktes ist nun der neue Rand von A . Die Menge A wird so entsprechend dieser Mittelpunkts-Kurve vergrößert. Das Gegenstück zur Addition ist die *Minkowski-Subtraktion* \ominus (*Erosion*):

$$A \ominus B = \{c \mid c + b \in A, b \in B\} \leftrightarrow A \ominus B = \{p \mid B_p \subseteq A\} \quad (8.37)$$

$$\leftrightarrow A \ominus B = \bigcap_{p \in B} A_p. \quad (8.38)$$

Man sagt: A wird durch B erodiert. Wählen wir wieder als Strukturelement einen Kreis mit dem Mittelpunkt im Ursprung, dann „rollen“ wir den Kreis von innen her auf dem Rand ab. Die Mittelpunktskurve ist der neue Rand von A , wodurch sich A verkleinert.

Wenn wir diese beiden Operationen auf digitale Bilder übertragen wollen, müssen wir uns zunächst auf Binärbilder beschränken, da die beiden Operationen nur für Punktmengen definiert sind. Alle schwarzen Punkte eines Binärbildes bilden dann die Menge A . Das Strukturelement B kann beliebig gewählt werden, wir beschränken uns hier auf die Vierer- oder Achternachbarschaft:

Zum Strukturelement B gehört das Pixel im Koordinatenursprung und seine Nachbarn entsprechend der Vierer- oder Achternachbarschaft, folglich ist B ein kleines Kreuz oder ein kleines Quadrat.

Nun können wir die Dilation und Erosion verbal beschreiben:

- **Dilation**

Alle schwarzen Pixel bleiben schwarz. Von den weißen Pixeln werden nur diejenigen schwarz, in deren Nachbarschaft mindestens ein schwarzes Pixel auftritt.

- **Erosion**

Alle weißen Pixel bleiben weiß. Von den schwarzen Pixeln werden nur diejenigen weiß, in deren Nachbarschaft mindestens ein weißes Pixel auftritt.

Wir erkennen eine Art Dualität von Erosion und Dilation. Wir bezeichnen das gesamte Binärbild mit BIN und führen die Operatoren **I** als Identitätsoperator, **C** als Komplementoperator (schwarz \leftrightarrow weiß), **D** als Dilationsoperator und **E** als Erosionsoperator ein. Eine wichtige Aussage ist nun:

Die beiden morphologischen Operationen Dilation (**D**) und Erosion (**E**) sind im Allgemeinen nicht invers zueinander, d. h. es gilt im Allgemeinen nicht:

$$\mathbf{ED}(BIN) = \mathbf{DE}(BIN) = BIN. \quad (8.39)$$

Es gilt aber eine Art Dualität bezüglich der Komplementbildung:

$$\mathbf{E} = \mathbf{CDC}, \quad \mathbf{CE} = \mathbf{DC}, \quad \mathbf{EC} = \mathbf{CD}, \quad \mathbf{CEC} = \mathbf{D}. \quad (8.40)$$

Öffnung und Schließung Nun führen wir zwei wichtige Operationen ein:

- Der Operator **S** = \mathbf{ED} heißt Schließungsoperator (*Fermeture, Closing*).
- Der Operator **O** = \mathbf{DE} heißt Öffnungsoperator (*Ouverture, Opening*).

Mit dem Schließungsoperator können kleine Bildlücken, Risse usw. geschlossen werden. Mit dem Öffnungsoperator können kleine Zipfel von Objekten, kleine Bildreste usw. beseitigt werden. Beide Operatoren sind Projektionsoperatoren, d. h. es gilt $\mathbf{OO} = \mathbf{O}$ und $\mathbf{SS} = \mathbf{S}$. Damit ist es sinnlos, ein Opening oder Closing mehrmals auf ein Bild anzuwenden. Man kann die beiden Operatoren auch sinnvoll auf Konturen anwenden, damit „Konturrisse“ schließen oder „Konturzipfel“ entfernen. Man muss dazu erst von der Kontur ein Objekt in einem Binärbild erzeugen. Auf das Binärbild werden die morphologischen Operatoren angewendet und anschließend wird mit dem Konturfolgeverfahren (siehe Abschn. 10.2) die neue Kontur detektiert. Eine ähnliche Operation kann man direkt auf den Konturpunkten ausführen, die Berechnung der limitierten konvexe Hülle (*LKH*), siehe Abschn. 1.2.3 und 10.4.

Thinning, Thickening, Skeleton Eine weitere, elementare Operation der Morphologie ist die *Hit- or Miss-*-Transformation. Mit ihr können solche Pixel detektiert werden, die gewisse geometrische Eigenschaften besitzen, z. B. Eckpunkte, Randpunkte oder ähnliche. Daher können mit dieser Operation mächtigere, morphologische Operationen wie Verdickungsoperationen (*Thickening*), Verdünnungsoperationen (*Thinning*) und Matchingoperationen eingeführt werden. Zunächst definieren wir eine Maske (z. B. der Größe 3×3) und translatieren diese über das Bild. Haben wir eine Pixelanordnung gefunden, die der Maske entspricht (Hit), so markieren wir das Pixel „schwarz“. War das Matching nicht erfolgreich (Miss), so markieren wir das Pixel „weiß“. Dieses Matching kann nun effizient auf morphologische Operationen zurückgeführt werden. Dazu muss die Matchingmaske in zwei disjunkte, zueinander komplementäre Strukturelemente *J* und *K* zerlegt werden. Dann wird die *Hit-or-Miss*-Transformation \otimes erklärt:

$$A \otimes (J, K) := (A \ominus J) \cap (\mathbf{CA} \ominus K). \quad (8.41)$$

Wir erodieren dabei A mit J , das Komplement von A mit K und bilden von beiden den Durchschnitt. Je nach Wahl der Strukturelemente kann man verschiedene Strukturen detektieren, z. B. Ecken oder isolierte Punkte. Für isolierte Punkte bezüglich der Vierernachbarschaft wählen wir als Strukturelemente $J = \{(0, 0)\}$ und $K = \{(0, 1), (0, -1), (1, 0), (-1, 0)\}$. Die Verdünnungsoperation steht in engem Zusammenhang mit der Skellettierung, d. h. der Berechnung des Skelettes (*skeleton*) eines Binärbildes. Das Skelett wird z. B. benötigt, wenn man in medizinischen Bildern Gefäße segmentiert hat und diese vermessen möchte. Das Skelett bildet dann die Mittelpunktlinien der Gefäße.

Grauwertbilder Nun wollen wir die beiden wichtigen Operatoren Dilation und Erosion auf Grauwertbilder übertragen. Der größte Grauwert sei G_{\max} . Wir können ein Grauwertbild f als Summe von Binärbildern B_i der „Dicke“ Eins interpretieren:

$$f = B_1 + B_2 + \dots + B_{G_{\max}}. \quad (8.42)$$

Damit werden die Operationen Erosion und Dilation auf f definiert:

$$\mathbf{Ef} \stackrel{\text{def}}{=} \mathbf{EB}_1 + \mathbf{EB}_2 + \dots + \mathbf{EB}_{G_{\max}}, \quad \mathbf{Df} \stackrel{\text{def}}{=} \mathbf{DB}_1 + \mathbf{DB}_2 + \dots + \mathbf{DB}_{G_{\max}}. \quad (8.43)$$

Nun kann man sich leicht überlegen, dass mit einer Umgebung $U(i, j)$ folgendes gilt:

$$(\mathbf{Ef})_{i,j} = \min_{(x,y) \in U(i,j)} f(x, y), \quad (\mathbf{Df})_{i,j} = \max_{(x,y) \in U(i,j)} f(x, y). \quad (8.44)$$

Damit haben wir die Erosion auf das Minimumfilter und die Dilation auf das Maximumfilter zurückgeführt. Das Minimum und Maximumfilter sind spezielle Rangordnungsfilter und lassen sich effektiv implementieren, siehe Abschn. 8.2.5.

Zylinderhut-Transformation Eine weitere morphologische Operation ist die Zylinderhut-Transformation, auch *top hat transform* genannt. Dabei wird vom Bild das geöffnete oder das geschlossene Bild subtrahiert:

$$WTHT(f) = f - \mathbf{Of}, \quad BTHT(f) = f - \mathbf{Sf}. \quad (8.45)$$

Das Ergebnis ist gerade der „Verlust“, der bei Öffnung oder Schließung auftritt. In praktischen Anwendungen unterscheidet man zwischen der WTHT (*write top hat transform*) und der BTHT (*black top hat transform*), je nachdem die Objekte hell oder dunkel sind. Die Größe des Strukturelementes ist entscheidend und sollte a priori ermittelbar sein. Alle Objekte, die in das Strukturelement „passen“, werden hervorgehoben. Die Verallgemeinerung auf Grauwertbilder ist noch interessanter, da dann auch Beleuchtungsschwankungen des Hintergrundes korrigiert werden können, siehe Abschn. 8.2.5.

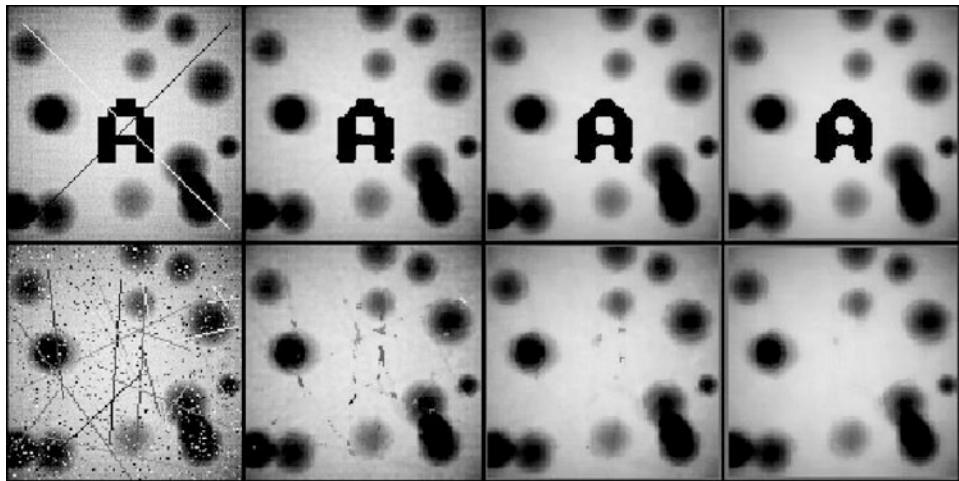


Abb. 8.5 Die erste Spalte zeigt zwei Eingabebilder, welche durch Anwendung von Medianfiltern der Größe 3×3 (Spalte 2), 5×5 (Spalte 3) und 7×7 (Spalte 4) gefiltert wurden

8.2.5 Rangordnungsfilter

Die Grauwerte in der Umgebung U eines Pixels denke man sich separat in einer gesonderten Liste aufsteigend sortiert. Die Position in dieser geordneten Liste nennen wir Rang bezüglich der Ordnung. Der an der ersten Stelle stehende Grauwert ist der kleinste Grauwert und besitzt den Rang eins usw. Der in der sortierten Liste entsprechend dem Rang an einer bestimmten Position stehende Grauwert ist nun der neue Filteroutput. Ist der Rang gleich eins, so erhalten wir ein *Minimumfilter*. Ist der Rang der größte aus U , so haben wir ein *Maximumfilter*. Das Minimum- und das Maximumfilter haben den Vorteil, dass sie sich separieren lassen. Damit können sie sehr effizient implementiert werden. Das Minimum- und Maximumfilter haben bei morphologischen Operationen große Bedeutung, siehe Abschn. 8.2.4. Weiterhin gibt es eine Reihe weiterer wichtiger Anwendungen. Sollen z. B. bezüglich einer Umgebung U die lokalen Minima/Maxima eines Bildes bestimmt werden, so wendet man auf das Bild ein Minimum/Maximum-Filter an. Man braucht nun nur noch pro Pixel im Bild nachzuschauen, ob der Grauwert gleich dem Wert in dem gefilterten Bild ist. Dies kann somit sehr effektiv implementiert werden, siehe als Anwendung Abschn. 10.6. Ist der Rang genau die mittlere Ordnungsposition, so sprechen wir vom *Medianfilter*, siehe Abschn. 21.2.5. Das Medianfilter kann zur Beseitigung kleiner, singulärer Störungen benutzt werden, z. B. bei „Pfeffer und Salz“-Rauschen. Kanten bleiben dabei erhalten, allerdings „frisst“ der Medianfilter Ecken an Objekten ab und beseitigt auch dünne Linien, siehe Abb. 8.5.

Gibt es nun außer dem Minimum-, Maximum- und Medianfilter auch Anwendungen für die anderen Rangordnungsfilter? Dies ist in der Tat so. Dazu betrachten wir einmal

Bilder, die als Objekte *blobs*, *spots* enthalten, die sich auch berühren dürfen. Wir nehmen einmal an, wir kennen die maximale *blob*-Größe, d. h. der größte *blob* ist ungefähr in einem Fenster der Größe $m \times m$ enthalten. Die Größe m könnte man als Strukturgröße bezeichnen. Das Bild könnte nun von Beleuchtungsschwankungen gestört sein, wobei wir annehmen, dass diese nicht zu abrupt sind. Ein typisches Beispiel wäre z. B. *shading* in Mikroskopaufnahmen. Nun wollen wir das Bild so normieren, dass die Objekte sich auf einen einheitlichen, homogenen Untergrund beziehen. Bei den Mirkoskopaufnahmen würde das bedeuten, dass wir das *shading* beseitigen. Dazu bearbeiten wir das Bild in drei Schritten und nennen das Filter *top min* Filter.

- Zunächst glätten wir das Bild etwas, z. B. mit einem $k \times k$ -Mittelwertfilter. Es darf k nicht zu groß gewählt werden, z. B. $k = 3$. Wir erhalten ein Bild $B1$.
- Wir legen eine Fenstergröße $n \times n$ fest. Die Größe n muss so gewählt werden, dass sich unabhängig von der Fensterlage im Bild immer Untergrundpunkte im Fenster befinden. Eine Wahl $n \geq m$ ist dann laut Voraussetzungen geeignet, z. B. $n = 2m$. Nun filtern wir das Bild mit einem Rangordnungsfilter des Ranges r und der Fenstergröße n und erhalten ein Bild $B2$.
- Wir berechnen die Differenz $B3 = B1 - B2$. Dabei können wir die negativen Filterwerte Null setzen.

Da wir auf den Untergrund normieren wollen, wäre folglich ein Minimumfilter mit $r = 1$ die eigentliche Wahl. Da aber auch im Untergrund trotz des Mittelwertfilters noch Ausreißer vorhanden sein können, ist ein Wert $r > 1$ besser, wobei r entsprechend der Fenstergröße n zu wählen ist. Auch wenn keine Beleuchtungsschwankungen vorhanden sind, hat dieser Filter einen Sinn. Wir normieren dann die Grauwerte der Objekte (*blobs*) absolut auf den Hintergrund und Hintergrundwerte bleiben auch Hintergrundwerte und erhalten sehr kleine Grauwerte. Ein Nachteil des *top min* Filters ist, dass bei großen Objekten, die als Untergrund zu werten sind, störende Randstreifen entstehen. Wenn die Ausreißer nicht genügend beseitigt wurden, entstehen um diese Ausreißer kleine sichtbare Artefakte. Dies sollte man korrigieren, indem man die „Versätze“ im Hintergrund korrigiert. Dies geschieht durch ein Maximumfilter. Folglich ersetzen wir das *top min* Filter durch das Filter *WTHT*, angewendet auf ein gegebenes Bild f bezüglich einer Fenstergröße n :

$$WTHT(f) := f - \max_n(\min_n(f)). \quad (8.46)$$

Da das Minimum-, Maximumfilter die Verallgemeinerung des Erosions-, Dilationsfilter (siehe Abschn. 8.2.4) auf Grauwertbilder darstellt, ist das Filter (8.46) die Verallgemeinerung des morphologischen *top hat* Filters (siehe (8.45)) auf Grauwertbilder. Folglich werden bei der BTHT die Filter *min* und *max* vertauscht. In der Abb. 10.13 ist die Gefäßstruktur eines Herzens zu sehen. Dabei erzeugen andere innere Organe Schatten und Kanten, die störend wirken. Auf dieses Bild wurde das *top min* Filter angewendet, siehe dazu das erste Bild in Abb. 8.6. Das zweite Bild in Abb. 8.6 zeigt das Ergebnis des *top-hat*-Filters. Im Unterschied zum *top hat* Filter erzeugt das *top min* Filter störende Streifen. Das

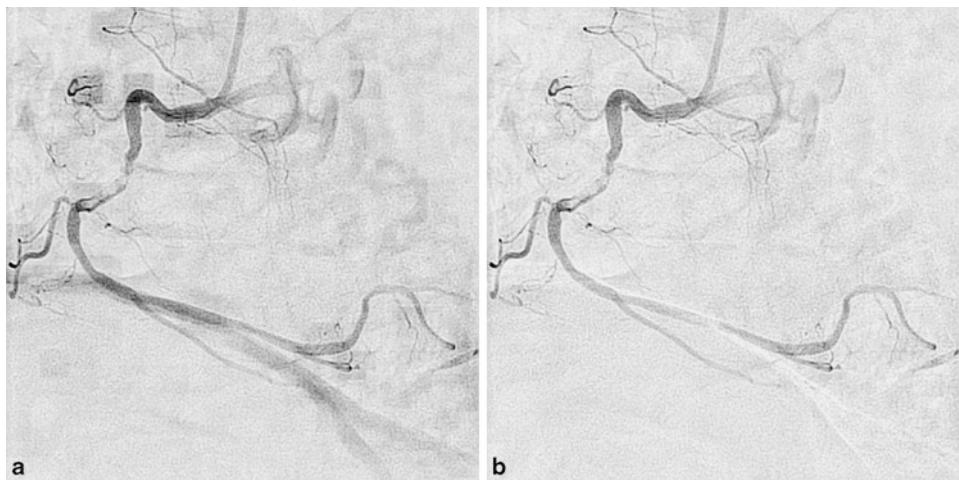


Abb. 8.6 a Anwendung eines *top-min*-Filter; b Anwendung eines *top-hat*-Filter. Als Eingabebild wurde Abb. 10.13 aus Kap. 10 verwendet

top-hat-Filter ist folglich hervorragend geeignet, Beleuchtungsschwankungen zu korrigieren. Kritisch ist die Wahl der Fenstergröße n bzw. die Wahl des Strukturelementes. Alle Objekte, die größer als das Strukturelement sind, werden als Hintergrund aufgefasst und eliminiert. Wenn z. B. dunkle Objekte, die nicht größer als das Strukturelement sind, als die eigentlichen zu betrachtenden Objekte aufgefasst werden, dann wird „weißer“ Hintergrund weiterhin als Hintergrund betrachtet und gleichzeitig werden größere dunkle Stellen eliminiert und auch als Hintergrund dargestellt. Objekte, die „kleiner“ als das Strukturelement sind, bleiben erhalten, z. B. *blobs*, *spots*. Es bleiben aber auch Objekte erhalten, bei denen unabhängig von der Lage des Strukturelementes über dem Objekt, stets Untergrundpixel im Strukturelement vorhanden sind, z. B. Linien, Gefäße usw., die nicht so „breit“ wie das Strukturelement sind. Ob man die WTHT oder die BTHT auf Grauwertbilder anwendet, hängt von der Interpretation der Objekte ab. Sind z. B. die gesuchten Objekte dunkel und „dunkel“ bedeutet hohe Grauwerte (bzw. die gesuchten Objekte sind hell und „hell“ bedeutet hohe Grauwerte), dann ist die WTHT anzuwenden, ansonsten die BTHT.

Da man Grauwerte $x_{i,j}$ als Realisierungen von Zufallsvariablen $X_{i,j}$ auffassen kann, kann man sich auch der Methoden der Stochastik bedienen. In diesem Sinne wird ein zweidimensionales Bild \mathbf{X} als stochastisches Feld (*random field*) bezeichnet. In der Regel brauchen wir dann noch ein Wahrscheinlichkeitsmaß, entweder $P(\mathbf{X} = \mathbf{x})$ für diskrete Zufallsvariable oder eine Dichtefunktion $f(\mathbf{x})$ für stetige Zufallsvariable. Das Hauptproblem liegt in Folgendem: wenn nur ein konkretes Bild gegeben ist, dann haben wir für das stochastische Feld nur eine Realisierung und mit einer Realisierung kann man in der Statistik ohne weitere Annahmen kaum etwas anfangen. Folglich reduziert man die Anzahl der Zufallsvariablen und gewinnt bei einem Bild immer mehr Realisierungen. Man nennt dies dann Statistik n -ter Ordnung, wobei n die Anzahl der Zufallsvariablen darstellt. Im Extremfall sind alle Grauwerte eines Bildes Realisierungen einer einzigen ($n = 1$) Zufallsvariablen X . Das Grauerthistogramm des Bildes beschreibt dann die Verteilung dieser einzigen Zufallsvariablen und wird Statistik erster Ordnung genannt.

9.1 Grundbegriffe der Informationstheorie

In der Bildverarbeitung werden häufig Grundbegriffe der Informationstheorie verwendet, die sich wiederum auf Verteilungen von Zufallsvariablen beziehen. Wir betrachten einmal binäre Zustände. Mit n Binärvariablen können wir $m = 2^n$ Zustände erfassen. Mit 8 Bit können wir z. B. 256 Zustände darstellen. Umgekehrt können wir m Zuständen $n = H = \log_2 m$ [bit] zuordnen. Wir bezeichnen dies auch als Unsicherheit pro Zustand, dabei sind alle Zustände gleichberechtigt. H nennt man auch die *Entropie* eines Zustandes. Wir schreiben

$$H = \log_2 m = -\log_2 \frac{1}{m}. \quad (9.1)$$

Da alle Zustände gleichberechtigt sind, haben sie alle die gleiche Wahrscheinlichkeit, nämlich $P(z_i) = \frac{1}{m} = p_i, i = 1, \dots, m$. Wenn die Zustände aber alle nicht die gleiche Wahrscheinlichkeit haben, dann können wir gleich verallgemeinern:

$$H_i = -\log_2 p_i, \quad \sum_i p_i = 1 \quad (9.2)$$

als Entropie des i -ten Zustandes. Da uns aber nicht die Entropie der einzelnen Zustände interessiert, sondern eine „mittlere“ Entropie aller Zustände, bilden wir ein gewichtetes Mittel aller Einzelentropien:

$$H_0 = -\sum_{i=1}^m p_i \log_2 p_i. \quad (9.3)$$

Diese mittlere Entropie aller Zustände wird schlechthin nach C.E. Shannon (1916–2001) als Entropie bezeichnet. Da wir m Zustände mit deren Wahrscheinlichkeiten gegeben haben, also eine endliche diskrete Zufallsvariable (ZV), spricht man auch von der Entropie einer diskreten ZV. Man kann nun leicht zeigen:

Die Entropie wird genau dann maximal, wenn die ZV gleichverteilt ist, d. h. wenn $p_i = \frac{1}{m}$ gilt. Dann ist:

$$H_0 = -\sum_{i=1}^m \frac{1}{m} \log_2 \frac{1}{m} = -\log_2 \frac{1}{m} = \log_2 m. \quad (9.4)$$

Damit sind wir wieder bei der Ausgangssituation angekommen.

Die Entropie wird genau dann minimal, wenn die ZV eine Null-Eins-Verteilung besitzt, d. h. wenn für ein j $p_j = 1, p_i = 0 \forall i \neq j$ gilt ($p_i = \delta_{i-j}$). In diesem Falle besteht die Entropie nur aus einem Term, die anderen Terme bestehen aus $0 \cdot \log_2 0$ und sind eigentlich undefiniert. Da wir aber diese Terme als Grenzwerte für $p_i \rightarrow 0$ auffassen können und dieser Grenzwert Null ist, definieren wir als Funktionswert für diesen Term generell:

$$0 \cdot \log_2 0 := 0. \quad (9.5)$$

Damit reduziert sich die Entropie bei einer Null-Eins-Verteilung der ZV zu

$$H_0 = -p_j \log_2 p_j = -1 \cdot \log_2 1 = 0. \quad (9.6)$$

Damit gilt generell die Ungleichung:

$$0 \leq H_0 \leq \log_2 m. \quad (9.7)$$

Beispiel 1 Wir haben ein Grauwertbild und pro Pixel 8 Bit zur Speicherung der Grauwerte zur Verfügung, also 256 Grauwerte, damit besteht die ZV aus den Zuständen $0, 1, \dots, 255$.

Dazu benötigen wir aber noch die Wahrscheinlichkeiten der Zustände, also der Grauwerte. Als Näherung entnehmen wir diese einem Grauerthistogramm, das wir von dem Bild berechnen müssen. Dann können wir die Entropie des Bildes berechnen. Wir nehmen einmal an, es sei $H_0 = 4$ bit. Die minimale Entropie ist dann 0 bit und die maximale 8 bit. Dann bedeuten informationstheoretisch die berechneten 4 bit: Wenn wir ein ideales Kompressionsverfahren hätten, könnten wir das Bild verlustfrei so kodieren, dass wir im Mittel nur 4 Bit pro Pixel brauchen würden. Mehr Information enthält eben das Bild bezüglich der Statistik der einzelnen Grauwerte nicht (Statistik 1. Ordnung).

Hätte das Bild die Entropie Null, dann läge eine Null-Eins-Verteilung vor, d. h. es kommt nur genau ein Grauwert im Bild vor. Welcher das ist wissen wir nicht, aber das Bild ist „eintönig“. Hätte das Bild die Entropie 8, dann läge eine Gleichverteilung aller Grauwerte vor. Das Bild wäre dann sehr kontrastreich und ließe sich nicht mit der Statistik 1. Ordnung komprimieren (z. B. ein sogenannter „Grauwertkeil“).

Wir sehen, die Entropie ist mehr ein Begriff der „Unsicherheit“ des anzutreffenden Zustandes des Gesamtsystems, z. B. des Bildes. Bei maximaler Entropie ist die Unsicherheit am größten, bei minimaler Entropie am kleinsten. Haben wir z. B. viele Bilder mit sehr kleiner Entropie, dann können wir auch sagen, diese Bilder sind sehr kontrastarm, können keine große Vielfalt ausdrücken.

Beispiel 2 Betrachten wir einmal sehr vereinfacht Sprachen des lateinischen Alphabets. Wir unterscheiden nicht zwischen Groß- und Kleinbuchstaben. Wenn wir das Leerzeichen als Trennzeichen noch berücksichtigen und die Ziffern weglassen, so besteht das Alphabet, also unsere ZV aus 27 Zuständen (26 Buchstaben und ein Leerzeichen). Wenn eine Sprache „kontrastreich“ oder ausdrucksstark ist, sollte die Entropie sehr groß sein. Die maximale Entropie wäre $H_0 = \log_2 27 = 4,76$ bit. Jetzt würden wir für verschiedene Sprachen „alle Texte dieser Welt“ durchgehen und die Histogramme berechnen. Die ausdruckstärkste Sprache wäre die mit der größten Entropie. Die Entropien der vier Sprachen Englisch, Deutsch, Französisch und Spanisch liegen dicht bei 4 bit, unterscheiden sich also kaum.

Wir betrachten jetzt zwei diskrete ZV X mit x_i , $i = 1, \dots, n$ und Y mit y_j , $j = 1, \dots, m$. Dann können wir je einzeln die beiden Entropien $H(X)$ und $H(Y)$ berechnen. Wir betrachten nun die sogenannte bedingte Entropie:

$$H(Y|x_i) = - \sum_j P(y_j|x_i) \log P(y_j|x_i). \quad (9.8)$$

Wir bilden daraus das gewichtete Mittel:

$$\begin{aligned} H(Y|X) &= - \sum_i \sum_j P(x_i)P(y_j|x_i) \log P(y_j|x_i) \\ &= - \sum_i \sum_j P(x_i, y_j) \log P(y_j|x_i). \end{aligned} \quad (9.9)$$

$H(Y|X)$ heißt bedingte Entropie von Y bezüglich X . Es gilt:

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y). \quad (9.10)$$

Einen Teil wollen wir zeigen:

$$\begin{aligned} H(X) + H(Y|X) &= -\sum_i P(x_i) \log P(x_i) - \sum_i \sum_j P(x_i, y_j) \log P(y_j|x_i) \\ &= -\sum_i \sum_j P(x_i, y_j) \log P(x_i) - \sum_i \sum_j P(x_i, y_j) \log P(y_j|x_i) \\ &= -\sum_i \sum_j P(x_i, y_j) [\log P(x_i) + \log P(y_j|x_i)] \\ &= -\sum_i \sum_j P(x_i, y_j) \log [P(x_i)P(y_j|x_i)] \\ &= -\sum_i \sum_j P(x_i, y_j) \log P(x_i, y_j) = H(X, Y). \end{aligned} \quad (9.11)$$

Definition 9.1 (Transinformation) *Die Größe*

$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \quad (9.12)$$

heißt *mutual information* oder auch *Transinformation*.

Es gilt:

$$\begin{aligned} I(X, Y) &= H(X) - H(X|Y) \\ &= -\sum_i P(x_i) \log P(x_i) - \left(-\sum_i \sum_j P(x_i, y_j) \log P(x_i|y_j) \right) \\ &= -\sum_i \sum_j P(x_i, y_j) \log P(x_i) + \sum_i \sum_j P(x_i, y_j) \log P(x_i|y_j) \\ &= \sum_i \sum_j P(x_i, y_j) \log \frac{P(x_i|y_j)}{P(x_i)} \\ &= \sum_i \sum_j P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(y_j)P(x_i)}. \end{aligned} \quad (9.13)$$

Wir sehen, die *mutual information* ist ein Maß für die Unabhängigkeit der beiden ZV X und Y (ist mehr als die Dekorreliertheit). Ist $I(X, Y) = 0$, so liegt totale Unabhängigkeit von X und Y vor. Ist $I(X, Y) = H(X)$, so herrscht maximale Abhängigkeit, d. h. es gibt eine Funktion f mit $X = f(Y)$. Bei einem Übertragungskanal sollte immer maximale Transinformation vorliegen, ansonsten gibt es Übertragungsfehler.

Im Folgenden wollen wir noch die sogenannte *differentielle Entropie* einführen, d. h. die Entropie für stetig verteilte ZV. Gegeben sei also eine stetig verteilte ZV X :

$$F(x) = \int_{-\infty}^x f(t)dt, \quad f(x) \geq 0 \quad \forall x, \quad \int_{-\infty}^{+\infty} f(t)dt = 1. \quad (9.14)$$

Dabei ist $f(x)$ die Dichte der ZV X .

Definition 9.2 (Differentielle Entropie) *Die differentielle Entropie einer Zufallsvariable X lautet:*

$$h(X) = - \int_S f(x) \log f(x) dx, \quad S = \{x \mid f(x) > 0\}. \quad (9.15)$$

Beispiel 1 X sei gleichverteilt von 0 bis a :

$$h(X) = - \int_0^a \frac{1}{a} \log \frac{1}{a} dx = \log a. \quad (9.16)$$

Man sieht, für $0 < a < 1$ ist $h(X) < 0$, d. h. negativ, was für die diskrete Entropie unmöglich war.

Beispiel 2 Wir betrachten die Normalverteilung mit dem Erwartungswert $\mu = 0$:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{x^2}{2\sigma^2}, \quad h(X) = \frac{1}{2} \log_2(2\pi e \sigma^2). \quad (9.17)$$

Jetzt gilt etwas sehr Interessantes: **Unter allen Verteilungen mit gleicher Varianz hat die Normalverteilung die größte Entropie.**

Wir betrachten jetzt einmal den Zusammenhang zwischen der differentiellen und diskreten Entropie. Gegeben sei eine beliebige Dichte $f(x)$. Wir unterteilen die x -Achse in äquidistante Intervalle mit der Intervallbreite Δ . In jedem Intervall i gibt es dann nach dem Mittelwertsatz irgendein x_i , so dass $p_i = \Delta f(x_i)$ ist, d. h. es ergibt sich genau die Fläche unter der Funktion $f(x)$ im i -ten Intervall. Damit können wir X nun eine diskrete ZV X^Δ zuordnen und damit deren Entropie berechnen:

$$\begin{aligned} H(X^\Delta) &= - \sum_{i=-\infty}^{+\infty} p_i \log_2 p_i = - \sum_{i=-\infty}^{+\infty} \Delta f(x_i) \log_2 \Delta f(x_i) \\ &= - \log_2 \Delta - \sum_{i=-\infty}^{+\infty} \Delta \cdot f(x_i) \log_2 f(x_i) = - \log_2 \Delta + \tilde{h}(X). \end{aligned} \quad (9.18)$$

Wir sehen der zweite Term, $\tilde{h}(X)$ ist eine Näherung für $h(X)$ und es gilt $\lim_{\Delta \rightarrow 0} \tilde{h}(X) = h(X)$. Der andere Term strebt gegen $+\infty$. Stellen wir die Gleichung um, so können wir schreiben:

$$\lim_{\Delta \rightarrow 0} \{H(X^\Delta) + \log_2 \Delta\} = h(X) \quad (9.19)$$

Wählen wir für die Intervalllänge $\Delta = 2^{-n}$, d. h. wir haben die stetige ZV mit n Bit quantisiert oder wir haben 2^n Intervalle, dann können wir schreiben:

$$H(X^\Delta) \approx h(X) + n. \quad (9.20)$$

Häufig wird in der Informationstheorie die sogenannte *Jensen-Ungleichung* benötigt:

Satz 9.3 (Jensen-Ungleichung) *Es sei f eine konvexe Funktion und X sei eine ZV, dann gilt*

$$E\{f(X)\} \geq f(E\{X\}). \quad (9.21)$$

Wenn f sogar streng konvex ist, dann gilt die Gleichheit nur wenn X eine Konstante ist.

Beweis Wir wollen dies kurz für diskrete ZV mit vollständiger Induktion beweisen. Als Induktionsanfang nehmen wir eine Zwei-Punkte-Verteilung:

$$p_1 f(x_1) + p_2 f(x_2) \geq f(p_1 x_1 + p_2 x_2) \quad (9.22)$$

Dies ist ja gerade die Definition einer konvexen Funktion. Wir nehmen nun an, die Ungleichung sei richtig für eine $(k-1)$ -Punkte-Verteilung und wollen zeigen, dass sie dann auch für eine k -Punkte-Verteilung richtig ist. Mit der Bezeichnung:

$$p'_i = \frac{p_i}{1-p_k}, \quad i = 1, \dots, k-1, \quad \sum_{i=1}^{k-1} p'_i = 1 \quad (9.23)$$

folgt:

$$\begin{aligned} \sum_{i=1}^k p_i f(x_i) &= p_k f(x_k) + (1-p_k) \sum_{i=1}^{k-1} p'_i f(x_i) \\ &\geq p_k f(x_k) + (1-p_k) f\left(\sum_{i=1}^{k-1} p'_i x_i\right) \\ &\geq f\left(p_k x_k + (1-p_k) \sum_{i=1}^{k-1} p'_i x_i\right) = f\left(\sum_{i=1}^k p_i x_i\right). \end{aligned} \quad (9.24)$$

Der Beweis für stetige ZV geht analog. □

Die sogenannte relative Entropie oder *Kullback-Leibler-Distanz* ist durch:

$$D(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = E_p \left(\log \frac{p(x)}{q(x)} \right) \quad (9.25)$$

definiert. Man kann sie sich als Abstandsmaß zweier Verteilungen p und q vorstellen, sie ist aber kein echtes Abstandsmaß. Es ist $D(p\|q) \geq 0$ und Null genau dann, wenn $p = q$ ist. Dies kann man mit der Jensen-Ungleichung relativ simpel zeigen. Wir sehen, die *mutual information* $I(X, Y)$ ist der Kullback-Leibler-Abstand der Verbund-Verteilung $p(x, y)$ zur Produktverteilung $p(x) \cdot p(y)$.

Weiter können wir schlussfolgern: Aus

$$0 \leq I(X, Y) = H(X) - H(X|Y) \quad (9.26)$$

folgt die Ungleichung $H(X|Y) \leq H(X)$. Die mittlere bedingte Entropie verringert sich also immer gegenüber der unbedingten Entropie. Dies gilt aber nur für das Mittel. Eine einzelne bedingte Entropie $H(X|y_j)$ kann dagegen größer oder kleiner werden. Dazu ein einfaches

Beispiel Gegeben sei (X, Y) mit der Verteilung $p_{11} = 0, p_{12} = \frac{3}{4}, p_{21} = \frac{1}{8}, p_{22} = \frac{1}{8}$. Es ist $H(X) = 0,544$, $H(X|Y=1) = 0$, $H(X|Y=2) = 1$, aber $H(X|Y) = 0,25$.

Informationsmaße werden in der Bildverarbeitung in vielen Algorithmen verwendet, insbesondere in der Bilddatenkompression. So spielt z. B. bei Bildregistrierungen in der Medizinischen Bildverarbeitung die *mutual information* eine große Rolle.

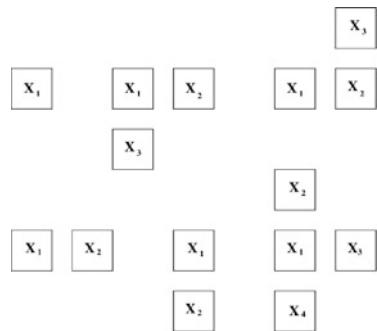
9.2 Statistiken n -ter Ordnung

Möchte man mit einem einzigen Bild Statistik betreiben, so gibt es sicher zwei Extrema:

- Alle Grauwerte x_i sind Realisierungen einer einzigen Zufallsvariablen X . Das „übliche“ Grauerthistogramm stellt dann eine statistische Beschreibung dieser einzigen Zufallsvariablen dar.
- Der Grauwert x_i ist eine Realisierung der Zufallsvariablen X_i , $i = 0, \dots, N - 1$. Damit ist das gesamte Bild nur eine Realisierung eines stochastischen Feldes.

Beide Extrema haben Vor- und Nachteile. Bei der Histogrammauswertung haben wir genügend Stichprobenmaterial zur Verfügung, die Ortsabhängigkeit der Grauwerte geht aber völlig verloren und damit jede Information über den Inhalt des Bildes. Beim stochastischen Feld beschreiben wir das Bild vollständig, können aber aus einer Realisierung ohne weitere Annahmen keine statistischen Aussagen treffen. Folglich scheint es sinnvoll zu sein, ein Modell zwischen beiden Extrema zu wählen. Dazu approximieren wir das gesamte

Abb. 9.1 Einige Teilstücke für Statistiken höherer Ordnung



stochastische Feld durch ein Teilstück. Das Teilstück aus Zufallsvariablen stellen wir uns als eine Schablone, eine Anordnung von Zufallsvariablen vor. Damit ist das Teilstück durch die Anzahl der Zufallsvariablen und durch deren gegenseitige Anordnung bestimmt. In der Abb. 9.1 sind einige Teilstücke zu sehen. Nun translatieren wir die Schablone über das konkrete Bild und betrachten die Grauwerte unter der Schablone als Realisierungen der Zufallsvariablen auf dem Teilstück. Die Anzahl der Zufallsvariablen auf dem Teilstück bestimmt dann eine n -dimensionale Verteilung oder auch Statistik n -ter Ordnung. Es gibt demnach mehrere Statistiken n -ter Ordnung in Abhängigkeit von der Topologie der Schablone. Bis zweiter Ordnung lassen sich die Histogramme noch sehr schön visualisieren. Die zweidimensionalen Histogramme werden oft *Co-occurrence-Matrizen* genannt.

9.3 Rauschmodelle

Unter Rauschen (*noise*) verstehen wir die stochastische Störung der „wahren“ Grauwerte. Die Art und Weise der Störung impliziert verschiedene Rauschmodelle. Wenn Rauschen als stochastische Störungen modelliert werden, dann ist es unmöglich mit einer oder einer endlichen Zahl von Aufnahmen das Rauschen zu beseitigen. Man kann eigentlich nur das Signal-Rausch-Verhältnis verbessern oder gar mit mehreren Aufnahmen das Rauschen absolut vermindern. Rauschvariablen werden oft mit N bezeichnet im Kontrast zu unserer bisherigen Bezeichnung als Anzahl von diskreten Funktionswerten. Weiterhin wird oft mittelwertfreies Rauschen ($E(N) = 0$) angenommen. Die Bezeichnung E bedeute den Erwartungswert einer Zufallsvariablen. Die Rauschstärke wird durch die Standardabweichung $\sigma(N)$ beschrieben. Im Zusammenhang mit Rauschen gibt es viele Begriffe. So wird oft signalunabhängiges Rauschen vorausgesetzt, d. h. die Verteilung der Rauschvariablen darf nicht vom eigentlichen Signalwert abhängen. In einfachen Modellen wird vom „weißen Rauschen“ gesprochen. Dies bedeutet $E(N_{i,j}) = 0$, $\sigma(N_{i,j}) = 1$ und alle $N_{i,j}$ sind paarweise dekorreliert. Damit wird die Autokorrelationsfunktion $E(\mathbf{N} * \mathbf{N}^R)$ zu:

$$E(\mathbf{N} * \mathbf{N}^R)_{m,n} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} E(N_{i,j} N_{i+m,j+n}) = M \cdot N \delta_{m,n}. \quad (9.27)$$

Das Spektrum des Einheitsimpulses δ ist $\alpha_{k,l} = \text{const}$. Da aber das Spektrum der Autokorrelationsfunktion das Leistungsspektrum der Originalfunktion ist, besitzen alle Frequenzen gleiche Intensitäten wie beim weißen Licht, daher auch der Name weißes Rauschen. Modelle zur Rauschminderung oder zum Verbessern des Signal-Rausch-Verhältnisses fallen mit in das Gebiet der Bildrestaurierung (siehe Abschn. 9.5), wobei in der Bildrestaurierung noch zusätzlich die Bildschärfe zu restaurieren ist. Das Signal-Rausch-Verhältnis (*signal noise ratio (SNR)*) ist dabei zu:

$$\text{SNR} = 10 \log_{10} \frac{\sum_i \sum_j (g_{i,j} - \bar{g})^2}{\sum_i \sum_j (n_{i,j} - \bar{n})^2} \quad (\text{dB}) \quad (9.28)$$

definiert. Dabei sind $g_{i,j}$ die Grauwerte eines Bildes und $n_{i,j}$ die Realisierungen des Rauschens. Insbesondere können die einfachen Verfahren, wie Glättungsfilter und Tiefpassfilter, nur das SNR beeinflussen. Das simpelste Rauschmodell ist:

$$\mathbf{X} = \mu + \mathbf{N} \quad (9.29)$$

mit den Forderungen $E(\mathbf{X}) = \mu$, d. h. mittelwertfreies Rauschen $E(\mathbf{N}) = \mathbf{0}$. Oft möchte man aus den Annahmen eines Rauschmodells die Verteilung des stochastischen Feldes \mathbf{X} ermitteln.

Beispiel 1 Wir nehmen das einfache Modell (9.29) und beschränken uns auf Zufallsvektoren, d. h. auf eindimensionale Bilder. Die Rauschvariablen N_i seien paarweise dekorreliert und identisch Gauß-verteilt mit $E(N_i) = 0$ und $\sigma^2(N_i) = \sigma^2$. Dann können wir die Verteilung von \mathbf{X} trivialerweise sofort aufschreiben:

$$f(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{1}{2\sigma^2} \sum_{i=0}^{N-1} (x_i - \mu_i)^2}. \quad (9.30)$$

In Abschn. 9.5.4 findet man ein Beispiel für die Verteilung, wenn \mathbf{N} eine Poisson-Verteilung besitzt.

Beispiel 2 Jetzt betrachten wir ein Modell, welches als Kanalrauschen bezeichnet wird und welches gleichzeitig ein Beispiel für multiplikatives Rauschen darstellt. Wir betrachten ein eindimensionales Binärbild \mathbf{y} mit $y_i \in \{-1, +1\}$. Nun kann sich y_i stochastisch ändern: Mit der Wahrscheinlichkeit p nimmt y_i seinen Komplementärwert an, mit der Wahrscheinlichkeit $1 - p$ bleibt er erhalten, daher ist

$$X_i = y_i \cdot N_i, \quad (9.31)$$

wobei die Rauschvariable N_i eine Bernoullische Zufallsvariable mit

$$P(N_i = -1) = p, \quad P(N_i = +1) = 1 - p \quad (9.32)$$

ist. Daher hat das stochastische Feld \mathbf{X} die diskrete Wahrscheinlichkeitsverteilung

$$P(\mathbf{X} = \mathbf{x}) = p^{\text{card}\{I \mid i \in I, y_i = -x_i\}} \cdot (1-p)^{\text{card}\{J \mid j \in J, y_j = x_j\}}. \quad (9.33)$$

Beispiel 3 Es gibt natürlich aus Rauschformen, die sich nur sehr schwer mit stochastischen Modellen beschreiben lassen. Ein typisches Beispiel dafür ist das Salz- und Pfeffer-Rauschen (*salt-and-pepper noise*). Es ist tatsächlich ähnlich einem Gemisch aus einzelnen Salz(weiß)- und Pfeffer(schwarz)-Körnern, die man mit einem Streuer über das Bild ausbringt. Da diese Körner im Bild singulär vorkommen (nicht in Clustern), ist das typische Werkzeug zur Beseitigung (bzw. Reduktion) des Rauschens ein geeignet zu wählender Rangordnungsfilter (z. B. Medianfilter), siehe Abschn. 8.2.5.

9.3.1 Rauschminderung mit mehreren Aufnahmen

Wir nehmen das einfache Modell (9.29) an, wobei uns die Kovarianzmatrix Σ_N des Rauschens nicht interessiert, wir betrachten nur die unbekannten Standardabweichungen $\sigma_{i,j}$ der Rauschvariablen $N_{i,j}$ als Maß für die Stärke des Rauschens. Aus der Stochastik ist bekannt:

Sind X und Y zwei unabhängige Zufallsvariable, dann gelten die einfachen Rechenregeln:

$$\begin{aligned} E(aX \pm bY) &= aE(X) \pm bE(Y), \quad a, b \in \mathbb{R} \\ \sigma^2(aX \pm bY) &= a^2\sigma^2(X) + b^2\sigma^2(Y), \quad a, b \in \mathbb{R}. \end{aligned} \quad (9.34)$$

Haben wir nun n unabhängige Realisierungen \mathbf{X}^j , $j = 1, \dots, n$ des Feldes \mathbf{X} zur Verfügung, dann berechnen wir pro Pixel den Mittelwert und erhalten mit den Rechenregeln (9.34):

$$E\left(\frac{1}{n} \sum_{j=1}^n X_i^j\right) = \mu_i, \quad \sigma^2\left(\frac{1}{n} \sum_{j=1}^n X_i^j\right) = \frac{1}{n} \sigma_i^2. \quad (9.35)$$

Demnach bleibt der „wahre“ Wert μ erhalten und die Standardabweichung verhält sich zu:

$$\sigma(\bar{X}_i) = \frac{1}{\sqrt{n}} \sigma(X_i). \quad (9.36)$$

Das Rauschen wird folglich mit dem Faktor $\frac{1}{\sqrt{n}}$ gedämpft. In der Praxis würde dies bedeuten: Wir müssen mit einer Kamera n -mal dieselbe Szene aufnehmen. Dabei darf mit der Kamera nicht „gewackelt“ werden, in der Szene darf sich nichts bewegen und es darf keinerlei Beleuchtungsschwankungen geben. In der Astronomie sind Aufnahmen mit langen Belichtungszeiten üblich, was genau diesem Modell entspricht, so dass rauscharme Bilder entstehen.

9.3.2 Rauschen und Bilddatenkompression

Ein verrauchtes Bild bereitet der Datenkompression ohne Informationsverlust große Schwierigkeiten, daher bestehen in der Datenkompression ohne Informationsverlust zwischen künstlichen Bildern (Computergrafik) und realen Bildern erhebliche Unterschiede. Man kann verbal sagen: Unabhängiges Rauschen lässt sich nicht verlustfrei komprimieren. Dazu ein einfaches Beispiel. Wenn wir ein 8-Bit-Bild mit einer Kamera aufnehmen, dann sind 2-Bit-Rauschen nicht außergewöhnlich. Folglich kann man das Bild nicht unter 25 Prozent verlustfrei komprimieren. Nun kommt aber noch die eigentliche Bildinformation dazu, so dass verlustfreie Kompressionsraten unter 50 Prozent nur in gutartigen Fällen zu finden sind. Dagegen lassen sich rauschfreie Bilder der Computergrafik sehr gut verlustfrei komprimieren. Ein weiteres Beispiel sind Prädiktor-Korrektor-Verfahren der Datenkompression. Mit einem möglichst einfachen Modell werden Daten vorhergesagt und nur die Differenz zu den tatsächlichen Daten wird kodiert. Je besser die Vorhersage, desto besser die Kompression. Wenn wir z. B. eine Bildfolge vorliegen haben, dann wäre die Einzelbildkompression sicher das schlechteste Verfahren. Der simpelste Prädiktor ist dann der identische Operator, d. h. man kodiert das erste Bild und dann nur noch die Differenzen zweier aufeinander folgender Bilder. Sind die Grauwerte als Zufallsvariablen pixelweise abhängig, dann wird die Entropie des Differenzbildes klein werden und man kann dieses gut komprimieren. Sind die Zufallsvariablen X und Y pro Pixel zweier Bilder aber unabhängig und haben z. B. die gleiche Varianz $\sigma^2(X) = \sigma^2(Y)$, so können wir (9.34) vereinfachen zu:

$$\sigma^2(X - Y) = \sigma^2(X) + \sigma^2(Y) = 2\sigma^2(X) \rightarrow \sigma(X - Y) = \sqrt{2} \cdot \sigma(X). \quad (9.37)$$

Wir sehen, die Standardabweichung der Differenz wird gegenüber den Einzelbildern um den Faktor $\sqrt{2}$ verstärkt. Damit wird die Entropie des Differenzbildes sogar größer und die Kompression damit schlechter. Da man bei Rauschen oft von pixelweise unabhängigem Rauschen ausgeht, tritt genau dieser Effekt ein und das Prädiktor-Korrektor-Modell kann im Extremfall sogar die Kompressionsraten verschlechtern. Genau dieser Effekt tritt auch bei der Kodierung eines Einzelbildes auf, wenn man z. B. die Differenz der Zeilen, bzw. der Spalten oder von Diagonalen kodieren möchte.

9.4 Energiefunktionen

Es sei \mathbf{x} der Vektor der Grauwerte aller Pixel eines Bildes als Realisierung des Stochastischen Feldes \mathbf{X} . In Anlehnung an die Physik führt man eine „Energiefunktion“ ein, die die Zustände des Bildes bewertet. Kleine Energien seien günstige Zustände und hohe Energien sollen ungünstige Zustände beschreiben, siehe auch [89]. Diese Energiefunktion $E(\mathbf{x})$ ist folglich das Modell zur Beschreibung der Bildzustände. Diesen Energiefunktionen kann man leicht Wahrscheinlichkeiten zuordnen:

$$p(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})}, \quad Z = \sum_{\mathbf{x}} e^{-E(\mathbf{x})}. \quad (9.38)$$

Z ist eine Normierungskonstante, wobei bei stetigen Variablen die Summation durch eine Integration zu ersetzen ist. In dieser Form nennt man \mathbf{X} Gibbs-verteilt. Gibbs-Verteilungen wurden erstmals durch Hassner und Sklansky in der Bildverarbeitung eingeführt.

Beispiel 1 Das einfachste und bekannteste Energiemodell ist das Ising-Modell zur Deutung des Ferromagnetismus:

$$E(\mathbf{x}) = -\alpha \sum_{i \in B} \sum_{k \in N_i} x_i x_k. \quad (9.39)$$

Dabei ist N_i eine Nachbarschaft von i , z. B. die Vierernachbarschaft. Weiter nehmen die Grauwerte nur Binärzustände $x_i = +1$ oder $x_i = -1$ an. Mit diesem Modell bewerten wir alle Binärbilder. Für $\alpha > 0$ gibt es nur zwei Minima der Energiefunktion, nämlich ein homogen weißes oder homogen schwarzes Bild. Obwohl man trivialerweise diese beiden globalen Minima sofort ablesen kann, ist es schwierig mit einer geeigneten Startlösung und einem geeigneten Optimierungsalgorithmus auch zu einem dieser beiden globalen Minima zu gelangen. Im Gegensatz zur Physik sieht man an diesem Beispiel, dass Energiefunktionen auch negative Werte haben dürfen. Die Verallgemeinerung des Isingmodells auf Grauwertbilder führt zum Potts-Modell:

$$E(\mathbf{x}) = -\alpha \sum_{i \in B} \sum_{k \in N_i} \delta_{x_i - x_k}. \quad (9.40)$$

Man kann beim Isingmodell auch jede Kombination einzeln bewerten:

$$E(\mathbf{x}) = \sum_{i \in B} \sum_{k \in N_i} a_{i,k} x_i x_k + b_i x_i. \quad (9.41)$$

Dies ist eine allgemeine quadratische Form, die nur unter bestimmten Konvexitätseigenschaften global zu minimieren ist.

Nun wollen wir mit einem MAP-Schätzer (siehe Abschn. 18.3) aus einem gegebenen, verrauschten Bild \mathbf{y} das wahrscheinlichste Bild \mathbf{x} bezüglich der A-priori-Gibbsverteilung oder das mit der kleinsten Energie bestimmen. Dazu müssen wir mit dem Rauschmodell die A-posteriori-Energiefunktion aus den A-priori-Wahrscheinlichkeiten und den Übergangswahrscheinlichkeiten bestimmen.

Beispiel 1 Das Stochastische Feld \mathbf{X} besitze eine Dichtefunktion $f(\mathbf{x})$ in der Gibbsschen Form mit der Energiefunktion $E(\mathbf{x})$. Das Bildmodell sei $\mathbf{Y} = \mathbf{X} + \mathbf{N}$. Dabei sind \mathbf{X} und \mathbf{N} unabhängig. Das Rauschen \mathbf{N} sei weißes, mittelwertfreies Gaußsches Rauschen. Dann ist

$$f(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})f(\mathbf{y} | \mathbf{x}) = f(\mathbf{x}) \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{\sum_i (y_i - x_i)^2}{2\sigma^2}}. \quad (9.42)$$

Damit können wir die neue Energiefunktion ablesen zu

$$E(\mathbf{x}, \mathbf{y}) = E(\mathbf{x}) + \frac{\|\mathbf{y} - \mathbf{x}\|^2}{2\sigma^2}. \quad (9.43)$$

Bei gegebenem Bild \mathbf{y} lösen wir die Optimierungsaufgabe $\min_{\mathbf{x}} E(\mathbf{x}, \mathbf{y})$. Dies ist identisch mit einem MAP-Schätzer, siehe Abschn. 18.3, wobei wir $E(\mathbf{x}|\mathbf{y})$ durch $E(\mathbf{x}, \mathbf{y})$ ersetzen können. Ist die A-priori-Verteilung bzw Gibbsverteilung konstant, d. h. $E(\mathbf{x}) = \text{constant}$, dann ist diese Aufgabe etwas unsinnig gestellt. Bei gegebenem \mathbf{y} führt dann die Minimierung von (9.43) immer auf die Lösung $\mathbf{x} = \mathbf{y}$, d. h. wir nehmen als Lösung das verrauschte Bild selbst. Wir benötigen folglich bei dieser simplen Form der Übergangswahrscheinlichkeiten $f(\mathbf{y}|\mathbf{x})$ in (9.42) eine sinnvolle, nichtkonstante A-priori-Energiefunktion $E(\mathbf{x})$. Die einfachste Idee ist: Wir suchen ein möglichst glattes restauriertes Bild. Daher wählen wir $E(\mathbf{x}) = \beta \sum_i \sum_{j \in N(i)} (x_i - x_j)^2$, $\beta > 0$. Folglich wird (9.43) zu

$$E(\mathbf{x}, \mathbf{y}) = \beta \sum_i \sum_{j \in N(i)} (x_i - x_j)^2 + \frac{\|\mathbf{y} - \mathbf{x}\|^2}{2\sigma^2}. \quad (9.44)$$

Mit dem Parameter β kann man nun die Stärke der Glattheit einstellen.

Beispiel 2 Wir nehmen einmal ein Binärbild und als A-priori-Energiefunktion das Ising Modell, siehe (9.39). Das Rauschmodell sei das Kanalrauschen, siehe (9.34). Dann lautet die A-posteriori-Energiefunktion

$$E(\mathbf{x}, \mathbf{y}) = -\beta \sum_i \sum_{j \in N(i)} x_i x_j - \frac{1}{2} \ln \frac{1-p}{p} \sum_i x_i y_i. \quad (9.45)$$

Mit β kann wieder der Einfluss der Glattheit gesteuert werden.

Dieser Abschnitt sollte verdeutlichen, dass man einen Zusammenhang zwischen Verteilungen und Energiefunktionen herstellen kann. Der Sinn besteht eigentlich nur darin, die Zustände maximaler Wahrscheinlichkeit mit den Zuständen minimaler Energie zu beschreiben. Die Zustände minimaler Energie kann man aber nur mit Optimierungsverfahren finden, die das globale Optimum berechnen können. Dazu gibt es recht aufwendige Verfahren, z. B. den *Gibbs-Sampler* oder die Methode des *Simulated Annealing*.

9.5 Bildrestaurierung

Die Bildrestaurationsmethoden werden nach den verschiedensten Zielkriterien eingeteilt. So unterscheidet man algebraische und stochastische Methoden, Methoden im Ortsraum oder Frequenzraum, direkte Methoden oder iterative Methoden, Regularisierungsmethoden usw., siehe z. B. [39].

Wir werden hier diese Einteilung nicht vornehmen, sondern nur die wichtigsten Methoden beschreiben. Wir bezeichnen wie in der Literatur üblich die Zufallsvariable des Rauschens mit N (*Noise*) und folglich eine Realisierung mit n . Bisher hatten wir aber immer mit N und n Anzahlen von Funktionswerten bezeichnet. Was von beiden gemeint ist, soll stets aus dem Zusammenhang eindeutig hervorgehen.

9.5.1 Invers-Filter

Das bekannteste Modell ist:

$$G = h * f + N. \quad (9.46)$$

Wir nehmen einmal an, die PSF h sei invertierbar. Wenn die Faltung $h * f$ durch den Rauschterm n „gestört“ wird, dann kann bei der Invertierung $\tilde{f} = h^{-1} * g = f + h^{-1} * n$ das Rauschen derart verstärkt werden, dass die Näherung \tilde{f} von f zu sehr abweicht und völlig unbrauchbar wird. Im Abschn. 2.2.2 hatten wir die Stabilität eines linearen Systems beschrieben und den Begriff der Konditionszahl eingeführt. Im Abschn. 4.12 hatten wir den Zusammenhang der Konditionszahl zu einer Faltungsgleichung und zu den Fourierkoeffizienten hergestellt und schließlich in (4.48) die Konditionszahl $K(h) = \|h\| \cdot \|h^{-1}\|$ bezüglich der Faltung:

$$K(h) = \frac{\max_j |\alpha_j(h)|}{\min_k |\alpha_k(h)|} \quad (9.47)$$

ausgerechnet. Daraus können wir nun umgekehrt ablesen, wann das Invers-Filter stabil ist. Die „Schwankungsbreite“ des Amplitudenspektrums von h ist die entscheidende Größe, die Phase spielt keine Rolle. Die Schwankungsbreite ist minimal, wenn das Amplitudenspektrum von h konstant ist. Wenn folglich das Amplitudenspektrum der PSF h nahezu konstant ist, dann kann man auch das Invers-Filter zur Restauration benutzen.

9.5.2 Restauration unter Zwang

Wir nehmen wieder als Bildentstehungsmodell:

$$G = h * f + N \quad (9.48)$$

an, wobei die PSF h wieder gegeben sein soll. Beim Invers-Filter im Abschn. 9.5.1 hatten wir vorausgesetzt, dass die PSF invertierbar sein soll. Diese Voraussetzung ist bei praktischen Problemen selten erfüllt. Daher stellen wir jetzt keine Forderung mehr an die PSF h . Über das Rauschen N wissen wir zunächst nichts. Wenn wir die konkrete Realisierung n des Rauschens N kennen würden, so schreiben wir einfach:

$$g - n = h * f \quad (9.49)$$

und berechnen f durch lösen der Gaußschenen Faltungs-Normalengleichungen, siehe (2.55). Dies wäre aber zu schön um war zu sein, wir wissen eigentlich immer nicht viel über das Rauschen. Aber so könnten wir wenigstens eine obere Schranke $\overline{E(\|N\|^2)}$ für die mittlere Rauschstärke $E(\|N\|^2) \leq \overline{E(\|N\|^2)}$ kennen, daher gehen wir zur Norm über und suchen ein f , das die Ungleichung:

$$E(\|G - h * f\|)^2 \leq \overline{E(\|N\|^2)} \quad (9.50)$$

erfüllt. Nun haben wir aber das Problem, dass es dafür unendlich viele Lösungen f geben kann, diese müssen wir irgendwie vernünftig bewerten. Wir suchen aus den vielen möglichen Lösungen eine aus, die einem Kriterium genügen soll. Dieses Kriterium verstecken wir abstrakt in einem Operator T . Folglich haben wir ein Optimierungsproblem zu lösen:

$$\|Tf\|^2 \rightarrow \text{Minimum} \quad \text{bei } E(\|G - h * f\|^2) \leq \overline{E(\|N\|^2)}. \quad (9.51)$$

Wenn wir nun annehmen, dass der Operator T linear ist, also $T = L$, dann kann man beweisen, dass das Minimum auf dem Rand angenommen wird. Wir müssen also lösen:

$$\|Lf\|^2 \rightarrow \text{Minimum} \quad \text{bei } E(\|G - h * f\|^2) = \overline{E(\|N\|^2)}. \quad (9.52)$$

Da die Restriktion nun in Gleichungsform vorliegt, können wir die Langrangesche Funktion bilden:

$$z(f, \alpha) = \|Lf\|^2 - \alpha(E(\|G - h * f\|^2) - \overline{E(\|N\|^2)}) \rightarrow \text{Minimum}. \quad (9.53)$$

Wir müssen nun den Gradienten von z nach f bilden und gleich Null setzen. Dazu nehmen wir weiter an, dass L sogar ein LSI-Operator ist, daher können wir L mit einer Zirkularmatrix C_l identifizieren. Weiterhin sei C_h die Zirkularmatrix zu h und wir interpretieren die diskreten Funktionen wieder als Spaltenvektoren. Der Einfachheit halber ersetzen wir G durch eine Realisierung g . Nun können wir den Gradienten bilden:

$$\nabla_f z(\mathbf{f}, \alpha) = 2C_l^T C_l \mathbf{f} - 2\alpha C_h^T (\mathbf{g} - C_h \mathbf{f}) = \mathbf{0}. \quad (9.54)$$

Wir lösen nach \mathbf{f} auf:

$$\mathbf{f} = \left(C_h^T C_h + \frac{1}{\alpha} C_l^T C_l \right)^{-1} C_h^T \mathbf{g}. \quad (9.55)$$

Für $\alpha \rightarrow \infty$ erhalten wir die Lösung mit der Pseudoinversen, siehe auch (22.17). Den Parameter α nennt man auch *Regularisierungsparameter* und das Verfahren als Regularisierung nach Tichinov und Arsenin, siehe Abschn. 22.2. Da wir die Lösung über Zirkularmatrizen hergeleitet haben, können wir die Lösung auch wieder in die Faltung übersetzen:

$$f * \left(h^R * h + \frac{1}{\alpha} l^R * l \right) = h^R * g. \quad (9.56)$$

Nun wenden wir darauf das Faltungstheorem (4.23) an, wobei unbedingt und genau auf die Faktoren zu achten ist (wir bezeichnen die Dimension jetzt ausnahmsweise mit N_d , um sie vom Rauschen N zu unterscheiden):

$$\sqrt{N_d} \cdot \alpha_k(f) \cdot \left(\sqrt{N_d} |\alpha_k(h)|^2 + \sqrt{N_d} \frac{1}{\alpha} |\alpha_k(l)|^2 \right) = \sqrt{N_d} \cdot \overline{\alpha_k(h)} \alpha_k(g) \quad (9.57)$$

Wir lösen auf:

$$\alpha_k(f) = \frac{\overline{\alpha_k(h)} \alpha_k(g)}{\sqrt{N_d} |\alpha_k(h)|^2 + \sqrt{N_d} \frac{1}{\alpha} |\alpha_k(l)|^2}. \quad (9.58)$$

Was für ein l könnte man sinnvollerweise nehmen? Sinnvoll ist z. B.: unter allen zulässigen f möchten wir das Glatteste. Dies bedeutet, die Summe der Betragsquadrate der Ableitungen von f soll minimal sein. Wir können für l irgendein Ableitungsfilter benutzen, z. B. den einfachen Laplacefilter, da wir die Ableitungen richtungsunabhängig benötigen:

$$l = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \quad (9.59)$$

Die einzige übrigbleibende Frage ist die Wahl des Regularisierungsparameters α . Man könnte probieren und das Restorationsergebnis bewerten oder man bestimmt diesen Parameter exakt nach der Abschätzung der mittleren Rauschstärke. Wir bilden die Funktion:

$$r(\alpha) = g - h * f = g - h * \left(h^R * h + \frac{1}{\alpha} l^R * l \right)^{-1} * h^R * g \quad (9.60)$$

und damit die Fehlerfunktion

$$\varphi(\alpha) = \|r(\alpha)\|^2 - \overline{E(\|N\|^2)}. \quad (9.61)$$

Wir müssen nun die Nullstelle dieser Funktion $\varphi(\alpha)$ finden. Man kann sogar zeigen, dass diese Funktion $\varphi(\alpha)$ monoton fallend in α ist. Damit können wir mittels Intervallschachtelung eine ausreichende Näherung der Nullstelle bestimmen.

Welche Forderung an f außer Glattheit könnte noch sinnvoll sein? Nehmen wir einmal den einfachsten Fall, der LSI-Operator ist der identische Operator $L = I$, d.h. $l = \delta$. Damit suchen wir unter allen zulässigen f dasjenige mit $\|f\|^2 \rightarrow \text{Minimum}$. Wir suchen also das Originalbild mit der kleinsten Norm. Ob diese Fragestellung gewünscht ist oder nicht, ist eine andere Frage. Die Lösung im Orts- und Spektralbereich können wir sofort aufschreiben:

$$f = \left(C_h^T C_h + \frac{1}{\alpha} I \right)^{-1} C_h^T g, \quad \alpha_k(f) = \frac{1}{\sqrt{N_d}} \cdot \frac{\overline{\alpha_k(h)} \alpha_k(g)}{|\alpha_k(h)|^2 + \beta}, \quad \beta = \frac{1}{\alpha} \cdot \frac{1}{N_d}. \quad (9.62)$$

Diese Lösung ist weiter nichts als die Lösung der regularisierten Faltungs-Normalengleichungen (2.69). Die Formel (9.62) können wir auch anders schreiben:

$$\alpha_k(f) = \frac{1}{\sqrt{N_d}} \cdot \frac{\overline{\alpha_k(h)}\alpha_k(g)}{|\alpha_k(h)|^2 + \beta} = \frac{1}{\sqrt{N_d}} \cdot \frac{\alpha_k(g)}{\alpha_k(h)} \cdot \left(\frac{|\alpha_k(h)|^2}{|\alpha_k(h)|^2 + \beta} \right). \quad (9.63)$$

Den in Klammer stehenden Ausdruck können wir als Korrekturfaktor pro Frequenz gegenüber der reinen Entfaltung interpretieren. Die Fehlerfunktion könnten wir jetzt gleich in Abhängigkeit von β betrachten. Man beachte: Der Term $\beta \geq 0$ ist eine nichtnegative, reelle Konstante und hängt demzufolge nicht von k ab, aber sie hängt von der Dimension N ab.

Dieses Restaurationsfilter liefert in der Praxis schon sehr gute Ergebnisse und wird als *Pseudoinvers-Filter* bezeichnet. Dieser Begriff könnte zu Verwechslungen mit der Pseudoinversen einer Matrix führen, hat aber direkt mit ihr nichts zu tun.

9.5.3 Wiener-Hellstrom-Optimal-Filter

Wie nehmen als Bildentstehungsmodell wie üblich

$$G = h * f + N \quad (9.64)$$

an, wobei die PSF h gegeben sein soll. $E(\dots)$ bedeute wieder den Erwartungswert einer Zufallsvariablen oder Zufallsfeldes. Über das Rauschen N machen wir einige Annahmen, z. B. sei $E(N) = 0$ und das Rauschen sei signalunabhängig. Wir wollen im Folgenden auch f als Zufallsfeld auffassen:

$$G = h * F + N. \quad (9.65)$$

Signalunabhängiges Rauschen bedeutet nun, dass $E(F_i \cdot N_j) = E(F_i) \cdot E(N_j)$ gilt. Beim Original Wiener-Filter (1942) wurde zunächst nur das Rauschen betrachtet. Erst später (1968) bezog Hellstrom die PSF h mit in seine Überlegungen ein. Wir suchen nun einen Operator T , sodass

$$E(\|F - TG\|^2) \rightarrow \text{Minimum} \quad (9.66)$$

gilt. Auf Grund dieses Ansatzes ist das Wiener-Filter ein MMSE-Filter (*Minimum Mean Square Error*), siehe auch Abschn. 18.3.3. Zwei Probleme fallen gleich auf: man müsste F kennen, um überhaupt das Problem zu lösen zu können, F wird aber doch gerade gesucht. Wir werden noch sehen, dass man dies tatsächlich abschwächen kann. Aber einen allgemeinen nichtlinearen Operator können wir so ohne weiteres nicht finden, wir müssen dies

drastisch einschränken, z. B. T sei ein linearer Operator. Wenn wir die diskreten Funktionen wieder als Spaltenvektoren interpretieren, dann können wir T als eine Matrix \mathbf{A} annehmen und wir müssen lösen:

$$E(\|F - \mathbf{A} \cdot G\|^2) \rightarrow \text{Minimum}. \quad (9.67)$$

Als notwendiges Kriterium, siehe auch Abschn. 21.2.1, gilt

$$E\{\mathbf{A} \cdot G - F\} \cdot G^T = \mathbf{0}. \quad (9.68)$$

Wir setzen für G die Beziehung (9.65) ein und erhalten

$$E(\mathbf{A} \cdot (\mathbf{C}_h F + N) \cdot (F^T \mathbf{C}_h^T + N^T)) = E(F(F^T \mathbf{C}_h^T + N^T)). \quad (9.69)$$

Nun multiplizieren wir aus. Bei einigen Termen können wir auf Grund der gemachten Voraussetzungen an das Rauschen die Erwartungswerte „hineinziehen“. Da der Erwartungswert von N mit Null angenommen wird, verschwinden einige Terme. Mit der Bezeichnung $\text{COV}_X := E(\mathbf{XX}^T)$ für die Kovarianzmatrix eines Zufallsvektors \mathbf{X} erhalten wir dann:

$$\mathbf{A} = \text{COV}_F \cdot \mathbf{C}_h^T \left[\mathbf{C}_h \cdot \text{COV}_F \cdot \mathbf{C}_h^T + \text{COV}_N \right]^{-1}. \quad (9.70)$$

Wir sehen nun, wir müssen tatsächlich nicht das Original F kennen, sondern nur die Kovarianzmatrix COV_F , was weitaus schwächer ist. Zusätzlich muss man noch die Kovarianzmatrix COV_N des Rauschens kennen. Selbst wenn wir diese Kovarianzmatrizen berechnen könnten, wird deren Dimension für praktische Aufgabenstellungen gewaltig sein. Daher machen wir an den Operator T außer der Linearität noch eine weitere Einschränkung: T soll zusätzlich noch verschiebungsinvariant sein. Daher können wir jetzt T als Faltung mit einer PSF w darstellen. Daher müssen wir nun lösen:

$$E(\|F - w * G\|^2) \rightarrow \text{Minimum}. \quad (9.71)$$

Dies ist aber nichts anderes als das lineare Gleichungssystem $E(\mathbf{C}_G \cdot \mathbf{w}w = \mathbf{F})$ im Sinne der kleinsten Quadrate zu lösen. In dieser Notation können wir also die üblichen Gaußschen Normalengleichungen benutzen:

$$E(\mathbf{C}_G^T \mathbf{C}_G \mathbf{w} = \mathbf{C}_G^T \mathbf{F}). \quad (9.72)$$

Dies übersetzen wir wieder in die Faltung

$$E(G^R * G * w = G^R * F) \quad (9.73)$$

und setzen nun die Modellgleichung für G ein:

$$\begin{aligned} & [h^R * h * E(F^R * F)] + [h^R * E(F^R * N) + h * E(N^R * F) + E(N^R * N)] * w \\ &= h^R * E(F^R * F) + E(N^R * F). \end{aligned} \quad (9.74)$$

Auf Grund unserer Voraussetzungen an das Rauschen verschwinden folgende drei Terme:

$$h^R * E(F^R * N), h * E(N^R * F), E(N^R * F). \quad (9.75)$$

Daher ergibt sich die Bestimmungsgleichung für das Filter w :

$$[h^R * h * E(F^R * F) + E(N^R * N)] * w = h^R * E(F^R * F). \quad (9.76)$$

Wir sehen, nun geht nicht mehr das Original F direkt ein, sondern wir brauchen nur noch deren Autokorrelation $E(F^R * F)$ zu kennen. Da uns eigentlich nicht das Wiener-Filter, sondern nur das geschätzte Bild $\hat{f} = w * g$ aus einer Realisierung g von G interessiert, falten wir beiden Seiten mit g durch und erhalten:

$$[h^R * h * E(F^R * F) + E(N^R * N)] * \hat{f} = h^R * E(F^R * F) * g. \quad (9.77)$$

Wenn wir nun das Faltungstheorem anwenden (siehe Abschn. 4.8), erhalten wir:

$$\alpha_k(\hat{f}) = \frac{1}{\sqrt{N_d}} \cdot \frac{\overline{\alpha_k(h)} \cdot E(|\alpha_k(F)|^2) \alpha_k(g)}{|\alpha_k(h)|^2 E(|\alpha_k(F)|^2) + \frac{E(|\alpha_k(N)|^2)}{N_d}}. \quad (9.78)$$

Wir benötigen also die mittleren Autokorrelationsfunktionen des Originals und des Rauschens bzw. die mittleren Leistungsspektren des Originalsignals und des Rauschens. Für das Rauschen wird oft weißes Rauschen angenommen, in diesem Falle ist:

$$E(N^R * N) = \delta N_d \sigma^2 \rightarrow |\alpha_k(N)|^2 = \sigma^2. \quad (9.79)$$

Wenn wir Zähler und Nenner durch $E(|\alpha_k(F)|^2)$ teilen und annehmen, dass der Quotient $\frac{E(|\alpha_k(N)|^2)}{E(|\alpha_k(F)|^2)} = \beta$ konstant ist, dann erhalten wir genau (9.63) bezüglich der Restauration unter Zwang. Wenn wir wieder weißes Rauschen annehmen, dann benötigen wir also im Normalfall außer der Rauschstärke σ^2 in (9.64) die Autokorrelation des Originalsignals $E((F^R * F)_n) = E(\sum_l F_l F_{l-n})$. Da wir aber im Normalfall diese aus dem unscharfen Bild g schätzen müssen und damit nur eine einzige Stichprobe vorliegt, führt dies zu Schwierigkeiten in der Zuverlässigkeit der Schätzung. Wenn wir aber z. B. annehmen, dass ein Summand $(K_F)_n = E(F_k \cdot F_{k-n})$ nur noch von n abhängt und nicht mehr von k , dann könnten wir die Autokorrelation tatsächlich auch aus einem Bild schätzen, da wir für diese Produkte genug Stichproben zur Verfügung hätten. In der Stochastik werden solche Zufallsfelder oder Prozesse als **stationär im erweiterten Sinne** bezeichnet. Nehmen wir dies einfach an, dann bleibt aber immer noch das Problem, dies aus dem unscharfen Bild g zu schätzen. Dazu nimmt man oft die Autokorrelation von g , restauriert das Bild (in der Hoffnung, dass es dem Originalbild besser entspricht) und wiederholt den Prozess bis zu einer Abbruchbedingung. Dazu gibt es in der Literatur zahlreiche Varianten dieser iterativen Wiener-Entfaltung.

9.5.4 Richardson-Lucy-Algorithmus

Im Folgenden sollen einige statistische Betrachtungen angestellt werden. Wir gehen wieder vom Modell (9.64) aus. Wenn wir nun annehmen, alle N_k seien unabhängig, normalverteilt mit dem Erwartungswert Null und der gleichen Varianz σ , dann können wir z. B. folgende Likelihoodfunktion (siehe Abschn. 18.3.1):

$$L(f) = \prod_{k=0}^{N-1} e^{-\frac{(g_k - (h * f)_k)^2}{2\sigma^2}} \quad (9.80)$$

zur Schätzung von f aufstellen. Statt $L(f)$ zu maximieren, minimieren wir $-\ln(L(f))$ und sehen sofort, dass dann:

$$\|g - h * f\|^2 \rightarrow \text{Minimum} \quad (9.81)$$

zu minimieren ist. Dieses Ergebnis ist insofern schlecht, da es die übliche inverse Filterung beschreibt. Über dieses Ergebnis brauchen wir uns nicht wundern: Für jedes Pixel k haben wir eine Normalverteilung mit einem anderen Erwartungswert $(h * f)_k$ angenommen. Damit wollen wir aus N Werten auch N Parameter schätzen, was natürlich sehr problematisch ist. Aber vielleicht bekommen wir ein „vernünftiges“ Ergebnis, wenn wir nicht die Normalverteilung annehmen. Dazu nehmen wir wieder an, die N_k seien unabhängig und es liege Poisson-Rauschen vor. Das heißt, die G_k sind unabhängig und haben eine **Poissonverteilung** mit dem Erwartungswert $(h * f)_k$. Eine Zufallsvariable X ist poissonverteilt, wenn:

$$P(X = l) = \frac{\lambda^l e^{-\lambda}}{l!} \quad (9.82)$$

gilt. λ ist dabei der Erwartungswert von X . Da wir von N ganzzahligen Grauwerten g_k ausgehen, ist diese Annahme durchaus möglich. Daher bilden wir wieder die Likelihood-Funktion:

$$L(f) = \prod_{k=0}^{N-1} \frac{(h * f)_k^{g_k} e^{-(h * f)_k}}{g_k!} \rightarrow \text{Maximum}. \quad (9.83)$$

Wir bilden wieder den Logarithmus

$$\begin{aligned} \ln(L(f)) &= \sum_{k=0}^{N-1} [g_k \ln(h * f)_k - (h * f)_k - \ln(g_k!)] \\ &= \sum_{k=0}^{N-1} \left[g_k \ln \left(\sum_{j=0}^{N-1} h_{k-j} f_j \right) - \sum_{j=0}^{N-1} h_{k-j} f_j - \ln(g_k!) \right]. \end{aligned} \quad (9.84)$$

Wir bilden wie üblich die partiellen Ableitungen

$$\frac{\partial \ln(L(F))}{\partial f_l} = \sum_{k=0}^{N-1} \frac{g_k}{\sum_{j=0}^{N-1} h_{k-j} f_j} h_{k-l} - \sum_{k=0}^{N-1} h_{k-l} = 0, \quad l = 0, \dots, N-1. \quad (9.85)$$

Diese Gleichung können wir kompakt als Faltung schreiben:

$$\sum_{k=0}^{N-1} \frac{g_k}{(h * f)_k} h_{l-k}^R = \left(\left(\frac{g}{h * f} \right) * h^R \right)_l = \sum_{k=0}^{N-1} h_{k-l}, \quad l = 0, \dots, N-1. \quad (9.86)$$

Die kompakte Schreibweise bedeutet immer eine pixelweise Operation. Diese nichtlineare Gleichung ist nun für f zu lösen. Dazu wird in der Literatur ein einfaches iteratives Vorgehen vorgeschlagen:

$$f_l^{(n+1)} = \frac{1}{(\sum_{k=0}^{N-1} h_{k-l})} f_l^{(n)} \left(\left(\frac{g}{h * f^{(n)}} \right) * h^R \right)_l, \quad l = 0, \dots, N-1, \quad n = 0, \dots \quad (9.87)$$

Diese Iteration wird Richardson-Lucy-Algorithmus genannt, siehe [58] und [43]. Es ist auffällig, dass dieser Algorithmus oft in der Astronomie verwendet wird. In der Iteration kann man die Nichtnegativität der Grauwerte $f_l \geq 0$ garantieren. Oft wird dieser Algorithmus auch als EM-Algorithmus mit Poisson-Statistik bezeichnet. In der Literatur gibt es eine Menge von Arbeiten, die sich mit Konvergenzuntersuchungen befassen und vergleiche zu den anderen Restaurationsmethoden anstellen. Der Vorteil ist, man braucht das Rauschen nicht extra modellieren und es zeigte sich, dass das Verfahren sehr robust gegenüber der Rauschstärke ist. Weiter ist bekannt, dass die Iteration sehr langsam konvergiert, daher ist der numerische Aufwand im Ortsraum beträchtlich. Als „Startbild“ $f^{(0)}$ kann g , aber auch ein „konstantes“ Bild, z. B. $f^{(0)} \equiv 1$, benutzt werden.

9.5.5 Bewegungsunschärfe (*motion blur*)

Wenn wir Bilder aufnehmen, in denen sich etwas bewegt und die Belichtungszeit nicht kurz genug ist, dann erhalten wir die allgemein bekannte Bewegungsunschärfe in den Bildern. Kann man nun diese Bilder nachträglich wieder scharf machen? Dazu müssen wir uns ein mathematisches Modell dieser Bewegungsunschärfe überlegen. Wir gehen zunächst davon aus, dass sich das gesamte Bild bewegt (und nicht nur Teile) und dass die Bewegung eine reine Translation des Bildes ist. Haben wir keine stetige Bewegung, sondern eine kurze abrupte Bewegung, so können wir diese als eine einzige Translation modellieren. Eindimensional ergibt sich dann das unscharfe Bild g :

$$g = f' + f = S_n f + f = f * (\delta + S_n \delta) = f * h = h * f. \quad (9.88)$$

Wir sehen, dies ist die gleiche Modellierung wie im Abschn. 4.22 zur Translationsberechnung mit Hilfe des Cepstrums. Da wir das scharfe Bild f berechnen wollen, haben wir eine inverse Faltung zu berechnen mit der PSF h . Da aber noch Rauschen und Störungen hinzukommen, müssen wir wie üblich das Modell (9.64) nutzen, wobei die Zufallsvariable N das Rauschen beschreibt. Bei dem obigen einfachen Modell einer einzigen Translation können wir die PSF tatsächlich mit der Cepstrum-Methode allein aus g berechnen. Unter Berücksichtigung einer Modellannahme für das Rauschen N , können wir dann die bisherigen Restaurationsmethoden (z. B. Wiener-Filter) anwenden.

Nun gehen wir einen Schritt weiter und nehmen an, die Bewegungsunschärfe ist nicht nur eine einzige Translation, sondern mehrere, wobei sich die Bilder additiv überlagern. Dazu nehmen wir der Einfachheit halber einmal an, die Translation erfolge nur horizontal oder nur vertikal, damit können wir wieder im Ortsraum direkt diskret rechnen und können die eindimensionale Schreibweise wieder benutzen. Wir bilden demzufolge:

$$\begin{aligned} g &= f + S_1 f + S_2 f + \dots + S_p f = f * (\delta + S_1 \delta + S_2 \delta + \dots + S_p \delta) \\ &= f * \left(\sum_{k=0}^p S_k \delta \right) = f * h = h * f. \end{aligned} \quad (9.89)$$

Darauf wenden wir das Faltungstheorem an:

$$\alpha_k(g) = \left(\sum_{l=0}^p e^{-2\pi i l \frac{k}{N}} \right) \alpha_k(f). \quad (9.90)$$

Damit können wir die Übertragungsfunktion zu $\alpha_k(h) = \sum_{l=0}^p e^{-2\pi i l \frac{k}{N}}$ ablesen.

Nun gehen wir zu Bewegungen in 2D-Bildern über, die nicht unbedingt nur horizontal oder vertikal erfolgen. Ziehen wir stetige Bewegungen in Betracht, dann müssen wir auch analoge Funktionen annehmen und über die Zeit integrieren:

$$g(x, y) = \int_0^T f(x - u(t), y - v(t)) dt = f(x, y) * \int_0^T (S_{u(t), v(t)} \delta) dt. \quad (9.91)$$

Wir verwenden wieder das Faltungstheorem oder das Verschiebungstheorem mit der IFT:

$$\begin{aligned} \alpha_g(v_1, v_2) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-2\pi i (v_1 x + v_2 y)} dx dy \\ &= \alpha_f(v_1, v_2) \int_0^T e^{-2\pi i (v_1 u(t) + v_2 v(t))} dt. \end{aligned} \quad (9.92)$$

Die Übertragungsfunktion $\alpha_h(v_1, v_2)$ von h lautet demnach:

$$\alpha_h(v_1, v_2) = \int_0^T e^{-2\pi i (v_1 u(t) + v_2 v(t))} dt \quad (9.93)$$

und ist durch die Bewegungstrajektorie $x = u(t), y = v(t), 0 \leq t \leq T$ vollständig bestimmt. Ein wichtiger Spezialfall ist sicher die geradlinig, gleichförmige Bewegung, wobei die Trajektorie ein Geradensegment darstellt, das im Koordinatenursprung beginnt. Wir wählen also für $u(t)$ und $v(t)$ einmal $u(t) = \frac{a \cdot t}{T}$ und $v(t) = \frac{b \cdot t}{T}$. Jetzt lautet die Übertragungsfunktion

$$\alpha_h(v_1, v_2) = \int_0^T e^{-2\pi i (v_1 \frac{at}{T} + v_2 \frac{bt}{T})} dt. \quad (9.94)$$

Dieses Integral können wir elementar ausrechnen, benutzen anschließend die Eulersche Formel und erhalten:

$$\begin{aligned} \alpha_h(v_1, v_2) &= \frac{T}{\pi(av_1 + bv_2)} \sin[\pi(av_1 + bv_2)] e^{-\pi i(av_1 + bv_2)} \\ &= T \operatorname{sinc}[\pi(v_1 a + v_2 b)] e^{-\pi i(av_1 + bv_2)}. \end{aligned} \quad (9.95)$$

Wir sehen phasenverschoben die „berühmte“ sinc-Funktion. Dies ist kein Wunder, da das Geradensegment als Rechteckfunktion deutbar ist. Die Phasenverschiebung benötigen wir eigentlich für die Restaurierung nicht. Wir verschieben das Trajektoriensegment um $(-a/2, -b/2)$ und wenden das Verschiebungstheorem auf die verschobene Funktion an und erhalten:

$$\alpha_h(v_1, v_2) = \frac{T}{\pi(av_1 + bv_2)} \sin[\pi(av_1 + bv_2)] = T \operatorname{sinc}[\pi(v_1 a + v_2 b)]. \quad (9.96)$$

Die Begründung ist recht einfach: Welches Bild wir aus der Serie der verschobenen Originalbilder restaurieren wollen, ist doch völlig egal. Deshalb legen wir den Koordinatenursprung in die Mitte des Geradensegmentes. Diese sinc-Funktion ist die Fouriertransformierte der Rechteckfunktion, die hier die Bewegungstrajektorie als Geradensegment beschreibt. Die Übertragungsfunktion (9.95) ist durch a, b und T vollständig bestimmt. In Abb. 9.2a ist ein künstlich erzeugtes, bewegungsunscharfes Bild zu sehen, wobei die Unschärfe horizontal mit 20 Pixeln erzeugt wurde. Das additiv überlagerte Bild wurde „exakt“ in der Gleitkommaarithmetik berechnet. In der Abb. 9.2b ist das Ergebnis der exakten inversen Faltung zu sehen, die Restaurierung scheint gelungen zu sein. Nun wurde das unscharfe Bild erzeugt, indem nach wie vor die verschobenen Bilder addiert werden, es wurde aber anschließend der Mittelwert pro Pixel gebildet und auf ganze Zahlen gerundet, d. h. optisch ist dieser Effekt im Bild nicht zu sehen, da es nur minimales Quantisierungsrauschen darstellt. Nun ist in Abb. 9.3a wieder das Ergebnis der inversen Faltung zu sehen, dieses ist völlig unbrauchbar. Benutzt man dagegen eine Restaurationsmethode, die das Rauschen berücksichtigt, dann ist das Ergebnis wieder brauchbar, siehe Abb. 9.3b. Das inverse Problem ist also numerisch instabil und bedarf auf jeden Fall der Regularisierung. In den Bildern der Abb. 9.4 sind die Restaurierungsergebnisse mit dem Wiener-Filter und

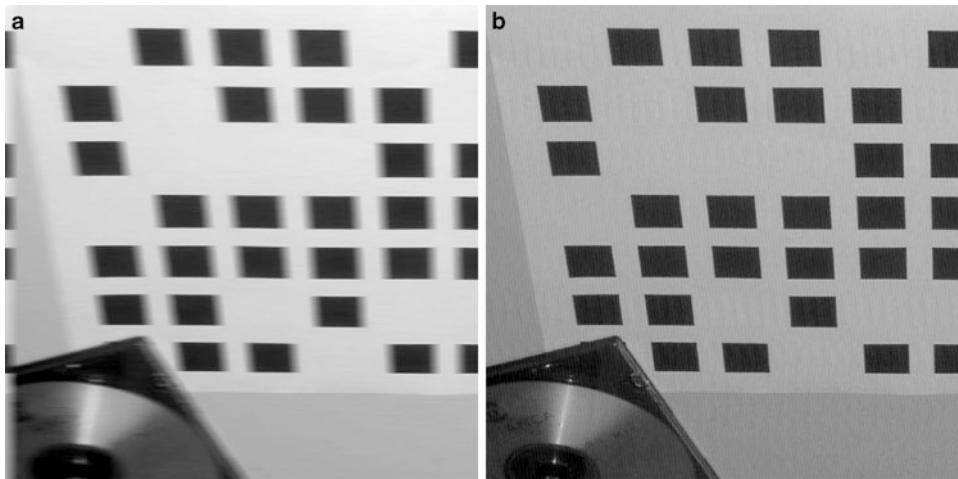


Abb. 9.2 **a** Simuliertes Bild mit Bewegungsunschärfe: Das Originalbild wurde mehrmals verschoben und die verschobenen Bilder wurden mittels Gleitkommaarithmetik addiert und ergeben das Gleitkommabild motion-floating-point-number. In der Abbildung ist die Visualisierung des Bildes motion-floating-point-number zu sehen (motion-visual). **b** Restaurationsmethode: die inverse Faltung, angewendet auf das Gleitkommabild motion-floating-point-number aus **a**, liefert ein brauchbares Ergebnis

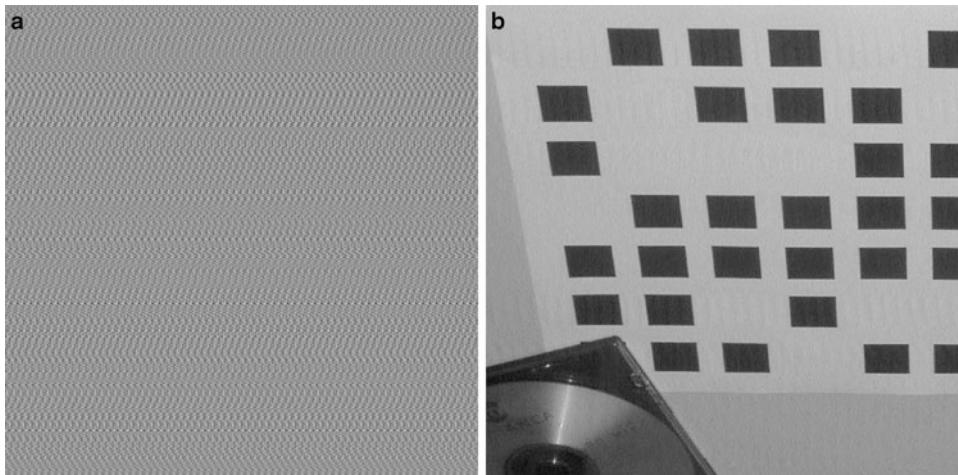


Abb. 9.3 **a** Restaurationsmethode: die inverse Faltung, angewendet auf das visualisierte Bild motion-visual aus Abb. 9.2, liefert nun ein völlig unbrauchbares Ergebnis. Das Bild motion-visual unterscheidet sich von motion-floating-point-number lediglich durch minimales Quantisierungsrauschen. **b** Restaurationsmethode: das Wiener-Filter, angewendet auf das visualisierte Bild motion-visual aus Abb. 9.2, liefert dagegen ein brauchbares Ergebnis

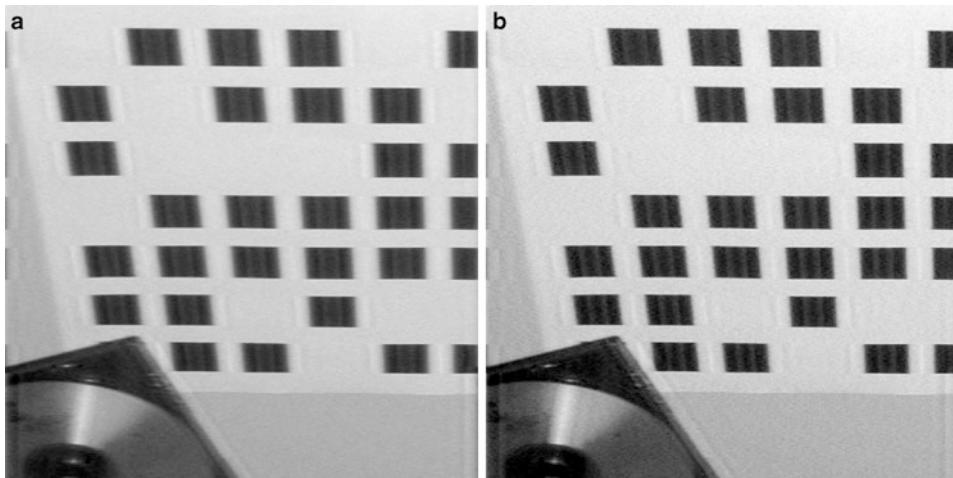


Abb. 9.4 **a** Restaurationsmethode: das Wiener-Filter wurde angewendet auf das Bild motion-visual aus Abb. 9.2, welches zusätzlich noch stark verrauscht wurde. Das Restaurierungsergebnis ist immer noch brauchbar. **b** Restaurationsmethode: der Richardson Lucy Algorithmus wurde angewendet auf das Bild motion-visual aus Abb. 9.2, welches zusätzlich noch stark verrauscht wurde. Das Restaurierungsergebnis ist ähnlich dem Wiener Filter

dem Richardson-Lucy-Algorithmus zu sehen, wobei das unscharfe Bild mit einem unabhängigen Gaußschen Rauschen ($\sigma = 5$) gestört wurde. Die Restaurierungsergebnisse sind ähnlich, man muss aber beim Wiener-Filter genau die Rauschstärke kennen. Im Gegensatz dazu darf man beim Richardson-Lucy-Algorithmus nicht zu lange iterieren, man braucht aber vorher die Rauschstärke nicht kennen.

Man sieht in der Abb. 9.5, dass die Übertragungsfunktion Nullstellen an parallelen Geraden besitzt. Diese Geraden haben die orthogonale Orientierung zur Bewegungstrajektorie. Da also die Übertragungsfunktion Nullstellen besitzt, ist es nicht möglich, das Original f exakt zu rekonstruieren, es werden Frequenzen ausgelöscht. Die Restaurierung muss sich also „rein theoretisch“ schon darauf beschränken, das Original möglichst gut zu approximieren.

Bevor die Bewegungsunschärfe durch eine Restaurationsmethode beseitigt wird (besser: reduziert wird), müssen wir „irgendwie“ a , b und T bestimmen. Wenn sich z. B. nur Teile in dem Bild bewegen, dann können wir nur die Unschärfe dieser Bildteile beseitigen. Wichtig ist, die Bewegung muss als Translation (zumindestens näherungsweise) modellierbar sein, und lokal ist das sehr häufig möglich.

Da wir die Übertragungsfunktion (9.95) von der Grundstruktur her kennen, d. h. dies ist die periodische sinc-Funktion, so müssen diese periodischen Nullstellen auch deutlich

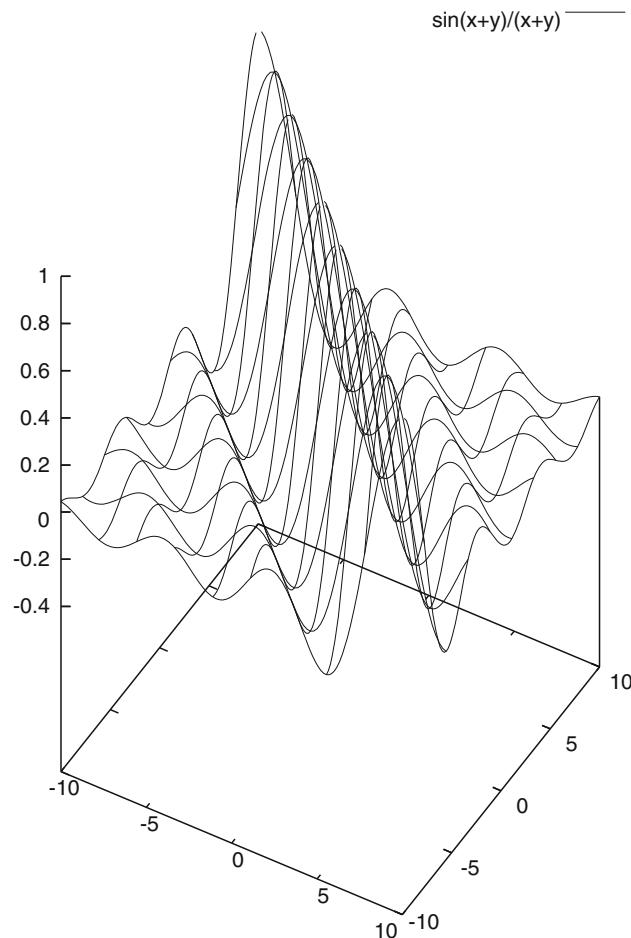


Abb. 9.5 sinc-Funktion

im Spektrum des bewegungsunscharfen Bildes zu sehen sein. In den Bildern der Abb. 9.6 sind die Amplitudenspektren der PSF und des bewegungsunscharfen Bildes zu sehen. Man „sieht“ die typische Streifenstruktur bedingt durch die sinc-Funktion. Die Breite der Streifen verhält sich dabei umgekehrt proportional zu der Länge der Bewegungstrajektorie, d. h. je länger die Bewegungstrajektorie ist, umso schmäler sind die Streifen. Die Orientierung der Streifen ist genau orthogonal zur Orientierung der Bewegungstrajektorie. Demzufolge könnte man die Bewegungstrajektorie und damit die PSF aus dem Spektrum des bewegungsunscharfen Bildes ermitteln. Man könnte auf das Amplituden-Spektrum die Radon-Transformation anwenden und daraus die Orientierung und die Breite ermitteln. Bei sehr kurzen Trajektorien sind die Streifen zu breit, daher gibt es zu wenige parallele „Nullstellen“-Geraden im Radon-Bild und die Bestimmung wird weitaus schwieriger.

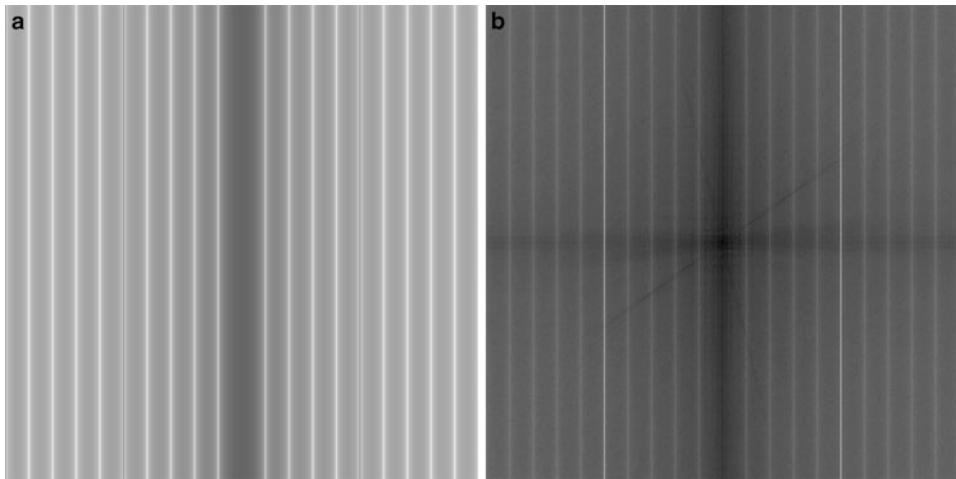


Abb. 9.6 **a** Amplitudenspektrum der PSF des Bildes aus Abb. 9.2a, dies entspricht dem Amplitudenspektrum der horizontal ausgerichteten sinc-Funktion. **b** Amplitudenspektrum des bewegungsunscharfen Bildes aus Abb. 9.2a

9.5.6 Wellenfront Kodierung (*wavefront coding*)

Bei einem „normalen“ optischen System ist auf Grund der Blendenöffnung die Tiefenschärfe begrenzt, was oft störend ist. Eine neue Idee ist in [13] *wavefront coding* genannt worden. Diese Idee besteht darin, das optische System so zu modifizieren, dass wir ein Bild einer gewissen Schärfe erhalten, wobei diese Schärfe nahezu unabhängig von der Fokussierung sein soll. Der Ausdruck Schärfe ist dann aber nicht richtig, es muss dann eher Unschärfe heißen. Anders ausgedrückt heißt dies, die PSF ist unabhängig (nahezu) von der Tiefenschärfe. Wenn diese PSF dann bezüglich der Faltung noch invertierbar ist, dann könnte man mit Restaurationsmethoden das echt scharfe Bild ausrechnen. Da man zeigen kann, dass die OTF (siehe Abschn. 4.1) tatsächlich keine Nullstellen hat, ist die PSF auch invertierbar. Natürlich ist die Unabhängigkeit von der Tiefenschärfe nicht unbegrenzt, aber ein relativ großer Bereich wird schon abgedeckt. Bei der praktischen Realisierung in einem optischen System wird in die Apertur eine Art „kleine Linse“ eingesetzt, man sagt eine *refraktiv wirkende kubische Phasenplatte*, siehe auch [13]. Weiterhin benötigen wir einen Signalprozessor, auf dem eine Bildrestaurationsmethode implementiert ist. Daher ordnet sich *wavefront coding* in das Gebiet der *computational photography* ein.

9.5.7 Kodierte Apertur (*coded aperture imaging*)

Eine klassische *pinhole camera* besitzt kein optisches System. Sie hat eine „unendlich kleine“ Blende und dies bedingt eine „unendlich große“ Tiefenschärfe. Der Vorteil der großen

Tiefenschärfe wird aber zunichte gemacht, weil durch diese kleine Blende kein Licht zu den Sensoren kommen kann. Dadurch ist diese klassische Kamera praktisch unbrauchbar, wir benötigen ein optisches System, welches größere Blenden erlaubt. Nun gibt es außer dem sichtbaren Licht noch energiereiche Strahlen, bei denen solch ein optisches System nicht funktioniert, z. B. Röntgenstrahlen, Gammastrahlen usw., sodass man wieder auf die klassische *pinhole camera* zurückgreifen muss. Dadurch entwickelte sich seit ca. 1965 im Zusammenhang mit der kosmischen Röntgenstrahlung eine modifizierte *pinhole camera*. Die Grundidee ist recht einfach:

- Wir installieren in eine Kamera mehrere *pinhole cameras*, d. h. unsere neue Kamera erhält eine Blende mit vielen „unendlich kleinen“ Löchern, den eigentlichen *pinhole cameras*. Es müssen so viele Löcher sein, damit genügend Licht in die neue Kamera eindringen kann.
- Der Sensor erhält dann als Bild die additive Überlagung aller Bilder der eigentlichen *pinhole cameras*, allerdings sind die Einzelbilder alle zueinander verschoben, da die kleinen Löcher versetzt in der Blende eingebracht wurden.
- Nun muss man dieses unscharfe Bild mit den Restaurationsmethoden wieder „scharf“ machen, wir benötigen also einen Prozessor dazu. Daher kann man die *coded aperture imaging* als Vorläufer der *computational photography* ansehen.
- Die Blende der Kamera können wir uns als Matrix vorstellen, wobei jedes Matrixelement entweder Null oder Eins ist, d. h. Licht wird durchgelassen (Loch) oder gesperrt. Wie solch eine Matrix (z. B. URA, MURA) aufzubauen ist, kann man z. B. in [24] nachlesen. Diese Matrix ist auch als diskrete Funktion mit Nullen und Einsen auffassbar. Diese ist nun bezüglich der inversen Faltung die PSF und muss numerisch stabil invertierbar sein. Siehe dazu auch in Abschn. 2.1.2 die Beispiele bezüglich der inversen Korrelation und Faltung.

Die Bildsegmentierung ist wohl eines der wichtigsten Gebiete der Bildverarbeitung. Wenn Bilder analysiert bzw. Szenen klassifiziert werden, setzt das in der Regel eine korrekte Segmentierung voraus. Klassische Bildverarbeitungsalgorithmen laufen oft in folgenden Schritten ab:

- a) Daten- oder Bildeingabe,
- b) Vorverarbeitung der Bilder mit Bildverbesserungsalgorithmen, z. B. diverse Filter zur Rauschunterdrückung, Filter zur Beleuchtungskorrektur, Verstärkung gewisser Eigenschaften, Detektion von „interessierenden“ Pixeln und vieles mehr,
- c) Segmentierung der Bilder in Regionen, Objekte mit geschlossenen Konturen oder Liniensegmente.
- d) Klassifikation und/oder Analyse der segmentierten Regionen, Objekte oder Liniensegmente.
- e) „Ausgabe“ der Analyseergebnisse.

Die Trennung der Segmentierung (Punkt c) von der anschließenden Klassifikation (Punkt d) ist genaugenommen so streng gar nicht möglich. Bei der Segmentierung klassifiziert man gewöhnlich schon etwas, ohne sich dessen bewusst zu sein. Dazu ein

Beispiel Ein OCR-Algorithmus soll auf einem Personalausweis die beiden ICAO-Zeilen lesen. Diese Zeilen enthalten zum Beispiel den Namen des Inhabers. Dazu wird man den Ausweis segmentieren und eine Liste von denjenigen segmentierten Objekten bestimmen, welche die Buchstaben/Sonderzeichen der ICAO-Zeilen sein könnten, um sie einer anschließenden Klassifikation zu unterwerfen. Da man aber gewisse Informationen über die Zeichen der ICAO-Zeilen hat, wird man gleich bei der Segmentierung die Objekte eliminieren, die keine Zeichen der ICAO-Zeilen sein können. Kriterien sind dazu z. B.: die Fläche ist zu groß, die Fläche ist zu klein, der Schwerpunkt liegt in einem völlig anderen Be-

reich des Ausweises usw. Die Verwendung dieser Kriterien kann als eine „festverdrahtete“ Klassifikation interpretiert werden und ist sehr von der Anwendung abhängig.

In aktuellen Ansätzen versucht man daher die Kriterien automatisch aus gegebenen Lerndaten zu ermitteln. Im Bereich der Segmentierung führt dies zum Gebiet der *semantischen Segmentierung*, bei der Segmentierung und semantisches Wissen (anhand von Lerndaten) gleich in einem Schritt vereint werden. In diesem Zusammenhang spricht man auch von der klassischen Segmentierung ohne Lernschritt als unüberwachte Segmentierung. Entscheidend für den Erfolg der Verfahren der semantischen Segmentierung sind geeignete annotierte Lerndaten. Sind diese bei einer Aufgabenstellung nicht verfügbar, so müssen Kritierien für die Segmentierung und dementsprechende Algorithmen manuell für eine Aufgabenstellung angepasst werden. Oft finden sich in der Praxis Mischformen, bei denen nur ein Teil der Parameter aus Lerndaten ermittelt werden.

In den folgenden Abschnitten sollen diejenigen Segmentierungsmethoden vorgestellt werden mit denen die Autoren erfolgreich gearbeitet haben. Da gewisse klassische Segmentierungsmethoden in vielen Büchern der Bildverarbeitung behandelt werden, soll hier auf die Darstellung dieser Algorithmen verzichtet werden. Dies sind insbesondere die *Split and Merge*-Strategien und die Wasserscheidentransformation.

10.1 Thresholding

Die einfachsten Segmentierungsmethoden beruhen auf globalen/lokalen Schwellwerten. Durch Berechnung oder Vorgabe einer Schwelle T wird ein Grauwertbild $f(x, y)$ in ein Binärbild $b(x, y)$ überführt:

$$b(x, y) = \begin{cases} 1, & \text{falls } f(x, y) \geq T \\ 0, & \text{falls } f(x, y) < T. \end{cases} \quad (10.1)$$

Legt man der Binarisierung das gesamte Bild zur Berechnung der Schwelle zugrunde, so redet man von globalem *thresholding*, ansonsten von lokalem *thresholding*. So einfach (10.1) im mathematischen Sinne ist, so schwierig ist es, die Schwelle T optimal für die jeweilige Anwendung zu berechnen. Es gibt simple iterative Verfahren, die meisten Verfahren berechnen die Schwelle nach einem Zielkriterium. Die Zielkriterien basieren oft auf dem Grauerhistogramm. Wenn das Grauerhistogramm typisch bimodal ist, dann ist es „anschaulich“ klar, dass man die Schwelle T genau in das Histogrammtal zwischen den beiden Gipfeln legt. In der Praxis sind die Histogramme aber nicht typisch bimodal, so dass man das Zielkriterium exakt mathematisch formulieren muss. Gibt man eine Schranke T vor, dann bezeichnen wir alle Pixel deren Grauwerte kleiner T sind als Hintergrund B , alle anderen Pixel als Vordergrund O oder als Objektpixel. Damit haben wir alle Pixel in zwei Klassen eingeteilt, wobei die Einteilung von der gewählten Schranke T abhängt. Nun bedienen wir uns einiger Hilfsmittel aus der Mustererkennung und zwar sogenannter Trennmaße von Klassen. Diese drücken die Separierbarkeit von Klassen aus, d. h. je größer

diese sind, umso besser lassen sich die Klassen unterscheiden. Es seien $P(B)$ und $P(O)$ die A-priori-Wahrscheinlichkeiten von Hintergrund und Vordergrund, $E(G|B)$ der Erwartungswert der Grauwerte des Hintergrundes, entsprechend sei $E(G|O)$ definiert. Weiter seien $\sigma^2(G|B)$ und $\sigma^2(G|O)$ die Varianzen der Grauwerte des Hintergrundes bzw. des Vordergrundes. Dann kann man ein Trennmaß der beiden Klassen Hintergrund und Vordergrund angeben:

$$D(T) = \frac{P(B)P(O)[E(G|B) - E(G|O)]^2}{P(B)\sigma^2(G|B) + P(O)\sigma^2(G|O)}. \quad (10.2)$$

Die A-priori-Wahrscheinlichkeiten $P(O)$ und $P(B)$ berechnen sich zu:

$$P(B) = \sum_{g=0}^{T-1} P(G=g), \quad P(O) = \sum_{g=T}^{g_{\max}} P(G=g), \quad P(B) + P(O) = 1. \quad (10.3)$$

Alle eingehenden Wahrscheinlichkeiten lassen sich simpel aus dem Grauerthistogramm berechnen, wobei $P(G=g|B) = \frac{P(G=g)}{P(B)}$ zu beachten ist. Nun ist diejenige Schranke T zu berechnen, für die das Trennmaß $D(T)$ aus (10.2) maximal ist. Die Methode ist weit verbreitet und wird als *Otsu thresholding method* (Otsu 1979) bezeichnet. Man kann natürlich noch andere Kostenfunktionen aufstellen, z. B. kann man Entropiemaße benutzen, siehe Abschn. 9.1.

10.2 Konturfolgeverfahren bei binärer Quantisierung

Das klassische Konturfolgeverfahren ist wohl das bekannteste und wichtigste elementare Segmentierungsverfahren. Ausgehend von einem Binärbild oder einem Grauwertbild mit berechneter Binarisierungsschwellen wird zunächst ein Randpunkt eines Objektes gesucht. Der Objektzusammenhang wird stets über die Vierernachbarschaft betrachtet. Ein Objektpunkt ist ein Randpunkt, wenn er in seiner Vierernachbarschaft mindestens einen Untergrundpunkt besitzt. Betrachtet man das Quadratgitter als orientierten Nachbarschaftsgraphen, dann ist die Kontur dasselbe wie eine Randmasche des Graphen. In der praktischen Bildverarbeitung macht man dies nicht, man definiert die Kontur als die geordnete Folge von Randpunkten des Objektes. Eine Randmasche enthält im Gegensatz dazu auch Objektpunkte, die keine Randpunkte sind. Um die Randpunkte in geordneter Reihenfolge zu finden, ist es wichtig, die sogenannten Randkanten zu betrachten. Eine Randkante ist ein Paar von Pixeln, die „viererbenachbart“ sind und das eine Pixel ein Randpixel und das andere ein Untergrundpixel ist, siehe Abb. 10.1a. Ausgehend von einem Randpunkt verfolgt man nun im mathematisch positivem Sinne die Randkanten und zählt dabei die Randpunkte auf und nicht die Randkanten selbst. Wird ein Randpunkt mehrfach hintereinander durch mehrere Randkanten erkannt, so wird er nur einmal aufgezählt. Wenn man die Punkte außerhalb des Bildes als Untergrundpunkte auffasst, so sind die Konturen stets geschlossen. Für Konturen werden oft spezielle Klassen oder Datenstrukturen

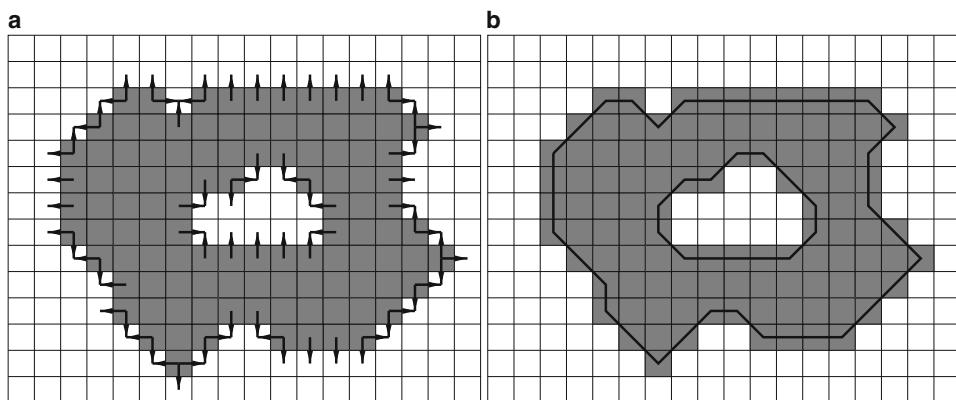


Abb. 10.1 a Objekt mit Randkanten bezüglich der Vierernachbarschaft. b Aus der Folge der Randkanten aus a wurde eine Innen- und eine Außenkontur berechnet

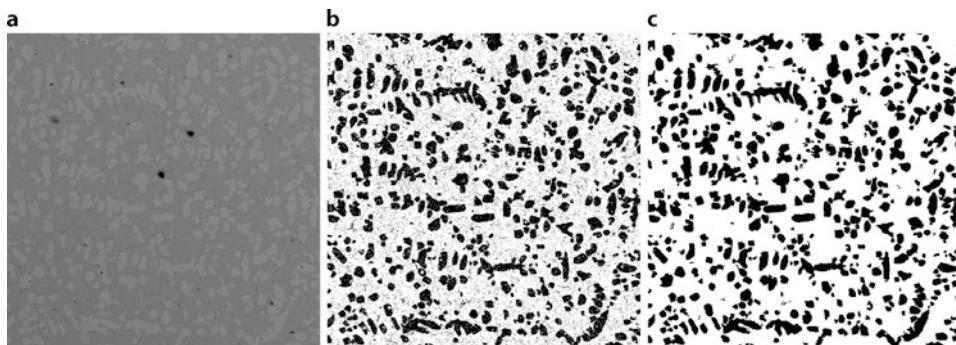


Abb. 10.2 a Originalbild: Anschliff einer Metalloberfläche. b Ergebnis: Globale Binarisierung nach Otsu, helle Stellen in a sind jetzt schwarz dargestellt. c Mit dem Konturfolgeverfahren wurden in b Objekte einer bestimmten Mindestgröße detektiert

gewählt. Es liegt z. B. nahe, nur den Startpunkt abzuspeichern und dann nur noch die Richtungskodes (Freeman-Kodes) Null bis Sieben, d. h. die Randpunkte der Kontur hängen formal über die Achternachbarschaft zusammen. Damit kann man eine Kontur als ein Polygon auffassen, wobei die Eckpunkte des Polygons die aufgezählten Randpunkte der Kontur darstellen. Da man beim Konturfolgeverfahren parallel viele Merkmale des Objektes berechnen kann (siehe Abschn. 19.7.1), so kann man an Hand der vorzeichenbehafteten Fläche des Objektes entscheiden, ob eine Außen- oder Innenkontur vorliegt, siehe Abb. 10.1b. In Abb. 10.2a ist die Mikroskopaufnahme einer Metalloberfläche zu sehen. Aufgabe ist es, die Fläche aller „hellen“ Objekte zu bestimmen. Diese Aufgabe scheint zunächst schwierig, erweist sich aber mit der Otsu-Binarisierung und dem Konturfolgeverfahren als recht simpel lösbar. Das Bild wird zunächst vorverarbeitet durch eine Histogrammausgleichung, dann wird eine Shadingkorrektur durchgeführt und schließlich nach Otsu

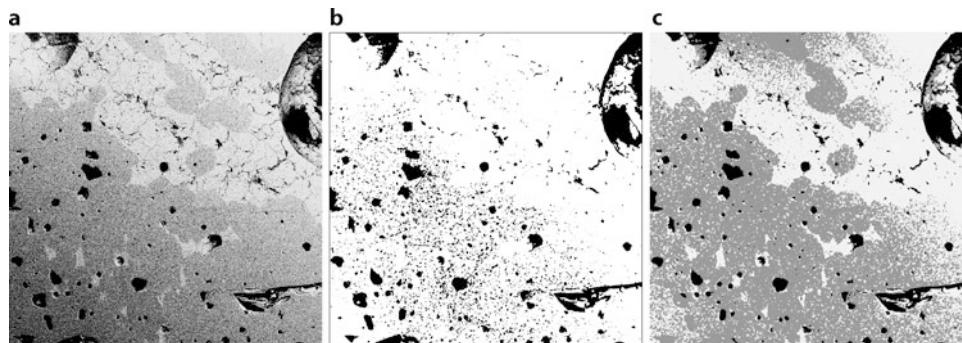


Abb. 10.3 **a** Originalbild: Anschliff einer Metalloberfläche mit Shadingeffekt, **b** Ergebnis: Problematische globale Binarisierung des Bildes aus **a** nach Otsu, **c** Besseres Ergebnis: Globale ternäre Quantisierung des Bildes aus **a** nach Otsu

binarisiert, das Ergebnis ist in Abb. 10.2b zu sehen. Die größeren Objekte haben Löcher und sind etwas zerklüftet, und es gibt jede Menge von „Rauschobjekten“. Mit dem Konturfolgeverfahren werden nun alle äußeren Konturen und deren Fläche bestimmt. Objekte mit einer „sehr kleinen“ Fläche werden eliminiert. In Abb. 10.2c sind alle schwarzen Pixel Objektpunkte und bestimmen somit die Gesamtfläche. In Abb. 10.3a ist noch ein etwas schwierigeres Segmentierungsproblem zu lösen. Zunächst ist das Shading auffällig, welches zu korrigieren ist. Eine Binarisierung zur Bestimmung der schwarzen Objekte gelingt nun nicht mehr so schön nach Otsu, siehe Abb. 10.3b. In der Abb. 10.3a ist aber eine typische Quantisierung des Bildes in grob drei Klassen zu erkennen. Wenn man dies als A-priori-Wissen kennt, dann kann man eine ternäre Einteilung mit zwei Schwellwerten ähnlich dem Otsuverfahren anwenden. Das Ergebnis ist in Abb. 10.3c zu sehen.

10.3 Konturfolgeverfahren bei lokaler ternärer Quantisierung

Das Konturfolgeverfahren mit lokaler ternärer Quantisierung basiert im Wesentlichen auf Ideen von Dr. Ortmann. Das Konturfolgeverfahren aus Abschn. 10.2 basiert auf einer vorherigen globalen Schwellwertberechnung. Alle Pixel des Bildes werden als Objekt- oder Hintergrundpixel klassifiziert und zwar in Abhängigkeit von einer globalen Schwelle, die z. B. nach Otsu berechnet wurde. Diese Methode versagt aber dann, wenn die Beleuchtung des Bildes stark ortsabhängig ist oder wenn ein Objekt selbst verschiedene Grauwerte besitzt. Diese Nachteile versucht man mit lokalen, adaptiven Methoden zu umgehen, wobei natürlich wieder andere Schwierigkeiten entstehen. Lokale Schwellen werden bestimmt durch eine Untersuchung der Umgebung $\mathcal{U}(x)$ eines Pixels x . Wenn aber alle Pixel aus der Umgebung zu einer Klasse gehören, Objekt oder Untergrund, dann ist keine binäre Entscheidung zwischen Objektpixel oder Untergrundpixel mehr möglich. Daher ist es sinnvoll, eine dritte Klasse von Pixeln einzuführen. Daher legen wir jetzt drei Label für ein Pixel fest:

```

1 Calculate  $g_{\max}$  and  $g_{\min}$ 
2 For each pixel  $\mathbf{x}$ :
    2.1 If  $g_{\max}(\mathbf{x}) - g_{\min}(\mathbf{x}) < \gamma$  then
        2.2 label point as unknown
    2.3 else
        2.4  $T = \frac{1}{2} (g_{\max}(\mathbf{x}) + g_{\min}(\mathbf{x}))$ 
        2.5 If  $g(\mathbf{x}) > T$  then label point as object
        2.6 else label point as background

```

Abb. 10.4 Pseudocode für die lokale ternäre Quantisierung

object, *background* und *unknown*. Wir müssen dann aber insbesondere für das Konturfolgeverfahren festlegen, wie diese *unknown* Pixel zu behandeln sind. Zur Berechnung der lokalen Schwelle könnte man alle bereits bekannten Methoden zur globalen Schwellwertberechnung anwenden, diese können aber für kleine Regionen ungeeignet sein. Weiterhin muss eine Schwelle für jedes Pixel berechnet werden, was zu langen Rechenzeiten führen würde. Daher muss man eine gesonderte und besonders effiziente Schwellwertberechnung benutzen. Eine geeignete Möglichkeit ist Berechnung des Maximums und Minimums der Grauwerte aus der Umgebung:

$$g_{\min}(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{U}(\mathbf{x})} g(\mathbf{y}), \quad g_{\max}(\mathbf{x}) = \max_{\mathbf{y} \in \mathcal{U}(\mathbf{x})} g(\mathbf{y}). \quad (10.4)$$

Die Berechnung dieser beiden Größen lässt sich auf Grund der Separierbarkeit der Masken effektiv implementieren. Im Falle, dass die Differenz zwischen Maximum und Minimum klein ist, gehören alle Pixel aus $\mathcal{U}(\mathbf{x})$ zur gleichen Klasse und deshalb interpretieren wir kleine Differenzen als die Folge von „Rauschen der Grauwerte“. Der Pseudokode dieses Vorgehens ist in Abb. 10.4 zu sehen. Es wird folglich das Grauwertbild nicht binarisiert, sondern ternär quantisiert mit drei Labeln.

Die lokale Quantisierung zeigt einige Besonderheiten:

- a) Kleine Objekte oder kleine Hintergrundbereiche werden ohne Probleme wie üblich detektiert. (Ein Objekt oder Hintergrundbereich nennen wir „klein“, wenn für jedes Pixel des Objektes in der Nachbarschaft mindestens ein Untergrundpixel liegt oder umgekehrt.)
- b) Die Detektion von großen Objekten mit hohem Kontrast ist nahezu perfekt, wobei innere Pixel des Objektes, die weit vom Rand entfernt liegen, als *unknown* markiert werden.
- c) Möglicherweise können Objekte nur teilweise detektiert werden, wenn ein Teil einen guten Kontrast zum Hintergrund hat und der andere Teil einen schlechten Kontrast.

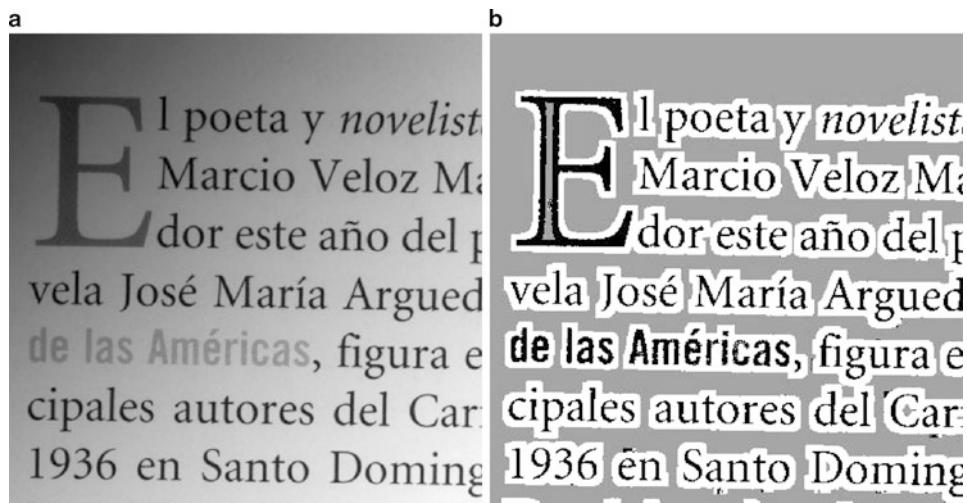


Abb. 10.5 a Originalbild: Ungleiche Ausleuchtung und Schrift mit verschiedenem Grauwertlevel. b Lokale ternäre Quantisierung des Bildes aus a. Objekt: schwarz, Untergrund: weiß, undefiniert: grau. Mit diesem Bild kann das Konturfolgeverfahren erfolgreich durchgeführt werden

Bei der Kontursuche in Binärbildern finden wir immer geschlossene Konturen, zumindest wenn wir die Pixel außerhalb des Bildes als Untergrundpunkte auffassen. Im Falle der Kontursuche auf ternären Bildern muss die Kontursuche stoppen, wenn sie auf Pixel mit „unknown“ trifft. Infolgedessen können die Konturen jetzt beliebig geschlossen oder offen sein. Dies bedeutet nun:

- In den Fällen a) und b) ist die Kontur geschlossen. Es bedarf keiner weiteren Überlegung.
- Im Fall c) können eine oder mehrere Konturen mit einem Objekt im Zusammenhang stehen. Diese Konturen müssen entweder aus der Segmentierung ausgeschlossen werden oder sie müssen korrigiert werden. Wenn z. B. nur ein kleiner Abstand zwischen Startpixel und Endpixel besteht, könnten diese einfach verbunden werden.
- Pixel außerhalb des Bildes könnte man als „unknown“ initialisieren. Das bedeutet, wenn Objekte den Bildrand berühren, werden deren Konturen nicht geschlossen. Andererseits kann dann aber der Nutzer entscheiden, ob diese geschlossen werden sollen oder nicht.

In Abb. 10.5a ist ein ungleichmäßig ausgeleuchtetes Bild mit Schriftzeichen zu sehen. So gar die Schrift besitzt unterschiedliche Grauwertlevel. In Abb. 10.5b ist die ternäre Bildquantisierung zu sehen. Dabei bedeutet: schwarz = *object*, weiß = *background* und grau = *unknown*. Man sieht, dass das Konturfolgeverfahren basierend auf dieser ternären Quantisierung alle Zeichen korrekt segmentieren wird.

10.4 Konturfolgeverfahren bei beliebigen Punktmengen

Wenn man nur typische, zunächst zusammenhangslose Punkte von Objekten detektieren kann, dann kann man das klassische Konturfolgeverfahren bezüglich der Vierernachbarschaft nicht anwenden. In der Abb. 10.6b ist eine Punktmenge zu sehen. Unser Auge integriert und glaubt Außen- und Innenkonturen der Punktmenge zu erkennen. Zunächst müssen wir dafür eine wesentliche Strukturkonstante *limit* definieren, die den Objektzusammenhang beschreibt. Wenn die Distanz eines Punktes zu einem anderen Punkt kleiner als *limit* ist, dann sind die beiden Punkte benachbart. Um „vernünftige“ Silhouetten mittels

Abb. 10.6 **a** Zufällig erzeugte Punktmenge P , die unserem Auge eine Silhouette suggeriert. **b** Auf die Punktmenge aus **a** wurde die parametrisierte Delaunay-Triangulation angewendet. Um die LKHD zu berechnen, müssen die Dreiecke noch gefüllt werden

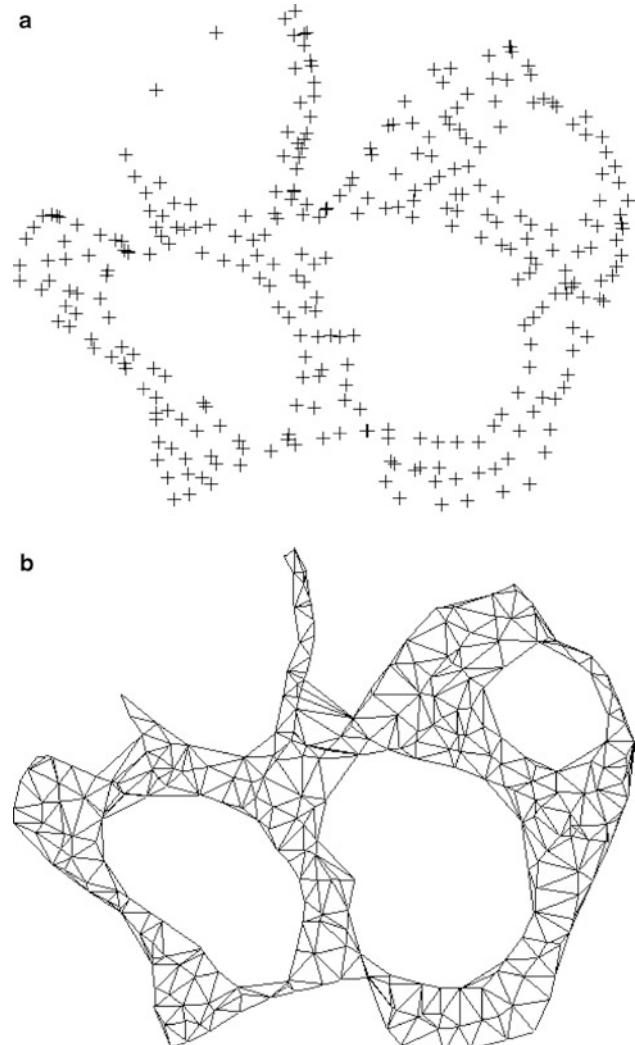
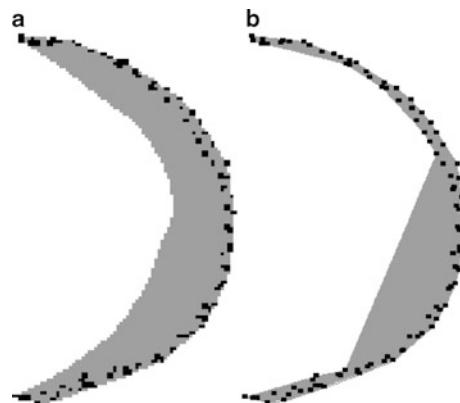


Abb. 10.7 a LKH: (grau) der Punktmenze (schwarze Punkte), b LKHD: (grau) Delaunay-Triangulation der gleichen Punktmenze aus a



des Konturfolgeverfahrens zu bestimmen, gibt es sicher viele Möglichkeiten. Eine davon ist die morphologische Operation Dilation, siehe Abschn. 8.2.4. Dabei hat das Strukturelement die Größe eines quadratischen Fensters der Seitenlänge *limit*. Man trägt nun die Punktmenze als Menge von schwarzen Punkten in ein Binärbild ein und dilatiert dieses. Anschließend bestimmt man mit dem Konturfolgeverfahren alle Außen- und Innenkonturen. Die Methode hat aber den entscheidenden Nachteil, dass bei großen Strukturelementen die Konturen zu „eckig“ werden und deshalb nicht zu gebrauchen sind. Dieser Effekt ändert sich auch nicht, wenn man andere Strukturelemente wie z. B. Kreise wählt.

Deshalb bietet sich jetzt an, die in Abschn. 1.2.4 eingeführte limitierte, konvexe Hülle (LKH) zu benutzen. In diesem Abschnitt wurde auch der Begriff der Silhouette einer Punktmenze bezüglich *limit* eingeführt. Wenn man die LKH direkt berechnet, müssen wir alle Tripel von Punkten auswählen, daher führt dies auf einen $O(n^3)$ -Algorithmus. Dieser ist folglich in der Praxis viel zu langsam. Eine Approximation der LKH ist die Anwendung einer modifizierten Delaunay-Triangulation, siehe auch Abschn. 10.7.3. Diese bezeichnen wir jetzt mit $LKHD(P)$. Zunächst führt man die Delaunay-Triangulation mit einem Standardalgorithmus durch und streicht dann alle diejenigen Dreiecke, bei denen mindestens eine Seite größer als *limit* ist. In Abb. 10.6 ist dies deutlich zu erkennen. Dann füllt man diese Dreiecke im Binärbild und bestimmt mittels des klassischen Konturfolgeverfahrens alle Außen- und Innenkonturen. Die Menge dieser Konturen nennen wir die Silhouette der Punktmenze. Der Parameter *limit* steuert diese modifizierte Delaunay-Triangulation. Kleine Werte erzeugen viele Einzelobjekte. Je größer *limit* wird, desto mehr nähern wir uns der konvexen Hülle der Punktmenze. Die $LKHD(P)$ verhält sich folglich ähnlich der $LKH(P)$. Die $LKHD$ reicht als Näherungslösung in der Praxis völlig aus, ist aber verschieden von der LKH . Die „Stetigkeit des Glättungseffektes“ ist bei der $LKHD$ nicht so gut wie bei der LKH , insbesondere bei größeren Werten von *limit*. In Abb. 10.7 repräsentieren die schwarzen Punkte die Menge P , wobei extrem eine „Konkavität“ dargestellt ist. Man sieht deutlich den Unterschied zwischen der LKH und der $LKHD$. In Abb. 10.8a ist eine Nervenzelle mit Zellkern zu sehen. Man möchte die Silhouette mit Zellkern bestimmen. Dazu bestimmt man zunächst typische Punkte der Zelle, selbst eine globale Binarisierung würde funktio-

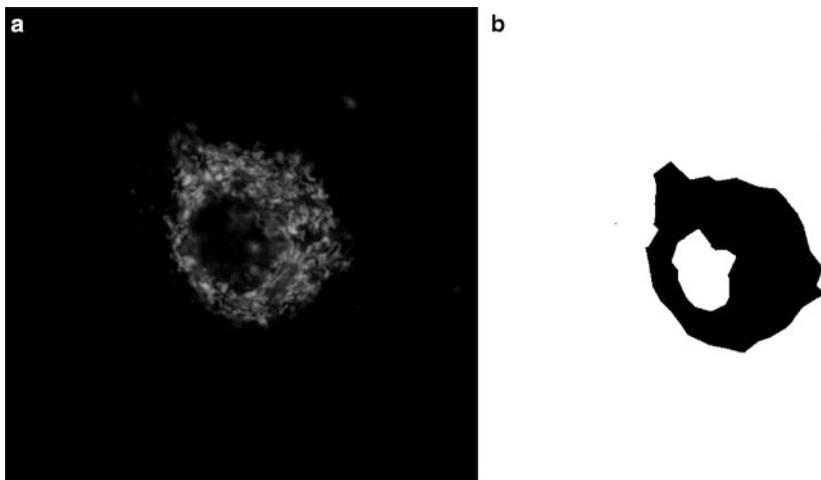


Abb. 10.8 **a** Eine Schicht aus einem Stack der Aufnahme einer Nervenzelle mit der konfokalen Laserscanning-Mikroskopie. Auch der Zellkern ist deutlich zu sehen. **b** Aus dem Bild in **a** wurden typische Punkte detektiert und von dieser Punktmenge mittels der parametrisierten Delaunay-Triangulation die *LKHD* berechnet

nieren. Diese Punkte sind aber nicht zusammenhängend. Nun muss man interaktiv oder durch A-priori-Wissen die Strukturkonstante *limit* ermitteln. Die modifizierte beschriebene Delaunay-Triangulation ergibt das gewünschte Ergebnis, siehe Abb. 10.8b. Wenn wir von einer Punktmenge als Silhouette nur die Außenkontur benötigen, dann können wir auch den in Abschn. 1.2.4 hergeleiteten, quadratischen Algorithmus benutzen.

10.5 Seeded region growing

Das „seeded region growing“-Prinzip besitzt eine gewisse Ähnlichkeit zur lokalen adaptiven Kontursuche. Oft kennt man einen „Saatpunkt“ eines Objektes, d. h. von bestimmten Punkten weiß man, dass sie auf jeden Fall zum Objekt gehören. Oft setzt man dazu Schwellwerte besonders hoch oder ermittelt mit Korrelationen Objektpunkte. Von diesem Startpunkt breitet sich nun „nach allen Seiten hin“ das Objekt aus. Die Ähnlichkeit zum Dijkstra-Algorithmus (siehe Abschn. 10.7.2) ist auffallend: vom Startpunkt ausgehend breite sich das Objekt wie ein Feuerbrand aus, man muss „nur“ festlegen, wann der Feuerbrand gestoppt wird. Die verbrannte Fläche stellt das Objekt dar. Natürlich stellen die Kostenfunktion und das Abbruchkriterium wieder die kritischen Punkte für die Anwendbarkeit dar. Wie bei Dijksta nimmt man zu dem bisherigen Objekt den Nachbarpunkt mit den geringsten Kosten hinzu. Eine einfache Kostenfunktion ist z. B. die Unähnlichkeit/Ähnlichkeit der Grauwerte. Man nimmt dasjenige neue Pixel aus der Nachbarschaft auf, das zum Grauwert des Startpunktes den ähnlichsten Grauwert besitzt. Abgebrochen wird das Wachstum wenn entweder eine bestimmte Fläche des Objektes erreicht wurde oder wenn bezüglich

einer Schwelle die Ähnlichkeit der Grauwerte nicht mehr gegeben ist. Es gibt auch parameterlose Abbruchkriterien, wie:

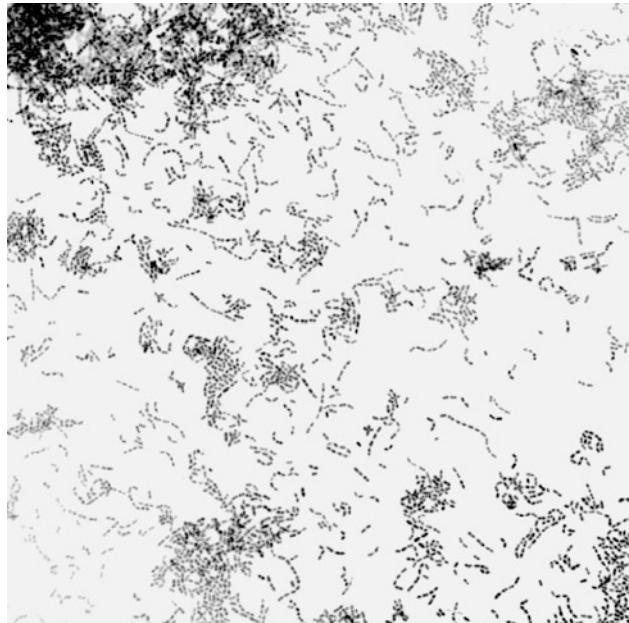
- Jedem Pixel im Bild ist ein Wert bezüglich einer Kostenfunktion zugeordnet.
- Man berechnet für alle Randpunkte des bisherigen Objektes das arithmetische Mittel aller Kosten K_1 .
- Man berechnet für alle Nachbarpixel des bisherigen Objektes das arithmetische Mittel aller Kosten K_2 .
- Für jedes neu aufgenommene Pixel wird die Kostendifferenz $|K_1 - K_2|$ als Funktion der Anzahl der aufgenommenen Pixel betrachtet. Diese Funktion wird bis zu einer bestimmten Maximalzahl *maxpixel* von Pixeln berechnet.
- Nun wird als Abbruchkriterium die Stelle der Funktion berechnet, wo diese ihr globales Maximum besitzt.

So ganz parameterlos scheint es dennoch nicht zu gehen. Der Parameter *maxpixel* ist zwar relativ robust wählbar, die Objekte zwischen kleinem und sehr großem Parameter unterscheiden sich schon wesentlich.

10.6 Anwendung in der Biofilmanalyse

Biofilme werden häufig mit der Konfokalen Laser Scanning Mikroskopie analysiert. Man erhält einen 3D Stack von 2D-Bildern, die man analysieren muss. In der Abb. 10.9 ist ein Bild aus einem Stack zu sehen, wobei Bakterien zu sehen sind. Eine Standardaufgabe ist nun diese Bakterien zu zählen und das Biovolumen zu berechnen. Neben dem Zählen muss man folglich auch alle Bakterien segmentieren, aber nur bezüglich ihrer Gesamtfläche. Ein einzelnes Bakterium zu segmentieren ist äußerst schwierig. Diese Aufgabe scheint mit dem klassischen Konturfolgeverfahren und einfachen Thresholdtechniken nicht lösbar zu sein. Eine einfache Idee ist das Zählen mit einer Startpunktsuche für das *region growing* zu verbinden. Die Startpunktsuche realisieren wir mit der Detektion von lokalen Maxima (oder Minima). Um ein lokales Maximum zu detektieren, benötigen wir die Ausdehnung des lokalen Maximums (eine Fenstergröße *window*) und die Höhe des lokalen Maximums (*high*) gegenüber dem lokalen Untergrund. Diese zwei Parameter muss man durch A-priori-Wissen kennen oder vom Nutzer interaktiv einstellen lassen. Zunächst muss man das Bild vorverarbeiten um *shading*, Beleuchtungsänderungen und Rauschen zu berücksichtigen. Dazu wenden wir auf das Bild das *top hat* Filter an, siehe (8.45) und (8.46). Die Fenstergröße des *top hat* Filters sollte größer als *window* sein. Damit haben wir den entscheidenden Vorteil, dass der lokale Hintergrund auf Null normiert wurde. Auf dieses Ergebnisbild wird nun die eigentliche lokale Maximumdetektion angewendet. Um die Komplexität der Maximumdetektion konstant zu implementieren, wenden wir zunächst ein Maximumfilter an, was mit konstanter Komplexität bezüglich der Größe *window* geschieht. Dann prüfen wir für jedes Pixel, ob der Grauwert gleich dem im Maximum-Bild ist und größer als *high* ist.

Abb. 10.9 Eine Ebene aus dem Stack einer Biofilmaufnahme



Die Liste der lokalen Maxima müssen wir nun noch auf Plateapunkte untersuchen. Wir löschen alle Punkte in der Liste, die zu einem Punkt einen kleineren Abstand als *window* besitzen und den gleichen Grauwert haben. Nun haben wir alle Bakterien gezählt und können diese Punkte für den *region growing*-Algorithmus als Startpunkte nutzen.

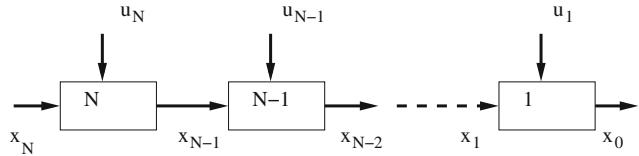
10.7 Liniendetektion

Eine weitere typische Aufgabe der Bildsegmentierung ist aus einer Menge von detektierten Einzelpunkten Linien zu gewinnen. Die Einzelpunkte können das Ergebnis einer Filtrierung mit anschließender Schwellwertberechnung sein oder vor vornherein bekannt sein. Es können Kantenpunkte oder auch direkt je nach Aufgabenstellung Linienpunkte sein. Andererseits ist es auch möglich direkt auf dem Grauwertbild Linien zu extrahieren oder Methoden zu kombinieren. Bei der Liniendetektion oder auch allgemein bei Segmentierungsmethoden bedient man sich oft Methoden aus der Graphentheorie. Eine Übersicht dazu findet man in [40].

10.7.1 Dynamische Programmierung

Dies ist eine allgemeine Methode um in einem Graphen bezüglich einer Kostenfunktion optimale Wege zu finden. Haben wir nur eine diskrete Punktmenge gegeben, dann kön-

Abb. 10.10 N -stufiger Entscheidungsprozess



nen wir die Punkte als Knoten in einem Graphen auffassen, allerdings fehlen die Kanten, die Nachbarschaften in dieser Punktmenge. Diese müssten erst mit anderen Verfahren ermittelt werden. Auf einem Grauwertbild könnten wir sofort einen Nachbarschaftsgraphen konstruieren:

Jedes Pixel ist ein Knoten und die Kanten sind durch die Vierer- oder Achternachbarschaft bestimmt. Aus den Grauwerten oder anderen Informationen könnten dann Kosten für die Kanten bzw. Knoten berechnet werden. Ausgehend von einem Startpixel suchen wir einen Weg zu einem vorgegebenen Zielpixel mit minimalen Kosten. Das ist die Grundidee zur Liniendetektion mittels der *Dynamischen Programmierung*. Der Name „Dynamische Programmierung“ ist eigentlich etwas irreführend und sollte besser *Rekursive Optimierung* heißen und drückt das *Bellmansche Optimalprinzip* (Bellman 1957) aus. Allgemein sei ein N -stufiger, deterministischer Entscheidungsprozess gegeben, siehe Abb. 10.10. Dabei werden

$$x_{i-1} = f_i(x_i, u_i), \quad x_i \in X_i, \quad u_i \in U_i, \quad i = 1, \dots, N \quad (10.5)$$

als Prozessgleichungen oder als Stufentransformationen bezeichnet. Dabei übernehmen die u_i die Aufgabe von Steuervariablen, beeinflussen folglich den Entscheidungsprozess. Die Steuervariablen sollen nun so gewählt werden, dass der Entscheidungsprozess im Sinne eines Zielkriteriums optimal ist:

$$z = z(g_N(x_N, u_N), g_{N-1}(x_{N-1}, u_{N-1}), \dots, g_1(x_1, u_1)) \rightarrow \text{Extremum.} \quad (10.6)$$

An die Zielfunktion z stellen wir die Forderung der „monotonen Separabilität“, die z. B. erfüllt ist, wenn z additiv ist $z = \sum_{i=1}^N g_i(x_i, u_i)$. Damit setzt sich die Zielfunktion additiv aus den einzelnen Bewertungen der Stufen zusammen, was wir im Folgenden immer annehmen werden. Da die Lösung des Problems rekursiv formuliert wird, sind die Indizes gegenläufig zur Flussrichtung des Prozesses gewählt. Die Wahl der optimalen Steuervariablen u_1, \dots, u_N wird optimale Politik genannt. Die Optimierung des Gesamtproblems kann nun nach Bellman rekursiv mit den Bellmanschen Funktionen w_i formuliert werden:

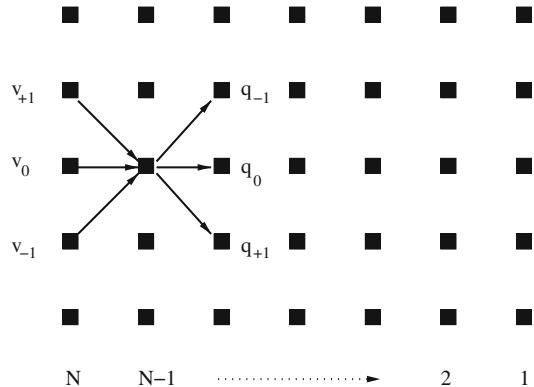
$$w_1(x_1) = \text{extremum}_{u_1}\{g_1(x_1, u_1)\}, \quad (10.7)$$

$$w_n(x_n) = \text{extremum}_{u_n}\{g_n(x_n, u_n) + w_{n-1}(x_{n-1})\}. \quad (10.8)$$

Dann ist $w_N(x_N)$ optimale Lösung des Gesamtproblems.

Dieses Grundprinzip soll nun an einem einfachen Beispiel der Bildsegmentierung gezeigt werden.

Abb. 10.11 Bildspalten als Stufen eines N -stufigen Entscheidungsprozesses. Die Pfeile geben die möglichen Vorgänger und Nachfolger eines Pixels an



Beispiel Ein Bild soll horizontal in ein „Oberteil“ und in ein „Unterteil“ segmentiert werden. Dazu bilden wir das Gradientenbild. Nun suchen wir einen Weg von der ersten Spalte des Bildes bis zur letzten Spalte des Bildes mit maximalen Gradientenkosten. Die Anzahl der Spalten des Bildes entspricht dann genau der Anzahl N der Stufen des Entscheidungsprozesses. Pro Spalte i bilden die Pixel die Zustände x_i . Die Steuerungen u_i sind die möglichen Fortsetzungen von einem Pixel ausgehend zur nächsten Stufe (Spalte). Die Steuerungen beschreiben somit die möglichen Vorgänger v und die möglichen Nachfolger q pro Pixel, siehe Abb. 10.11. Durch die Vorgänger- und Nachfolgerrestriktion garantieren wir immer N Stufen, d. h. einen Weg genau der Länge N . Es bezeichne $g_i(p_j)$ die Kosten eines Pixels p_j in der Stufe i . Dann wollen wir einen Weg, beginnend in der ersten Spalte N und endend in der letzten Spalte 1 der Länge N finden, sodass die Bewertung

$$z = \sum_{i=1}^N g_i(p_{j_i}) \rightarrow \text{Extremum} \quad (10.9)$$

minimal/maximal wird. Entsprechend Abb. 10.11 ergeben sich dann die Bellmanschen Funktionen zu:

$$w_i(p_j) = \text{extremum}_{l \in \{-1, 0, +1\}} \{g_i(p_j) + w_{i-1}(q_l)\} \quad (10.10)$$

$$= g_i(p_j) + \text{extremum}_{l \in \{-1, 0, +1\}} \{w_{i-1}(q_l)\}, \quad i = N, \dots, 1. \quad (10.11)$$

Außer dem optimalen Kostenwert interessiert natürlich der optimale Weg. Daher ist es günstig, den optimalen Nachfolger pro Pixel abzuspeichern. Daher kann man auch die Rekursion auflösen und iterativ in einer Vorwärtsrechnung die Werte w pro Pixel berechnen und den optimalen Nachfolger abspeichern. Dann braucht man nur noch in der Rückwärtsverfolgung, beginnend beim optimalen Startpixel, die Kette der optimalen Nachfolgepixel zu extrahieren und nichts mehr zu berechnen. Bei dem Beispiel der horizontalen Bildsegmentierung könnte es z. B. passieren, dass ein Weg von links oben nach rechts unten gefunden wird. Man möchte aber eine näherungsweise horizontale Segmentierung, d. h.

der Weg sollte nicht zu stark schwanken. Dann muss man sich Gedanken über die Kostenfunktion machen. Wir müssten simple lokale Krümmungsmaße $\text{kr}(v_m, p, p_l)$, $m, l \in \{-1, 0, +1\}$ mit in die Kosten einfließen lassen, die wir mit einem Faktor belohnen oder bestrafen können, daher ist

$$z = \sum_{i=1}^N g_i(p_{j_i}) + \alpha \sum_{i=2}^{N-1} \text{kr}(v_m, p_{j_i}, q_l) \rightarrow \text{Extremum} \quad (10.12)$$

zu berechnen. Falls wir maximieren ist $\alpha < 0$ und falls wir minimieren ist $\alpha > 0$ zu setzen. Entsprechend Abb. 10.11 würden wir die Krümmungswerte je nach Vorgänger/Nachfolger-Konstellation im Wertebereich von $\{0, 1, 2\}$ annehmen oder völlig andere Bewertungen wählen. Da wir pro Pixel nun drei Nachfolger bzw. drei Vorgänger berücksichtigen müssen, benötigen wir auch drei Bellmansche Funktionen. Diesen Index hängen wir an w oben an und notieren diesen analog der Richtung $\{-1, 0, +1\}$ wie in Abb. 10.11. Die Bellmanschen Funktionen (10.11) werden dann mit $k \in \{-1, 0, +1\}$ zu:

$$w_i^k(p_j) = \text{extremum}_{l \in \{-1, 0, +1\}} \{g_i(p_j) + \alpha \cdot \text{kr}(v_k, p_j, q_l) + w_{i-1}^l(q_l)\}. \quad (10.13)$$

Für die Vorwärtsrechnung benötigen wir für die Bellmanschen Funktionen drei Matrizen, wobei der optimale Nachfolger getrennt zu speichern ist.

Bemerkungen

- Es kann das Bild natürlich auch vertikal segmentiert werden.
- Man kann in einer beliebigen Spalte oder Zeile starten und sich die Anzahl der Spalten/Zeilen vorgeben.
- Man kann in genau einem Pixel starten und sich die Anzahl der Spalten/Zeilen vorgeben.
- Man muss ein Bild nicht unbedingt segmentieren. In einigen Anwendungen sind zwei Aufnahmen eines Objektes vorhanden, wobei beide Bilder einen gewissen Überlappungsbereich besitzen. Durch die Aufnahmen sind aber im Überlappungsbereich gewisse nichtlineare Verzerrungen vorhanden. Die Bilder sind nun vertikal zusammenzufügen. Man möchte vertikal eine Schnittlinie berechnen, so dass der Zusammenfügefehler minimal wird. Im Überlappungsbereich wird somit pro Pixel eine Fehlerfunktion zwischen beiden Bildern berechnet, die eine Kostenfunktion darstellt. Nun wird ein vertikaler Pfad mittels dynamischer Programmierung berechnet, so dass die Kosten der Schnittlinie minimal sind.
- Häufig werden genau nach diesem Prinzip Indizes aus einer Kostenmatrix ermittelt, die durch den Pfad minimaler Kosten berechnet werden. Dazu ist ein ausführliches Beispiel in Abschn. 19.8.4 angegeben. Ein weiteres Beispiel zur Berechnung einer Folge von Indizes ist die Berechnung des *Levenshtein-Abstandes* zweier Strings. Dieser Abstand stellt dann die minimalen Überführungskosten zwischen den beiden Strings dar.

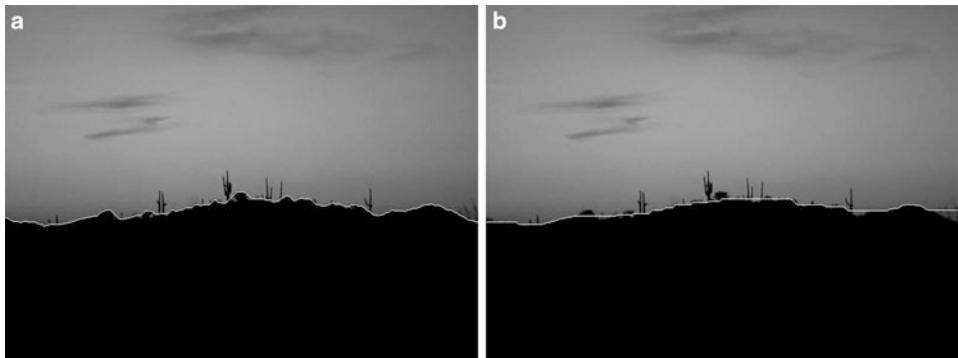


Abb. 10.12 Dynamische Programmierung: Horizontale Segmentierung – weiße Linie. **a** Es wurde kein Strafterm bezüglich lokaler Krümmungen angewendet. **b** Es wurde ein hoher Strafterm bezüglich lokaler Krümmungen angewendet

- In manchen Anwendungen der Mustererkennung/Bildverarbeitung werden Algorithmen mit eigenständigen Namen versehen, obwohl sie auch nur das Grundprinzip der dynamischen Programmierung benutzen. Ein Beispiel dazu ist der *Viterbi-Algorithmus* in der Sprachverarbeitung.

Das Entscheidende für eine sinnvolle Anwendung ist natürlich die Wahl der Kostenfunktion. In der Abb. 10.12b soll horizontal der Horizont „glatt“ abgetrennt werden. In diesem einfachen Fall reicht der Betrag des Gradienten als Kostenfunktion aus, da oberhalb und unterhalb der „Segmentübergangszone“ keine „wesentlichen“ Kanten vorhanden sind.

10.7.2 Dijkstras Algorithmus

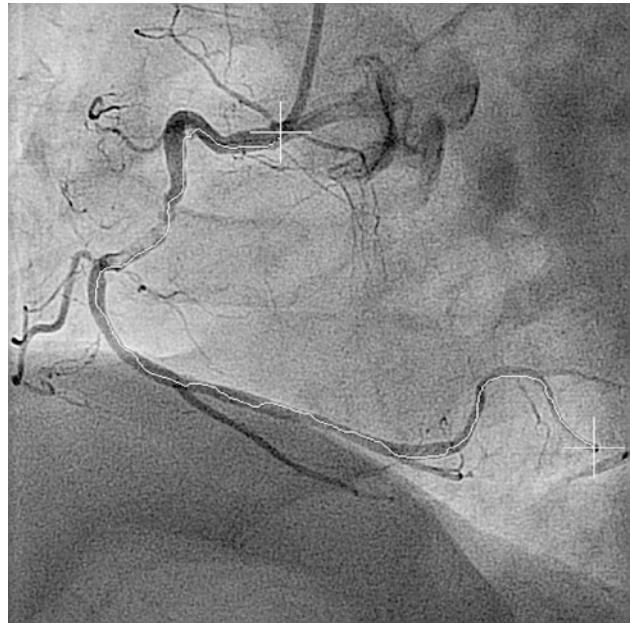
Dijkstra stellte diesen Algorithmus erstmals 1959 zur Berechnung des kürzesten Pfades in einem Graphen vor. Dieser Algorithmus wird häufig als Greedy-Algorithmus bezeichnet, seine Ähnlichkeiten zur Dynamischen Programmierung sind auffällig, siehe [68]. Wir wollen diesen hier ausschließlich zur Liniendetektion in Grauwertbildern beschreiben. Dazu benutzen wir das Quadratgitter mit der Achternachbarschaft, siehe Abschn. 1.2. Gegeben sei ein Grauwertbild, d. h. eine Menge von Pixeln $B \subseteq \mathbb{Z}^2$ mit einer Grauwertfunktion $f(i, j)$. Aus diesem Grauwertbild wird eine Kostenfunktion $g(i, j)$, $(i, j) \in B \subseteq \mathbb{Z}^2$ berechnet. Bezuglich dieser Kostenfunktion suchen wir nun einen Weg im Quadratgitter bezüglich der Achternachbarschaft mit minimalen Kosten. Dabei starten wir von einem festen Pixel p ausgehend und enden in einem fest vorgegebenen Endpixel q . Statt des Endpixels kann man auch ein „Endgebiet“, d. h. eine Menge von Pixeln Q vorgeben. Der Weg ist beendet, sobald er das Endgebiet erreicht hat, egal bei welchem Pixel. Der wesentliche Unterschied zur dynamischen Programmierung besteht darin, dass die Weglänge nicht

vorgeschrieben ist und es auch keine Richtungsbeschränkungen für den Weg gibt. Einen Weg mit maximalen Kosten (positive Kosten) zu suchen ist unsinnig, da sofort der längste Weg die Lösung wäre. Wir setzen für die Kostenfunktion im Folgenden stets $g_{i,j} > 0$ voraus, ansonsten addieren wir einfach zu allen Kosten ein $\varepsilon > 0$. Wir bauen nun schrittweise vom Startpixel p ausgehend ein zusammenhängendes Objekt M auf. Alle Pixel aus M nennen wir Objektpixel, ansonsten Untergrundpixel. Randpunkte sind die Pixel aus M , in deren Achternachbarschaft sich mindestens ein Untergrundpixel befindet. Eine Randkante von M ist ein Paar von Pixeln und zwar bestehend aus einem Randpunkt und einem seiner Nachbaruntergrundpunkte.

Algorithmus

- Zu Beginn besteht das Objekt M nur aus dem Startpixel p .
- Wir ordnen allen Pixeln des Bildes B eine summarische Kostenmatrix K zu, die die gleiche Dimension wie das Grauwertbild besitzt. Die Kostenmatrix K wird zu Beginn auf Null initialisiert und bekommt an der Position des Startpixels p die Kosten $g(p)$ zugewiesen.
- Wir nehmen alle Randkanten von M in eine Datenstruktur DAT auf. Jede Randkante erhält als Bewertung die Summe der Kosten der beiden Pixel, die die Randkante verbindet, wobei die Kosten für das Randpixel von M aus der summarischen Kostenmatrix K zu entnehmen ist.
- Wir suchen in DAT die Randkante mit minimaler Bewertung. Die ausgewählte Randkante wird in DAT gelöscht. Der Untergrundpunkt der gelöschten Randkante wird zur Menge M hinzugefügt, gleichzeitig werden in der summarischen Kostenmatrix für diesen Untergrundpunkt die Kosten der gelöschten Randkante eingetragen. Ist die Kante mit minimaler Bewertung aus DAT keine Randkante, obwohl sie in DAT eingetragen war, wird sie entfernt und der Vorgang wiederholt, bis eine „echte“ Randkante mit minimaler Bewertung gefunden wurde.
- Für das neu in M aufgenommene Pixel werden die Randkanten ermittelt und in DAT mit den entsprechenden Kosten neu aufgenommen.
- Nun wird wieder in DAT die Randkante mit minimaler Bewertung gesucht usw. Ist der Untergrundpunkt der Randkante das Zielpixel q oder ein Pixel aus dem Zielgebiet, dann wird die „Vorwärtsrechnung“ beendet.
- Die Vorwärtsrechnung gleicht einem von dem Startpixel ausgehenden Flächenbrand, der dann abbricht, wenn dieser das Zielpixel oder das Zielgebiet erreicht hat.
- Ausgehend vom Zielpixel q beginnt nun die Rückwärtsrechnung zur Berechnung des optimalen Weges. In der Nachbarschaft des Zielpixels wird das Pixel ermittelt, das die minimalen summarischen Kosten in K hat. Von diesem wird wieder das nächste in der Nachbarschaft ermittelt usw. bis man am Startpixel p angekommen ist.
- Man kann auch bei der Vorwärtsrechnung die optimalen Vorgänger abspeichern, dann entfällt die Suche in der Nachbarschaft mit den minimalen Kosten.

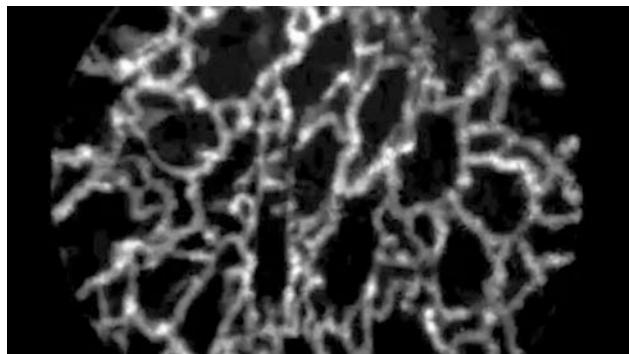
Abb. 10.13 Liniendetektion mittels Dijkstra-Algorithmus: Der Verlauf eines Gefäßes des Herzens wurde durch Anklicken des Startpunktes und des Endpunktes gefunden



- Die Rückwärtsrechnung gleicht anschaulich dem Abrollen einer Kugel, die wir auf das Zielpixel legen. Die Höhe über den Pixeln beschreiben die summarischen Kosten. Die Kugel rollt dann automatisch auf dem optimalen Weg zum Startpixel.
- Der optimale Weg ist genau dann nicht eindeutig, wenn es Kosten $g_{i,j} = 0$ gibt, dann entsteht in der summarischen Kostenfunktion ein Plateau. Daher addieren wir einfach zu allen Kosten ein $\varepsilon > 0$.
- Auf eine genaue Implementierung dieses Algorithmus gehen wir hier nicht ein, so wird z. B. für die Datenstruktur DAT ein *Fibonacci-Heap* empfohlen.

Der erfolgreiche Einsatz des Dijkstra-Algorithmus hängt natürlich wieder von der Kostenfunktion ab, diese ist gemäß der Aufgabenstellung sehr sorgfältig zu wählen. Dazu ein Beispiel eines Röntgenbildes des Herzens, siehe Abb. 10.13. Der Kardiologe begutachtet die großen Gefäße, ob diese eventuell Stenosen aufweisen, also Gefäßverengungen. Dazu könnte er sich interaktiv ein großes Gefäß aussuchen. Dies kann man realisieren durch Anklicken eines Start- und Endpunktes eines Gefäßes und der Dijkstra Algorithmus detektiert den Verlauf des Gefäßes. Wenn viele Gefäße bzw. Gefäßverzweigungen vorliegen, kann man auch die Start/Endpunkte kaskadieren, also Zwischenpunkte anklicken, um so die Detektion eines bestimmten Gefäßes zu erzwingen. In Abb. 10.13 ist der detektierte Weg zu sehen. Als Kostenfunktion wurde das Ergebnisbild des in Abschn. 8.2.2 eingeführten Richtungs-DoB-Filter benutzt, wobei dieses Bild noch „geschickt“ zu normalisieren ist. Die „Kunst“ der erfolgreichen Anwendung des Dijkstra-Algorithmus besteht tatsächlich in der „richtigen“ Konstruktion der Kostenfunktion.

Abb. 10.14 Aufnahme mittels *Confocal Laser Endomicroscopy*: Gefäßstruktur des Rektums (ca. 1 mm^2)



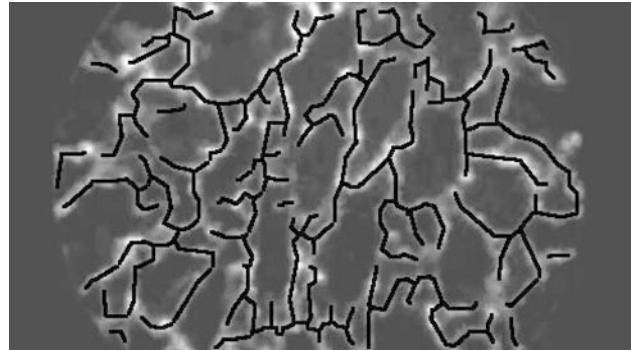
10.7.3 Graphbasierte Methoden

Wir nehmen einmal an, aus einer Vorverarbeitung und Binarisierung resultieren eine Menge von Pixeln (Punktmenge PM), die zu irgendwelchen Linien gehören. Dies muss nicht eine Linie sein, sondern kann sogar ein „Liniengeflecht“ sein. Wie kann man allein aus der Kenntnis dieser Pixel (ohne weiteres Zusatzwissen) das Liniengeflecht berechnen. Dies ist nur möglich über die geometrische Lage der Pixel zueinander. Allein aus der geometrischen Lage muss dann eine Art Kostenfunktion ableitet werden. Die Pixel bilden nun die Knoten eines Graphen, wobei die Kanten unbekannt sind. Diese Kanten sollen nun allein über die geometrische Lage der Pixel zueinander berechnet werden. Anschließend stellt der Graph mit seinen Knoten und Kanten das Liniengeflecht dar. Zunächst baut man gedanklich einen vollständigen Graphen auf, d. h. jedes Pixel ist mit jedem anderen verbunden. Die Kanten werden mit dem Euklidischen Abstand der Pixel zueinander bewertet. Aus diesem Graphen soll nun ein Untergraph ausgewählt werden. In der Abb. 10.14 ist eine Aufnahme mittels *Confocal Laser Endomicroscopy* eines mikroskopisch kleines Ausschnittes des Rektums zu sehen. Die weißen Strukturen sind das Liniengeflecht, welches zu detektieren ist und von welchem Maßzahlen abzuleiten sind. Eine klassische Vorgehensweise ist nun die Folgende:

- Das Bild muss zunächst vorverarbeitet werden indem Linienelemente besonders verstärkt werden. Dazu dienen in hervorragender Weise die in Abschn. 8.2.2 aufgeführten Richtungs-DoB-Filter.
- Sequentiell werden nun die besten Output-Pixel des DoB-Filters bestimmt und als Startpunkte für eine *region-growing*-Segmentierung benutzt.
- Auf das Segmentierungsergebnis, welches ein Binärbild darstellt, wird nun ein Skelettierungsalgorithmus angewendet. Diese Punkte des Skelettes dienen nun als Ausgangspunkt für die graphenorientierten Methoden zur Bestimmung des Liniengeflechtes.

Wald minimal aufspannender Bäume Wir betrachten einmal die Pixel des Skelettes als Knoten eines Graphen. Jeder Knoten ist mit jedem anderen verbunden, es liegt also ein vollständiger Graph vor. Jede Kante wird mit Kosten belegt, z. B. mit der Entfernung der

Abb. 10.15 Liniengeflechtdetektion des Bildes aus Abb. 10.14: Die schwarzen Punkte stellen die Knoten eines Waldes von minimal aufspannenden Bäumen dar



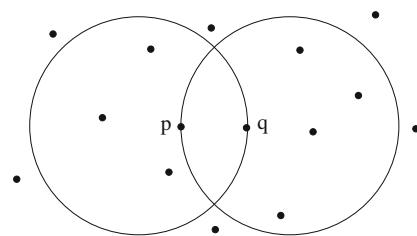
beiden Pixel bezüglich irgendeiner Metrik. Der minimal aufspannende Baum (*Minimal Spanning Tree*, (MST)) ist ein Teilgraph, der folgendermaßen aufgebaut wird:

- Er enthält weiterhin alle Knoten, d. h. Pixel.
- Es werden soviele Kanten gestrichen, dass der Teilgraph weiterhin zusammenhängend ist und dass keine Zyklen mehr enthalten sind.
- Der MST ist dann der Teilgraph, der bezüglich der Kosten aller übrigbleibenden Kanten die minimalen Kosten liefert.

Zur algorithmischen Berechnung sei hier auf die einschlägige Literatur verwiesen [64], da dies ein Standardalgorithmus ist. Den MST kann man weiter zerlegen in Komponenten, die trivialerweise selbst wieder Bäume darstellen. Eine Zerlegungsstrategie ist z. B., dass man sukzessive die Kanten mit den größten Kosten streicht und aufhört mit streichen, wenn aus der Anwendung heraus eine bestimmte Schwelle erreicht ist. Es entsteht dann ein Wald von minimal aufspannenden Bäumen. Diese Vorgehensweise ähnelt einem Clusteralgorithmus, die minimal aufspannenden Bäume stellen die Cluster dar. Dieser Algorithmus wird daher in der Literatur als graphbasierter Clusteralgorithmus eingestuft. Auf das Bild in Abb. 10.14 wurde bezüglich des ermittelten Skelettes ein MST-Algorithmus angewendet. Dieser lieferte einen Wald, d. h. mehrere MST. Von diesen Bäumen wurden die besonders kurzen Äste entfernt. Auf die Liniensegmente zwischen zwei Knoten wurde nun noch ein *Split and Merge* Algorithmus angewendet, um diese in robuster Weise zu approximieren. Die Länge aller dieser Segmente bildet nun die eigentliche Länge des Liniengeflechtes. In Abb. 10.15 ist das recht gut gelungene Ergebnis zu sehen.

Relativer Nachbarschaftsgraph (RNG) Manchmal ist es störend, dass der minimal aufspannende Baum keine Zyklen enthalten darf, man möchte aber ein ähnlich gutes Ergebnis haben. Dazu kann der relative Nachbarschaftsgraph dienen. Es werden in dem vollständigen Graphen nur die Kanten erhalten, deren beide Pixel relative Nachbarn sind. Zwei Pixel/Punkte p und q sind relative Nachbarn, wenn es innerhalb des durch p und q bestimmten Kreiszweiecks keine weiteren Pixel/Punkte gibt, siehe Abb. 10.16.

Abb. 10.16 Definition des relativen Nachbarschaftsgraphen



Delaunay-Graph (DG) Ein noch etwas umfassender Graph ist der Delaunay-Graph, der ebenfalls das Liniengeflecht sinnvoll beschreiben kann. Dieser Graph basiert auf dem Voronoi-Graphen. Zur Definition sei auf eine umfangreiche Standardliteratur verwiesen [64]. Wenn man die Punktmenge PM als Graphen ohne Kanten interpretiert, dann gilt die Beziehung:

$$PM \subset MST \subseteq RNG \subseteq DG. \quad (10.14)$$

Der Delaunay-Graph wird auch zur Delaunay-Triangularisierung genutzt, d. h. eine gegebene Punktmenge PM wird mit Dreiecken triangularisiert. Dies kann man z. B. nutzen, um die limitierte, konvexe Hülle einer Punktmenge zu berechnen, siehe Abschn. 1.2.3.

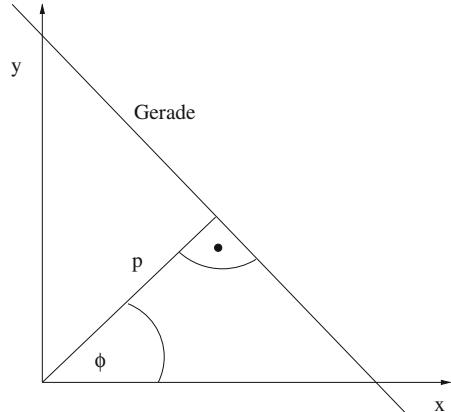
10.8 Akkumulations- oder Votingmethoden

Wenn wir im Bild an eine Menge von Punkten eine Gerade oder ein anderes geometrisches Primitivum fitten, dann wissen wir a priori, dass die Punkte bis auf einige Ausreißer zu dem geometrischen Primitivum gehören. Dies ist aber oft nicht der Fall. Oft weiß man, dass mehrere Geraden im Bild vorhanden sind und muss diese detektieren. Erst wenn man die Zugehörigkeit von Punkten zu einer Geraden bestimmt hat, kann man nachträglich eine Gerade fitten. Zunächst wollen wir an Hand von Geraden das Grundprinzip demonstrieren und anschließend verallgemeinern bzw. die allgemeinen Probleme beschreiben.

10.8.1 Primale Voting-Methode zur Detektion von Geraden

- In einem Bild seien n Punkte detektiert worden. Wir bilden nun alle Paare von Punkten, d. h. es gibt $\binom{n}{2} = \frac{n(n-1)}{2}$ solcher Punktpaare. Durch zwei Punkte ist eindeutig eine Gerade bestimmt, wir bestimmen diese in der Hesseschen Normalform mit den Parametern p, φ , siehe Abb. 10.17. Nun wollen wir in einem Parameterraum, d. h. in einem Koordinatensystem, an der Stelle p, φ die Zahl Eins inkrementieren. Das setzt voraus, dass wir zu Beginn den Parameterraum mit Null initialisiert haben.
- Wenn wir nun den Parameterraum mit einer Datenstruktur verwalten wollen, dann müssen wir vorher den Parameterraum p, φ diskretisieren. Die Wertebereiche können

Abb. 10.17 Hessesche Normalform einer Geraden



wir leicht abschätzen. Den Winkel legen wir von Null bis 180 Grad fest, da eine Gerade keinen Orientierungssinn besitzt. Die Größe p ist beschränkt durch die Länge der Diagonale des Originalbildes, wobei p auch negativ sein kann. Die Diskretisierung wird durch praktische Belange festgelegt, so könnten wir den Winkel z. B. in Ein-Grad-Schritten festsetzen und p mit der Zahl Eins, d. h. eine Verschiebung der Geraden ist bis auf ein Pixel festgelegt.

- Wir inkrementieren nun entsprechend für alle $\binom{n}{2} = \frac{n(n-1)}{2}$ Punktepaare. Liegen nun z. B. m Punkte auf einer Geraden im Bild, so muss sich im Parameterraum ein **Peak** der Höhe m ausbilden. Wenn wir nur eine typische Gerade detektieren wollen, bestimmen wir einfach die Position des Maximums im Parameterraum. Wenn wir l Geraden detektieren wollen, dann bestimmen wir die l größten Peaks im Parameterraum.
- Praktische Probleme entstehen aber nun infolge der Diskretisierung. Wir müssen einen Punkt im Parameterraum mit einer **Akkumulatorzelle** identifizieren. Dies können wir tun, indem wir vor der Maximumbestimmung einen Glättungsfilter von der Größe der Akkumulatorzelle auf das Parameterbild anwenden. Wir können auch direkt beim akkumulieren nicht nur an der Stelle p , ϕ inkrementieren, sondern auch an den Nachbarstellen entsprechend der Größe der Akkumulatorzelle. Man sollte aber keinen Mittelwertfilter benutzen oder die Nachbarstellen gleichwertig inkrementieren, sondern einen Gaußfilter. Dazu stellen wir uns ein Bild vor, in dem tatsächlich nur m Geraden vorkommen und deshalb sich im Akkumulatorraum m Peaks ausprägen. Diese m Peaks seien ideal, d. h. haben die Form des Einheitsimpulses. Der Mittelwertfilter wird nun mit dem Einheitsimpuls gefaltet, es ergibt sich als Resultat wieder der Mittelwertfilter, d. h. eine „Säule“, die oben ein „glattes“ Plateau hat. Wenn wir nun anschließend das Maximum suchen, ist dies nicht mehr eindeutig, wir müssten aufwendig die „Mitte“ des Plateaus bestimmen. Dies kann man vermeiden, wenn man den Gaußfilter benutzt, bzw. eine gewichtete Akkumulation in der Akkumulatorzelle durchführt.
- Wenn Punkte im Originalbild bezüglich der Achternachbarschaft benachbart sind, dann liegen nur vier Geradenrichtungen vor, nämlich 0 Grad, 45 Grad, 90 Grad und

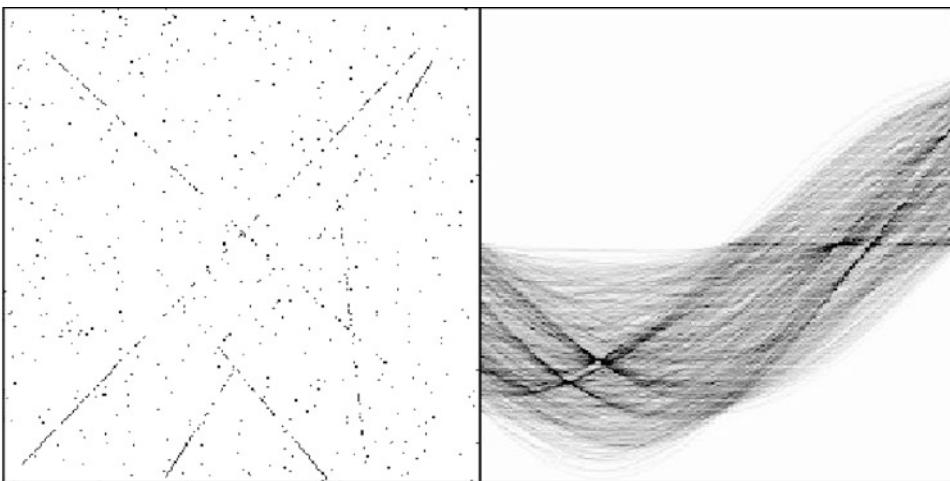


Abb. 10.18 Punktmenge \leftrightarrow Hough-Transformation

135 Grad. Wenn folglich viele Punkte benachbart sind, dann werden infolge dieser Diskretisierung diese vier Richtungen im Parameterraum bevorzugt inkrementiert. Daher sollte man Punktpaare aus einer Achternachbarschaft nicht in die Akkumulation einbeziehen.

- Wie man leicht sieht, hat diese primale Methode zur Geradendetektion die Komplexität $O(n^2)$.

10.8.2 Duale Voting-Methode: Hough-Transformation

- Im Gegensatz zur primalen Methode kann man auch die Dualität von Punkt und Gerade nutzen, d. h. einem Punkt entspricht genau ein Geradenbüschel und umgekehrt. Diese Idee geht auf Hough (1962) zurück und wird deshalb **Hough-Transformation** genannt. Die erste praktische Anwendung der Hough-Transformation wird Duda und Hart (1973) zugeschrieben.
- Anstelle der Punktpaare werden die n Punkte einzeln betrachtet und für jeden Punkt (x_i, y_i) das Geradenbüschel berechnet:

$$p(\varphi) = x_i \cos \varphi + y_i \sin \varphi. \quad (10.15)$$

Für einen festen Punkt (x_i, y_i) ist $p(\varphi)$ eine Kurve im Parameterraum, den sogenannten Houghraum. Über dieser Kurve wird die Zahl Eins inkrementiert. Wenn infolge der notwendigen Diskretisierung des Winkels φ dieser in q Stufen eingeteilt wird, so sind pro Punkt (x_i, y_i) des Originalraumes q Inkrementierungen im Houghraum durchzuführen.

- Die Probleme bezüglich der Akkumulatorzellen und der Diskretisierung sind die gleichen wie bei der primalen Methode. Die Geraden werden detektiert, indem man wieder die lokalen, wesentlichen Maxima im Houghraum bestimmt.
- Wie man leicht sieht, ist die Komplexität $O(q \cdot n)$, folglich linear in n .

10.8.3 Zeit- und Speicher-Effizienz

Wichtig für Anwendungen ist natürlich die Zeitkomplexität und der benötigte Speicher:

- Die Zeitkomplexität ist natürlich ein wesentliches Gütekriterium in der Anwendung. Bei der Gerdendetektion hat die primale Methode eine quadratische Komplexität $O(n^2)$. Die duale Houghmethode ist dagegen linear $O(n)$, allerdings mit einem großen Vorfaktor. Der Vorfaktor bei der Houghmethode wird eindeutig durch q bestimmt. Daher kann man sogar genau feststellen, wann man die primale und wann die duale Methode anwenden sollte, allerdings nur auf die Zeitkomplexität bezogen. Daher folgt: haben wir weniger Punkte als Diskretisierungsstufen q , dann ist bezüglich der Zeitkomplexität die primale Methode der Houghmethode vorzuziehen.
- Einen wichtigen Einfluß auf die Zeit- und Speichereffizienz hat die Beschränkung des Suchraumes. Man hat fast bei jeder Anwendung A-priori-Wissen über die zu detektierenden geometrischen Primitiva. Bei Geraden sucht man oft nur Geraden mit minimalen oder maximalen Anstiegen. Mit diesem Wissen sollte man von vornherein den Akkumulatorraum beschränken.
- Weiterhin kann man die Zeiteffizienz durch zufällige Auswahlen verbessern. Die Grundidee geht auf Fischler und Bolles (1981) zurück und wird RANSAC-Methode genannt (**R**andom **S**ample **C**onsensus). Bei der primalen Methode braucht man nicht unbedingt alle Punktpaare zu benutzen, sondern man wählt zufällig eine Untermenge aus. Theoretisch müsste man aber dann untersuchen, mit welcher Wahrscheinlichkeit man im Akkumulatorraum die richtigen Peaks bestimmen kann. Oder ein ähnliches Verfahren ist das Folgende:

Man wählt zufällig zwei Punkte aus und bestimmt die Gerade durch die beiden Punkte. Jetzt betrachtet man einen „Schlauch“ einer bestimmten Breite längs der Geraden und zählt die Punkte im Bild, die innerhalb des Schlauches liegen. Wenn die Anzahl eine bestimmte Schranke übersteigt, dann hat man bereits eine Gerade gefunden. Natürlich hängt die Güte von der Breite des Schlauches und dieser Schranke wesentlich ab. Die Effizienz dieses Verfahrens hängt von der Verteilung der Geraden im Bild ab. Haben wir z. B. ein sehr langes und ein kurzes Geraendsegment im Bild, dann ist dieses Vorgehen bezüglich der Bestimmung des langen Geraendsegmentes äußerst effektiv. Haben wir dagegen viele kurze Geraendsegmente im Bild, dann verringert sich die Effizienz wesentlich.

- In manchen Fällen kann man den Akkumulatorraum in einen Unterraum projizieren, dann gewinnt man in erster Linie an Speichereffizienz. Allerdings muss man sich der Konsequenzen bewusst sein. Dazu ein einfaches

Beispiel: Ein Personalausweis, der auf einem Scanner aufgenommen wurde, soll horizontal ausgerichtet werden, damit die OCR-Software rotationsinvariant funktionieren kann. In der Vorverarbeitung wurden die Konturen derjenigen Zeichen ermittelt, die in den beiden ICAO-Zeilen stehen. Man weiß allerdings nicht, welches Zeichen in welcher Zeile steht, aber man weiß, es gibt nur diese beiden (langen) Zeilen. Von jedem Zeichen wurde nun der Schwerpunkt ermittelt. Der Anstieg einer Geraden durch alle Schwerpunkte der Zeichen einer Zeile bestimmt damit den Rotationswinkel des Ausweises, also des gesamten Dokumentes. Wir benötigen also nur den Anstieg dieser beiden parallelen Geraden. Daher wählen wir z. B. die primale Methode und akkumulieren nur die Anstiege der Geraden durch jeweils zwei Punkte. Damit ist der Akkumulatorraum nur eindimensional. Würden wir allerdings diese Methode auf ein Dokument übertragen, das mehr als zwei Zeilen hat, dann wird dies nur noch funktionieren, wenn die Anzahl aller Zeilen wesentlich kleiner ist als die Anzahl der Zeichen einer Zeile.

10.8.4 Hough-Transformation und Konvexgeometrie

Es gibt einen interessanten Zusammenhang zwischen der Hough-Transformation und der Konvexgeometrie:

Definition 10.1 (Stützgerade) Eine Gerade G heißt **Stützgerade** einer ebenen, konvexen Figur K , wenn G Berührende an K ist. Wenn der Rand von K differenzierbar ist, dann ist die Berührende gerade die Tangente an K . Auch wenn der Rand nicht differenzierbar ist, gibt es Stützgeraden. Dies sind dann diejenigen Geraden, die mindestens einen Randpunkt schneiden und alle anderen Punkte von K liegen nur auf einer Seite der Geraden.

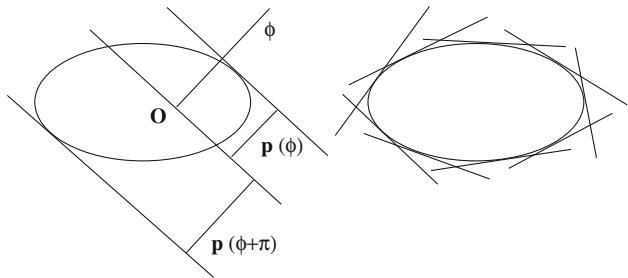
Alle Stützgeraden von K bilden eine Kurvenschar, deren Enveloppe (Einhüllende) der Rand ∂K der konvexen Figur K ist. Mit dem Ursprung O des Koordinatensystems (der hier der Einfachheit halber im Innern von K liegen möge) kann eine Gerade G im Randpunkt (x, y) durch ihre Normalenrichtung φ und ihren Stützabstand p vom Ursprung O beschrieben werden:

$$p(\varphi) = x \cos \varphi + y \sin \varphi. \quad (10.16)$$

Da der Ursprung im Innern von K liegt, legen wir $0 \leq \varphi \leq 2\pi$ fest, womit immer $p \geq 0$ gilt. Wir sehen, das ist die gleiche Funktion wie bei der Hough-Transformation. Wir wollen nun eine Parameterdarstellung $x(\varphi), y(\varphi)$ der Kontur der konvexen Figur mit Hilfe von (10.16) herleiten. Die Formel (10.16) stellt aber nur eine Gleichung für die beiden Unbekannten $x(\varphi), y(\varphi)$ dar, daher benötigen wir noch eine zweite Gleichung. Wir benutzen den üblichen Trick in der Mathematik, neben $p(\varphi)$ muss auch die Ableitung $p'(\varphi)$ gegeben sein. Wir differenzieren daher (10.16) nach φ :

$$p'(\varphi) = x' \cos \varphi - x \sin \varphi + y' \sin \varphi + y \cos \varphi. \quad (10.17)$$

Abb. 10.19 Stützgeraden und Enveloppe einer konvexen Figur



Da (x', y') der Tangentenvektor und $(\cos \varphi, \sin \varphi)$ der zugehörige Normalenvektor ist, verschwindet das zugehörige Skalarprodukt aus beiden und (10.17) reduziert sich zu:

$$p'(\varphi) = -x \sin \varphi + y \cos \varphi. \quad (10.18)$$

Beide Gleichungen (10.16), (10.18) lösen wir nach $x(\varphi), y(\varphi)$ auf:

$$\begin{aligned} x(\varphi) &= p(\varphi) \cos \varphi - p'(\varphi) \sin \varphi \\ y(\varphi) &= p(\varphi) \sin \varphi + p'(\varphi) \cos \varphi. \end{aligned} \quad (10.19)$$

Dies ist die gewünschte Parameterdarstellung für die Kontur der Konvexität. Da die Stützfunktion mit 2π periodisch ist, sind auch alle Ableitungen von $p(\varphi)$ periodisch in 2π . Aus der Stützfunktion lassen sich interessante geometrische Beziehungen für konvexe Figuren herleiten:

- Wir wollen den Umfang von K berechnen:

$$\begin{aligned} U_K &= \int_{\partial K} ds = \int_{\partial K} \sqrt{dx^2 + dy^2} = \int_0^{2\pi} (p(\varphi) + p''(\varphi)) d\varphi \\ &= \int_0^{2\pi} p(\varphi) d\varphi. \end{aligned} \quad (10.20)$$

Man beachte, dass infolge der Periodizität der Ableitungen gilt:

$$\int_0^{2\pi} p''(\varphi) d\varphi = p'(2\pi) - p'(0) = 0. \quad (10.21)$$

- Die Breite $b(\varphi)$ einer konvexen Figur bezüglich einer Richtung φ ist der Abstand der beiden parallelen Stützgeraden G und G' , die senkrecht zur Richtung φ verlaufen. Für die mittlere Breite B_K ergibt sich dann die verblüffend einfache Beziehung:

$$B_K = \frac{1}{2\pi} \int_0^{2\pi} b(\varphi) d\varphi = \frac{1}{2\pi} \int_0^{2\pi} (p(\varphi) + p(\varphi + \pi)) d\varphi = \frac{U_K}{\pi}. \quad (10.22)$$

Man kann diese Formel auch interpretieren: Die „Anzahl“ $N_K(\varphi)$ der Geraden, die senkrecht zur Richtung φ verlaufen und die konvexe Figur K schneiden, ist proportional zur Breite $b(\varphi)$. Also ist B_K bzw. U_K bis auf einen Faktor ein Maß für die „Anzahl“ aller die Figur K schneidenden Geraden. Für die *Stereologie* ist dies von besonderer Bedeutung, da diese Schnitte im 3D Raum den Schnitten von Ebenen mit Körpern entsprechen, d. h. sie haben etwas mit Anschliffen oder Anschnitten zu tun.

- Auch die Fläche einer konvexen Figur kann man durch die Stützfunktion ausdrücken:

$$\begin{aligned} F_K &= \frac{1}{2} \int_{\partial K} p(\varphi) ds = \frac{1}{2} \int_0^{2\pi} p(\varphi) [p(\varphi) + p''(\varphi)] d\varphi \\ &= \frac{1}{2} \int_0^{2\pi} [p(\varphi)^2 - p'(\varphi)^2] d\varphi. \end{aligned} \quad (10.23)$$

- Nun bleibt nur noch die Frage, wie berechnet man numerisch die Stützfunktion? Natürlich mit der Hough-Transformation. Um Verwechslungen zu vermeiden, bezeichnen wir jetzt die p -Koordinate im Hough-Raum mit P . Wir berechnen für jeden Konturpunkt der konvexen Figur die Kurve $P(\varphi)$ und akkumulieren längs dieser Kurve im Hough-Raum. Den Akkumulatorwert pro Koordinate (P, φ) bezeichnen wir mit $A(P, \varphi)$. Die Stützfunktion sfkt ergibt sich dann zu:

$$\text{sfkt}(\varphi) = \max \{ P \mid A(P, \varphi) > 0 \}. \quad (10.24)$$

Die Stützfunktion ist somit die obere Randfunktion aller positiven Akkumulatorwerte und hat nichts mit den lokalen Maxima im Houghraum zu tun.

10.8.5 Verallgemeinerungen

Andere geometrische Primitiva Nicht nur Geraden können mit der primären oder dualen Akkumulatormethode detektiert werden, sondern auch andere geometrische Primitiva. Im Allgemeinen erhöht sich dann die Dimension des Akkumulatorraumes, sodass die Zeiteffizienz zum Hauptproblem wird. Nehmen wir als Beispiel die Detektion von Kreisen:

1. **Primale Methode:** Ein Kreis mit den Parametern Mittelpunkt (x_0, y_0) und Radius r ist eindeutig durch drei Punkte beschrieben. Daher wählen wir alle Tripel von Punkten oder via RANSAC eine zufällige Auswahl davon, bestimmen jeweils den Mittelpunkt und den Radius des Kreises und akkumulieren an diesem Punkt (x_0, y_0, r) im dreidimensionalen Akkumulatorraum.
2. **Duale Methode (Hough-Transformation):** Wir wählen wieder die Punkte einzeln und akkumulieren im dreidimensionalen Houghraum längs der dualen Parameterglei-

chung. Diese lautet bei einem fest vorgegebenem Punkt (x, y) :

$$(x - x_0)^2 + (y - y_0)^2 = r^2. \quad (10.25)$$

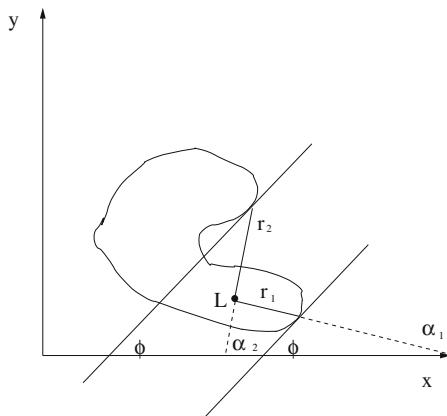
Man sieht, (10.25) beschreibt bei festen x, y in den Unbekannten x_0, y_0, r einen *Doppelkegel*. Daher ist im Akkumulatorraum (x_0, y_0, r) längs dieses Doppelkegels zu inkrementieren.

Haben wir A-priori-Wissen zur Verfügung, dann können wir den Akkumulatorraum einschränken. Wissen wir z. B., dass wir nur Kreise eines bestimmten festen Radius R suchen, dann besteht der Akkumulatorraum nur aus den Mittelpunkten x_0, y_0 und ist demzufolge nur zweidimensional.

Beliebige Freiformobjekte Bei beliebigen Freiformobjekten wird die Akkumulation noch aufwendiger. Da wir das Referenzobjekt nicht in analytischer Form beschreiben können, müssen wir es in tabellarischer Form versuchen. Zunächst legen wir einen Referenzpunkt L im Inneren des Objektes fest. Dann werden in eine Tabelle, die „R-table“ genannt wird, die Stützgeraden „eingetragen“. Dies sind aber keine „echten Stützgeraden“, da das Objekt nicht konvex sein muss. Für eine Richtung φ kann es mehrere „Stützgeraden“ geben und für alle diese werden in die Tabelle nicht die Abstände p der Stützgeraden zum Koordinatenursprung O eingetragen, sondern der Abstand $r(\varphi)$ des Konturpunktes zum Referenzpunkt L und der Winkel $\alpha(\varphi)$ der Verbindungsgeraden zwischen dem Referenzpunkt L und dem Konturpunkt. Die Idee liegt darin, dass wir allein aus dem Winkel φ der Stützgeraden und dem Konturpunkt den Referenzpunkt L wieder berechnen können, siehe Abb. 10.20. Nun werden die Konturpunkte eines unbekannten Objektes bestimmt. Man kann davon ausgehen, dass die Gradientenrichtung des Konturpunktes bekannt ist und damit auch der Winkel φ . Nun berechnen wir in dieser Richtung vom Konturpunkt ausgehend mit den Informationen der „R-table“ einen Referenzpunkt. Wenn in der R-Table für diese Richtung mehrere Einträge vorhanden sind, müssen wir eben mehrere Referenzpunkte berechnen. Der Akkumulationsraum hat die Dimension der Referenzpunkte, ist also zweidimensional. Nun inkrementieren wir im Akkumulationsraum an den Koordinaten der berechneten Referenzpunkte eine Eins. Wenn wir für alle Konturpunkte eines unbekannten Objektes die berechneten Referenzpunkte akkumuliert haben, dann muss sich, wenn das unbekannte Objekt das Referenzobjekt ist, ein typischer Peak für den Referenzpunkt L ausbilden. Diese Akkumulationsmethode heißt *Generalized Hough Transform*.

Es kommt sicher jetzt die Frage auf, warum ist der Akkumulatorraum nur zweidimensional obwohl bei einfacheren Objekten wie Kreisen der Akkumulatorraum sogar dreidimensional ist? Man erkennt sofort aus der Abb. 10.20, dass die gesuchten Objekte gegenüber dem Referenzobjekt nicht einmal rotiert oder skaliert sein dürfen. Es gilt lediglich die Invarianz gegenüber Verschiebungen. Kreise durften sogar skaliert sein, dies beschreibt der gesuchte Radius. Natürlich kann man die *Generalized Hough Transform* diesbezüglich noch verallgemeinern, der Aufwand steigt aber beträchtlich an.

Abb. 10.20 Generalized Hough-Transform



Radon-Transformation Eine völlig andere Verallgemeinerung der Akkumulation ist die Übertragung auf Grautonbilder. Bisher sind wir davon ausgegangen, dass wir die Koordinaten einzelner Punkte zur Verfügung haben, insbesondere wenn das Bild binarisiert ist, oder wenn mit anderen Deskriptoren einzelne Punkte detektiert worden sind. Wenn wir aber direkt auf dem Grautonbild arbeiten wollen, dann bietet die **Radon-Transformation** eine Möglichkeit der Akkumulation. Statt einzelner Punkte betrachten wir alle diejenigen Geraden (p, φ) , die das gegebene Bild „schneiden“. Längs dieser Geraden summieren (integrieren) wir alle Grauwerte und bezeichnen die Summe (Integral) mit $I(p, \varphi)$. Wenn wieder eine Geradengleichung in der Hesseschen Normalform $p = x \cos \varphi + y \sin \varphi$ gegeben ist, dann ist

$$x(t) = p \cos \varphi - t \sin \varphi, \quad y(t) = p \sin \varphi - t \cos \varphi \quad (10.26)$$

eine Parameterdarstellung dieser Geraden. Die Summe entlang einer Geraden lässt sich als Kurvenintegral schreiben und wir benötigen dafür das differentiale Bogenelement:

$$ds = \sqrt{dx^2 + dy^2} \rightarrow ds = dt. \quad (10.27)$$

Nun können wir das Kurvenintegral längs dieser Geraden berechnen:

$$\begin{aligned} I(p, \varphi) &= \int_{\text{Gerade}} f(x, y) ds \\ &= \int_{-\infty}^{+\infty} f(p \cdot \cos \varphi - t \cdot \sin \varphi, p \cdot \sin \varphi - t \cdot \cos \varphi) dt. \end{aligned} \quad (10.28)$$

In der Praxis wird man $0 \leq \varphi \leq 2\pi$ festlegen und p so wählen, dass die Geraden auch das Bild „schneiden“. Für theoretische Betrachtungen kann man $0 \leq p \leq \infty$ setzen und erhält somit eine Integraltransformation, genannt Radon-Transformation. Wenn wir diese auf ein

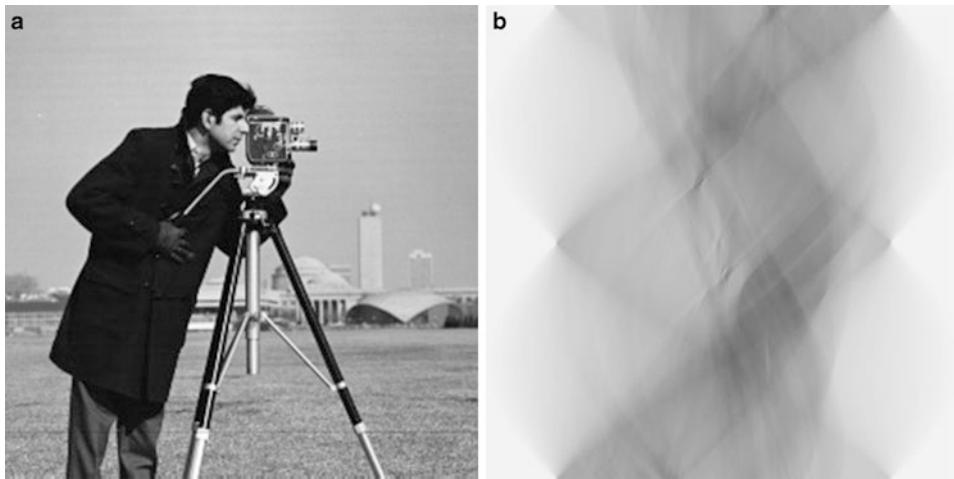


Abb. 10.21 a Originalbild, b Radon-Transformierte

Binärbild anwenden, dann ist diese identisch mit der Hough Transformation aller Objektpunkte. In Abb. 10.21a ist ein Bild und in Abb. 10.21b dessen Radon-Transformierte zu sehen. Die Radon-Transformation besitzt große Bedeutung in der Computertomographie. Es werden „Schnittebenen“ durch den Körper berechnet, wobei pro Ebene die Absorptionsraten von Röntgenstrahlen gemessen werden. Dies geschieht pro Ebene aus verschiedenen Richtungen, so dass pro Ebene die Funktion $I(p, \varphi)$ vorliegt. Pro Ebene muss man nun aus der Funktion $I(p, \varphi)$ die inverse Funktion $f(x, y)$ berechnen. Viele Ebenen im Körper mit den Grauwertbelegungen $f(x, y)$ ergeben das 3D-Computertomographiebild. Wie kann man aber die inverse Funktion $f(x, y)$ aus $I(p, \varphi)$ berechnen? Dazu nutzt man das *Fourier-Slice-Theorem*, siehe Abschn. 16.1. Es gibt auch die Methode der gefilterten Rückprojektion, die auch als Akkumulation deutbar ist, siehe Abschn. 16.1.

Oft werden in der Bildverarbeitung nur Grauwertbilder betrachtet. Mit der Entwicklung billiger Farbkameras ist es heute üblich auch Farbbilder zu betrachten, da die Farbe zusätzliche Informationen liefert. In diesem Zusammenhang hat sich der Begriff *Color Vision* herausgebildet. Im Zusammenhang mit Farben treten eine Menge von elementaren Grundbegriffen auf. Oft spricht man von Rot, Grün- und Blauauszügen, damit meint man eigentlich schon das RGB-Farbmodell. Es werden folglich drei Kanäle für ein Farbbild benötigt, d. h. pro Pixel benötigt man drei „Grautöne“. Dies ergibt mathematisch pro Pixel einen dreidimensionalen Vektor, folglich ist ein Farbbild ein Vektorfeld und man kann sich der Methoden aus der Vektoranalysis bedienen. Weiterhin kann man ein Farbbild als spezielles mehrkanaliges Bild auffassen, wobei für die Anzahl der Kanäle $n = 3$ gilt. Es gibt auch tatsächlich Multispektralkameras, die Bilder mit $n > 3$ liefern. Häufig wird auch der Begriff *Echtfarbendarstellung* benutzt. In der Computergrafik wird dieser Begriff benutzt, wenn pro Pixel 24 Bit zur Verfügung stehen, damit sind ca. 16 Millionen Farben darstellbar, und zwar alle, die das menschliche Auge unterscheiden kann. Ein *Falschfarbenbild* entspricht im Wesentlichen einem Echtfarbenbild, jedoch mit dem Unterschied, dass dem Rot-, Grün- und Blaukanal beliebige Wellenbereiche zugeordnet werden. Ein *Pseudofarbbild* ist dagegen etwas völlig anderes. Hier haben wir ein Grautonbild zur Verfügung, bei dem bestimmte Pixel oder Grauwerte künstlich eingefärbt werden, z. B. durch setzen von Farbtabellen. Diese Pseudofarbbilder treten häufig im Zusammenhang mit *overlay*-Bildern oder Markierungsbildern auf. Diese Markierungsbilder haben zwei Funktionen:

- Eine „Verwaltungsfunktion“: Sie markieren bestimmte Pixel als „bereits abgearbeitet“.
- Eine „Visualisierungsfunktion“: Die „abgearbeiteten“ Pixel sollen farblich so dargestellt werden, dass sie sich deutlich vom Untergrund abheben. Die Grafik soll dann dieses Pseudofarbbild darstellen.

11.1 Spektrale Farben

Wenn Licht als elektromagnetische Strahlung auf einen Körper trifft, werden bestimmte Wellenbereiche absorbiert und bestimmte reflektiert. Das menschliche Sehsystem reagiert darauf und ordnet diesem Farben zu. Diese Farben werden *Körperfarben* genannt. Im Gegensatz dazu unterscheidet man Farben von *selbstleuchtenden Lichtquellen*. Für das menschliche Auge kommt nur das sichtbare Licht mit einer Wellenlänge λ von 380 nm bis 780 nm (Nanometer, $1\text{nm} = 10^{-9}\text{m}$) in Frage. Dies entspricht einer Frequenz ν zwischen $4,3 \cdot 10^{14}$ und $7,5 \cdot 10^{14}$ Hz. Dieser Zusammenhang ist durch

$$\lambda \cdot \nu = c, \quad c = 2,9979246 \cdot 10^8 \text{ m/s} \quad (11.1)$$

gegeben. In der Abb. 11.1 ist die spektrale Verteilung der elektromagnetischen Strahlen zu sehen. Die spektralen Farben des sichtbaren Lichtes erhält man durch Zerlegung des Sonnenlichtes (weißes Licht) mit Hilfe eines Prismas.

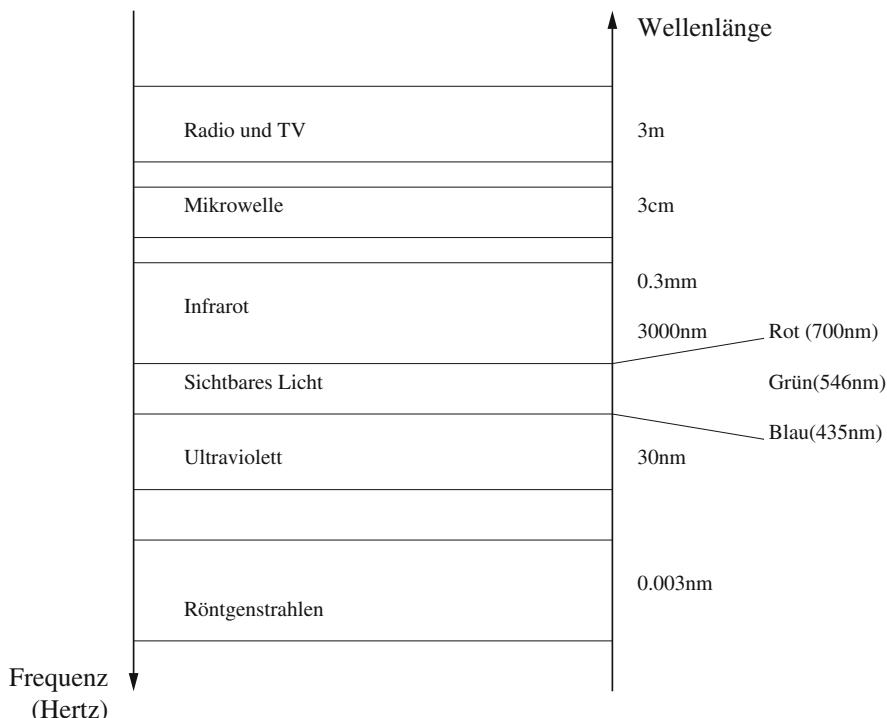


Abb. 11.1 Spektrale Verteilung

11.2 Visuelle Farbwahrnehmung

Das optische System des Auges – der dioptische Apparat – ist ein zusammengestztes, nicht ganz exakt zentriertes Linsensystem, das auf der Netzhaut ein umgekehrtes und stark verkleinertes Bild der Umwelt entwirft. Die Netzhaut besteht aus Photorezeptoren, den Gliazellen, den Pigmentzellen und vier verschiedenen Klassen von Nervenzellen. Die Photorezeptoren lassen sich morphologisch in zwei Klassen einteilen: in die Stäbchen (ca. 120 Millionen) und in die Zapfen (ca. 6 Millionen). Das Sehen am Tage nennt man *photopisches Sehen*, die Zapfen dienen als Rezeptoren. Das Sehen in der Dämmerung nennt man *mesopisches Sehen*, dabei dienen Stäbchen und Zapfen gleichberechtigt als Rezeptoren. Das Sehen in einer sternklaren Nacht nennt man *skotopisches Sehen*, die Stäbchen dienen als Rezeptoren. Beim photopischen Sehen ist die Sehschärfe und das Farbensehen sehr gut ausgeprägt, während beim skotopischen Sehen die Sehschärfe geringer ist und keine Farben wahrgenommen werden können, es herrscht funktionale Farbenblindheit.

Die Farben unserer Umwelt teilen wir ein in bunte und unbunte. Die Grautöne von schwarz bis weiß werden als unbunte Farben bezeichnet. Ein normalsichtiger Mensch kann bei Tageslicht etwa 30 bis 40 Graustufen unterscheiden. Die bunten Farben lassen sich durch drei Komponenten charakterisieren:

- Der Farbton oder auch Bunton genannt (*hue*).
- Die Buntheit oder auch Sättigung genannt (*saturation*).
- Die Helligkeit oder Dunkelstufe genannt (*brightness*).

Der Farbton beschreibt die Art der Farbe wie z. B. rot, grün, blau, violett usw. Farbtöne können in einem Farbenkreis in geschlossener Reihe von rot, orange, gelb, grün, blau, violett, purpur bis der Kreis sich wieder bei rot schließt, dargestellt werden.

Die Sättigung gibt an wie „rein“ oder „ausgewaschen“ dem Menschen die Farbe erscheint. Je enger der Wellenzahlbereich der Farbe ist, desto „reiner“ erscheint sie. Monochromatisches Licht erzeugt die Empfindung von „total reinen“ Farben.

Die Helligkeit beschreibt die Stärke der Lichtempfindung. Die wahrgenommene Helligkeit entspricht aber nicht direkt der Energie, da das Auge auf verschiedene Wellenlängen die Helligkeit anders empfindet. Im „grünen“ Wellenzahlbereich ist das Auge am empfindlichsten. Alle Farbtöne des Farbenkreises können durch Spektralfarben oder durch additive Mischung von zwei Spektralfarben repräsentiert werden. Außerdem lässt sich zu jeder Farbe des Farbenkreises eine Komplementärfarbe finden, sodass deren additive Mischung die Farbe Weiß ergibt. So sind z. B. Rot und Cyan, Zitronengelb und Violett, Blau und Gelb, Grün und Magenta Komplementärfarben.

Auf der Retina im Auge findet eine additive Farbmischung statt. Mit drei geeigneten Grundfarben kann für den normal Farbtüchtigen jede Farbe, die durch selbstleuchtende Lichtquellen herstellbar ist, erzeugt werden. Die Wellenlängen der Grundfarben sind genormt, z. B. rot (700nm), grün (546 nm) und blau (435 nm). Die Farbwahrnehmung, basierend auf diesen drei Grundfarben, nennt man Trichromatie. Etwa 2% der Bevölke-

rung sind Dichromaten. Die wahrgenommenen Farben sind bei Dichromaten wesentlich weniger differenziert als bei Trichromaten.

Die häufigste „Farbkrankheit“ ist die „Rot-Grün-Blindheit“, d. h. diese Personen können nicht mehr Rot von Grün unterscheiden. Monochromat, also farbenblind, sind weniger als 0,005% der Bevölkerung.

Wenn man mit Farben umgeht, dann muss man auch Farben genormt benennen und bezeichnen können. Da es hier international andere Bezeichnungen gibt (z. B. herbstgolden oder profi-schwarz), muss man für wissenschaftliche Zwecke die Benennungen standardisieren. Dazu gibt es in der Computer Grafik ein englischsprachiges Farbbezeichnungssystem CNC (*Color Naming System*).

11.3 Farbtafeln

Bei additiver Farbmischung der drei Grundfarben RGB kann man nicht alle Farben, insbesondere reine Spektralfarben, durch Farbmischung erzeugen, weil man nichtnegative Farbwerte der drei Komponenten voraussetzt. Um dennoch alle Farben normieren zu können, wurden Farbsysteme mit virtuellen Primärvalenzen X, Y, Z entwickelt. Das im Jahre 1931 festgelegte CIE-Farbmesssystem nutzt solche virtuellen Primärvalenzen (CIE – Commission Internationale de l’Eclairage). Man benutzt dazu drei Normspektralwertkurven, die durch Mittelungen von genormten Beobachtungen bestimmt wurden und wobei jeder Spektralfarbe, d. h. jeder Wellenlänge λ ein Zahlentriple $u(\lambda), v(\lambda), w(\lambda)$ zuordnet. Dabei ist z. B. $v(\lambda)$ proportional zur Helligkeitsempfindlichkeitskurve des menschlichen Auges. Ist nun eine Strahlung gegeben, die einen Farbreiz im Auge hervorruft, wird sie als Farbreizfunktion $\varphi(\lambda)$ bezeichnet. Diese setzt sich zusammen aus:

$$\varphi(\lambda) = \begin{cases} S(\lambda) & \text{für Selbstleuchter} \\ S(\lambda) \cdot R(\lambda) & \text{für Körperfarben} \\ S(\lambda) \cdot R(\lambda) + S_F(\lambda) & \text{für Fluoreszenzfarben.} \end{cases} \quad (11.2)$$

Als Fluoreszenz bezeichnet man das Leuchten von Stoffen, wenn sie mit bestimmten Formen von Licht angestrahlt werden. Das Leuchten hört sofort auf, wenn die Lichtquelle versiegt. Phosphoreszenz ist ein ähnlicher Effekt, wenn allerdings die Lichtquelle versiegt, liegt ein längeres Nachleuchten vor. Die Normfarbwerte berechnet man nun zu:

$$\begin{aligned} X &= k \cdot \int_{380}^{780} \varphi(\lambda) \cdot u(\lambda) d\lambda, \\ Y &= k \cdot \int_{380}^{780} \varphi(\lambda) \cdot v(\lambda) d\lambda, \\ Z &= k \cdot \int_{380}^{780} \varphi(\lambda) \cdot w(\lambda) d\lambda, \end{aligned} \quad (11.3)$$

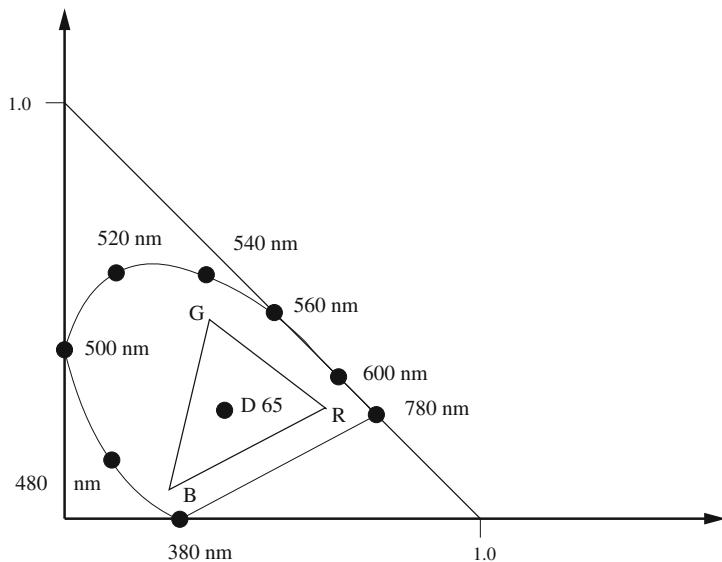


Abb. 11.2 Normfarbwerttafel-CIE Farbsystem

wobei der Faktor k zu Normierungszwecken benutzt wird (vollkommen mattweißer Körper). Die Darstellung in einer Normfarbwerttafel geschieht allerdings nur zweidimensional ohne die Information über die Helligkeit durch die Normierung:

$$x = \frac{X}{X + Y + Z}, \quad y = \frac{Y}{X + Y + Z}, \quad z = \frac{Z}{X + Y + Z}. \quad (11.4)$$

Damit gilt $x + y + z = 1$ und es sind nur noch zwei Koordinaten notwendig. Geometrisch bedeutet dies, wir projizieren die dreidimensionalen Punkte (X, Y, Z) in die Ebene $x + y + z = 1$. Für $x \geq 0, y \geq 0, z \geq 0$ bildet diese Ebene ein Dreieck, wobei jede Achse bei $x = 1, y = 1$ und $z = 1$ geschnitten wird. Dies hat aber den Nachteil, dass wir aus den Punkten der Ebene die originalen Punkte (X, Y, Z) nicht eindeutig rekonstruieren können. Dazu muss man eine der Koordinaten, z. B. die Helligkeit Y , vorgeben. Dazu lösen wir das homogene lineare Gleichungssystem (11.4) nach X und Z in Abhängigkeit von Y auf:

$$X = x \cdot \frac{Y}{y}, \quad Y = y \cdot \frac{Y}{y}, \quad Z = z \cdot \frac{Y}{y}. \quad (11.5)$$

Zur grafischen Darstellung des Dreiecks $x + y + z = 1, x \geq 0, y \geq 0, z \geq 0$ projizieren wir das Dreieck in die x, y -Ebene und stellen das projizierte Dreieck dar. Dies ist die „übliche“ Darstellung der Normfarbtafel, siehe Abb. 11.2. Auf der Kontur der Region liegen die „reinen“ Spektralfarben, die untere Gerade wird als Purpurgerade bezeichnet. Alle anderen Farbtarten liegen im Inneren der Region. In der Mitte ist ein normierter Punkt, das

Normweiß, eingetragen. Dieser wird auch als Unbuntpunkt bezeichnet. Betrachtet man eine Gerade, die durch den Unbuntpunkt geht, dann schneidet diese die Kontur der Region in zwei Punkten. Diese beiden Schnittpunkte bilden *Komplementärfarben*. Wählt man nun drei Grundfarben aus, z. B. Rot, Grün und Blau, dann kann man nach dem Prinzip der additiven Farbmischung alle Farben innerhalb des Dreieckes erzeugen, welches die drei Farbpunkte aufspannen, siehe Abb. 11.2. Man sieht, dass dadurch tatsächlich nicht alle Farben erzeugbar sind. Dieses Dreieck wird auch als *Maxwellsches Farbdreieck* bezeichnet.

In der CIE-Farbtafel werden quasi die Bunntöne in linearer Abhängigkeit von der Wellenlänge dargestellt. Diese „Linearität“ stimmt aber nicht mit der menschlichen Empfindung von Bunton und Sättigung überein, z. B. kann der Mensch im Grünbereich viel kleinere Farbnuancen unterscheiden als im Blaubereich. Zur Verbesserung dieser „Nichtlinearitäten“ wurden andere Farbtafeln bzw. Farträume entwickelt, sogenannte uniforme Farbenräume. Die CIE hat daher bereits 1976 den LUV und den LAB-Farbenraum empfohlen. Dafür haben sich die Bezeichnungen CIELAB und CIELUV eingebürgert. Der CIELAB-Raum wird zur Beschreibung von Körperfarben benutzt, der CIELUV-Raum dagegen zur Beschreibung von Lichtfarben.

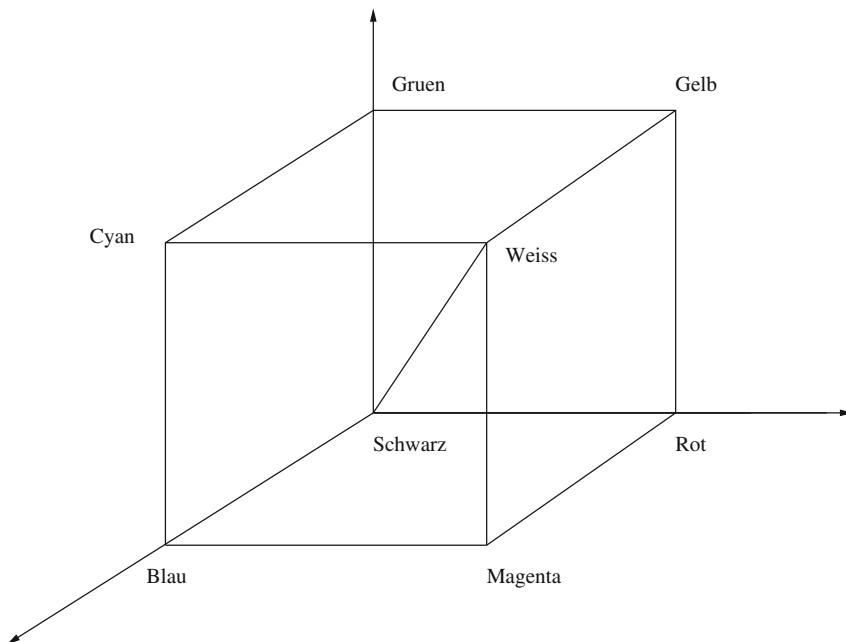
11.4 Farbmodelle

11.4.1 Technikorientierte Farbmodelle

RGB-Modell Dies ist wohl das bekannteste Farbmodell. Man legt drei Grundfarben für Rot, Grün und Blau in der CIE-Normfarbtafel fest und kann dann alle Farben innerhalb dieses Dreieckes additiv aus den drei Grundfarben mischen. Bei diesem Modell werden Lichtintensitäten addiert, damit ergibt die additive Mischung aus gleichen Anteilen von Rot, Grün und Blau die Farbe Weiß. Die drei Primärfarben können wir als Basisvektoren eines dreidimensionalen Farb-Vektorraumes auffassen, siehe Abb. 11.3. Der Ursprung des Würfels ist der Schwarzpunkt, auf der Hauptdiagonalen liegen alle unbunten Farben, die Grautöne. Auf RGB-Systemen basieren z. B. die Farbfernsehempfänger. Für das PAL-System und das NTSC-System werden verschiedene Primärvalenzen und verschiedene Referenzweisse benutzt, welche in Tab. 11.1 dargestellt sind.

Es soll nun kurz die Beziehung zwischen CIE-System und RGB-System für das PAL-System nach obiger Tabelle demonstriert werden, siehe auch [51]. Die z -Werte für die Koordinaten sind aus $x+y+z=1$ zu entnehmen. Folglich wird das Normweiß durch $x = 313$, $y = 0,329$ und $z = 0,358$ beschrieben. Bei einer vorzugebenden Luminanz $Y_w = 1,000$ ergeben sich nach (11.5) die CIE-Koordinaten zu $X_w = 0,951$, $Y_w = 1,000$, $Z_w = 1,088$. Die Koordinaten der drei Grundfarben bezeichnen wir mit (X_i, Y_i, Z_i) , $i = r, g, b$. Die Vektoraddition dieser drei Grundfarben ergibt gerade das Normweiß, also:

$$\begin{aligned} X_w &= X_r + X_g + X_b \\ Y_w &= Y_r + Y_g + Y_b \\ Z_w &= Z_r + Z_g + Z_b. \end{aligned} \tag{11.6}$$

**Abb. 11.3** RGB-Einheitswürfel

Aus dieser Beziehung wollen wir die Luminanzen der drei Grundfarben ausrechnen. Mit (11.4) erhalten wir dann:

$$\begin{aligned} X_w &= x_r \frac{Y_r}{y_r} + x_g \frac{Y_g}{y_g} + x_b \frac{Y_b}{y_b} \\ Y_w &= y_r \frac{Y_r}{y_r} + y_g \frac{Y_g}{y_g} + y_b \frac{Y_b}{y_b} \\ Z_w &= z_r \frac{Y_r}{y_r} + z_g \frac{Y_g}{y_g} + z_b \frac{Y_b}{y_b} \end{aligned} \quad (11.7)$$

Tab. 11.1 PAL- und NTSC-System

	NTSC Primärfarben	NTSC Primärfarben	PAL Primärfarben	PAL Primärfarben
CIE	x	y	x	y
Rot	0,67	0,33	0,64	0,33
Grün	0,21	0,71	0,29	0,60
Blau	0,14	0,08	0,15	0,06
	NTSC Normlichtart C	NTSC Normlichtart C	PAL D65 Normlichtart	PAL D65 Normlichtart
Weiß	0,310	0,316	0,313	0,329

Tab. 11.2 sRGB-Standard

	sRGB	sRGB
	Primärfarben	Primärfarben
CIE	x	y
Rot	0,6400	0,3300
Grün	0,3000	0,6000
Blau	0,1500	0,0600
	Normlichtart C	Normlichtart C
	D65	D65
Weiß	0,3127	0,3290

Der gesuchte Lösungsvektor ergibt sich aus dem linearen Gleichungssystem:

$$\begin{aligned} 0,951 &= 1,939 Y_r + 0,483 Y_g + 2,500 Y_b \\ 1,000 &= 1,000 Y_r + 1,000 Y_g + 1,000 Y_b \\ 1,088 &= 0,091 Y_r + 0,183 Y_g + 13,167 Y_b \end{aligned} \quad (11.8)$$

zu $Y_r = 0,223$, $Y_g = 0,706$, $Y_b = 0,071$. Ist nun eine Farbe im RGB-System gegeben, dann kann man durch Linearkombination der Basisvektoren die CIE-Koordinaten ausrechnen:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} + G \begin{pmatrix} X_g \\ Y_g \\ Z_g \end{pmatrix} + B \begin{pmatrix} X_b \\ Y_b \\ Z_b \end{pmatrix}. \quad (11.9)$$

Mit dieser Beziehung kann man auch invers aus den CIE-Koordinaten die RGB-Koordinaten ausrechnen.

Betrachten wir im RGB-Würfel den Schnitt dieses Würfels mit der Ebene $R + G + B = 1$, so ergibt sich das *Maxwellsche* Farbdreieck. Mit der Darstellung von RGB-Dateien an beliebigen Monitoren ergeben sich gewisse Standardisierungsprobleme, deshalb hat sich der sRGB-Standard herausgebildet (Tab. 11.2).

CMY-Modell Dem RGB-Modell liegt die additive Farbmischung zugrunde, dies lässt sich bei Monitoren, Projektoren usw. anwenden. Beim Drucken dagegen lässt sich dieses Modell nicht anwenden. Die Idee ist nun ähnlich der Farbfilter, die Farbschichten werden nacheinander aufgetragen und wirken als Farbfilter bezüglich des einfallenden weißen Lichtes. Natürlich ist es noch ein technisches Problem welche Farben als Filter dienen und wie dick/dünn sie aufgetragen werden müssen. Als Filter bieten sich die Komplementärfarben zu Rot, Grün und Blau an. Diese sind zu Rot Cyan, zu Grün Magenta und zu Blau Yellow, daher auch der Name CMY-Modell. Folglich gilt die simple Umrechnung

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} C \\ M \\ Y \end{pmatrix} \quad (11.10)$$

zum additiven RGB-Modell. Entsprechend dieser Beziehung spricht man auch vom *subtraktiven* Farbmodell im Gegensatz zum additiven RGB-Modell. Überlagert sich folglich rotes, grünes und blaues Licht, ergibt sich weißes Licht. Mischt man dagegen Rot, Grün und Blau als flüssige Farben, so ergibt sich Schwarz.

Beim Farbdruck wird oft noch eine weiter Grundfarbe verwendet, um die Farbenvielfalt zu erhöhen. Weiterhin wird Schwarz beigemischt, um den Kontrast etwas zu erhöhen.

Weitere technikorientierte Farbmodelle sind die der Fernsehübertragungstechniken. Die bekanntesten sind das YIQ- und das YUV-System.

11.4.2 Wahrnehmungsorientierte Farbmodelle

Diese Modelle orientieren sich an der menschlichen Farbwahrnehmung. Die Farben lassen sich wie die technikorientierten Modelle auch durch drei Größen charakterisieren:

- Den Farbton oder auch Bunnton genannt (*Hue*).
- Die Buntheit oder Sättigung (*Saturation*).
- Die Helligkeit (*Intensity*).

Daher wird für dieses Farbmodell der Name HSI-Modell benutzt. Farbe, Sättigung und Helligkeit werden in Zylinderkoordinaten dargestellt. Dabei wird die z -Achse zur Intensitätsachse, der Abstand zu dieser Achse ist dann die Sättigung, der Winkel (Polarkoordinaten in der Ebene) repräsentiert den Bunnton.

Die Grundidee besteht nun darin, die Unbuntachse im RGB-Würfel in die Intensitätsachse zu transformieren. Dazu rotieren wird das RGB-Koordinatensystem so, dass die neue Blauachse B' mit der Unbuntachse identisch ist. Eine weitere Forderung ist, dass die Achsen R , R' und B' komplanar sind. Damit sind zwei Eulersche Winkel eindeutig bestimmt:

- a) Eine Rotation des Würfels um 45° bezüglich der Rotachse R .
- b) Eine anschließende Rotation des Würfels um $-35,264^\circ$ bezüglich der (alten) Grünachse G .

Bezüglich absoluter und relativer 3D-Rotationen siehe den Abschn. 13.2. Folglich ergibt sich für die Rotationsmatrix:

$$\mathbf{R}_{\text{rot}} = \begin{pmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & -\frac{1}{\sqrt{3}} \\ 0 & 1 & 0 \\ \frac{1}{\sqrt{3}} & 0 & \frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix} \quad (11.11)$$

und damit:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \mathbf{R}_{\text{rot}} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \quad (11.12)$$

Einzelne aufgeschrieben, ergibt sich:

$$R' = \frac{1}{\sqrt{6}} (2R - G - B), \quad G' = \frac{1}{\sqrt{2}} (G - B), \quad B' = \frac{1}{\sqrt{3}} (R + G + B). \quad (11.13)$$

Die Zylinderkoordinaten bwezeichnen wir nun mit ρ , θ und h . Dann ist:

$$\rho = \sqrt{R'^2 + G'^2}, \quad \theta = \arctan\left(\frac{G'}{R'}\right), \quad h = B'. \quad (11.14)$$

Nun setzen wir den Farbton H gleich $H = \theta$, die Intensität I gleich $I = \frac{\sqrt{3}}{3}h$ und für die Sättigung könnten wir eigentlich ρ benutzen. Nun geht aber leider in diese Definition der Sättigung die Intensität mit ein, denn: wir nehmen einmal an, wir hätten die reine Farbe $(R, G, B) = (0, 4, 0)$, dann hat die reine Farbe $(R, G, B) = (0, 6, 0)$ doch offensichtlich die gleiche Sättigung, nur die Intensität hat sich geändert. Bei der obigen Definition der Sättigung hätte sich aber der Abstand zur Unbuntachse vergrößert, womit die Sättigung größer wäre. Wir müssen folglich die Sättigung unabhängig von der Intensität definieren, z. B. zu $S = \frac{\rho}{\rho_{\max}}$. Dies entspricht dann der Beziehung

$$S = 1 - \frac{\min(R, G, B)}{R + G + B}. \quad (11.15)$$

Das HSI-Modell ist typisch für die Bildverarbeitung, während in der Computergrafik häufig die HSV- und HLS-Modelle benutzt werden. Diese kann man als Approximation des HSI-Modells auffassen, damit ergibt sich ein etwas geringererer Rechenaufwand für die nichtlinearen Transformationen.

11.5 Operationen auf Farbbildern

Wie bei der „gewöhnlichen“ Bildverarbeitung, sollen auch alle Operationen auf Farbbilder anwendbar sein. Da Farbbilder Zusatzinformationen tragen, gilt es geschickt diese auszunutzen. Die einfachsten Fälle sind die folgenden:

- Man verarbeitet z. B. beim HSI-Modell nur die Helligkeitskomponente I als gewöhnliches Grautonbild. Oder man sucht z. B. „reine“ Farbkanten in der Farbtonkomponente H .
- Man verarbeitet z. B. beim RGB-Modell irgendeinen Auszug, oder alle drei Auszüge getrennt, und vereinigt zum Schluss die Ergebnisse.

Eine etwas aufwendigere Version ist die, Farbbilder als Vektorfunktionen zu verarbeiten. So kann man z. B. Kanten durch Richtungsableitungen in Vektorfunktionen finden, siehe z. B.

Abschn. 8.2.1. Damit im Zusammenhang steht auch der Strukturtensor für Farbbilder, genannt Farbtensor, siehe Abschn. 17.5.2. Das Histogramm eines „normalen“ Grautonbildes ist eindimensional, das Histogramm eines Farbbildes folglich dreidimensional. Vergleicht man folglich Farbhistogramme, entsteht ein großer numerischer Aufwand. Daher werden Farbhistogramme oft einer Clusterung unterworfen, das Clusterergebnis nennt man eine Signatur, siehe Abschn. 19.8.3. Diese Signaturen kann man nun mit geeigneten Abstandsmaßen vergleichen, z. B. der Earth Mover’s Distance (EMD), siehe auch Abschn. 19.8.3.

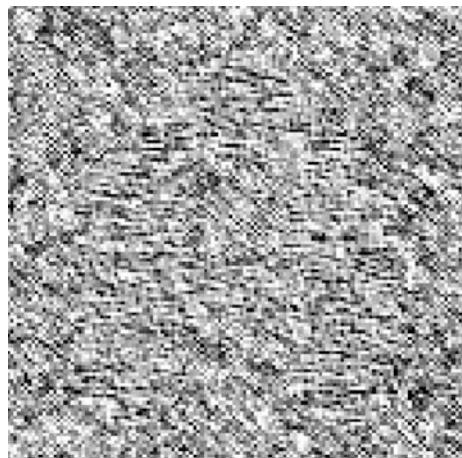
Weitere praktische Anwendungen sind das Auffinden von Farbfehlern und die Farbkalibrierung, siehe dazu [34].

12.1 Einführung

Textur ist ein Begriff, den man den Eigenschaften der Oberfläche eines Objektes zuschreibt. Dieser Begriff ist nicht klar mathematisch definierbar, aber man hat intuitiv eine Vorstellung von einer Textur. Eine Textur kann aus Texturprimitiva oder Texturelementen zusammengesetzt sein, oft auch Texel genannt. Viele Beispiele für Texturen findet man im Brodatz-Album [11]. Eine Texturbeschreibung ist auf jeden Fall skalenabhängig. Vergrößern wir z. B. eine Textur so, dass im Ausschnitt nur noch ein Texel zu sehen ist, dann würden wir dies nicht mehr als Textur bezeichnen. Verkleinern wir dagegen die Textur extrem, dass die Texel kaum noch aufgelöst werden, haben wir den Eindruck eines verrauschten Bildes. Verrauschten Bildern ordnen wir aber intuitiv keine Textur mehr zu. Texturen aus Texeln sind strukturierte, geordnete Texturen. Außerdem gibt es noch stochastische Texturen. Diese sind irregulär, aber dennoch visuell homogen. Hauptziele in der Bildverarbeitung von Texturen sind:

- Texturerkennung, Texturzuordnung bzw. Texturklassifikation. Häufig ist diese Aufgabenstellung bei Satelliten- und Luftaufnahmen zu finden. Dabei ist grundsätzlich zu beachten, ob die Erkennung in einem gewissen Bereich skalierungs invariant oder rotations invariant ist.
- Textursegmentierung. Ein Bild enthält mehrere verschiedene Texturen und es sind die Texturgrenzen zu ermitteln. Oft sind auch Störungen in Texturen bei industriellen Produkten zu segmentieren, obwohl die Störungen selbst keine Texturen sein müssen. In Abb. 12.1 ist ein „A“ als Textur in einer anderen Textur enthalten. In der Abb. 12.2 sind vom Bild 12.1 verschiedene Texturmerkmale pro Pixel berechnet und visualisiert worden. Anhand dieser Bilder kann man nun die eigentliche Segmentierung durchführen.
- *Shape from texture*. Nichtstochastische Texturen werden auf die Oberfläche eines 3D-Objektes projiziert. Aus der Formänderung der Textur (Texturgradient) kann man Tiefeinformationen gewinnen.

Abb. 12.1 Zwei verschiedene Texturen: die Textur des „A“ unterscheidet sich von der Textur des „Untergrundes“



Umgangssprachlich ordnet man Texturen Eigenschaften zu wie: körnig, glatt, länglich, fein usw. Mit diesen Eigenschaften können wir aber algorithmisch nichts anfangen. Zur Texturbeschreibung gibt es im Wesentlichen drei Zugänge:

- Statistische Texturbeschreibung.
- Syntaktische oder Strukturelle Texturbeschreibung.
- Hybride Methoden der Texturbeschreibung.

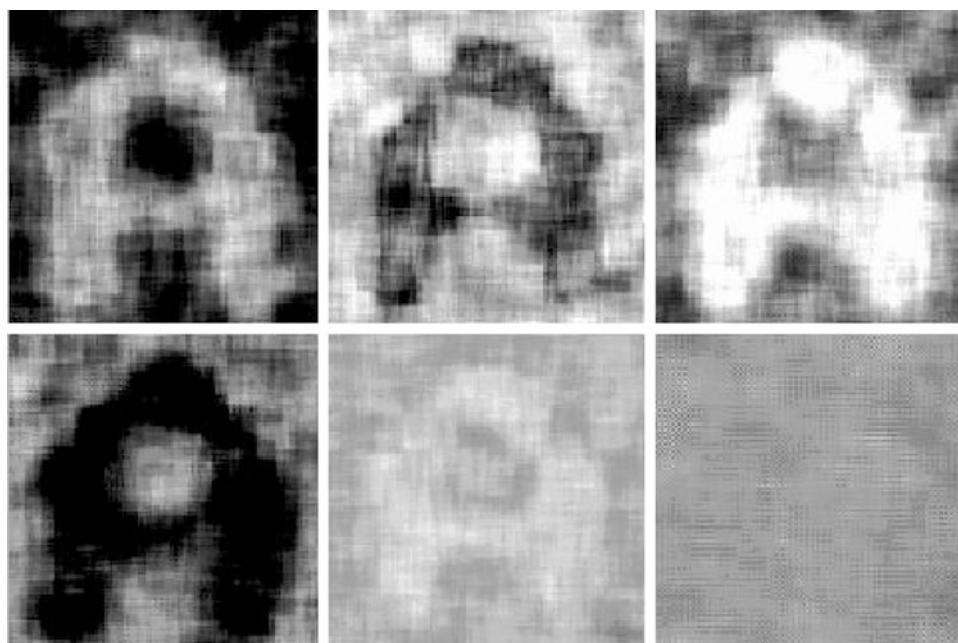


Abb. 12.2 Sechs verschiedene Texturmerkmale sind in sechs Bildern dargestellt

Im Folgenden soll aber nur auf elementare statistische Merkmale eingegangen werden. Zusätzlich werden wir autoregressive Prozesse vorstellen, weil man damit synthetische Texturen generieren kann.

12.2 Elementare statistische Merkmale

Im Folgenden sollen Texturen durch elementare statistische Merkmale beschrieben werden.

- Einen ersten, elementaren Merkmalsvektor erhalten wir, indem wir die Autokorrelationsfunktion (siehe auch (2.30)) eines Texturbildes berechnen, das nur eine Textur enthält. Da die übliche periodische Fortsetzung dem Modell nicht entspricht, „kürzen“ wir einfach die Berechnung und ersparen uns die periodische Fortsetzung. Dadurch werden die Funktionswerte bei größeren Verschiebungen immer unsicherer. Wenn wir noch geeignet normieren, dann erhalten wir eine „Pseudoautokorrelation“:

$$(\text{pseudo-auto})_{p,q} = \frac{MN}{(M-p)(N-q)} \frac{\sum_{i=1}^{M-p} \sum_{j=1}^{N-q} f_{i+p,j+q} f_{i,j}}{\sum_{i=1}^M \sum_{j=1}^N f_{i,j}^2}. \quad (12.1)$$

- Die klassischen, statistischen Texturmerkmale werden bei Haralick [30] beschrieben und beziehen sich auf die Statistik zweiter Ordnung. Ein zweidimensionales Grauwert-Histogramm ist die klassische *co-occurrence-matrix*, siehe Abschn. 9.2. Dabei bezieht sich die Statistik auf benachbarte Paare von Pixeln, wobei noch verschiedene Entfernung und Richtungen zu berücksichtigen sind. Elementare Merkmale sind dann z. B. die Energie, die Entropie, der Kontrast, die Normierte Kovarianz (Korrelation) usw., siehe [30].
- Ein einfaches Maß für die „Löchrigkeit“ einer binären Textur, genannt *lacunarity*, wird folgendermaßen berechnet. Wir wählen ein Quadrat der Seitenlänge l und schieben dieses Quadrat über das Bild, so ähnlich wie bei den linearen Filtern. Wir zählen jedesmal, wieviele schwarze Punkte der binären Textur in dem Quadrat liegen. Nach einem Bilddurchlauf berechnen wir von diesen Zählwerten den Mittelwert μ_l und die Standardabweichung σ_l . Wir normieren ähnlich den Variationskoeffizienten und berechnen $\text{lacunarity}(l) = \left(\frac{\sigma_l}{\mu_l}\right)^2$. Nun wiederholen wir dies für ein Quadrat mit einer anderen Seitenlänge l , wobei wir l systematisch in einem gewissen Bereich vergrößern bzw. verkleinern, wobei l insgesamt M verschiedene Werte annehme. Damit erhalten wir eine *lacunarity*-Funktion. Da die Beurteilung einer Textur mit Funktionen etwas schwierig ist, nimmt man einfach den Mittelwert aller Funktionswerte. Folglich wird die eigentliche *lacunarity* als Mittelwert

$$\overline{\text{lacunarity}} = \frac{1}{M} \sum_l \text{lacunarity}(l) \quad (12.2)$$

berechnet. Je „gleichmäßiger“ eine Textur ist, umso kleiner ist das Maß. Große Werte deuten auf große Lücken in der binären Textur hin. Dieses Maß wird oft in der Medizin

zur numerischen Beurteilung von Gefäßstrukturen herangezogen, ähnlich der fraktalen Kästchendimension. Für eine schnelle Implementierung dieses Maßes können wir wieder die Mittelwertfilter nutzen, da ein Mittelwertfilter bis auf die Skalierung in einem Binärbild die Anzahl der schwarzen Punkte zählt, siehe auch Abschn. 8.1.2.

12.3 Autoregressive Prozesse (AR)

Wir stellen uns ein eindimensionales Bild vor, d. h. eine Folge von Zufallsvariablen X_n , $n = 0, \dots, N - 1$, wobei wir uns die Entstehung des Bildes von „links“ nachts „rechts“ vorstellen. Diese Entstehung beschreiben wir durch einen autoregressiven Prozess

$$X_i = \sum_{k=1}^M a_k X_{i-k} + N_i, \quad i = M, \dots, N - 1. \quad (12.3)$$

Dabei sind geeignete Anfangsbedingungen zu wählen und die N_i sind unabhängige, mittelwertfreie Zufallsvariablen. Der Prozess ist damit kausal und hängt somit nur von der „zeitlich“ vorangegangenen Umgebung ab. Dieses autoregressive Modell ist geeignet synthetische Texturen zu erzeugen. Allerdings muss man nun diese „zeitliche“ Entstehung auf zweidimensionale Bilder verallgemeinern. Als Beispiel betrachten wir das einfache Modell in Abb. 12.3. Der jeweils aktuelle Grauwert g lässt sich aus den vier Grauwerten g_{lo}, g_o, g_{ro}, g_l durch zeilenweise Bearbeitung des Bildes bestimmen. Als Startwerte dieses zweidimensionalen Modells dienen die Grauwerte der ersten Bildzeile und der beiden Randspalten des Bildes. Die Berechnung erfolgt dann durch

$$g_{i,j} = a_{lo}g_{i-1,j-1} + a_og_{i,j-1} + a_{ro}g_{i+1,j-1} + a_lg_{i-1,j} + n_{i,j}. \quad (12.4)$$

Der Term $n_{i,j}$ ist eine Realisierung einer mittelwertfreien Zufallsvariablen $N_{i,j}$, die durch Pseudozufallszahlen zu erzeugen sind. Ebenso sei $g_{i,j}$ eine Realisierung der Zufallsvariablen $G_{i,j}$. In der Abb. 12.4 sind einige Beispiele von Texturen zu sehen, die durch obiges Beispielmodell berechnet wurden.

Abb. 12.3 Einfaches AR-Modell: Festlegung der Vorgänger

		g_{lo}	g_o	g_{ro}	
		g_l	g		

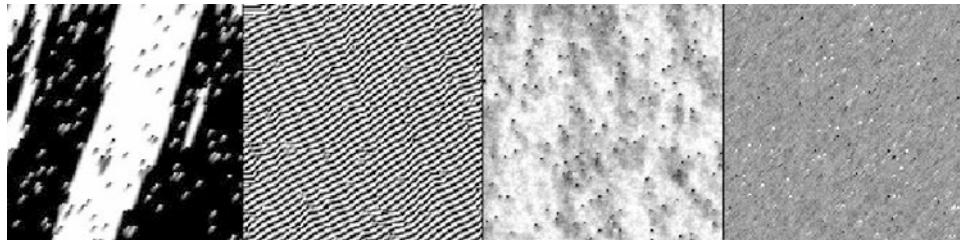


Abb. 12.4 Durch AR-Prozesse erzeugte Texturen

Haben wir dagegen eine Textur gegeben, so können wir zumindestens versuchen, die Parameter des AR-Modells aus der Textur mit der MMSE-Methode (siehe Abschnitte 21.2, 9.5.3) zu schätzen:

$$S = E \left(\sum_{(i,j) \in B} [a_{lo} G_{i-1,j-1} + a_o G_{i,j-1} + a_{ro} G_{i+1,j-1} + a_l G_{i-1,j} + N_{i,j} - G_{i,j}]^2 \right) \rightarrow \text{Minimum.} \quad (12.5)$$

Dabei ist B ein geeignet zu wählender Bereich der Textur im Bild. Wenn wir berücksichtigen, dass wir mittelwertfreie Zufallsvariablen $N_{i,j}$ annehmen und diese Zufallsvariablen unabhängig zwischen den Pixeln sind, dann verschwinden einige Erwartungswerte und es bleiben als zu bestimmende Daten nur noch Erwartungswerte der Form $\sum_{(i,j) \in B} E(G_{i-x,j-y} \cdot G_{i-u,j-v})$ übrig, die aus einem einzigen Bild der Textur zu schätzen sind. Diese Schätzungen sind aber nur realistisch, wenn die Erwartungswerte von den Koordinatendifferenzen abhängen. Dies ist z. B. bei stationären Feldern G der Fall, was wir im Folgenden annehmen. In diesem Falle brauchen wir nur noch die Schätzungen $A_{k,l} = \sum_{(i,j) \in B} g_{i,j} g_{i-k,j-l}$ aus dem Bild B zu berechnen. Dies ist gerade die zyklische Autokorrelationsfunktion des Bildes B . Die Gaußschen Normalengleichungen lauten dann:

$$\begin{pmatrix} A_{0,0} & A_{1,0} & A_{2,0} & A_{-2,1} \\ A_{1,0} & A_{0,0} & A_{1,0} & A_{-1,1} \\ A_{2,0} & A_{1,0} & A_{0,0} & A_{-2,1} \\ A_{-2,1} & A_{-1,1} & A_{-2,1} & A_{0,0} \end{pmatrix} \cdot \begin{pmatrix} a_{lo} \\ a_o \\ a_{ro} \\ a_l \end{pmatrix} = \begin{pmatrix} A_{1,1} \\ A_{0,1} \\ A_{-1,1} \\ A_{1,0} \end{pmatrix}. \quad (12.6)$$

Teil II

3D-Bildverarbeitung

Im folgenden Kapitel werden wir einige Basiskonzepte der 3D-Geometrie vorstellen, welche für die nachfolgenden Verfahren der 3D-Rekonstruktion wesentlich sind. Vorgestellt werden die notwendigen Werkzeuge zur Modellierung von projektiven Transformationen und Rotationen mittels homogener Koordinaten und Quaternionen.

13.1 Geometrische Transformationen und homogene Koordinaten

Bevor wir die wichtigsten Grundprinzipien der 3D-Bildverarbeitung erläutern, müssen wir geometrische Transformationen und ihre algebraischen Beschreibungen näher betrachten. Wir werden dabei in diesem Abschnitt das Konzept von homogenen Koordinaten kennenlernen, welches für die Beschreibung von projektiven Transformationen sinnvoll ist.

13.1.1 Geometrische Transformationen in der Ebene

Die wichtigsten algebraischen Hilfsmittel zur Beschreibung der Transformationen sind homogene Koordinaten. Zuvor sei nochmals an die elementaren geometrischen Transformationen erinnert, siehe auch Abschn. 1.4.3.

- Translationen:

$$\mathbf{x}' = \mathbf{x} + \mathbf{a}. \quad (13.1)$$

Die zwei Parameter der Verschiebung können durch eine Punktreferenz, d. h. einen Quellpunkt und einen Zielpunkt bestimmt werden.

- Rotationen:

$$\mathbf{x}' = \mathbf{R} \cdot \mathbf{x}. \quad (13.2)$$

Der eine Parameter der Rotation (Rotationswinkel) kann durch eine Punktreferenz, d. h. einen Quellpunkt und einen Zielpunkt bestimmt werden. Es wird stets um den Koordinatenursprung rotiert.

- Euklidische Transformationen (Kongruenztransformationen):

$$\mathbf{x}' = \mathbf{R} \cdot \mathbf{x} + \mathbf{a}. \quad (13.3)$$

Eigentlich gehören auch Spiegelungen dazu, diese lassen wir aber hier bewusst weg. Die drei Parameter der Euklidischen Transformation (ohne Spiegelung) können durch zwei Punktreferenzen, d. h. zwei Quellpunkte und zwei Zielpunkte bestimmt werden.

- Ähnlichkeitstransformationen:

$$\mathbf{x}' = s \cdot \mathbf{R} \cdot \mathbf{x} + \mathbf{a}. \quad (13.4)$$

Die vier Parameter der Ähnlichkeitstransformationen (auch ohne Spiegelungen) können durch zwei Punktreferenzen, d. h. zwei Quellpunkte und zwei Zielpunkte bestimmt werden.

- Affine Transformationen:

$$\mathbf{x}' = \mathbf{A} \cdot \mathbf{x} + \mathbf{a}. \quad (13.5)$$

Da der Verschiebungsvektor \mathbf{a} eine besondere Bedeutung besitzt, bezeichnen wir seine Komponenten mit a_{i0} und nicht mit a_{i3} . Ansonsten müssten wir die Verschiebungen im 3D-Raum mit a_{i4} bezeichnen.

Die sechs Parameter der affinen Transformationen können durch drei Punktreferenzen, d. h. drei Quellpunkte und drei Zielpunkte bestimmt werden. Komponentenweise aufgeschrieben ist dann:

$$\begin{aligned} x' &= a_{11}x + a_{12}y + a_{10} \\ y' &= a_{21}x + a_{22}y + a_{20}. \end{aligned} \quad (13.6)$$

Es ist zu bemerken, dass Geraden in Geraden übergehen und parallele Geraden auch weiterhin parallel bleiben. Ein Rechteck kann folglich in ein Parallelogramm überführt werden. Weiterhin ist das sogenannte Teilverhältnis invariant, siehe Abschn. 14.3.1. Die affinen Transformationen sind die allgemeinsten linearen Koordinatentransformationen im Sinne von „linearen Funktionen“ und nicht von linearen Operatoren. Weiterhin hängen sie eng mit dem Begriff des affinen Raumes und dessen Basis zusammen. Eine affine Basis besteht aus drei Punkten, weil eine affine Transformation durch drei Punktreferenzen bestimmt ist.

- Projektive Transformationen:

$$x' = \frac{a_{11}x + a_{12}y + a_{10}}{a_{31}x + a_{32}y + a_{30}}, \quad y' = \frac{a_{21}x + a_{22}y + a_{20}}{a_{31}x + a_{32}y + a_{30}}. \quad (13.7)$$

Projektive Transformationen der Ebene werden auch als Homographien bezeichnet. Eine projektive Transformation der Ebene wird durch acht Parameter bestimmt, folglich bestimmen vier Quellpunkte und vier Zielpunkte eindeutig eine projektive Transformation der Ebene. Formal zählt man neun Parameter, da wir aber Zähler und Nenner mit einer Konstanten multiplizieren können ohne dass sich die Transformation ändert, sind es eben „echt“ nur acht Parameter. Diese Transformation ist offenbar nicht mehr linear, aber Geraden gehen immer noch in Geraden über, deshalb werden sie auch oft **Kollineationen** genannt. Die Parallelität von Geraden bleibt allerdings nicht mehr erhalten. Ein Rechteck kann demnach in ein beliebiges Viereck überführt werden. Die projektiven Transformationen (13.7) werden durch gebrochen rationale Funktionen beschrieben, allerdings ist der Zähler und Nenner für sich wieder linear. Setzen wir den Nenner gleich Null, so erhalten wir eine Gerade, dessen Punkte (sofern der Zähler ungleich Null ist) ins Unendliche abgebildet werden. Wir nennen diese spezielle Gerade die „projektive Gerade“. Da vier Punktreferenzen eine projektive Transformation eindeutig bestimmen, bestimmen vier Punkte auch die projektive Basis der projektiven Ebene. Für die analytische projektive Geometrie ist es unerlässlich zu sogenannten **homogenen Koordinaten** überzugehen, da wir in der projektiven Geometrie unter anderem auch den Schnittpunkt zweier paralleler Geraden ausrechnen wollen.

13.1.2 Homogene Koordinaten

Eine „Tilde“ über einem Symbol soll im Folgenden ausdrücken, dass es homogene und keine kartesischen Koordinaten sind. Wir führen jetzt formal folgende Substitution ein:

$$x = \frac{\tilde{x}}{\tilde{z}}; \quad y = \frac{\tilde{y}}{\tilde{z}}. \quad (13.8)$$

Dann ergibt sich für die projektive Transformation:

$$x' = \frac{a_{11}\frac{\tilde{x}}{\tilde{z}} + a_{12}\frac{\tilde{y}}{\tilde{z}} + a_{10}}{a_{31}\frac{\tilde{x}}{\tilde{z}} + a_{32}\frac{\tilde{y}}{\tilde{z}} + a_{30}}, \quad y' = \frac{a_{21}\frac{\tilde{x}}{\tilde{z}} + a_{22}\frac{\tilde{y}}{\tilde{z}} + a_{20}}{a_{31}\frac{\tilde{x}}{\tilde{z}} + a_{32}\frac{\tilde{y}}{\tilde{z}} + a_{30}}. \quad (13.9)$$

Damit ist:

$$x' = \frac{a_{11}\tilde{x} + a_{12}\tilde{y} + a_{10}\tilde{z}}{a_{31}\tilde{x} + a_{32}\tilde{y} + a_{30}\tilde{z}}, \quad y' = \frac{a_{21}\tilde{x} + a_{22}\tilde{y} + a_{20}\tilde{z}}{a_{31}\tilde{x} + a_{32}\tilde{y} + a_{30}\tilde{z}}. \quad (13.10)$$

Nun substituieren wir ebenso:

$$x' = \frac{\tilde{x}'}{\tilde{z}'}; \quad y' = \frac{\tilde{y}'}{\tilde{z}'}, \quad (13.11)$$

dann ist offensichtlich:

$$\begin{aligned}\tilde{x}' &= a_{11}\tilde{x} + a_{12}\tilde{y} + a_{10}\tilde{z} \\ \tilde{y}' &= a_{21}\tilde{x} + a_{22}\tilde{y} + a_{20}\tilde{z} \\ \tilde{z}' &= a_{31}\tilde{x} + a_{32}\tilde{y} + a_{30}\tilde{z}.\end{aligned}\quad (13.12)$$

In Matrixschreibweise ist nun:

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}} = \begin{pmatrix} a_{11} & a_{12} & a_{10} \\ a_{21} & a_{22} & a_{20} \\ a_{31} & a_{32} & a_{30} \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix}. \quad (13.13)$$

Multiplizieren wir mit einer Konstanten c den Zähler und Nenner, so ist:

$$x' = \frac{c \cdot \tilde{x}}{c \cdot \tilde{z}} = \frac{\tilde{x}}{\tilde{z}}, \quad y' = \frac{c \cdot \tilde{y}}{c \cdot \tilde{z}} = \frac{\tilde{y}}{\tilde{z}}. \quad (13.14)$$

Das heißt aber:

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}} \quad (13.15)$$

ist äquivalent zu:

$$\tilde{\mathbf{x}}' = c \cdot \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}, \quad (13.16)$$

d. h. beschreibt die gleiche projektive Transformation. Die Form:

$$\begin{pmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{z}' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{10} \\ a_{21} & a_{22} & a_{20} \\ a_{31} & a_{32} & 1 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad (13.17)$$

entspricht der (13.13) für ein spezielles c , nämlich: $c = \frac{1}{a_{30}}$. Damit ist sie aber nicht völlig äquivalent, da $a_{30} \neq 0$ nicht erfüllt sein muss. Wir bezeichnen dabei die Matrixkoeffizienten in (13.17) auch mit a_{ij} , obwohl es nicht die gleichen a_{ij} wie in (13.13) sind.

Die homogenen Koordinaten bieten einen weiteren Vorteil, man kann statt eines affinen Raumes jetzt einen projektiven Raum analytisch beschreiben. Einen Punkt in kartesischen Koordinaten erweitern wir einfach zu:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad (13.18)$$

und können damit jeden Punkt der projektiven Ebene beschreiben. Allerdings können wir wieder mit einer beliebigen Konstanten multiplizieren:

$$c \cdot \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad (13.19)$$

und dieser Vektor von homogenen Koordinaten beschreibt denselben Punkt im projektiven Raum.

Punkte mit $\tilde{z} = 0$ sind sogenannte uneigentliche Punkte, die zur projektiven Ebene, aber nicht zur affinen Ebene gehören, da sie keine kartesischen Koordinaten besitzen. Wir betrachten nun einmal eine beliebige Gerade in kartesischen Koordinaten:

$$ax + by + c = 0. \quad (13.20)$$

Wir substituieren wieder durch die homogenen Koordinaten mit

$$x = \frac{\tilde{x}}{\tilde{z}}; \quad y = \frac{\tilde{y}}{\tilde{z}}. \quad (13.21)$$

Damit erhalten wir für eine Gerade im projektiven Raum in homogenen Koordinaten:

$$a\tilde{x} + b\tilde{y} + c\tilde{z} = 0. \quad (13.22)$$

Wir setzen nun $a = 0, b = 0, c = 1$ und erhalten eine spezielle Gerade mit der Gleichung $\tilde{z} = 0$. Diese Gerade heißt uneigentliche projektive Gerade und besteht nur aus uneigentlichen Punkten. Umgekehrt liegen auch alle uneigentlichen Punkte der projektiven Ebene auf dieser uneigentlichen Geraden.

Nun wollen wir den Schnittpunkt einer beliebigen Geraden (13.22) mit der uneigentlichen Geraden $\tilde{z} = 0$ ausrechnen. Es ergibt sich trivialerweise $a\tilde{x} + b\tilde{y} = 0$. Daher ist $\frac{\tilde{x}}{\tilde{y}} = -\frac{b}{a}$ und folglich:

$$S_g = \lambda \cdot \begin{pmatrix} -b \\ a \\ 0 \end{pmatrix} \quad (13.23)$$

der eindeutig bestimmte Schnittpunkt beider Geraden. Der Schnittpunkt scheint sinnvollerweise nur von der Richtung der Geraden abzuhängen, nicht aber von deren Verschiebung. Zwei parallele Geraden:

$$\begin{aligned} a\tilde{x} + b\tilde{y} + c\tilde{z} &= 0 \\ a\tilde{x} + b\tilde{y} + d\tilde{z} &= 0 \end{aligned} \quad (13.24)$$

haben den gleichen Schnittpunkt wie mit der uneigentlichen Geraden, nämlich exakt S_g nach (13.23). Alles bisher Formulierte gilt natürlich auch für den projektiven 3D-Raum und dessen projektive Ebenen, d. h.:

$$a\tilde{x} + b\tilde{y} + c\tilde{z} + d\tilde{u} = 0 \quad (13.25)$$

ist die Gleichung einer projektiven Ebene und $\tilde{u} = 0$ beschreibt die uneigentliche Ebene. Alle uneigentlichen Punkte des projektiven Raumes bilden die uneigentliche projektive Ebene. Eine Ebene schneidet die uneigentliche Ebene in einer uneigentlichen Geraden, die auch Schnittgerade zweier paralleler Ebenen ist.

Sind in der projektiven Ebene zwei Punkte $\tilde{\mathbf{x}}_1$ und $\tilde{\mathbf{x}}_2$ gegeben, so geht eindeutig eine Gerade durch die beiden Punkte mit:

$$\tilde{\mathbf{x}} = \lambda \tilde{\mathbf{x}}_1 + \mu \tilde{\mathbf{x}}_2, \quad \lambda, \mu \in R. \quad (13.26)$$

Dies bedeutet wiederum:

$$\tilde{\mathbf{x}}^T (\tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2) = 0, \quad (13.27)$$

d. h. $\tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2$ charakterisiert in homogenen Koordinaten eindeutig eine Gerade, die durch zwei Punkte geht. Mit der Schreibweise (13.22) nennen wir $\mathbf{a} = (a, b, c)^T$ den charakteristischen Vektor einer Geraden. Nun betrachten wir zwei Geraden mit den beiden charakteristischen Vektoren \mathbf{a}_1 und \mathbf{a}_2 . Der Schnittpunkt $\tilde{\mathbf{x}}$ dieser beiden Geraden erfüllt $\tilde{\mathbf{x}}^T \cdot \mathbf{a}_1 = 0$, $\tilde{\mathbf{x}}^T \cdot \mathbf{a}_2 = 0$, d. h. $\tilde{\mathbf{x}}$ steht senkrecht auf den Vektoren \mathbf{a}_1 und \mathbf{a}_2 . Damit ist:

$$\tilde{\mathbf{x}} = \mathbf{a}_1 \times \mathbf{a}_2 \quad (13.28)$$

der Schnittpunkt beider Geraden in homogenen Koordinaten. Dies zeigt beeindruckend die Dualität von Punkt und Gerade. Nochmals zusammengefasst:

- $\tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2$ beschreibt eine Gerade durch zwei Punkte,
- $\mathbf{a}_1 \times \mathbf{a}_2$ ist der Schnittpunkt zweier Geraden.

Nun gehen wir noch weiter und betrachten eine beliebige Gerade \mathbf{a} , die durch den Schnittpunkt $\mathbf{a}_1 \times \mathbf{a}_2$ der beiden Geraden \mathbf{a}_1 und \mathbf{a}_2 geht. Für jeden Punkt $\tilde{\mathbf{x}}$ dieser Geraden \mathbf{a} muss natürlich $\mathbf{a}^T \cdot \tilde{\mathbf{x}} = 0$ gelten, also auch $\mathbf{a}^T \cdot (\mathbf{a}_1 \times \mathbf{a}_2) = 0$, was als Beschreibung eines **Geradenbüschels** dient. Daher ist:

$$\mathbf{a} = \lambda \mathbf{a}_1 + \mu \mathbf{a}_2 \quad (13.29)$$

als homogene Koordinaten interpretiert die Beschreibung einer projektiven Geraden. Damit ist ein Geradenbüschel ebenfalls ein eindimensionaler projektiver Raum.

In einem n -dimensionalen, projektiven Raum besteht die projektive Basis aus $n + 2$ Punkten, wobei beliebige n Punkte der Basis linear unabhängig sein müssen. In homogenen Koordinaten ist es deshalb auch möglich, uneigentliche Punkte mit in die Basis aufzunehmen. Die sogenannte projektive Standardbasis besteht aus den $n + 2$ Punkten:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}. \quad (13.30)$$

Betrachten wir diese nochmals für die projektive Ebene, so ist:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \quad (13.31)$$

die projektive Standardbasis in homogenen Koordinaten. Der erste Punkt ist der Schnittpunkt der x -Achse mit der uneigentlichen Geraden, der zweite Punkt der Schnittpunkt der y -Achse mit der uneigentlichen Geraden, der dritte Punkt ist der Koordinatenursprung (kartesische Koordinaten $(0, 0)^T$) und der vierte hat die kartesischen Koordinaten $(1, 1)^T$.

Transformationen werden wir im Zusammenhang mit Kameramodellen immer in der Matrixschreibweise bezüglich homogener Koordinaten angeben, so lautet nun z. B. eine 2D \rightarrow 2D Euklidische Bewegung (ohne Spiegelung):

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}} = \begin{pmatrix} \cos \varphi & -\sin \varphi & a_{10} \\ \sin \varphi & \cos \varphi & a_{20} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix}. \quad (13.32)$$

Eine interessante Frage ist oft: Eine Gerade oder Ellipse wird affin oder projektiv transformiert, wie transformieren sich dann die Koeffizienten der Geraden bzw. Ellipse? Dazu betrachten wir wieder die Gleichung einer Geraden $\mathbf{g}^T \tilde{\mathbf{x}} = 0$ und eine projektive Transformation mit $\tilde{\mathbf{x}}' = \tilde{\mathbf{A}} \tilde{\mathbf{x}}$. In die Gleichung der transformierten Geraden $\mathbf{g}'^T \tilde{\mathbf{x}}' = 0$ setzen wir die Transformation ein und erhalten $\mathbf{g}'^T \tilde{\mathbf{x}}' = \mathbf{g}^T \tilde{\mathbf{A}} \tilde{\mathbf{x}} = 0$. Damit können wir ablesen: $\gamma \cdot \mathbf{g} = \mathbf{A}^T \mathbf{g}'$. Ähnlich können wir mit Kurven zweiter Ordnung verfahren. Eine Kurve zweiter Ordnung beschreibt:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0. \quad (13.33)$$

Nun führen wir die Abkürzungen

$$\mathbf{E} = \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (13.34)$$

ein und können in homogenen Koordinaten notieren $S(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^T \mathbf{E} \tilde{\mathbf{x}} = 0$. Für die transformierte Kurve gilt dann $S'(\tilde{\mathbf{x}}', \tilde{\mathbf{x}}') = \tilde{\mathbf{x}}'^T \mathbf{E}' \tilde{\mathbf{x}}' = \tilde{\mathbf{x}}^T \mathbf{A}^T \mathbf{E}' \mathbf{A} \tilde{\mathbf{x}}$. Daher gilt: $\alpha \mathbf{E} = \mathbf{A}^T \mathbf{E}' \mathbf{A}$. Nun betrachten wir zwei beliebige Punkte $\tilde{\mathbf{y}}$ und $\tilde{\mathbf{z}}$. Punkte auf der Geraden durch die beiden Punkte werden durch $\tilde{\mathbf{x}} = \tilde{\mathbf{y}} + \lambda \tilde{\mathbf{z}}$ beschrieben. Wir betrachten nun den Schnitt dieser Geraden mit einer Kurve zweiter Ordnung. Dazu setzen wir den Punkt $\tilde{\mathbf{x}}$ der Geraden in $S(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = 0$ ein:

$$S(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = (\tilde{\mathbf{y}} + \lambda \tilde{\mathbf{z}})^T \mathbf{E} (\tilde{\mathbf{y}} + \lambda \tilde{\mathbf{z}}) = S(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}) + 2\lambda S(\tilde{\mathbf{y}}, \tilde{\mathbf{z}}) + \lambda^2 S(\tilde{\mathbf{z}}, \tilde{\mathbf{z}}) = 0. \quad (13.35)$$

Dies ist offensichtlich eine quadratische Gleichung in λ , welche im Allgemeinen zwei Lösungen besitzt. Wir schreiben die Lösung einmal auf und betrachten den Radikanten unter der Wurzel. Es fallen beide Lösungen zu einer zusammen, wenn der Radikant Null ist. Dies ist offenbar dann der Fall, wenn die Gerade eine Tangente an die Kurve ist. Die Bedingung dafür, dass der Radikant gleich Null ist, lautet:

$$S(\tilde{\mathbf{y}}, \tilde{\mathbf{z}})^2 - S(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}) \cdot S(\tilde{\mathbf{z}}, \tilde{\mathbf{z}}) = 0. \quad (13.36)$$

Dies kann nun so interpretiert werden: Der Punkt $\tilde{\mathbf{y}}$ sei fest vorgegeben. Dann wird ein Punkt $\tilde{\mathbf{z}}$ derart gesucht, dass (13.36) erfüllt ist. Da (13.36) quadratisch ist, gibt es zwei grundlegend verschiedene Lösungen, d. h. zwei Tangenten an die Kurve zweiter Ordnung. Wir nehmen nun an, $\tilde{\mathbf{y}}$ liege selbst schon auf der quadratischen Form, dann folgt trivialerweise $S(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}) = 0$ und damit als Bestimmungsgleichung $S(\tilde{\mathbf{y}}, \tilde{\mathbf{z}}) = 0$. Wenn $\tilde{\mathbf{z}}$ irgendwo liegt, haben wir zwei Schnittpunkte, wobei einer $\tilde{\mathbf{y}}$ selbst sein muss. Wenn es also nur einen Schnittpunkt geben soll, geht dies nur, wenn $\tilde{\mathbf{z}}$ selbst auf der Tangente an $\tilde{\mathbf{y}}$ liegt. Dies können wir auch interpretieren zu $\tilde{\mathbf{y}}^T \mathbf{E} \tilde{\mathbf{z}} = \tilde{\mathbf{y}}^T \tilde{\mathbf{I}} = 0$ für alle $\tilde{\mathbf{z}}$ auf der Tangente in $\tilde{\mathbf{y}}$. Es gibt folglich einen Geradenvektor $\tilde{\mathbf{I}}$, der die Gleichung erfüllt. Es kann nur einen geben, d. h. $\tilde{\mathbf{I}} = \mathbf{E} \tilde{\mathbf{z}}$ ist die Lösung und als einzige Gerade kann dies nur die Tangente im Punkte $\tilde{\mathbf{y}}$ der quadratischen Form sein. Da wir die Punkte $\tilde{\mathbf{z}}$ auf der Tangente nicht kennen, nehmen wir einen, den wir kennen, nämlich $\tilde{\mathbf{y}}$. Wir lösen nun auf und erhalten $\tilde{\mathbf{y}} = \mathbf{E}^{-1} \tilde{\mathbf{I}}$ und setzen dies in die quadratische Form $\tilde{\mathbf{y}}^T \mathbf{E} \tilde{\mathbf{y}} = 0$ ein und erhalten:

$$\tilde{\mathbf{I}}^T \mathbf{E}^{-1} \tilde{\mathbf{I}} = 0. \quad (13.37)$$

Dies ist eine Bedingungsgleichung für alle Tangenten an eine quadratische Form. Dies nennt man die *duale quadratische Form*, d. h. die quadratische Form wird beschrieben als Enveloppe (Einhüllende) aller Tangenten. Daher können wir nun ableiten:

Lemma 13.1 Eine Gerade durch zwei Punkte $\tilde{\mathbf{y}}, \tilde{\mathbf{z}}$ ist genau dann tangent an eine Kurve zweiter Ordnung mit der Matrix \mathbf{E} , wenn die Gleichung:

$$(\tilde{\mathbf{y}} \times \tilde{\mathbf{z}})^T \mathbf{E}^{-1} (\tilde{\mathbf{y}} \times \tilde{\mathbf{z}}) = 0 \quad (13.38)$$

erfüllt ist.

Die quadratische Form mit der inversen Matrix ist, wie oben dargelegt, die duale quadratische Form. Der Vektor $\tilde{\mathbf{l}}$, der eine Gerade durch zwei Punkte beschreibt, errechnet sich, wie schon bekannt, zu $\tilde{\mathbf{l}} = \tilde{\mathbf{y}} \times \tilde{\mathbf{z}}$.

13.2 Beschreibung von 3D-Rotationen

In der 3D-Bildverarbeitung spielen $3D \leftrightarrow 3D$ Ähnlichkeitstransformationen oder Euklidische Transformationen eine große Rolle, z. B. bei der Beschreibung der äußeren Parameter einer Kamera. Wesentlicher Bestandteil dieser Transformationen sind 3D-Rotationen. Daher soll in den folgenden Abschnitten genauer auf 3D-Rotationen eingegangen werden.

13.2.1 Beschreibung durch Rotationsmatrizen

Wir betrachten als Koordinatentransformation eine „reine“ Rotation:

$$\mathbf{y} = \mathbf{R}\mathbf{x}. \quad (13.39)$$

Die Matrix \mathbf{R} heißt *Rotationsmatrix* oder *Drehmatrix*, wenn

1. die Matrix \mathbf{R} eine Orthogonalmatrix ist, d. h. wenn die Zeilenvektoren und die Spaltenvektoren je für sich ein Orthonormalsystem bilden, damit gilt $\mathbf{R}^{-1} = \mathbf{R}^T$,
2. die Beziehung $\det(\mathbf{R}) = +1$ gilt, d. h. es dürfen in der Abbildung außer Drehungen keine Spiegelungen enthalten sein.

In der Ebene wird eine Rotationsmatrix simpel durch:

$$\mathbf{R} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \quad (13.40)$$

ausgedrückt, dazu reicht ein Rotationswinkel φ aus. Man vermutet nun sicher, dass im 3D-Raum zwei Winkel benötigt werden. Dies ist aber nicht der Fall, wir benötigen (schon Euler hat dies bewiesen) **drei** Größen, entweder

- Drei *Eulersche Winkel* (Rotationen um die Koordinatenachsen), oder
- Eine *Drehachse*, die durch den Koordinatenursprung geht (zwei Parameter) und einen *Rotationswinkel* bezüglich dieser Drehachse.

Diese drei Parameter sind immer in einer *Rotationsmatrix* \mathbf{R} enthalten, d. h. aus den Parametern kann stets eine Rotationsmatrix berechnet werden und umgekehrt, aus der Rotationsmatrix können die Parameter wieder berechnet werden. Beschreibt eine Rotationsmatrix nun eine Rotation des Koordinatensystems oder die eines Punktes in einem gegebenen

Koordinatensystem? Je nach Schreibweise und Interpretation der Formel (13.39) ergibt sich Folgendes. In erster Linie bedeutet die Schreibweise (13.39) eine Rotation des Punktes \mathbf{x} im gleichen Koordinatensystem, d. h. \mathbf{y} sind die Koordinaten des rotierten Punktes im gleichen Koordinatensystem. Nun nehmen wir einmal an, das Koordinatensystem sei rotiert. Dann haben wir drei neue Basisvektoren $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$, die auch ein Orthonormalsystem bilden müssen. Diese drei Basisvektoren fassen wir als Spaltenvektoren auf und schreiben sie als Spalten in eine Matrix \mathbf{R} . Dann können wir den Punkt \mathbf{x} nach dieser Basis entwickeln und nennen die Koeffizienten vor den Basisvektoren $\mathbf{y}^T = (y_1, y_2, y_3)$:

$$\mathbf{x} = y_1 \mathbf{r}_1 + y_2 \mathbf{r}_2 + y_3 \mathbf{r}_3 = \mathbf{R} \cdot \mathbf{y}. \quad (13.41)$$

Damit gilt:

$$\mathbf{x} = \mathbf{R} \cdot \mathbf{y} \rightarrow \mathbf{y} = \mathbf{R}^T \cdot \mathbf{x}. \quad (13.42)$$

Vergleichen wir (13.42) mit der Punkttransformation (13.39), so müssen wir für die Interpretation bezüglich der Rotation des Koordinatensystems die inverse Rotation verwenden. Wir rotieren entweder den Punkt „vorwärts“, oder die Koordinatenachsen „rückwärts“. Wenn folglich ein Koordinatensystem und eine Rotationsmatrix gegeben sind, dann sind die neuen Basisvektoren immer die Zeilenvektoren der Rotationsmatrix.

Nun denken wir uns ein x, y, z -Referenzkoordinatensystem und ein körpereigenes u, v, w -Koordinatensystem, welche zu Beginn identisch sein sollen. Alle Koordinatensysteme sollen Rechtssysteme bilden. Nun führen wir bezüglich des starren x, y, z -Referenzkoordinatensystems eine Drehung α um die z -Achse, anschließend eine Drehung β um die y -Achse und anschließend eine Drehung γ um die x -Achse aus. Die Drehwinkel α, β, γ heißen *Eulersche Winkel*. Die Rotationsmatrix \mathbf{R} ergibt sich dann:

$$\begin{aligned} \mathbf{R} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{pmatrix} \cdot \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \cdot \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \mathbf{R}_x \cdot \mathbf{R}_y \cdot \mathbf{R}_z. \end{aligned} \quad (13.43)$$

Die positive Orientierung impliziert, dass bei der Rotation um die y -Achse die Vorzeichen wechseln. Da die Matrixmultiplikation nicht kommutativ ist, dürfen wir nicht einfach die Reihenfolge vertauschen. Schauen wir von der „Spitze“ einer Drehachse auf die rotierende Ebene, dann wird immer „links herum“ um die Drehachse gedreht, d. h. in Pfeilrichtung wird immer „rechts herum“ gedreht. Diese Rotationen bezogen sich bisher auf ein starres, gegebenes Referenzkoordinatensystem, deshalb werden diese Art Rotationen oft als *absolute Rotationen* bezeichnet. Die Rotation (13.43) lässt sich, obwohl die Formel diesselbe bleibt, noch anders deuten. Wir haben diese Formel bisher im Sinne einer Punkttransformation von „rechts nach links“ interpretiert, d. h. zuerst Drehung um die z -Achse, dann um die (alte) y -Achse, dann um die (alte) x -Achse. Wir können (13.43) tatsächlich auch von „links nach rechts“ interpretieren:

- a) Eine Drehung um γ bezüglich der x -Achse, oder was zu Beginn dasselbe ist, um die u -Achse.
- b) Eine Drehung um β bezüglich der v -Achse, oder was dasselbe ist, der neuen y' -Achse, wenn die beiden Koordinatensysteme wieder zur Deckung gebracht werden. Daher gibt es auch eine neue z' -Achse.
- c) Eine Drehung um α bezüglich der w -Achse, oder was dasselbe ist, bezüglich der neuen z'' -Achse, die aus der z' -Achse hervorgegangen ist.

Da sich diese Rotationen von „links nach rechts“ immer auf die zuletzt erzeugte neue Achse beziehen, nennt man diese auch *relative Rotationen*. Wie kann man nun beweisen, dass diese Interpretation auch richtig ist? Wir zeigen die Idee:

Der Einfachheit halber betrachten wir nur zwei von den drei Rotationen, nämlich:

$$\begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \cdot \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathbf{D} \cdot \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix}. \quad (13.44)$$

Auf der linken Seite von (13.44) steht die uns bekannte Interpretation von „rechts nach links“ und auf der rechten Seite natürlich auch. Aber auf der rechten Seite beginnen wir mit der Rotation um die y -Achse, mit der wir auf der linken Seite enden. Da die Kommutativität nicht gilt, ist die Matrix \mathbf{D} auch nicht identisch mit der Rotationsmatrix bez. der z -Achse. Die Matrix \mathbf{D} ist aber auf jeden Fall irgendeine Rotationsmatrix. Von jeder Rotationsmatrix können wir eindeutig die

- a) Rotations- oder Drehachse und
- b) den Rotations- oder Drehwinkel bestimmen.

Den Drehwinkel bestimmen wir in mathematisch positivem Sinne, d. h. die Drehachse bekommt eine Orientierung und wir drehen rechts herum, wenn wir in Richtung der Orientierung der Drehachse schauen. Berechnen wir nun die Drehachse und den Drehwinkel der Matrix \mathbf{D} , so erhalten wir:

- a) die Drehachse ist gerade die neue z' -Achse,
- b) der Drehwinkel ist identisch mit α .

Dies lässt sich recht einfach ausrechnen, dazu sollte der Leser erst den Abschn. 13.2.2 studieren. Die Beziehung (13.44) lautet in Matrixform $\mathbf{R}_y \cdot \mathbf{R}_z = \mathbf{D} \cdot \mathbf{R}_y$, d. h. $\mathbf{D} = \mathbf{R}_y \cdot \mathbf{R}_z \cdot \mathbf{R}_y^T$. Daraus folgt sofort $\text{Spur}(\mathbf{D}) = \text{Spur}(\mathbf{R}_z)$, womit der Drehwinkel gleich α ist.

Die Punkte \mathbf{z} der Drehachse von \mathbf{D} erfüllen die Gleichung $\mathbf{z} = \mathbf{R}_y \mathbf{R}_z \mathbf{R}_y^T \mathbf{z}$, d. h. es gilt $\mathbf{z}' = \mathbf{R}_z \mathbf{z}'$ mit $\mathbf{z}' = \mathbf{R}_y^T \mathbf{z}$. Daher sind \mathbf{z}' die Punkte der Rotationsachse von \mathbf{R}_z , also der z -Achse. Aus $\mathbf{z} = \mathbf{R}_y \mathbf{z}'$ folgt, dass \mathbf{z} die Punkte der um die y -Achse gedrehten z -Achse sind. Daher ist die Rotationsachse diese gedrehte z -Achse, was wir zeigen wollten.

Da es bezüglich der Eulerwinkel viele Reihenfolgen und Möglichkeiten gibt, muss man eigentlich immer genau sagen, was man konkret unter den Eulerwinkeln versteht. Dazu gibt es regelrecht Konventionen. Unsere obige Zerlegung entspricht z. B. der Luftfahrt-norm, der *Yaw-Pitch-Roll-Norm*. In der Reihenfolge als absolute Rotationen schreiben wir zyx (zuerst Rotation um die z -Achse, dann um die alte y -Achse, dann um die alte x -Achse, $R = R_x R_y R_z$), oder als relative Rotationen $xy'z''$ (zuerst um die alte x -Achse, dann um die neue y -Achse y' , dann um die neueste z -Achse z'' mit der gleichen Rotationsmatrix $R = R_x R_y R_z$). Die *Yaw-Pitch-Roll-Eulerwinkel* heißen

- Gierwinkel,
- Nickwinkel,
- Rollwinkel.

Es gibt aber auch andere Konventionen, z. B. die *x -Konvention*. Diese wird beschrieben durch zxz oder $zx'z''$ mit $R = R_z R_x R_z$. Für die Rotationen um die z -Achsen brauchen wir natürlich zwei verschiedene Winkel. Die *y -Konvention* ist dann analog zyz bzw. $zy'z''$. Bei Angaben von Eulerwinkeln muss man genau darauf achten, was für eine Konvention verwendet wird.

13.2.2 Bestimmung der Drehachse und des Drehwinkels

Bestimmung der Drehachse Die Drehachse muss durch den Koordinatenursprung gehen, da keine Translation vorliegt. Alle Punkte \mathbf{x} auf der Drehachse sind invariant bezüglich der Drehung, müssen also die Fixpunktgleichung $\mathbf{x} = \mathbf{Rx}$ erfüllen. Dies ist eine lineares homogenes Gleichungssystem mit drei Gleichungen und drei Unbekannten. Daraus berechnen wir einen Einheitsvektor auf der Drehachse und legen noch dessen Orientierung fest. Numerisch bestimmen wir von der Matrix $\mathbf{I} - \mathbf{R}$ den normierten Eigenvektor zum Eigenwert Null.

Bestimmung des Drehwinkels Wir legen nun ein neues kartesisches Koordinatensystem mit drei orthonormalen Basisvektoren $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ fest, dabei ist dann \mathbf{B} eine Drehmatrix. Der Basisvektor \mathbf{b}_3 sei dabei identisch mit dem berechneten Drehachseneinheitsvektor, dieser bilde nun die z' -Achse. Die beiden anderen Basisvektoren $\mathbf{b}_1, \mathbf{b}_2$ als x' - und y' -Achse werden so berechnet, dass sie zusammen mit \mathbf{b}_3 ein Orthonormalsystem mit Rechtsorientierung bilden. Es sei nun ein Punkt \mathbf{x} im ursprünglichen Koordinatensystem gegeben. Dieser besitze im neuen Koordinatensystem die Koordinaten $\mathbf{x}' = (x'_1, x'_2, x'_3)^T$, d. h. es gilt $\mathbf{x} = x'_1 \mathbf{b}_1 + x'_2 \mathbf{b}_2 + x'_3 \mathbf{b}_3 = \mathbf{Bx}'$. Für die rotierten Punkte $\mathbf{y} = \mathbf{Rx}$ gelte analog $\mathbf{y} = \mathbf{By}'$, d. h. $\mathbf{y}' = (y'_1, y'_2, y'_3)^T$ sind die Koordinaten von \mathbf{y} bezüglich der Basis $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$. Damit folgt aus $\mathbf{y} = \mathbf{Rx}$ sofort $\mathbf{By}' = \mathbf{RBx}'$ und damit $\mathbf{y}' = \mathbf{B}^T \mathbf{RBx}'$, womit wir die gleiche Rotation im

neuen Koordinatensystem beschrieben haben. Da wir aber in diesem Koordinatensystem als z' -Achse gerade die Drehachse gewählt haben, muss die Drehachse folgende Gestalt haben:

$$\begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathbf{B}^T \mathbf{R} \mathbf{B}. \quad (13.45)$$

Daraus können wir den Drehwinkel θ ausrechnen, wir nutzen dazu Rechenregeln bezüglich der Spur einer Matrix. Für ähnliche Matrizen \mathbf{A}, \mathbf{C} mit $\mathbf{C} = \mathbf{B}^{-1} \mathbf{A} \mathbf{B}$ gilt $\text{Spur}(\mathbf{C}) = \text{Spur}(\mathbf{A})$. Diese Beziehung wenden wir nun an und es folgt $\cos \theta + \cos \theta + 1 = \text{Spur}(\mathbf{R})$, damit ist schließlich:

$$\cos \theta = \frac{\text{Spur}(\mathbf{R}) - 1}{2}. \quad (13.46)$$

13.2.3 Drehmatrix aus Drehachse und Drehwinkel

Diese Aufgabe ist sogar noch leichter. Aus der Drehachse, die jetzt gegeben ist, bestimmen wir wie im Abschn. 13.2.2 die Matrix \mathbf{B} . Da der Drehwinkel jetzt gegeben ist, lösen wir die Beziehung (13.45) nach der Rotationsmatrix \mathbf{R} auf:

$$\mathbf{R} = \mathbf{B} \cdot \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{B}^T. \quad (13.47)$$

13.2.4 Beschreibung von 3D-Rotationen durch Quaternionen

In diesem Abschnitt geht es nicht darum eine neue Theorie der Rotationen zu entwickeln, sondern nur darum, das Bekannte übersichtlicher und daher handlicher zu beschreiben. Das Analogon liegt im Zweidimensionalen. Statt der x, y -Ebene können wir auch die komplexe Ebene nutzen und versuchen eine Rotation in der komplexen Ebene zu beschreiben, und siehe, das funktioniert recht elegant:

In Polarkoordinaten lässt sich eine komplexe Zahl zu $z_1 = a + ib = re^{i\varphi}$ schreiben. Betrachten wir nun die komplexen Zahlen auf dem Einheitskreis $z_2 = e^{i\psi}$, wir wollen sie Einheits- \mathbb{C} -Zahlen nennen, dann ergibt sich bei der komplexen Multiplikation:

$$z_1 \cdot z_2 = re^{i\varphi} \cdot e^{i\psi} = re^{i(\varphi+\psi)}. \quad (13.48)$$

Damit haben wir die Rotation auf eine simple komplexe Multiplikation mit einer Einheits- \mathbb{C} -Zahl zurückgeführt. Ist diese Idee nun auf den 3D-Raum übertragbar? Bevor wir das

klären, müssen noch ein paar Begriffe eingeführt werden. Bezuglich der Multiplikation können wir jeder komplexen Zahl eine Matrix zuordnen und die Multiplikation von komplexen Zahlen durch diese Matrixmultiplikation ersetzen. Es ist also $z = x + iy$ identifizierbar mit der speziellen Matrix:

$$z = x + iy = \begin{pmatrix} x & y \\ -y & x \end{pmatrix}. \quad (13.49)$$

Mit $z_1 = x_1 + iy_1, z_2 = x_2 + iy_2$ ist $z_1 \cdot z_2 = (x_1x_2 - y_1y_2) + i(y_1x_2 + x_1y_2)$. Andererseits gilt:

$$z_1 \cdot z_2 = \begin{pmatrix} x_1 & y_1 \\ -y_1 & x_1 \end{pmatrix} \cdot \begin{pmatrix} x_2 & y_2 \\ -y_2 & x_2 \end{pmatrix} = \begin{pmatrix} x_1x_2 - y_1y_2 & x_1y_2 + y_1x_2 \\ -y_1x_2 - x_1y_2 & -y_1y_2 + x_1x_2 \end{pmatrix}. \quad (13.50)$$

Die Matrixmultiplikation ist im Allgemeinen nicht kommutativ, speziell bei diesen Matrizen ist aber die Kommutativität garantiert. Eine andere Darstellung ist die Matrix-Vektor-Darstellung im Sinne eines linearen Operators. Dazu fassen wir die komplexen Zahlen $z = x + iy$ als Vektoren $\mathbf{z} = (x, y)^T$ auf. Dann können wir die Multiplikation durch:

$$z = z_1 \cdot z_2 \Leftrightarrow \mathbf{z} = \mathbf{z}_1 \cdot \mathbf{z}_2 = \begin{pmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \quad (13.51)$$

beschreiben. Weiterhin sei an die Trivialität $|z|^2 = z \cdot \bar{z}$ erinnert, wobei \bar{z} die zu z konjugiert komplexe Zahl ist.

Bei der Verallgemeinerung auf den 3D-Raum benötigen wir „komplexe Zahlen“ des 3D-Raumes. Aus der Algebra ist aber bekannt, dass es 3D-komplexe Zahlen „mit vernünftigen Eigenschaften“ nicht geben kann, sondern die nächst höhere Dimension sind 4D-Zahlen, genannt *Quaternionen*. Diese hat schon *Hamilton* 1843 eingeführt. Quaternionen bilden einen **Schiefkörper**, da die Multiplikation nicht kommutativ ist. Eine Quaternion schreiben wir zu $z = a + ib + jc + kd$ und bezeichnen die Menge aller Quaternionen mit \mathbb{H} . Daran sehen wir, dass wir dies als Erweiterung einer komplexen Zahl ansehen können. Andererseits erinnert diese Schreibweise auch an die 3D-Einheitsvektoren aus der linearen Algebra, daher können wir auch Quaternionen abstrakt auffassen als Summe einer skalaren Grösse und eines 3D-Vektors, was natürlich in der linearen Algebra als Unsinn anzusehen wäre. In der Menge der Quaternionen müssen wir nun eine Addition und eine Multiplikation einführen. Die Addition ist recht einfach zu definieren, diese definieren wir *komponentenweise*. Die Multiplikation sollte eine „kompatible“ Erweiterung der komplexen Multiplikation sein, daher scheint es nahe zu liegen, dass $i \cdot i = j \cdot j = k \cdot k = -1$ sein soll. Bei den gemischten Produkten kommt allerdings eine neue Qualität hinzu, was soll $i \cdot j$ bedeuten? Da wir bei i, j, k an die Basisvektoren denken, benutzen wir für diese gemischten Vektorprodukte das uns bekannte *Vektorprodukt*, wobei stets zu beachten ist, dass ein

Rechtssystem entsteht. Folglich können wir jetzt die Verknüpfungstafel exakt aufschreiben:

	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Die Multiplikation zweier Quaternionen wollen wir mit \otimes bezeichnen. Die Addition ist damit kommutativ und assoziativ, die Multiplikation ist nur assoziativ, die Kommutativität gilt infolge der enthaltenen Vektorprodukte nicht:

$$z_1 \otimes z_2 \neq z_2 \otimes z_1.$$

Multiplikation und Addition sind aber distributiv. Unter Beachtung der Vorzeichen kann aus der Verknüpfungstafel jeder Quaternion ähnlich der komplexen Zahlen (13.50) eine Matrix zugeordnet werden, so dass die Matrizenmultiplikation zum Quaternionenprodukt algebraisch äquivalent ist:

$$z = a + ib + jc + kd \Leftrightarrow Z = \begin{pmatrix} a & b & c & d \\ -b & a & -d & c \\ -c & d & a & -b \\ -d & -c & b & a \end{pmatrix}. \quad (13.52)$$

Mit $\bar{z} = a - ib - jc - kd \Leftrightarrow Z^T$ bezeichnen wir die zu z konjugierte Quaternion. Wie man leicht überprüfen kann, gilt:

$$\overline{z_1 \otimes z_2} = \overline{z_2} \otimes \overline{z_1}, \quad z \otimes \bar{z} = a^2 + b^2 + c^2 + d^2 = a^2 + |\mathbf{a}|^2 = |z|^2, \quad \mathbf{a}^T = (b, c, d). \quad (13.53)$$

Mit $z^{-1} = \frac{1}{z} \Leftrightarrow Z^{-1}$ gilt:

$$(z_1 \otimes z_2)^{-1} = z_2^{-1} \otimes z_1^{-1}, \quad z^{-1} = \frac{1}{z} = \frac{1 \otimes \bar{z}}{z \otimes \bar{z}} = \frac{\bar{z}}{|z|^2}. \quad (13.54)$$

Das Quaternionenprodukt können wir bezüglich der Definition noch übersichtlicher schreiben zu:

$$z_1 \otimes z_2 = (a_1 + \mathbf{a}_1) \otimes (a_2 + \mathbf{a}_2) = a_1 \cdot a_2 + \mathbf{a}_1 \cdot \mathbf{a}_2 + a_2 \cdot \mathbf{a}_1 + \mathbf{a}_1 \otimes \mathbf{a}_2. \quad (13.55)$$

Durch diese kompakte Form reicht es also aus, das Quaternionenprodukt bezüglich zweier Vektoren zu definieren, dies folgt aber sofort aus unserer Definition zu:

$$\mathbf{a}_1 \otimes \mathbf{a}_2 = -\mathbf{a}_1^T \cdot \mathbf{a}_2 + \mathbf{a}_1 \times \mathbf{a}_2. \quad (13.56)$$

Das erste Produkt ist das gewöhnliche Skalarprodukt und das zweite ist das Vektorprodukt, man beachte $\mathbf{a}_1 \times \mathbf{a}_2 = -\mathbf{a}_2 \times \mathbf{a}_1$. Daher wird oft die Definition der Multiplikation zweier Quaternionen gleich in der Form:

$$z_1 \otimes z_2 = (\mathbf{a}_1 + \mathbf{a}_1) \otimes (\mathbf{a}_1 + \mathbf{a}_2) = [\mathbf{a}_1 \cdot \mathbf{a}_2 - \mathbf{a}_1^T \cdot \mathbf{a}_2] + [\mathbf{a}_1 \cdot \mathbf{a}_2 + \mathbf{a}_2 \cdot \mathbf{a}_1 + \mathbf{a}_1 \times \mathbf{a}_2] \quad (13.57)$$

angegeben. Eine andere Darstellung des Quaternionenproduktes ist die Matrix-Vektor-Darstellung im Sinne eines linearen Operators. Dazu fassen wir die Quaternionen $z_1 = a_1 + \mathbf{a}_1 = a_1 + ia_{1,1} + ja_{1,2} + ka_{1,3}$ als Vektoren $\mathbf{z} = (a_1, a_{1,1}, a_{1,2}, a_{1,3})^T$ auf. Dann können wir die Multiplikation analog zu (13.51) durch

$$z = z_1 \otimes z_2 \leftrightarrow \mathbf{z} = \mathbf{Z}_1 \cdot \mathbf{z}_2 = \begin{pmatrix} a_1 & a_{1,1} & a_{1,2} & a_{1,3} \\ -a_{1,1} & a_1 & -a_{1,3} & a_{1,2} \\ -a_{1,2} & a_{1,3} & a_1 & -a_{1,1} \\ -a_{1,3} & -a_{1,2} & a_{1,1} & a_1 \end{pmatrix} \cdot \begin{pmatrix} a_2 \\ a_{2,1} \\ a_{2,2} \\ a_{2,3} \end{pmatrix} \quad (13.58)$$

ausdrücken. Wenn wir eine analoge Darstellung für $z_2 \otimes z_1$ haben wollen und als Vektor statt z_1 wieder z_2 benutzen wollen, dann müssen wir beachten, dass die Multiplikation nicht kommutativ ist. Es ergibt sich dann

$$z = z_2 \otimes z_1 \leftrightarrow \mathbf{z} = \mathbf{Z}_1^* \cdot \mathbf{z}_2 = \begin{pmatrix} a_1 & a_{1,1} & a_{1,2} & a_{1,3} \\ -a_{1,1} & a_1 & a_{1,3} & -a_{1,2} \\ -a_{1,2} & -a_{1,3} & a_1 & a_{1,1} \\ -a_{1,3} & a_{1,2} & -a_{1,1} & a_1 \end{pmatrix} \cdot \begin{pmatrix} a_2 \\ a_{2,1} \\ a_{2,2} \\ a_{2,3} \end{pmatrix}. \quad (13.59)$$

Man sieht, die rechte untere 3×3 -Matrix ist jetzt transponiert. Mit einfaches Nachrechnen kann man nun folgende Rechenregel für beliebige Quaternionen beweisen:

$$|z_1 \otimes z_2| = |z_1| \cdot |z_2|. \quad (13.60)$$

Nun wollen wir die Einheitsquaternionen beschreiben. Wie bei komplexen Zahlen nennen wir diejenigen Quaternionen Einheitsquaternionen, für die

$$|z|^2 = |a|^2 + |\mathbf{a}|^2 = 1 \quad (13.61)$$

gilt. Daher können wir formal $|a| = \cos \varphi$ und $|\mathbf{a}| = \sin \varphi$ setzen. Folglich bilden die folgenden Quaternionen

$$z_\varphi = \cos \varphi + \boldsymbol{\mu} \cdot \sin \varphi \quad (13.62)$$

Einheitsquaternionen. Dabei ist $\boldsymbol{\mu}$ irgendein Einheitsvektor, d. h. es muss nur gelten $|\boldsymbol{\mu}| = 1$. Diese Darstellung erinnert schon einmal an die Eulersche Formel, die Verallgemeinerung lautet nun:

$$e^{\mu\varphi} = \cos \varphi + \boldsymbol{\mu} \sin \varphi. \quad (13.63)$$

Multiplizieren wir nun eine Einheitsquaternion mit einem beliebigen Vektor (die Quaternion besteht nur aus dem vektoriellen Anteil), so ergibt sich:

$$z_\varphi \otimes \mathbf{a} = (\cos \varphi + \boldsymbol{\mu} \sin \varphi) \otimes \mathbf{a} = \cos \varphi \cdot \mathbf{a} - \sin \varphi \cdot (\boldsymbol{\mu}^T \cdot \mathbf{a}) + \sin \varphi \cdot (\boldsymbol{\mu} \times \mathbf{a}) \quad (13.64)$$

$$\mathbf{a} \otimes z_\varphi = \mathbf{a} \otimes (\cos \varphi + \boldsymbol{\mu} \sin \varphi) = \cos \varphi \cdot \mathbf{a} - \sin \varphi \cdot (\boldsymbol{\mu}^T \cdot \mathbf{a}) - \sin \varphi \cdot (\boldsymbol{\mu} \times \mathbf{a}) \quad (13.65)$$

Steht der Einheitsvektor $\boldsymbol{\mu}$ senkrecht auf \mathbf{a} , d. h. $\boldsymbol{\mu}^T \mathbf{a} = 0$, dann reduzieren sich die Formeln auf:

$$z_\varphi \otimes \mathbf{a} = (\cos \varphi + \boldsymbol{\mu} \sin \varphi) \otimes \mathbf{a} = \cos \varphi \cdot \mathbf{a} + \sin \varphi \cdot (\boldsymbol{\mu} \times \mathbf{a}) \quad (13.66)$$

$$\mathbf{a} \otimes z_\varphi = \mathbf{a} \otimes (\cos \varphi + \boldsymbol{\mu} \sin \varphi) = \cos \varphi \cdot \mathbf{a} - \sin \varphi \cdot (\boldsymbol{\mu} \times \mathbf{a}). \quad (13.67)$$

In diesem Spezialfall gilt:

$$|z_\varphi \otimes \mathbf{a}|^2 = |\mathbf{a} \otimes z_\varphi|^2 = \cos^2 \varphi |\mathbf{a}|^2 + \sin^2 \varphi |\boldsymbol{\mu} \times \mathbf{a}|^2 = (\cos^2 \varphi + \sin^2 \varphi) \cdot |\mathbf{a}|^2 = |\mathbf{a}|^2. \quad (13.68)$$

Aus den letzten beiden Gleichungen folgt dann:

$$z_\varphi \otimes \mathbf{a} = \mathbf{a} \otimes \bar{z}_\varphi = \mathbf{a} \otimes z_\varphi^{-1}, \quad \boldsymbol{\mu}^T \mathbf{a} = 0. \quad (13.69)$$

In diesem Falle ist das Produkt also zerlegt worden in eine Kosinuskomponente in \mathbf{a} -Richtung und eine Sinuskomponente in $\boldsymbol{\mu} \times \mathbf{a}$ -Richtung, d. h. aber, der Vektor \mathbf{a} beschreibt eine Drehung um α bezüglich der Drehachse $\boldsymbol{\mu}$! Dies ist ja eigentlich das, was wir analog der komplexen Zahlen herleiten wollten, nur mit dem Unterschied, dass wir noch die entscheidende Restriktion haben, dass \mathbf{a} senkrecht auf $\boldsymbol{\mu}$ stehen muss.

Daher betrachten wir nun den allgemeinen Fall ohne Restriktion an \mathbf{a} und versuchen diesen auf den bisherigen Fall zurückzuführen. Wir zerlegen \mathbf{a} in eine parallele und eine orthogonale Komponente bezüglich der Drehachse $\boldsymbol{\mu}$:

$$\mathbf{a} = \mathbf{a}_{\parallel} + \mathbf{a}_{\perp}. \quad (13.70)$$

Da das Distributivgesetz gilt, bilden wir folgenden Ausdruck:

$$z_\varphi \otimes (\mathbf{a}_{\parallel} + \mathbf{a}_{\perp}) \otimes z_\varphi^{-1} = z_\varphi \otimes \mathbf{a}_{\parallel} \otimes z_\varphi^{-1} + z_\varphi \otimes \mathbf{a}_{\perp} \otimes z_\varphi^{-1} = \mathbf{a}_{\parallel} + z_\varphi \otimes \mathbf{a}_{\perp} \otimes z_\varphi^{-1}. \quad (13.71)$$

Die Identität $z_\varphi \otimes \mathbf{a}_{\parallel} \otimes z_\varphi^{-1} = \mathbf{a}_{\parallel}$ ist nahezu trivial, zumindestens ist sie einfach und formal nachzurechnen. Der Ausdruck $z_\varphi \otimes \mathbf{a}_{\perp} \otimes z_\varphi^{-1}$ ist leicht interpretierbar. Dazu setzen wir Klammern $(z_\varphi \otimes \mathbf{a}_{\perp}) \otimes z_\varphi^{-1}$. Die erste Klammer beschreibt eine Rotation der orthogonalen Komponente um φ bezüglich der Drehachse $\boldsymbol{\mu}$. Anschließend wird dieses Ergebnis (welches ja wieder orthogonal zu $\boldsymbol{\mu}$ ist) wieder um φ gedreht, so dass schließlich die orthogonale Komponente insgesamt um $2 \cdot \varphi$ gedreht wird. Die Addition dieser mit \mathbf{a}_{\parallel} ergibt schließlich die eigentliche Rotation. Also beschreibt der Ausdruck $z_\varphi \otimes \mathbf{a} \otimes z_\varphi^{-1}$ die Rotation von \mathbf{a}

um den Winkel $2 \cdot \varphi$ bezüglich der Drehachse μ . Daher setzen wir als Einheitsquaternion $z_{\frac{\varphi}{2}} = \cos \frac{\varphi}{2} + \sin \frac{\varphi}{2} \mu = a + \mathbf{a}$ an und dann beschreibt

$$\mathbf{x}' = z_{\frac{\varphi}{2}} \otimes \mathbf{x} \otimes z_{\frac{\varphi}{2}}^{-1} \quad (13.72)$$

die gewünschte Rotation um φ und ist äquivalent der Transformation $\mathbf{x}' = \mathbf{R} \cdot \mathbf{x}$. Wir können diese Transformation auch mit der Eulerschen Formel aufschreiben:

$$\mathbf{x}' = e^{\mu \frac{\varphi}{2}} \otimes \mathbf{x} \otimes e^{-\mu \frac{\varphi}{2}}. \quad (13.73)$$

In vektorieller Form aufgelöst ergibt sich zunächst:

$$\mathbf{x}' = a^2 \mathbf{x} + \mathbf{a} \mathbf{x}^T \mathbf{a} - \mathbf{a} \mathbf{x} \times \mathbf{a} - a \mathbf{a}^T \mathbf{x} + (\mathbf{a}^T \mathbf{x}) \mathbf{a} + a \mathbf{a} \times \mathbf{x} + (\mathbf{a} \times \mathbf{x})^T \mathbf{a} - (\mathbf{a} \times \mathbf{x}) \times \mathbf{a}. \quad (13.74)$$

Ein Term ist Null und andere heben sich auf. Wenn wir für mehrfache Vektorprodukte die Regel:

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b}(\mathbf{a}^T \mathbf{c}) - \mathbf{c}(\mathbf{a}^T \mathbf{b}) \quad (13.75)$$

beachten, dann ist:

$$(\mathbf{a} \times \mathbf{x}) \times \mathbf{a} = -\mathbf{a}(\mathbf{a}^T \mathbf{x}) + \mathbf{x}(\mathbf{a}^T \mathbf{a}). \quad (13.76)$$

Folglich ergibt sich insgesamt:

$$\mathbf{x}' = (a^2 - \mathbf{a}^T \mathbf{a}) \mathbf{x} + 2\mathbf{a}(\mathbf{a} \times \mathbf{x}) + 2(\mathbf{a}^T \mathbf{x}) \mathbf{a}, a^2 + \mathbf{a}^2 = 1. \quad (13.77)$$

Wenn wir nun konkret in diese Transformation die obige Winkelschreibweise der Einheitsquaternion einsetzen und die Beziehungen $\sin 2\varphi = 2 \sin \varphi \cos \varphi$, $\cos 2\varphi = \cos^2 \varphi - \sin^2 \varphi$, $1 - \cos 2\varphi = 2 \sin^2 \varphi$ benutzen, so erhalten wir die Rotationsdarstellung nach Rodrigues:

$$\mathbf{x}' = \cos \varphi \mathbf{x} + \sin \varphi (\mu \times \mathbf{x}) + (1 - \cos \varphi) (\mu^T \mathbf{x}) \mu. \quad (13.78)$$

Als nächstes ist die Beschreibung der Hintereinanderausführung zweier Drehungen interessant:

- 1. Drehung: $z_{\mu_1, \frac{\varphi_1}{2}}$
- 2. Drehung: $z_{\mu_2, \frac{\varphi_2}{2}}$.

Damit ist:

$$\begin{aligned} \mathbf{a}_{rot} &= z_{\mu_2, \frac{\varphi_2}{2}} \otimes \left(z_{\mu_1, \frac{\varphi_1}{2}} \otimes \mathbf{a} \otimes z_{\mu_1, \frac{\varphi_1}{2}}^{-1} \right) \otimes z_{\mu_2, \frac{\varphi_2}{2}}^{-1} \\ &= \left(z_{\mu_2, \frac{\varphi_2}{2}} \otimes z_{\mu_1, \frac{\varphi_1}{2}} \right) \otimes \mathbf{a} \otimes \left(z_{\mu_2, \frac{\varphi_2}{2}} \otimes z_{\mu_1, \frac{\varphi_1}{2}} \right)^{-1}. \end{aligned} \quad (13.79)$$

Aus dieser Formel ist ersichtlich: Die Einheitsquaternion

$$z_{\mu_{12}, \psi} = z_{\mu_2, \frac{\varphi_2}{2}} \otimes z_{\mu_1, \frac{\varphi_1}{2}} \quad (13.80)$$

beschreibt die neue Drehachse und den neuen Drehwinkel, d. h. die Hintereinanderausführung ist die Multiplikation der beiden Einheitsquaternonen in umgekehrter Reihenfolge. Durch einfaches Ausrechnen finden wir:

$$\cos \frac{\psi}{2} = \cos \frac{\varphi_1}{2} \cos \frac{\varphi_2}{2} - (\mu_2^T \mu_1) \sin \frac{\varphi_1}{2} \sin \frac{\varphi_2}{2} \quad (13.81)$$

als skalare Komponente der Einheitsquaternion und

$$\mu_{12} \sin \frac{\psi}{2} = \sin \frac{\varphi_1}{2} \cos \frac{\varphi_2}{2} \mu_1 + \cos \frac{\varphi_1}{2} \sin \frac{\varphi_2}{2} \mu_2 + (\mu_2 \times \mu_1) \sin \frac{\varphi_1}{2} \sin \frac{\varphi_2}{2} \quad (13.82)$$

als vektorielle Komponente der neuen Einheitsquaternion.

13.2.5 Exponentielle Form einer Quaternion

Im Zusammenhang mit den Rotationen hatten wir die Eulersche Beziehung verwendet, insbesonders im Zusammenhang mit Einheitsquaternonen können wir schreiben:

$$z = a + \mathbf{a} = |z| \cdot e^{\mu \varphi}, \quad \mu = \frac{\mathbf{a}}{|\mathbf{a}|}, \quad \cos \varphi = \frac{a}{|z|}, \quad \sin \varphi = \frac{|\mathbf{a}|}{|z|}. \quad (13.83)$$

Dabei wird oft μ als *Eigenachse* und φ als *Eigenwinkel oder Phase* bezeichnet. Was bedeutet aber allemein e^z ? Wir können dies leicht auf die obige Darstellung zurückführen, aber man versteht grundsätzlich unter dieser Notation eine abkürzende Schreibweise für die Taylorreihe bezüglich der Quaternionen-Operationen:

$$e^z = \sum_{k=0}^{\infty} \frac{z^k}{k!}. \quad (13.84)$$

Für $z = a + ib + jc + kd$ schreiben wir nun:

$$e^z = e^{a+ib+jc+kd} = e^a \cdot e^{|\mathbf{a}|(\frac{\mathbf{a}}{|\mathbf{a}|})} = e^a \cdot e^{|\mathbf{a}|\mu}. \quad (13.85)$$

Ist $z_1 \otimes z_2 = z_2 \otimes z_1$ erfüllt, dann gilt auch die folgende Rechenregel:

$$e^{z_1} \otimes e^{z_2} = e^{z_1 \oplus z_2}. \quad (13.86)$$

Im Allgemeinen gilt dies aber auf keinen Fall. Auf jeden Fall gilt dann aber immer:

$$e^{\mu \varphi_1} \otimes e^{\mu \varphi_2} = e^{\mu(\varphi_1 + \varphi_2)}. \quad (13.87)$$

Als Gegenbeispiel für den ganz allgemeinen Fall dient:

$$e^{\pi i} \otimes e^{\pi j} = 1. \quad (13.88)$$

Andererseits ist aber

$$e^{\pi i \oplus \pi j} = \cos \pi\sqrt{2} + \frac{i+j}{\sqrt{2}} \sin \pi\sqrt{2}. \quad (13.89)$$

13.2.6 Quaternionen über komplexen Zahlen

Quaternionen bestehen aus vier Größen, komplexe Zahlen aus zwei Größen. Daher könnte man denken, zwei komplexe Zahlen könnten auch eine Quaternion bilden. Tatsächlich gibt es eine solche Herleitung, die wir kurz erläutern wollen.

Dazu fassen wir eine Quaternion als ein Paar von komplexen Zahlen auf:

$$z_1 = (q_1, q_2), \quad z_2 = (q_3, q_4), \quad q_i \in \mathbb{C}. \quad (13.90)$$

Die Addition und Multiplikation sind dann:

$$z_1 \oplus z_2 = (q_1 + q_3, q_2 + q_4) \quad (13.91)$$

$$z_1 \otimes z_2 = (q_1 q_3 - q_2 \overline{q_4}, q_1 q_4 + q_2 \overline{q_3}). \quad (13.92)$$

Das neutrale Element der Addition ist $n = (0, 0)$ und der Multiplikation ist $e = (1, 0)$. Die Kommutativität der Multiplikation gilt natürlich nicht, wenn es die gleichen Quaternionen sein sollen, die wir schon kennen. Eine beliebige Quaternion $z = (a + bi, c + di)$ lässt sich als Linearkombination der Elemente $(1, 0), (i, 0), (0, 1), (0, i)$ darstellen:

$$\begin{aligned} z &= (a + bi, c + di) \\ &= [(a, 0) \otimes (1, 0)] \oplus [(b, 0) \otimes (i, 0)] \oplus [(c, 0) \otimes (0, 1)] \oplus [(d, 0) \otimes (0, i)]. \end{aligned} \quad (13.93)$$

Mit der symbolischen Abkürzung:

- $(1, 0) \rightarrow 1$
- $(i, 0) \rightarrow i$
- $(0, 1) \rightarrow j$
- $(0, i) \rightarrow k$

können wir eine beliebige Quaternion durch $z = a + ib + jc + kd$ bezeichnen und erhalten unsere alte bekannte Darstellung.

13.2.7 Dot-Produkte

Das Wort Dot-Produkt ist ein Synonym für Skalarprodukt. Im Sinne einer Verallgemeinerung für Quaternionen gibt es mehrere Möglichkeiten, die in bestimmten Anwendungen benötigt werden. Mit $z_1 = a_1 + \mathbf{a}_1$ und $z_2 = a_2 + \mathbf{a}_2$ führen wir ein erstes Dot-Produkt ein:

$$z_1 \odot z_2 = \frac{1}{2} ((z_1 \otimes \overline{z_2}) \oplus (z_2 \otimes \overline{z_1})) = a_1 a_2 + \mathbf{a}_1^T \mathbf{a}_2. \quad (13.94)$$

Das Ergebnis ist also eine Quaternion mit ausschließlich dem Realteil oder eine skalare Zahl. Ein zweites Dot-Produkt ist folgendes:

$$z_1 \bullet z_2 = \frac{1}{2} ((z_2 \otimes \overline{z_1}) \oplus (z_1 \otimes z_2)) = a_1 a_2 + a_1 \mathbf{a}_2 + \mathbf{a}_1 \times \mathbf{a}_2. \quad (13.95)$$

Betrachten wir zwei rein vektorielle Quaternionen, also $z_1 = \mathbf{a}_1$, $z_2 = \mathbf{a}_2$, dann ergeben die beiden Dot-Produkte:

$$z_1 \odot z_2 = \mathbf{a}_1^T \mathbf{a}_2, \quad (13.96)$$

$$z_1 \bullet z_2 = \mathbf{a}_1 \times \mathbf{a}_2, \quad (13.97)$$

sie sind folglich identisch mit den beiden bekannten Produkten, dem Skalarprodukt und dem Kreuzprodukt.

13.2.8 Beispiele

Beispiel 1 Zunächst ein Trivialbeispiel. Folgende Rotation

1. Drehung: $\frac{\pi}{2}$ um x-Achse
2. Drehung: $\frac{\pi}{2}$ um y-Achse

soll durch die Drehachse und den Drehwinkel beschrieben werden. Natürlich können wir nun die Rotationsmatrix bestimmen und daraus die Drehachse und den Drehwinkel. Mit Hilfe der Quaternionen geht das aber viel schneller und eleganter. Es ist

$$x\text{-Achse: } \mathbf{u}_1 = i, \quad y\text{-Achse: } \mathbf{u}_2 = j.$$

Daher können wir sofort die beiden Einheitsquaternionen aufschreiben:

$$z_{\mathbf{u}_1, \frac{\varphi_1}{2}} = \cos \frac{\varphi_1}{2} + i \sin \frac{\varphi_1}{2} = \frac{1}{2} \sqrt{2} + i \frac{1}{2} \sqrt{2}, \quad (13.98)$$

$$z_{\mathbf{u}_2, \frac{\varphi_2}{2}} = \cos \frac{\varphi_2}{2} + j \sin \frac{\varphi_2}{2} = \frac{1}{2} \sqrt{2} + j \frac{1}{2} \sqrt{2}. \quad (13.99)$$

Für die Hintereinanderausführung brauchen wir nun noch das Quaternionenprodukt in umgekehrter Reihenfolge berechnen:

$$\begin{aligned} z_{\mu_2, \frac{\varphi_2}{2}} \otimes z_{\mu_1, \frac{\varphi_1}{2}} &= \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 \cdot j + \frac{1}{4} \cdot 2 \cdot i - \frac{1}{4} \cdot 2 \cdot k = \frac{1}{2}(1 + i + j - k) \\ &= \cos \frac{\pi}{3} + \frac{1}{\sqrt{3}}(i + j - k) \cdot \sin \frac{\pi}{3}. \end{aligned} \quad (13.100)$$

Also erfolgte eine Drehung bezüglich der Drehachse: $\mu_{12} = \frac{1}{\sqrt{3}}(i + j - k)$ mit dem Drehwinkel $\frac{2}{3}\pi = 120^\circ$.

Beispiel 2 Man berechne die Rotationsmatrix, die der Einheitsquaternion

$$z_{\mu, \frac{\varphi}{2}} = \frac{1}{2}(1 + i - j + k) \quad (13.101)$$

entspricht. Da wir mit der Einheitsquaternion sofort Drehachse und Drehwinkel ablesen können, könnten wir sofort die Rotationsmatrix berechnen. Mit den Quaternionen geht das aber wieder viel eleganter. Dazu nutzen wir direkt unsere Quaternionen-Drehformel:

$$\mathbf{x}_{rot} = \mathbf{x}' = z_{\mu, \frac{\varphi}{2}} \otimes \mathbf{x} \otimes z_{\mu, \frac{\varphi}{2}}^{-1}. \quad (13.102)$$

Daher setzen wir direkt an:

$$(0 + ix' + jy' + kz') = \frac{1}{2}(1 + i - j + k) \otimes (0 + ix + jy + kz) \otimes \frac{1}{2}(1 - i + j - k). \quad (13.103)$$

Wir rechnen die rechte Seite aus und erhalten:

$$ix' + jy' + kz' = -iy - jz + kx. \quad (13.104)$$

Daraus lesen wir die Rotationsmatrix ab:

$$\mathbf{R} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix}. \quad (13.105)$$

Beispiel 3 Umgekehrt sei nun die Rotationsmatrix:

$$\mathbf{R} = \begin{pmatrix} +\frac{2}{3} & +\frac{2}{3} & +\frac{1}{3} \\ +\frac{1}{3} & -\frac{2}{3} & +\frac{2}{3} \\ +\frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \end{pmatrix} \quad (13.106)$$

gegeben, gesucht sind Drehachse und Drehwinkel. Wenn wir nun die Einheitsquaternion ausrechnen, sind diese sofort ablesbar. Da dies die inverse Aufgabe zu Beispiel 2 ist, müssen wir die Einheitsquaternion als Unbekannte ansetzen und ansonsten genauso wie in Beispiel 2 verfahren. Also ist aufzuschreiben:

$$0 + ix' + jy' + kz' = (a + ib + jc + kd) \otimes (0 + ix + jy + kz) \otimes (a - ib - jc - kd) \quad (13.107)$$

und unter Berücksichtigung von $a^2 + b^2 + c^2 + d^2 = 1$ sind a, b, c, d auszurechnen.

13.2.9 Kanonisch exponentielle Darstellung einer Drehmatrix

Eine große Rolle werden dabei sogenannte *schiefsymmetrische* Matrizen \mathbf{U} spielen, für die $\mathbf{U}^T = -\mathbf{U}$ gilt. Die Hauptdiagonale ist also immer mit Nullen belegt und nur drei unbekannte Parameter bestimmen die (3×3) -Matrix \mathbf{U} (ebenso wie eine Rotationsmatrix). Wenn wir einen dreidimensionalen Vektor $\mathbf{u} = (u_1, u_2, u_3)^T \in R^3$ wählen, dann können wir diesem sofort eine schiefsymmetrische Matrix zuordnen:

$$\mathbf{u} \leftrightarrow \mathbf{U} = \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix}. \quad (13.108)$$

Wir haben die schiefsymmetrische Matrix offenbar so gewählt, dass sie mit dem Kreuzprodukt konsistent ist, d. h. bilden wir $\mathbf{w} = \mathbf{u} \times \mathbf{v}$, so ist ebenso $\mathbf{w} = \mathbf{U} \cdot \mathbf{v}$. Wir stellen uns einmal vor, die Rotationsmatrix hänge von der Zeit t ab, d. h. es gelte für alle t :

$$\mathbf{R}(t)\mathbf{R}^T(t) = \mathbf{I}. \quad (13.109)$$

Die Zeit können wir hier direkt mit dem Winkel im Bogenmaß identifizieren, wenn wir um eine feste Drehachse rotieren. Zu Beginn ist $t = 0$, damit der Winkel Null und bei $t = 2\pi$ haben wir eine komplette Rotation um die Dreiechse. Wir bilden die Ableitung nach der Zeit t und erhalten:

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) + \mathbf{R}(t)\dot{\mathbf{R}}^T(t) = 0 \rightarrow \dot{\mathbf{R}}(t)\mathbf{R}^T(t) = -(\dot{\mathbf{R}}(t)\mathbf{R}^T(t))^T, \quad (13.110)$$

d. h. die Matrix $\dot{\mathbf{R}}(t)\mathbf{R}^T(t)$ ist eine schiefsymmetrische Matrix. Damit können wir in unserer Notation auch schreiben

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) = \mathbf{U}(t) \rightarrow \dot{\mathbf{R}}(t) = \mathbf{U}(t) \cdot \mathbf{R}(t). \quad (13.111)$$

Wir nehmen nun einmal an, \mathbf{U} sei konstant, hänge also nicht von der Zeit t ab:

$$\dot{\mathbf{R}}(t) = \mathbf{U} \cdot \mathbf{R}(t). \quad (13.112)$$

Bevor wir dieses Differentialgleichungssystem lösen, betrachten wir dieses zunächst eindimensional:

$$\frac{dx}{dt} = K \cdot x, \quad K = \text{const} \rightarrow \frac{dx}{x} = K \cdot dt. \quad (13.113)$$

Die Lösung lautet:

$$\ln|x| = K \cdot t + C \rightarrow x(t) = e^{Kt} \cdot C' = e^{Kt} \cdot x(0). \quad (13.114)$$

Die nächste Verallgemeinerung ist die auf Vektoren. Daher schreiben wir das Differentialgleichungssystem für jeden Spaltenvektor der Rotationsmatrix getrennt auf, also für irgendeinen Vektor $\mathbf{x}(t)$:

$$\dot{\mathbf{x}}(t) = \mathbf{U} \cdot \mathbf{x}(t). \quad (13.115)$$

Dieses lineare System mit konstanten Koeffizienten besitzt in Verallgemeinerung des eindimensionalen Systems die Lösung:

$$\mathbf{x}(t) = e^{\mathbf{U}t} \mathbf{x}(0), \quad (13.116)$$

wobei

$$e^{\mathbf{U}t} = \mathbf{I} + \mathbf{U}t + \frac{(\mathbf{U}t)^2}{2!} + \dots + \frac{(\mathbf{U}t)^n}{n!} + \dots \quad (13.117)$$

ist. Die Lösung für die Rotationsmatrix wird von den drei Anfangsbedingungen für die drei Spaltenvektoren bestimmt. Wir wählen deshalb die Einheitsmatrix \mathbf{I} als die drei Startvektoren und erhalten somit:

$$\mathbf{R}(t) = e^{\mathbf{U}t}. \quad (13.118)$$

An Hand der Reihendarstellung und der Eigenschaft, dass \mathbf{U} schiefsymmetrisch ist, kann man leicht zeigen, dass

$$(e^{\mathbf{U}t})^{-1} = e^{-\mathbf{U}t} = e^{\mathbf{U}^T t} = (e^{\mathbf{U}t})^T \quad (13.119)$$

ist und wir damit tatsächlich eine Rotationsmatrix erhalten haben ($\det(e^{\mathbf{U}t}) = +1$ gilt auch noch). Dazu brauchen wir der Exaktheit halber noch zwei Eigenschaften, nämlich: Wenn $\mathbf{AB} = \mathbf{BA}$ gilt, dann gilt auch $e^{\mathbf{A}}e^{\mathbf{B}} = e^{\mathbf{B}}e^{\mathbf{A}} = e^{\mathbf{A}+\mathbf{B}}$. Außerdem gilt $\det(e^{\mathbf{A}}) = e^{\text{Spur}(\mathbf{A})}$. Die Interpretation unserer Darstellung ist die folgende:

Ist $\|\mathbf{u}\| = 1$, dann ist \mathbf{u} die Drehachse und t der Drehwinkel im Bogenmaß ist also ähnlich den Quaternionen eine kompakte Darstellung der Rotation mit Drehachse und Drehwinkel. Wir haben bisher gezeigt: Unter der Annahme, dass \mathbf{U} konstant ist, können wir aus Drehachse und Drehwinkel eine Rotationsmatrix berechnen, d. h. aber nicht, dass wir aus einer gegebenen Rotationsmatrix Drehachse und Drehwinkel in der exponentiellen Schreibweise angeben können.

Satz 13.2 Für jede beliebige Rotationsmatrix \mathbf{R} existiert ein nicht notwendig eindeutiges \mathbf{U} , sodass

$$\mathbf{R} = e^{\mathbf{U}} \quad (13.120)$$

gilt.

Beweis Der Beweis ist recht simpel zu führen. Wir berechnen wie im Abschn. 13.2.2 aus einer gegebenen Rotationsmatrix den Drehwinkel und Drehachse. Damit können wir die exponentielle Darstellung aufschreiben. Wir sehen gleichzeitig, dass alle Vektoren $2\pi k\mathbf{u}$ die gleiche Rotationsmatrix ergeben. \square

Wenn nun \mathbf{u} gegeben ist, wie berechnen wir die Rotationsmatrix ohne die Reihenentwicklung nutzen zu müssen? Wir könnten die Methoden der letzten Abschnitte benutzen, aber es gibt noch die Formel von **Rodrigues**:

$$\mathbf{R} = e^{\mathbf{U}} = \mathbf{I} + \frac{\mathbf{U}}{|\mathbf{u}|} \sin(|\mathbf{u}|) + \frac{\mathbf{U}^2}{|\mathbf{u}|^2} (1 - \cos(|\mathbf{u}|)). \quad (13.121)$$

Der Beweis ist auch recht einfach zu führen. Zunächst setzen wir $t = |\mathbf{u}|$ und anschließend benutzen wir $\mathbf{v} = \frac{\mathbf{u}}{|\mathbf{u}|}$ als Einheitsvektor. Ganz leicht nachzurechnen sind die beiden Beziehungen:

$$\mathbf{V}^2 = \mathbf{v}\mathbf{v}^T - \mathbf{I}, \quad \mathbf{V}^3 = -\mathbf{V}. \quad (13.122)$$

Daher kann man die Potenzreihe (13.117) mit dieser Eigenschaft vereinfachen zu:

$$e^{\mathbf{V}t} = \mathbf{I} + \left(t - \frac{t^3}{3!} + \frac{t^5}{5!} - \dots \right) \mathbf{V} + \left(\frac{t^2}{2!} - \frac{t^4}{4!} + \frac{t^6}{6!} - \dots \right) \mathbf{V}^2. \quad (13.123)$$

Wir sehen, die beiden (Teil-)Potenzreihen in (13.123) sind die Taylorreihen für $\sin(t)$ und $1 - \cos(t)$, daher erhalten wir:

$$e^{\mathbf{V}t} = \mathbf{I} + \mathbf{V} \sin(t) + \mathbf{V}^2(1 - \cos(t)). \quad (13.124)$$

An dieser Formel nach Rodrigues (13.121) erkennen wir sofort die Periodizität, d.h. $e^{\mathbf{V}2k\pi} = \mathbf{I}$. Weiterhin ist wichtig zu bemerken, dass im Allgemeinen:

$$e^{\mathbf{U}_1} e^{\mathbf{U}_2} \neq e^{\mathbf{U}_2} e^{\mathbf{U}_1} \neq e^{\mathbf{U}_1 + \mathbf{U}_2} \quad (13.125)$$

gilt. Aber: aus $\mathbf{AB} = \mathbf{BA}$, folgt $e^{\mathbf{A}} e^{\mathbf{B}} = e^{\mathbf{B}} e^{\mathbf{A}} = e^{\mathbf{A} + \mathbf{B}}$. Außerdem gilt $\det(e^{\mathbf{A}}) = e^{\text{Spur}(\mathbf{A})}$. Wir benutzen jetzt die Formel (13.121) nach Rodrigues. Dann können wir wie üblich die Rotation $\mathbf{x}' = \mathbf{Rx}$ nun in der Form:

$$\mathbf{x}' = \mathbf{x} + \sin(t)\mathbf{Ux} + (1 - \cos(t))\mathbf{U}^2\mathbf{x} \quad (13.126)$$

beschreiben. Die schiefsymmetrische Matrix \mathbf{U} korrespondiert mit dem Kreuzprodukt, folglich $\mathbf{U}\mathbf{x} = \mathbf{u} \times \mathbf{x}$, $\mathbf{U}^2\mathbf{x} = \mathbf{u} \times (\mathbf{u} \times \mathbf{x}) = (\mathbf{u}^T\mathbf{x})\mathbf{u} - \mathbf{x}$ mit $\mathbf{u}^T\mathbf{u} = 1$. Deshalb folgt:

$$\mathbf{x}' = \cos(t)\mathbf{x} + \sin(t)(\mathbf{u} \times \mathbf{x}) + (1 - \cos(t))(\mathbf{u}^T\mathbf{x})\mathbf{u}. \quad (13.127)$$

Vergleichen wir diese Formel mit der Formel (13.78) nach Rodrigues, so sehen wir, dass beide mit $t = \varphi$, $\mathbf{u} = \boldsymbol{\mu}$ gleich sind. Daher noch nachträglich die Erkenntnis, dass \mathbf{u} mit $|\mathbf{u}| = 1$ die normierte Drehachse und t der Drehwinkel im Bogenmaß sind.

13.2.10 Vor- und Nachteile

Euler-Transformation Diese hat den Vorteil, dass man intuitiv eine Vorstellung hat und die Eulerwinkel sich leicht interpretieren lassen. Außerdem kann man mit den Eulerwinkeln eine Interpolation durchführen.

Als Nachteile sind Mehrdeutigkeiten anzusehen, d. h. ist irgendeine 3D-Rotation gegeben (Rotationsmatrix), so lassen sich die Eulerwinkel im Allgemeinen nicht eindeutig bestimmen. Es treten sogar Entartungen auf: Betrachten wir irgendeine Rotation um die z -Achse, dann eine 90 Grad-Rotation um die y -Achse, so ist die folgende Rotation um die x -Achse redundant, d. h. man hätte sie auch gleich mit der Rotation um die z -Achse beschreiben können. In diesem Falle fallen also die Rotation um die z -Achse und um die x -Achse eigentlich zu einer Rotation zusammen, der Freiheitsgrad wird plötzlich um Eins reduziert. Diese Entartung nennt man in der Literatur **Gimbal Lock**. Dieser Begriff ist direkt vom Verschluss (Versagen, Blockade) einer Kardanischen Aufhängung abgeleitet. Der Körper in der Kardanischen Aufhängung ist oft ein Stabilisierungskreisel, Kreiselkompass oder Ähnliches.

Quaternionen Die Vorteile liegen auf der Hand: Es treten keine Mehrdeutigkeiten und Entartungen mehr auf. Als Nachteil wäre die sehr kompakte Form der Beschreibung und „man benötigt Wissen über die Quaternionen“ anzusehen.

13.2.11 Eine Anwendung zur Bestimmung einer Rotation

Wir wollen eine Rotation mittels der LSE-Methode (siehe Abschn. 21.2) bestimmen. Dazu haben wir eine Liste von 3D-Referenzpunkten gegeben, folglich:

$$\mathbf{x}'_i = \mathbf{R}\mathbf{x}_i, \quad i = 1, 2, \dots, n. \quad (13.128)$$

Was ist die minimale Anzahl von 3D-Referenzpunkten zur Bestimmung einer „reinen“ 3D Rotation aus (13.128)? Man könnte vermuten ein Referenzpaar, denn ein Referenzpaar liefert 3 Gleichungen und wir haben 3 Unbekannte, das müsste doch schon funktionieren.

Dies ist aber nicht so. Folgende Überlegung: wir haben im 3D-Raum einen Punkt und den rotierten Punkt gegeben, also zwei Punkte. Wir legen eine Drehachse durch den Koordinatenursprung und legen durch die zwei Punkte einen Kreis, der Kreis bestimmt eine Ebene und die Drehachse müsste senkrecht auf der Ebene stehen und durch den Kreismittelpunkt gehen. Dann haben wir eine Rotation des Punktes bezüglich der Drehachse gefunden, diese gedankliche Konstruktion muss richtig sein, denn es soll ja eine Rotation vorliegen, also muss es solch eine Drehachse mit dem Kreis auch geben. Nun drehen wir die Ebene bezüglich der Geraden, die durch die beiden Punkte geht und korrigieren die Drehachse so, dass sie weiterhin senkrecht auf der Ebene steht. Dadurch bleibt aber der Abstand des Durchstoßpunktes der Drehachse durch die Ebene zu den beiden Punkten untereinander gleich, d. h. nur untereinander gleich, absolut werden sie größer oder kleiner. Das heißt aber, die Drehachse durchstößt den Mittelpunkt eines anderen Kreises der durch die beiden Punkte geht. Folglich haben wir eine andere Drehachse mit einem anderen Drehwinkel konstruiert. Damit ist die Rotation nicht eindeutig bestimmbar. Wir benötigen folglich mindestens zwei Punktreferenzen.

Gesucht ist nun die Rotationsmatrix \mathbf{R} aus den n gegebenen Referenzpunkten. Da die Punkte fehlerbehaftet sind, haben wir das Ausgleichsproblem

$$S = \sum_{i=1}^n |\mathbf{x}'_i - \mathbf{Rx}_i|^2 \rightarrow \text{Minimum mit } \mathbf{R}^T \mathbf{R} = \mathbf{E}, \det(\mathbf{R}) = +1 \quad (13.129)$$

zu lösen. Durch die Restriktionen liegt ein kompliziertes nichtlineares Optimierungsproblem vor. Wir versuchen nun, die Rotation durch eine Einheitsquaternion zu beschreiben und zwar in der Hoffnung, dass das Problem numerisch gutartiger wird. Wie wir nun von den Einheitsquaternonen wissen, gilt:

$$S = \sum_{i=1}^n |\mathbf{x}'_i - z \otimes \mathbf{x}_i \otimes \bar{z}|^2, \quad (13.130)$$

wobei z eine Einheitsquaternion beschreibe mit $|z| = 1$. Damit gilt weiter:

$$S = \sum_{i=1}^n |\mathbf{x}'_i - z \otimes \mathbf{x}_i \otimes \bar{z}|^2 = \sum_{i=1}^n |\mathbf{x}'_i - z \otimes \mathbf{x}_i \otimes \bar{z}|^2 |z|^2 = \sum_{i=1}^n |\mathbf{x}'_i \otimes z - z \otimes \mathbf{x}_i|^2. \quad (13.131)$$

Wir haben dabei die wichtige Rechenregel (13.60) benutzt. Die Quaternionenmultiplikation ist eine lineare Operation in z , so dass wir wie üblich eine Matrix \mathbf{Y}_i finden können, so dass die gleiche Darstellung erreicht wird, d. h. es gilt:

$$\mathbf{x}'_i \otimes z - z \otimes \mathbf{x}_i \cong \mathbf{Y}_i \cdot \mathbf{z}, \quad (13.132)$$

wobei die Einheitsquaternion z als vier-elementiger Spaltenvektor \mathbf{z} geschrieben wird. Die Form von \mathbf{Y}_i ergibt sich aus den Darstellungen (13.58) und (13.59) zu:

$$\mathbf{Y}_i = \mathbf{X}'_i - \mathbf{X}_i^*. \quad (13.133)$$

Daraus folgt nun:

$$|\mathbf{x}'_i \otimes z - z \otimes \mathbf{x}_i|^2 \cong \mathbf{z}^T \mathbf{Y}_i^T \mathbf{Y}_i \cdot \mathbf{z}. \quad (13.134)$$

Damit haben wir folgendes Problem zu lösen:

$$S = \mathbf{z}^T \mathbf{Y} \mathbf{z} \rightarrow \text{Minimum mit } |\mathbf{z}| = 1 \quad (13.135)$$

mit

$$\mathbf{Y} = \sum_{i=1}^n \mathbf{Y}_i^T \mathbf{Y}_i. \quad (13.136)$$

Die Lösung ist recht einfach: Wir bestimmen den Eigenvektor von \mathbf{Y} zum kleinsten Eigenwert.

Genau dieses Ausgleichsproblem entsteht, wenn wir die sogenannte E -Matrix in die äußeren Kameraparameter zerlegen wollen, folglich in die Translation und die Rotation, siehe Problem (14.182). Um diese elegante Lösung auch auf Euklidische Transformationen, also Rotationen und Translationen zu übertragen, müssen wir einige Erweiterungen der Quaternionen durchführen. Dazu dienen die nächsten Abschnitte.

13.2.12 Duale Zahlen, Duplex-Zahlen

Diese wurden bereits im 19. Jahrhundert von Clifford eingeführt:

$$D = \{d \mid d = a + b\epsilon, a, b \in \mathbb{R}, \epsilon^2 = 0\}. \quad (13.137)$$

Man nennt a den Realteil und b den Dualteil. Addition und Multiplikation von $d_1 = a_1 + b_1\epsilon$, $d_2 = a_2 + b_2\epsilon$ ergeben sich dann:

$$d_1 + d_2 = (a_1 + a_2) + (b_1 + b_2)\epsilon \quad (13.138)$$

$$d_1 \cdot d_2 = (a_1 + b_1\epsilon) \cdot (a_2 + b_2\epsilon) = a_1 a_2 + (a_1 b_2 + a_2 b_1)\epsilon. \quad (13.139)$$

$(D, +, \cdot)$ ist ein kommutativer Ring, aber kein Körper, denn das inverse Element

$$d^{-1} = \frac{a - b\epsilon}{a^2} \quad (13.140)$$

von $d = a + b\epsilon$ ist für $a = 0$ nicht existent.

13.2.13 Duale Quaternionen

Diese werden analog den dualen Zahlen eingeführt:

$$\hat{\mathbb{H}} = \{\hat{q} \mid \hat{q} = q + q'\varepsilon, q, q' \in \mathbb{H}, \varepsilon^2 = 0\}. \quad (13.141)$$

Die Addition ist dann

$$\hat{q}_1 + \hat{q}_2 = (q_1 \oplus q_2) + (q'_1 \oplus q'_2)\varepsilon \quad (13.142)$$

mit $\hat{n} = 0 + 0\varepsilon$. Ähnlich den dualen Zahlen ist dann die Multiplikation:

$$\hat{q}_1 \cdot \hat{q}_2 = (q_1 + q'_1\varepsilon) \cdot (q_2 + q'_2\varepsilon) = q_1 \otimes q_2 + (q_1 \otimes q'_2 \oplus q'_1 \otimes q_2)\varepsilon. \quad (13.143)$$

Die konjugierte duale Quaternion ist definiert zu:

$$\overline{\hat{q}} = \overline{q} + \overline{q'}\varepsilon. \quad (13.144)$$

Daraus lassen sich die Eigenschaften

- $\overline{(\hat{q}_1 + \hat{q}_2)} = \overline{\hat{q}_1} + \overline{\hat{q}_2}$
- $\hat{q}_1 \cdot \overline{\hat{q}_2} = \overline{\hat{q}_2} \cdot \hat{q}_1$
- $\hat{q} \cdot \overline{\hat{q}} = (q \otimes \overline{q}) + (q \otimes \overline{q'} \oplus q' \otimes \overline{q})\varepsilon$

ableiten. Insbesondere, da die letzte Eigenschaft etwas aus der Reihe fällt, erzeugen wir mit:

$$\|\hat{q}\|^2 = \hat{q} \cdot \overline{\hat{q}} \quad (13.145)$$

keinen Betrag oder eine „richtige Norm“ und bezeichnen dies deshalb als Quasinorm. Trotzdem bezeichnen wir eine duale Quaternion $\hat{q} = q + q'\varepsilon$ mit der Eigenschaft:

$$\hat{q} \cdot \overline{\hat{q}} = 1 \quad (13.146)$$

als duale Einheitsquaternion. Diese Eigenschaft gilt also genau dann, wenn:

$$q \otimes \overline{q} = 1, \quad q \otimes \overline{q'} \oplus q' \otimes \overline{q} = 0. \quad (13.147)$$

Die inversen Elemente verhalten sich ebenso wie bei den dualen Zahlen. Die inverse duale Quaternion zu $\hat{q} = q + q'\varepsilon$ ist:

$$\hat{q}^{-1} = q^{-1} - (q^{-1} \otimes q' \otimes q^{-1})\varepsilon. \quad (13.148)$$

Das heißt, für duale Quaternionen mit $q = 0$ oder $\hat{q} = 0 + q'\epsilon$, existiert keine inverse duale Quaternion \hat{q}^{-1} .

Falls zwei duale Quaternionen invertierbar sind, gilt:

$$(\hat{q}_1 \cdot \hat{q}_2)^{-1} = \hat{q}_2^{-1} \cdot \hat{q}_1^{-1}. \quad (13.149)$$

13.2.14 Euklidische Transformationen und duale Quaternionen

Für gewöhnliche Spaltenvektoren können wir eine Ähnlichkeitstransformation durch

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} \quad (13.150)$$

beschreiben. Fassen wir diese Spaltenvektoren als reine Quaternionen auf, dann ist:

$$\mathbf{p}' = (\mathbf{q} \otimes \mathbf{p} \otimes \bar{\mathbf{q}}) \oplus \mathbf{t}. \quad (13.151)$$

Wir hatten die konjugierte duale Quaternion $\tilde{\hat{q}}$ eingeführt. Eine ähnliche konjugierte Operation bezeichnen wir jetzt mit dem Tilde-Operator. Mit $\hat{q} = q + q'\epsilon$ ist dann:

$$\tilde{\hat{q}} = q - q'\epsilon. \quad (13.152)$$

Für diese konjugierte Operation gelten die Rechenregeln:

- $\tilde{\hat{g}} = \bar{\tilde{\hat{q}}}$
- $(\hat{q}_1 + \hat{q}_2) = \tilde{\hat{q}}_1 + \tilde{\hat{q}}_2$
- $(\hat{q}_1 \cdot \hat{q}_2) = \tilde{\hat{q}}_1 \cdot \tilde{\hat{q}}_2.$

Nun können wir Euklidische Transformationen mit dualen Einheitsquaternonen beschreiben:

$$\hat{p}' = 1 + p'\epsilon = \hat{q} \cdot \hat{p} \cdot \tilde{\hat{q}}. \quad (13.153)$$

Dabei ist \hat{q} eine duale Einheitsquaternion mit den Bedingungen (13.146) und (13.147). Ein Punkt wird durch $\hat{p} = 1 + p\epsilon$ dargestellt und der transformierte Punkt durch $\hat{p}' = 1 + p'\epsilon$, wobei p und p' rein vektorielle Quaternionen sind, die die Punkte repräsentieren. In der dualen Einheitsquaternion muss nun die Rotation und die Translation kodiert sein. Diese ist:

$$\hat{q} = q + \frac{1}{2}(t \otimes q)\epsilon, \quad (13.154)$$

wobei q eine Einheitsquaternion ist, die die Rotation beschreibt. Damit werden wir jetzt die Transformation (13.153) nachrechnen und zeigen, dass sie stimmt. Es ist:

$$\begin{aligned}
 \hat{p}' &= 1 + p' \varepsilon = \hat{q} \cdot \hat{p} \cdot \tilde{\hat{q}} = \left(q + \frac{1}{2}(t \otimes q) \varepsilon \right) \cdot (1 + p \varepsilon) \cdot \left(\bar{q} - \frac{1}{2}(\bar{t} \otimes \bar{q}) \varepsilon \right) \\
 &= \left[q \otimes \bar{q} + \left[q \otimes p \otimes \bar{q} \oplus \frac{1}{2}(t \otimes q) \otimes \bar{q} \ominus \frac{1}{2}q \otimes \bar{q} \otimes \bar{t} \right] \varepsilon \right] \\
 &= 1 + [(q \otimes p \otimes \bar{q}) \oplus t] \varepsilon \\
 &= 1 + p' \varepsilon.
 \end{aligned} \tag{13.155}$$

Nun müssen wir nur noch zeigen, dass \hat{q} eine duale Einheitsquaternion ist:

$$\begin{aligned}
 \hat{q} \cdot \bar{q} &= \left(q + \frac{1}{2}(t \otimes q) \varepsilon \right) \cdot \left(\bar{q} + \frac{1}{2}(\bar{t} \otimes \bar{q}) \varepsilon \right) \\
 &= q \otimes \bar{q} + \left[\frac{1}{2}(q \otimes \bar{q} \otimes \bar{t}) \oplus \frac{1}{2}(t \otimes q \otimes \bar{q}) \right] \varepsilon \\
 &= 1 + \frac{1}{2}(\bar{t} \oplus t) \varepsilon = 1 + 0\varepsilon = 1.
 \end{aligned} \tag{13.156}$$

Damit haben wir (13.153) bewiesen, d. h. dass sich (genau wie Rotationen durch Einheitsquaternionen) Euklidische Transformationen durch duale Einheitsquaternionen beschreiben lassen.

14.1 Kameramodelle

Alle Modellvorstellungen über eine sinnvolle $3D \rightarrow 2D$ Abbildung werden als *Kameramodelle* bezeichnet. Es bezeichne $\mathbf{x}_W = (x_W, y_W, z_W)^T$ einen Punkt in einem *Weltkoordinatensystem*. Weiterhin sei $\mathbf{x}_K = (x_K, y_K, z_K)^T$ ein Punkt in einem 3D-Kamera-Koordinatensystem, wobei stets die z_K -Achse senkrecht zur eigentlichen (u, v) -Bildebene sei. Die Transformation vom Weltkoordinatensystem in das Kamerakoordinatensystem geschieht stets durch eine $3D \rightarrow 3D$ Bewegung, also durch eine Rotation (Rotationsmatrix \mathbf{R}) und eine Translation \mathbf{t} . Der Sinn dieser Bewegung besteht darin: Die eigentlichen Abbildungsgleichungen sollen in einem festen Kamerakoordinatensystem beschrieben werden. Dazu benötigen wir die Koordinaten eines realen, abzubildenden Punktes in diesem Kamerakoordinatensystem. Dies ist aber in der Regel nicht möglich, sondern der abzubildende Punkt wird in einem völlig anderen Bezugssystem dargestellt, dem *Weltkoordinatensystem*. Daher müssen wir jetzt die Koordinaten umrechnen und dies geschieht durch Bewegung des Koordinatensystems. Die Rotation verlangt drei Parameter und die Translation ebenfalls drei Parameter. Diese sechs Parameter werden als *Parameter der äußeren Orientierung* einer Kamera bezeichnet:

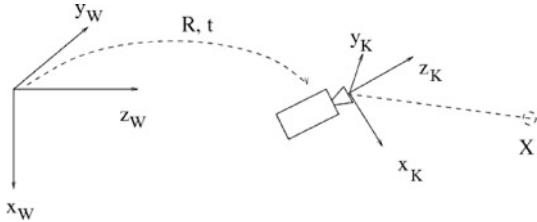
$$\mathbf{x}_K = \mathbf{R} \cdot \mathbf{x}_W + \mathbf{t}. \quad (14.1)$$

Wir schreiben diese Bewegung in homogenen Koordinaten auf:

$$\begin{pmatrix} x_K \\ y_K \\ z_K \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} = \tilde{\mathbf{R}}\tilde{\mathbf{T}} \cdot \tilde{\mathbf{x}}_W. \quad (14.2)$$

Man beachte: Ist die letzte Koordinate gleich 1, dann stimmen kartesische mit den homogenen Koordinaten überein. Ist dies nicht der Fall, so setzen wir eine „Schlange“ über das Symbol.

Abb. 14.1 Lage Weltkoordinatensystem \leftrightarrow Kamerakoordinatensystem: äußere Orientierung einer Kamera



14.1.1 Orthografische Projektion

Das einfachste Kameramodell ist die Beschreibung der Abbildung durch eine orthogonale Parallelprojektion, genannt *orthografische Projektion* (*orthographic projection*). Die Abbildungsgleichungen sind in diesem Falle simpel durch

$$u = x_K, \quad v = y_K \quad (14.3)$$

gegeben. Damit wird dieses einfachste Kameramodell durch fünf Parameter beschrieben:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}. \quad (14.4)$$

Obwohl acht Parameter in den Abbildungsgleichungen stehen, sind es nur fünf, denn wir müssen daran denken, dass die r_{ij} Elemente einer Rotationsmatrix sind, d. h. es gilt:

$$r_{11}^2 + r_{12}^2 + r_{13}^2 = 1, \quad r_{21}^2 + r_{22}^2 + r_{23}^2 = 1, \quad r_{11}r_{21} + r_{12}r_{22} + r_{13}r_{23} = 0. \quad (14.5)$$

Acht Parameter und diese drei Restriktionen ergeben genau die fünf Parameter dieses Kameramodells. Bei diesem einfachsten Modell ist sogar ein Parameter der äußeren Orientierung überflüssig.

14.1.2 Skalierte orthografische Projektion

Das orthografische Abbildungs-Modell (14.4) berücksichtigt keinerlei Skalierungen und ist deshalb völlig unrealistisch für praktische Anwendungen. Eine Skalierung bewirkt eigentlich der Sensor immer, aber auch die Abbildung durch telezentrische Objektive kann eine Skalierung bewirken. Deshalb ist die nächste Stufe das erste realistische Kameramodell, die Hintereinanderausfuehrung einer orthografischen Projektion und einer isotropen Skalierung:

$$u = s \cdot x_K, \quad v = s \cdot y_K, \quad (14.6)$$

wobei s ein positiver Skalierungsfaktor sei. Dieses Kameramodell gilt näherungsweise für Objekte, die „unendlich entfernt sind“ von der Kamera, oder wenn wir spezielle *telezentrische* Objektive verwenden. Oft wird dieses Kameramodell als Schwache Perspektive (*weak perspective camera*) bezeichnet. Damit wird dieses Kameramodell mit sechs Parametern beschrieben. Im Sinne von Transformationen realisieren wir eine spezielle $3D \rightarrow 2D$ affine Abbildung des Weltkoordinatensystems in die (u, v) -Bildecke. Diese können wir nun auch einfach aufschreiben:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} s \cdot r_{11} & s \cdot r_{12} & s \cdot r_{13} & s \cdot t_1 \\ s \cdot r_{21} & s \cdot r_{22} & s \cdot r_{23} & s \cdot t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}. \quad (14.7)$$

Bisher konnten wir in der Darstellung noch kartesische Koordinaten verwenden, in homogenen Koordinaten lässt sich die Abbildung auch schreiben zu:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ 0 & 0 & 0 & \frac{1}{s} \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}; \quad u = \frac{\tilde{u}}{\tilde{w}}, \quad v = \frac{\tilde{v}}{\tilde{w}}. \quad (14.8)$$

Aus dieser Darstellung können die sechs Parameter abgelesen werden.

14.1.3 Anisotrop skalierte orthografische Projektion

Das nächsthöhere Modell hat sieben Parameter und ist deutbar als Hintereinanderausführung einer orthografischen Projektion mit anschliessender anisotroper Skalierung:

$$u = s_1 \cdot x_K, \quad v = s_2 \cdot y_K, \quad (14.9)$$

Die Abtastung eines Signals kann zu anisotropen Skalierungen führen, deshalb modelliert diese Kamera den „Gesamtabbildungssprozess“ besser als die *weak perspective*. Die Abbildung lautet:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 \cdot r_{11} & s_1 \cdot r_{12} & s_1 \cdot r_{13} & s_1 \cdot t_1 \\ s_2 \cdot r_{21} & s_2 \cdot r_{22} & s_2 \cdot r_{23} & s_2 \cdot t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}. \quad (14.10)$$

Dieses Modell ist in der Praxis kaum anzutreffen.

14.1.4 Affine Kamera

Das nächsthöhere Modell enthält dann acht Parameter. Dieses Modell wird als *affine Kamera* bezeichnet. Die affine Kamera enthält acht unabhängige Parameter. Es ist allerdings unklar, welcher realistische optische Abbildungsprozess eigentlich zur affinen Kamera führt. Da wir acht unabhängige Parameter zur Verfügung haben, ist es günstiger, wir benennen die Parameter neu:

Eine allgemeine $3D \rightarrow 2D$ affine Abbildung wird in homogenen Koordinaten durch

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{10} \\ a_{21} & a_{22} & a_{23} & a_{20} \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \mathbf{A} \cdot \mathbf{x}_w \quad (14.11)$$

beschrieben. Weit entfernte Objekte können näherungsweise mit dem affinen Modell beschrieben werden.

14.1.5 Spezielle affine Kameras

Die *weak perspective* war eine spezielle affine Kamera, nämlich eine orthografische Projektion mit anschließender Skalierung. Wir können uns dies auch anschaulich mit der Zentralprojektion (siehe Abb. 14.3) vorstellen. Dazu haben wir eine Bildebene, ein Projektionszentrum, und betrachten eine Objektebene, die parallel zur Bildebene liegt. Nun bilden wir irgendeinen 3D-Punkt ab. Dazu projizieren wir ihn senkrecht auf die Objektebene, und nun bilden wir diesen Punkt in der Objektebene mit einem Sehstrahl in die Bildebene ab. Durch den Sehstrahl erreichen wir die gewünschte Skalierung.

Nun können wir aber auch die 3D-Punkte durch **Parallelprojektion** in die Objektebene abbilden, und dann mit dem Sehstrahl in die Bildebene abbilden. Diese spezielle affine Kamera nennt man **Paraperspektive**.

14.1.6 Pinhole camera

Die nächsthöhere Stufe eines Kameramodells beinhaltet die zentralprojektive Abbildung $3D \rightarrow 2D$. Dazu müssen wir eine perspektivische Projektion des Kamerakoordinatensystems in die Bildebene beschreiben. Als Kameramodell dient dazu die ideale Lochkamera (*pinhole camera*). Das Wort Lochkamera ist dabei fast wörtlich zu nehmen, wir stellen uns einen dunklen Kasten mit einem sehr kleinen Loch vor, durch das das Licht einströmen kann. Im Kasten befindet sich eine Fotoplatte, die belichtet wird. Unsere Lochkamera hat also keine Optik, nur ein Loch. Das Loch ist somit vergleichbar mit einer sehr kleinen Blendenöffnung, damit haben wir einen sehr großen Tiefenschärfebereich. Der einzige Nachteil

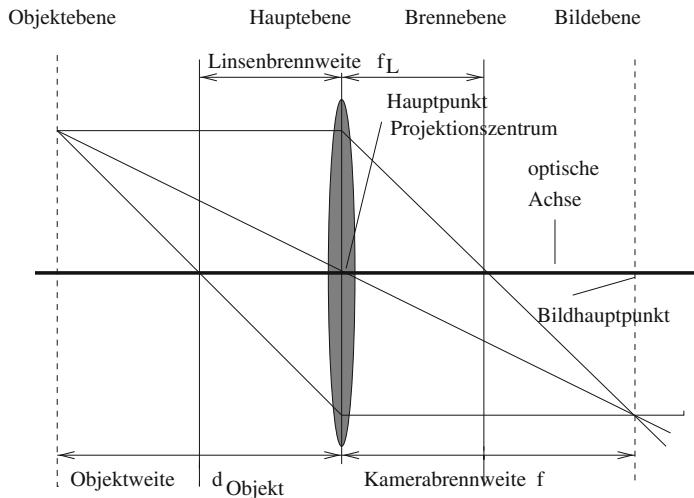


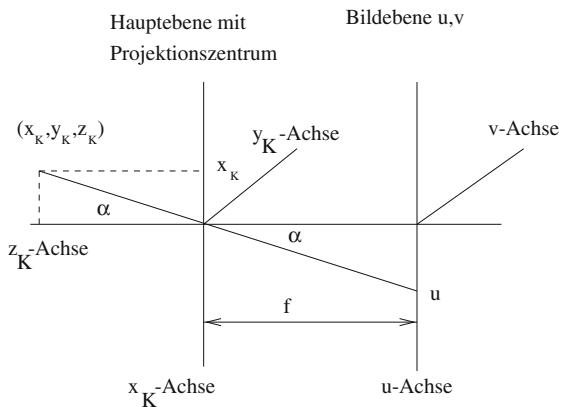
Abb. 14.2 Modell der dünnen Linse mit Bezeichnungen

ist, dass genügend Licht eintreten muss, was in der Praxis nicht der Fall sein wird. Deshalb benötigen wir ein Objektiv, um die Blendenöffnung zu vergrößern, damit genug Licht eintreten kann. Im Folgenden demonstrieren wir dies an dem Modell der „dünnen Linse“ und zeigen, dass wir dennoch am Lochkameramodell festhalten können. Als erstes Modell nehmen wir an, die optische Achse sei identisch mit der z -Achse des Kamera-Koordinatensystems und stehe senkrecht auf der Bildebene. Damit ist die Projektionsebene parallel zur Bildebene. Der Abstand f des Projektionszentrums zur Bildebene wird als *Kamerabrennweite* oder *Kamerakonstante*, manchmal auch als *Kammerkonstante* bezeichnet. Die Kamerabrennweite f ist i. Allg. nicht identisch mit der *Linsenbrennweite*. Für eine dünne Linse gilt vielmehr die Beziehung:

$$\frac{1}{f_L} = \frac{1}{f} + \frac{1}{d_{\text{Objekt}}}. \quad (14.12)$$

Dabei ist f_L die Linsenbrennweite und d_{Objekt} der Abstand der Linse zur Objektebene (Gegenstandsweite). Für weit entfernte Objekte können wir näherungsweise die Linsenbrennweite benutzen, siehe Abb. 14.2. In der Abbildung ist auch ersichtlich: Das Modell der Lochkamera abstrahiert eigentlich auf eine Blende als „Punkt“, besitzt also eine unendliche Tiefenschärfe. Aus der Abbildung geht hervor, dass eine endliche Blende, also mit einer dünnen Linse, das Modell dennoch benutzbar ist, auch wenn keine unendliche Tiefenschärfe mehr vorliegt. Unschärfen stören das Modell in unserem Sinne nicht. In der Regel werden sogenannte sphärische Linsen eingesetzt, die wie jede andere Linse auch, natürlich Linsenfehler besitzt. Die wichtigsten Linsenfehler sind sogenannte Verzeichnungen der Objektive, die wir erst später modellieren wollen. Legen wir nun entsprechend Abb. 14.2 für das Kamerakoordinatensystem folgendes fest:

Abb. 14.3 Lochkameramodell: Zentralprojektion mit Welt- und Kamerakoordinatensystem

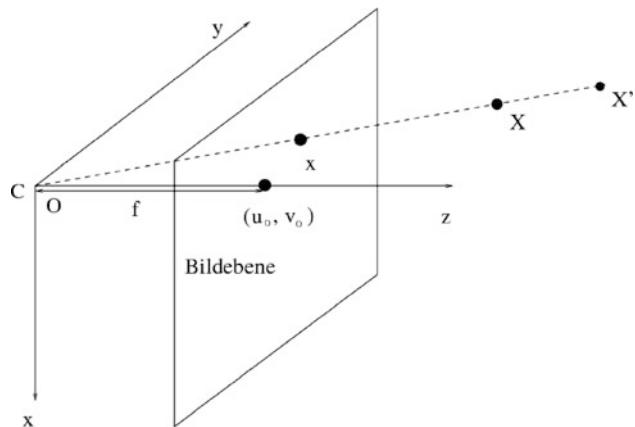


- a) Der Ursprung ist das Projektionszentrum.
- b) Die (x_K, y_K) -Ebene ist parallel zur (u, v) -Bildebene.
- c) Die orthogonale Projektion der x_K -Achse stimmt mit der u -Achse überein und die orthogonale Projektion der y_K -Achse stimmt mit der v -Achse überein.
- d) Die z_K -Achse ist die optische Achse.

Wie wir in Abb. 14.3 sehen, sind dann die x_K, y_K -Koordinaten und die u, v -Koordinaten gerade spiegelbildlich (wie in unserem Auge auf der Retina) und wir müssten mit negativen Vorzeichen rechnen. Wie haben nun zwei Möglichkeiten:

- Wir vertauschen die Orientierungen der u, v -Achsen gespiegelt gegenüber den x_K, y_K -Achsen.
- Wir verschieben die u, v -Ebene im Abstand f vor das Projektionszentrum, um die Spiegelung zu vermeiden.

Abb. 14.4 Lochkameramodell: andere Darstellung von Abb. 14.3



Durch eine der beiden Möglichkeiten vermeiden wir nun die „Mitführung eines Minuszeichens“. Folglich können wir leicht nach dem $\tan(\alpha)$ die Verhältnisse (ohne Minuszeichen)

$$\frac{u}{f} = \frac{x_K}{z_K}, \quad \frac{v}{f} = \frac{y_K}{z_K} \quad (14.13)$$

ableiten. Daraus erhalten wir die Abbildungsgleichungen (ohne Minuszeichen):

$$u = \frac{f \cdot x_K}{z_K}, \quad v = \frac{f \cdot y_K}{z_K}. \quad (14.14)$$

Oft findet man auch in der Literatur, dass die (x_K, y_K) -Ebene im Abstand f „links“ oder „rechts“ vom Projektionszentrum festgelegt wird. Nach „links“ bedeutet, dass das Projektionszentrum die Koordinaten $(0, 0, -f)$ besitzt und die Abbildungsgleichungen werden zu:

$$u = \frac{z_K \cdot x_K}{f + f}, \quad v = \frac{z_K \cdot y_K}{f + f}. \quad (14.15)$$

Nach „rechts“ bedeutet, dass das Projektionszentrum die Koordinaten $(0, 0, +f)$ besitzt. Die Abbildungsgleichungen werden dann zu:

$$u = \frac{f \cdot x_K}{z_K - f}, \quad v = \frac{z_K \cdot y_K}{f - f}. \quad (14.16)$$

Wir legen uns im Folgenden auf die erste Form fest. Als *äußere Kameraparameter (extrinsic parameters)* werden einheitlich die sechs Parameter der Translation und Rotation des Weltkoordinatensystems in das Kamerakoordinatensystem bezeichnet. Diese hängen von der Lage der Kamera im Weltkoordinatensystem ab und verändern sich natürlich bei Bewegung der Kamera. Die Abbildungsparameter der Abbildung des Kamerakoordinatensystems in die Bildebene ändern sich nicht bei Bewegung der Kamera und werden demzufolge als *innere Kameraparameter (intrinsic parameters)* bezeichnet oder als Parameter der inneren Orientierung. Zweifelsohne ist dies die Kamerabrennweite f und der Bildhauptpunkt (u_0, v_0) , der i. Allg. nicht mit der Bildmitte identisch ist. Dies sind die drei wesentlichen Parameter der inneren Orientierung. Bis hierher ist die Abbildung ausschließlich optisch erklärt und verständlich. Im Bild messen wir aber Pixelkoordinaten, die mit den Sensorkoordinaten im Kamerakoordinatensystem übereinstimmen müssten. Dies ist aber nicht der Fall. Nun gibt es zwei Möglichkeiten, erstens, der Hersteller von Sensoren gibt uns die exakte Sensorgeometrie an und wir rechnen die Pixelkoordinaten auf Grund der Sensorgeometrie um, oder zweitens, was einfacher ist, diese Umrechnung nehmen wir mit in den Abbildungsprozess auf, obwohl es mit dem bisher beschriebenen, optischen Abbildungsprozess nichts zu tun hat. Mindestens haben wir eine isotrope Skalierung der u - und v -Achse durch den Sensor zu berücksichtigen. Dann ergibt sich:

$$u = \frac{s \cdot f \cdot x_K}{z_K}, \quad v = \frac{s \cdot f \cdot y_K}{z_K}. \quad (14.17)$$

Bei diesem Modell ist der dritte Parameter das Produkt $s \cdot f$, die Kamerakonstante f ist demnach in diesem Modell nicht mehr explizit bestimbar, nur das Produkt aus dieser mit dem positiven Skalierungsfaktor s . Das liegt nun daran, dass wir bei einer Skalierung nicht mehr wissen, welcher Anteil optisch und welcher durch den Sensor bedingt ist. Diesen Nachteil hätten wir nicht, wenn wir fordern, dass der Hersteller die Sensorgeometrie angibt. Das Kameramodell kann nun noch dahin erweitert werden, dass man annimmt, die Bildachsen u und v werden vom Sensor unterschiedlich, also anisotrop skaliert:

$$u = \frac{s_{x_K} \cdot f \cdot x_K}{z_K}, \quad v = \frac{s_{y_K} \cdot f \cdot y_K}{z_K}. \quad (14.18)$$

Dann haben wir vier innere Parameter, nämlich den Bildhauptpunkt und die beiden Produkte aus Skalierungsfaktor und Kamerakonstante. Wir sehen, es lassen sich nur diese Produkte angeben und nicht mehr explizit die Kamerakonstante. Schliesslich kann man noch annehmen, dass die Sensorgeometrie eine Scherung verursachen kann. Damit stehen die Basisvektoren nicht unbedingt mehr senkrecht aufeinander. Der Winkel zwischen diesen Basisvektoren in der Bildebene ist dann der fünfte innere Kameraparameter. Für den Fall der Zentralprojektion unterscheiden wir also im wesentlichen drei Kameramodelle:

- Sechs äußere Parameter, drei innere (Bildhauptpunkt und Produkt aus Kamerakonstante und isotropen Skalierungsfaktor), also insgesamt neun unabhängige Parameter,
- Sechs äußere Parameter, vier innere (Bildhauptpunkt, Produkte aus anisotropen Skalierungsfaktoren mit der Kamerakonstante), also insgesamt zehn unabhängige Parameter
- Sechs äußere Parameter, fünf innere (Bildhauptpunkt, Produkte aus anisotropen Skalierungsfaktoren mit der Kamerakonstante, Winkel zwischen den gescherten Basisvektoren der Bildebene), also insgesamt 11 unabhängige Parameter.

Für das erste Modell wollen wir nun die Gesamtabbildung aufschreiben. Da wir den Bildhauptpunkt (u_0, v_0) bezüglich eines von uns willkürlich gewählten Koordinatenursprungs in der Bildebene nicht kennen, müssen wir in den Abbildungen u und v durch die Relativkoordinaten $u - u_0$ und $v - v_0$ ersetzen und erhalten:

$$u - u_0 = \frac{\alpha \cdot x_K}{z_K}, \quad v - v_0 = \frac{\alpha \cdot y_K}{z_K} \quad \text{mit } \alpha = s \cdot f. \quad (14.19)$$

Mit einer formal eingeführten Grösse d kann man dies auch in der Form

$$\begin{aligned} d \cdot u &= \alpha \cdot x_K + u_0 \cdot z_K \\ d \cdot v &= \alpha \cdot y_K + v_0 \cdot z_K \\ d &= z_K \end{aligned} \quad (14.20)$$

schreiben, was zu homogenen Koordinaten führt. Dies schreiben wir nun in Matrixform auf:

$$\begin{pmatrix} d \cdot u \\ d \cdot v \\ d \end{pmatrix} = \begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \alpha & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_K \\ y_K \\ z_K \\ 1 \end{pmatrix} = \tilde{\mathbf{K}} \cdot \tilde{\mathbf{x}}_K = \mathbf{K} \cdot \mathbf{x}_K. \quad (14.21)$$

Die Matrix $\tilde{\mathbf{K}}$ bzw. die (3×3) -Matrix \mathbf{K} bezeichne stets die Projektion aus dem Kamerakoordinatensystem in die Bildebene, sie beinhaltet damit stets die inneren Kameraparameter. Daher wird oft auch die Matrix \mathbf{K} als interne Projektionsmatrix bezeichnet. Die Bewegung des Weltkoordinatensystems in das Kamerakoordinatensystem können wir simpel durch

$$\mathbf{x}_K = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \mathbf{R} \cdot \mathbf{x}_W + \mathbf{t} \quad (14.22)$$

beschreiben. Um wieder mit homogenen Koordinaten zu rechnen, schreiben wir statt dessen:

$$\begin{pmatrix} x_K \\ y_K \\ z_K \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} = \tilde{\mathbf{R}} \tilde{\mathbf{T}} \cdot \tilde{\mathbf{x}}_W. \quad (14.23)$$

Diese Beziehung setzen wir nun in die Abbildungsvorschrift für die Kamerakoordinaten ein und erhalten:

$$\tilde{\mathbf{u}} = \tilde{\mathbf{K}} \cdot \tilde{\mathbf{R}} \tilde{\mathbf{T}} \cdot \tilde{\mathbf{x}}_W = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_W \quad (14.24)$$

$$\tilde{\mathbf{u}} = \mathbf{K} \cdot (\mathbf{R} \mathbf{x}_W + \mathbf{t}) = \mathbf{K} \cdot \mathbf{R} \cdot \mathbf{x}_W + \mathbf{K} \cdot \mathbf{t}. \quad (14.25)$$

Ausführlich aufgeschrieben ergibt sich:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \alpha \cdot r_{11} + u_0 \cdot r_{31} & \alpha \cdot r_{12} + u_0 \cdot r_{32} & \alpha \cdot r_{13} + u_0 \cdot r_{33} & \alpha \cdot t_1 + u_0 \cdot t_3 \\ \alpha \cdot r_{21} + v_0 \cdot r_{31} & \alpha \cdot r_{22} + v_0 \cdot r_{32} & \alpha \cdot r_{23} + v_0 \cdot r_{33} & \alpha \cdot t_2 + v_0 \cdot t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}. \quad (14.26)$$

Betrachten wir noch einmal das perspektivische Modell (14.14) in Kamerakoordinaten und lassen den Index K weg, dafür betrachten wir aber n Punkte \mathbf{x}_i , $i = 1, \dots, n$:

$$u_i = \frac{f \cdot x_i}{z_i}, \quad v_i = \frac{f \cdot y_i}{z_i}. \quad (14.27)$$

Es sei $\bar{z} = \frac{1}{n} \sum_i z_i$, dann schreiben wir:

$$u_i = \frac{f \cdot x_i}{\bar{z} + \Delta z_i}, \quad v_i = \frac{f \cdot y_i}{\bar{z} + \Delta z_i}. \quad (14.28)$$

Beide Ausdrücke entwickeln wir bez. der Tiefe z in eine Taylorreihe:

$$u_i = \frac{f \cdot x_i}{\bar{z} + \Delta z_i} = \frac{f}{\bar{z}} \left(1 - \frac{\Delta z_i}{\bar{z}} + \frac{(\Delta z_i)^2}{\bar{z}^2} - \dots \right) \cdot x_i, \quad (14.29)$$

$$v_i = \frac{f \cdot y_i}{\bar{z} + \Delta z_i} = \frac{f}{\bar{z}} \left(1 - \frac{\Delta z_i}{\bar{z}} + \frac{(\Delta z_i)^2}{\bar{z}^2} - \dots \right) \cdot y_i. \quad (14.30)$$

Betrachten wir nur das Glied Nullter Ordnung, dann liegt das Modell 2 (Schwache Perspektive) vor, wobei die Skalierung $s = \frac{f}{\bar{z}}$ ist. Wir können also \bar{z} als mittlere Tiefe aller Punkte auffassen. Wenn folglich alle Punkte in einer Ebene liegen und diese Ebene parallel zur Bildebene ist, dann liegt bei einer Kamera tatsächlich die Schwache Perspektive vor.

14.1.7 Pinhole camera – Erweiterung 1

Im Modell (14.26) hatten wir nur neun Parameter, damit bestehen also Abhängigkeiten zwischen den eigentlich 11 Matrixkoeffizienten. Das erweiterte Modell mit nun 10 Parametern lässt sich sofort auch aufschreiben, es folgt mit

$$\tilde{\mathbf{K}} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (14.31)$$

wieder $\tilde{\mathbf{u}} = \tilde{\mathbf{K}} \cdot \tilde{\mathbf{R}} \tilde{\mathbf{T}} \cdot \tilde{\mathbf{x}}_W = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_W$, oder ausführlich:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} \alpha_u \cdot r_{11} + u_0 \cdot r_{31} & \alpha_u \cdot r_{12} + u_0 \cdot r_{32} & \alpha_u \cdot r_{13} + u_0 \cdot r_{33} & \alpha_u \cdot t_1 + u_0 \cdot t_3 \\ \alpha_v \cdot r_{21} + v_0 \cdot r_{31} & \alpha_v \cdot r_{22} + v_0 \cdot r_{32} & \alpha_v \cdot r_{23} + v_0 \cdot r_{33} & \alpha_v \cdot t_2 + v_0 \cdot t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}. \quad (14.32)$$

14.1.8 Pinhole camera – allgemeiner Fall

Das dritte Modell mit nun 11 Parametern entspricht formal der allgemein bekannten zentralprojektiven Abbildung in homogenen Koordinaten, also mit

$$\tilde{\mathbf{K}} = \begin{pmatrix} \alpha_u & \alpha_s = \alpha_v \cdot \cot \varphi & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (14.33)$$

und wieder $\tilde{\mathbf{u}} = \tilde{\mathbf{K}} \cdot \tilde{\mathbf{R}}^T \cdot \tilde{\mathbf{x}}_W = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_W$ folgt allgemein:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{10} \\ a_{21} & a_{22} & a_{23} & a_{20} \\ a_{31} & a_{32} & a_{33} & a_{30} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}. \quad (14.34)$$

Da man einen der 12 Koeffizienten normieren kann, haben wir 11 unabhängige Parameter, dies entspricht dem dritten Modell mit sechs äußeren und fünf inneren Kameraparametern. Werden im Folgenden die Indizes weggelassen, dann handelt es sich immer um Weltkoordinaten. Wenn wir die Matrix der inneren Parameter (14.31) beachten sowie (14.34) und (14.32) vergleichen, sehen wir, dass die dritte Zeile in der allgemeinen Projektionsmatrix (14.34) spezifiziert werden kann zu:

$$\mathbf{a}_3^T = (a_{31}, a_{32}, a_{33}) = \mathbf{r}_3^T = (r_{31}, r_{32}, r_{33}), a_{30} = t_3. \quad (14.35)$$

14.1.9 Normalisierte Koordinaten

Zerlegen wir nun z. B. die „interne Projektionsmatrix $\tilde{\mathbf{K}}$ “ in $\tilde{\mathbf{K}} = (\mathbf{K}, 0)$ und gehen zu neuen Koordinaten über

$$\tilde{\mathbf{u}}_N = \mathbf{K}^{-1} \tilde{\mathbf{u}} = \mathbf{K}^{-1} \cdot \tilde{\mathbf{K}} \cdot \tilde{\mathbf{R}}^T \cdot \tilde{\mathbf{x}}_W, \quad (14.36)$$

so nennt man diese Koordinaten $\tilde{\mathbf{u}}_N$ *normalisierte Koordinaten*. Wir abstrahieren damit von den inneren Parametern, wir tun so, als ob wir eine *ideale Kamera* zur Verfügung hätten. Alle inneren Parameter dieser idealen Kamera sind Null bzw. Eins. Demzufolge ist die interne Projektionsmatrix die Einheitsmatrix. Wir können also schreiben:

$$\tilde{\mathbf{u}}_N = \tilde{\mathbf{E}} \cdot \tilde{\mathbf{x}}_K = (\mathbf{E}, 0) \cdot \tilde{\mathbf{x}}_K = \mathbf{E} \cdot \mathbf{x}_K = \mathbf{x}_K. \quad (14.37)$$

Damit sind die homogenen Koordinaten der Bildpunkte identisch mit den 3D-Koordinaten des Kamerakoordinatensystems. Die Identität gilt natürlich nur bis auf einen unbekannten Faktor, da wir somit nur den Sehstrahl zur Verfügung haben und nicht den absoluten 3D-Punkt. Bei einer idealen Kamera ergänzen wir folglich die 2D-Punkte einfach mit einer Eins und erhalten die Richtung des Sehstrahles. Mit normalisierten Koordinaten kann man recht erfolgreich rechnen, man setzt dann stillschweigend voraus, dass die inneren Kameraparameter bestimmt wurden und die homogenen Pixelkoordinaten durch Multiplikation mit der inversen internen Projektionsmatrix umgerechnet wurden.

Lage des Weltkoordinatensystems Manchmal (z. B. in der Robotik) ist die Lage des Weltkoordinatensystems äußerst wichtig. Oft soll aber nur die Oberfläche eines Objektes rekon-

struiert werden, d. h. deren Form ist wichtig und nicht die Lage in einem Welkoordinaten- system. In solch einem Falle können wir das Weltkoordinatensystem „hinlegen wo wir wollen“. Wir müssen nur darauf achten, dass dann die äußereren Kameraparameter durch Kalibrierung auch bestimmbar sind. Bezuglich photogrammetrischer Rekonstruktion brauchen wir normalerweise zwei Kameras, dies nennen wir eine Stereo-Anordnung. Mit dieser können wir durch Triangulation zweier Bildpunkte den 3D-Punkt berechnen. Eine besonders einfache Stereoanordnung zweier Kameras ist der sogenannte Stereo-Normalfall. Zwei identische Kameras (d. h. mit gleichen inneren Parametern) sind nur bezüglich der x -Achse verschoben, nicht verdreht gegeneinander. Die optischen Achsen beider Kameras sind dann parallel. Wenn wir mit dieser Stereo-Anordnung eine 3D-Oberfläche rekonstruieren wollen, dann legen wir einfach das Weltkoordinatensystem in das Kamerakoordinatensystem der ersten Kamera. Damit lauten in diesem Weltkoordinatensystem die Abbildungsgleichungen der ersten Kamera:

$$\tilde{\mathbf{u}} = \mathbf{K} \cdot \mathbf{x}_W. \quad (14.38)$$

Die Abbildung der zweiten Kamera müssen wir nun in dem gleichen Weltkoordinatensystem beschreiben. Wir wissen, die zweite Kamera ist bezüglich der x -Achse um d verschoben, alles andere bleibt identisch. Daher ergibt sich für die zweite Kamera:

$$\tilde{\mathbf{v}} = \mathbf{K} \cdot \mathbf{x}_W + \mathbf{K} \cdot (-d, 0, 0)^T. \quad (14.39)$$

Die unbekannten Parameter sind nun die inneren Parameter und die Verschiebung d . Die inneren Parameter kann man durch Kalibrierung bestimmen und das d muss durch die Konstrukteure des Stereopaars bekannt sein bzw. irgendwie vermessen worden ein. Solch eine Kamerapaar ist sicher teuer in der Herstellung, da die Parallelität exakt garantiert werden muss.

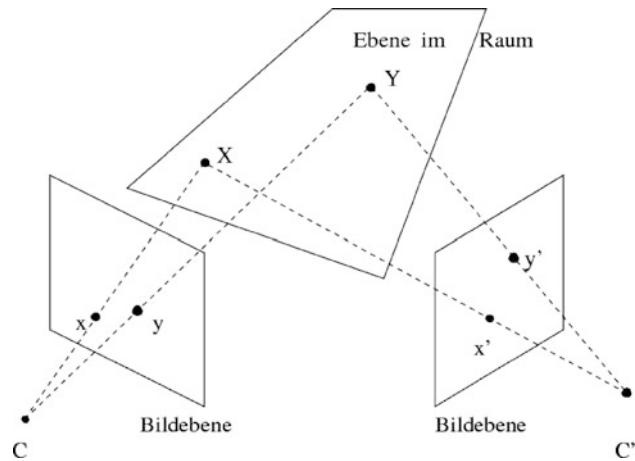
14.1.10 Homographien

Es liege von einer Kamera zwei Aufnahmen vor. In der 3D-Szene liegen Objekte in einer Ebene, z. B. Fenster in einer Häuserwand. Dann existiert zwischen den Abbildungen der Ebene in beiden Bildern eine Homographie, also eine $2D \leftrightarrow 2D$ projektive Abbildung, siehe Abb. 14.5. Zur Berechnung der Homographie benötigen wir mindestens vier Punktreferenzpaare.

14.1.11 Kamerainvarianten

Je nach Abbildungsmodell (dabei wird höchstens die perspektivische Abbildung zugelassen, keine Verzeichnungen usw.) gibt es sechs realistische, verschiedene Kameramodelle,

Abb. 14.5 Homographie:
2D \leftrightarrow 2D-projektive Abbil-
dung bezüglich der beiden
Bildebenen



sechs äußere Kameraparameter gilt es zu bestimmen. Es gibt maximal fünf innere Kameraparameter, die *invariant* gegenüber Bewegungen der Kamera sind und bei einer neuen Kameraposition eigentlich nicht wieder berechnet zu werden brauchen. Bei den zentralperspektivischen Modellen ist es unrealistisch, eine Skalierung s durch den Sensor auszuschließen. Damit kann aber nur das Produkt $s \cdot f$ bestimmt werden, aber nicht die Kammerkonstante f selbst. Gehen wir von fünf inneren Kameraparametern aus, so müssen wir diese als *Invariante*n der zentralperspektivischen Abbildung gegenüber Bewegungen aus den Abbildungsparametern wieder berechnen können. Bei fünf inneren Parametern müssen wir Modell (14.34) annehmen. Die Abbildungsgleichungen schreiben wir jetzt pro homogener Koordinate

$$\begin{aligned}\tilde{u} &= \mathbf{a}_1^T \cdot \mathbf{x} + a_{10} \\ \tilde{v} &= \mathbf{a}_2^T \cdot \mathbf{x} + a_{20} \\ \tilde{w} &= \mathbf{a}_3^T \cdot \mathbf{x} + a_{30},\end{aligned}\tag{14.40}$$

oder eleganter in Matrixschreibweise:

$$\tilde{\mathbf{A}} = (\mathbf{A}, \mathbf{a}), \quad \tilde{\mathbf{u}} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{a}.\tag{14.41}$$

Eine Bewegung beschreiben wir durch $\mathbf{x} = \mathbf{R} \cdot \mathbf{x}' + \mathbf{t}$, diese setzen wir in die Transformationsformeln (14.40) ein und erhalten:

$$\begin{aligned}\tilde{u} &= \mathbf{a}_1^T \cdot \mathbf{R} \cdot \mathbf{x}' + \mathbf{a}_1^T \cdot \mathbf{t} + a_{10} \\ \tilde{v} &= \mathbf{a}_2^T \cdot \mathbf{R} \cdot \mathbf{x}' + \mathbf{a}_2^T \cdot \mathbf{t} + a_{20} \\ \tilde{w} &= \mathbf{a}_3^T \cdot \mathbf{R} \cdot \mathbf{x}' + \mathbf{a}_3^T \cdot \mathbf{t} + a_{30}\end{aligned}\tag{14.42}$$

Die uns interessierenden Transformationen können wir nun ablesen:

$$\mathbf{a}_1'^T = \omega \cdot \mathbf{a}_1^T \cdot \mathbf{R}; \quad \mathbf{a}_2'^T = \omega \cdot \mathbf{a}_2^T \cdot \mathbf{R}; \quad \mathbf{a}_3'^T = \omega \cdot \mathbf{a}_3^T \cdot \mathbf{R},\tag{14.43}$$

wobei ω infolge der homogenen Koordinaten ein unbekannter Faktor ist. Damit können wir nun sechs Skalarprodukte bilden:

$$\mathbf{a}'_i^T \mathbf{a}'_j = \omega^2 \cdot \mathbf{a}_i^T \mathbf{a}_j, \quad i, j = 1, 2, 3, \quad i \leq j. \quad (14.44)$$

Um Invarianten zu erhalten, müssen wir die Faktoren ω^2 eliminieren. Dies ist leicht durch 15 Quotientenbildungen möglich. Allerdings können nur 5 von diesen 15 Quotienten unabhängig voneinander sein, denn es gibt nur 5 innere Parameter. Wir entscheiden uns willkürlich für die folgenden fünf:

$$I_{11} = \frac{\mathbf{a}_1^T \mathbf{a}_1}{\mathbf{a}_3^T \mathbf{a}_3}, \quad I_{12} = \frac{\mathbf{a}_1^T \mathbf{a}_2}{\mathbf{a}_3^T \mathbf{a}_3}, \quad I_{13} = \frac{\mathbf{a}_1^T \mathbf{a}_3}{\mathbf{a}_3^T \mathbf{a}_3}, \quad I_{22} = \frac{\mathbf{a}_2^T \mathbf{a}_2}{\mathbf{a}_3^T \mathbf{a}_3}, \quad I_{23} = \frac{\mathbf{a}_2^T \mathbf{a}_3}{\mathbf{a}_3^T \mathbf{a}_3}. \quad (14.45)$$

Diese fünf Invarianten müssen zumindestens Funktionen der fünf inneren Kameraparameter sein, wenn nicht sogar einige identisch mit inneren Kameraparametern sind.

Eine weitere Schlussfolgerung aus Modell (14.34) ist folgende: Sind alle homogenen Koordinaten $\tilde{u} = 0, \tilde{v} = 0, \tilde{w} = 0$ gleich Null, dann heißt dies, die kartesischen Koordinaten sind „unbestimmte Ausdrücke“, der einzige Punkt, wo Bildkoordinaten unbestimmt sein müssen, ist das Projektionszentrum $\mathbf{x}_c = (x_c, y_c, z_c)^T$ selbst:

$$\begin{aligned} 0 &= \mathbf{a}_1^T \cdot \mathbf{x}_c + a_{10} \\ 0 &= \mathbf{a}_2^T \cdot \mathbf{x}_c + a_{20} \\ 0 &= \mathbf{a}_3^T \cdot \mathbf{x}_c + a_{30}. \end{aligned} \quad (14.46)$$

In Matrixschreibweise ergibt sich

$$\mathbf{A}\mathbf{x}_c + \mathbf{a} = \mathbf{0} \rightarrow \mathbf{x}_c = -\mathbf{A}^{-1}\mathbf{a}. \quad (14.47)$$

Auf Grund dieser Eigenschaft können wir die Abbildungsgleichungen auch in der Form

$$\begin{aligned} \tilde{u} &= \mathbf{a}_1^T \cdot (\mathbf{x} - \mathbf{x}_c) \\ \tilde{v} &= \mathbf{a}_2^T \cdot (\mathbf{x} - \mathbf{x}_c) \\ \tilde{w} &= \mathbf{a}_3^T \cdot (\mathbf{x} - \mathbf{x}_c) \end{aligned} \quad (14.48)$$

schreiben. In Matrixschreibweise ist dann:

$$\tilde{\mathbf{u}} = \mathbf{A}(\mathbf{x} - \mathbf{x}_c). \quad (14.49)$$

Betrachten wir nochmals die dritte Ebene: $0 = \mathbf{a}_3^T \cdot (\mathbf{x} - \mathbf{x}_c)$. Diese geht durch das Projektionszentrum und jeder weitere Punkt dieser Ebene wird auf den unendlich fernen Punkt

abgebildet. Aus diesen beiden Eigenschaften folgt, dass diese Ebene parallel zur Bildebene sein muss. Damit ist der Vektor \mathbf{a}_3 ein *Normalenvektor* aller Ebenen, die zur Bildebene parallel sind. Den Bildhauptpunkt $(u_0, v_0)^T$ können wir relativ leicht ausrechnen. Dazu betrachten wir die optische Achse, die durch das Projektionszentrum geht und senkrecht auf der Bildebene steht, daher lautet deren Parameterdarstellung $\mathbf{x} - \mathbf{x}_c = \lambda \mathbf{a}_3$. Diese setzen wir in die Abbildungsgleichungen (14.48) ein und erhalten:

$$\begin{aligned}\tilde{u}_0 &= \mathbf{a}_1^T \cdot (\lambda \mathbf{a}_3) \\ \tilde{v}_0 &= \mathbf{a}_2^T \cdot (\lambda \mathbf{a}_3) \\ \tilde{w}_0 &= \mathbf{a}_3^T \cdot (\lambda \mathbf{a}_3).\end{aligned}\quad (14.50)$$

Nun interessieren uns die kartesischen Koordinaten des Bildhauptpunktes, diese sind:

$$u_0 = I_{13} = \frac{\mathbf{a}_1^T \mathbf{a}_3}{\mathbf{a}_3^T \mathbf{a}_3}, \quad v_0 = I_{23} = \frac{\mathbf{a}_2^T \mathbf{a}_3}{\mathbf{a}_3^T \mathbf{a}_3}, \quad (14.51)$$

welche zwei der fünf Kameravarianten darstellen und gleichzeitig zwei der fünf inneren Kameraparameter sind. Als nächsten inneren Parameter wollen wir den Winkel zwischen den gesuchten Basisvektoren in der Bildebene berechnen. Wir berechnen somit im x, y, z -Koordinatensystem zunächst die Vektoren, die die Geraden $v = 0$ bzw. $u = 0$ bestimmen. Dies ist recht einfach, denn \mathbf{a}_3 steht senkrecht auf der Bildebene und aus obigen Abbildungsgleichungen ist ersichtlich, dass $u = 0$ bzw. $v = 0$ Ebenen beschreiben, deren Normalenvektoren senkrecht auf den Achsen stehen und sofort ablesbar sind. Damit kann man über Vektorprodukte sofort die Richtungsvektoren der gesuchten Achsen ausrechnen. Mit Hilfe des Skalarproduktes berechnen wir deren Schnittwinkel, benutzen die sogenannte Lagrangesche Identität für Mehrfachprodukte und erhalten:

$$\cos \varphi = \frac{I_{12} - I_{13}I_{23}}{\sqrt{(I_{11}^2 - I_{13}^2)(I_{22}^2 - I_{23}^2)}}. \quad (14.52)$$

Bei diesem Ausdruck zur Berechnung der Scherung ist es numerisch besser, das Produkt $\mathbf{a}_3^T \mathbf{a}_3$ in Zähler und Nenner wieder auszuklammern und zu kürzen, da sonst erhebliche numerische Probleme auftreten. Damit haben wir einen weiteren inneren Kameraparameter bestimmt. Durch ähnliche, einfache Überlegungen lassen sich auch die Skalierungsfaktoren berechnen, es ergibt sich:

$$s_{x_K} \cdot f = \sqrt{I_{11} - I_{13}^2}, \quad s_{y_K} \cdot f = \sqrt{I_{22} - I_{23}^2}, \quad (14.53)$$

womit wir die letzten beiden inneren Kameraparameter berechnet haben. In der praktischen 3D-Rekonstruktion benötigen wir natürlich die Parameter eines Kamera-Modells. Die Bestimmung der Parameter eines Kamera-Modells nennt man *Kalibrierung*.

14.1.12 Homogene Koordinaten via Kartesische Koordinaten

Je nach Anwendung ist die eine Form gegenüber der anderen effektiver bzw. führt zur kompakteren Schreibweise. Mit kartesischen Koordinaten können wir unsere Vorstellungen über die analytische Geometrie des 3D-Raumes besser darstellen, während die homogenen Koordinaten zur einer besseren algebraischen Darstellung führen. Die bessere algebraische Darstellung sei kurz an der Berechnung eines Sehstrahlvektors gezeigt. Der Sehstrahl geht durchs Projektionszentrum, daher lautet die Geradendarstellung des Sehstrahles:

$$\mathbf{x} = \mathbf{x}_c + \lambda \mathbf{t}. \quad (14.54)$$

Diesen setzen wir in die Abbildungsgleichung $\tilde{\mathbf{u}} = \mathbf{A}\mathbf{x} + \mathbf{a}$ ein und erhalten:

$$\tilde{\mathbf{u}} = \mathbf{A}(\mathbf{x}_c + \lambda \mathbf{t}) + \mathbf{a} = [\mathbf{A}\mathbf{x}_c + \mathbf{a}] + \lambda \mathbf{A}\mathbf{t} = \lambda \mathbf{A}\mathbf{t}. \quad (14.55)$$

Daraus folgt nun simpel der Richtungsvektor \mathbf{t} bis auf einen Faktor zu:

$$\mathbf{t} = \mathbf{A}^{-1} \tilde{\mathbf{u}}. \quad (14.56)$$

In diesem Falle müssen wir uns mit kartesischen Koordinaten etwas mehr plagen:

Dazu schreiben wir die Abbildungsgleichungen kartesisch auf:

$$u = \frac{\langle \mathbf{a}_1, \mathbf{x} - \mathbf{x}_c \rangle}{\langle \mathbf{a}_3, \mathbf{x} - \mathbf{x}_c \rangle}, \quad v = \frac{\langle \mathbf{a}_2, \mathbf{x} - \mathbf{x}_c \rangle}{\langle \mathbf{a}_3, \mathbf{x} - \mathbf{x}_c \rangle}. \quad (14.57)$$

Durch leichte Umformungen sehen wir:

$$\langle u\mathbf{a}_3 - \mathbf{a}_1, \mathbf{t} \rangle = 0, \langle v\mathbf{a}_3 - \mathbf{a}_2, \mathbf{t} \rangle = 0. \quad (14.58)$$

Damit haben wir zwei Vektoren gefunden, die senkrecht auf dem Richtungsvektor \mathbf{t} stehen, damit ist:

$$\mathbf{t} = (u\mathbf{a}_3 - \mathbf{a}_1) \times (v\mathbf{a}_3 - \mathbf{a}_2) = \mathbf{a}_1 \times \mathbf{a}_2 + u\mathbf{a}_2 \times \mathbf{a}_3 + v\mathbf{a}_3 \times \mathbf{a}_1. \quad (14.59)$$

Mit der Abkürzung

$$\mathbf{C} = (\mathbf{a}_2 \times \mathbf{a}_3, \mathbf{a}_3 \times \mathbf{a}_1, \mathbf{a}_1 \times \mathbf{a}_2) \quad (14.60)$$

können wir auch schreiben:

$$\mathbf{t} = \mathbf{C} \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \quad (14.61)$$

womit wir die Beziehung zu den homogenen Koordinaten wieder hergestellt haben, denn es gilt allgemein für (3×3) Matrizen:

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} (\mathbf{a}_2 \times \mathbf{a}_3, \mathbf{a}_3 \times \mathbf{a}_1, \mathbf{a}_1 \times \mathbf{a}_2). \quad (14.62)$$

Im Zusammenhang mit dem Kreuzprodukt wollen wir noch an die Operatorschreibweise erinnern, also mit $\mathbf{a}^T = (a_1, a_2, a_3), \mathbf{b}^T = (b_1, b_2, b_3)$ ist:

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}. \quad (14.63)$$

Ein weiteres schönes Beispiel zur Nutzung von homogenen Koordinaten wäre es, die Bildgerade einer im Raum verlaufenden Geraden $\mathbf{x} = \mathbf{x}_p + \lambda \mathbf{t}$ zu bestimmen. Dazu wählen wir einfach im Raum zwei Punkte auf der Geraden aus, z. B. für $\lambda = 0, \lambda = 1$ und berechnen die Bildpunkte:

$$\tilde{\mathbf{u}}_0 = \mathbf{Ax}_p + \mathbf{a}, \quad \tilde{\mathbf{u}}_1 = \mathbf{Ax}_p + \lambda \mathbf{At} + \mathbf{a}. \quad (14.64)$$

Die in der Bildebene durch zwei Punkte gehende Gerade wird in homogenen Koordinaten eindeutig durch das Kreuzprodukt bestimmt $\tilde{\mathbf{u}}_0 \times \tilde{\mathbf{u}}_1$. Nun können wir auch noch leicht den Normalenvektor der durch die Bildgerade und das Projektionszentrum verlaufende Ebene bestimmen zu $\mathbf{A}^{-1} \tilde{\mathbf{u}}_0 \times \mathbf{t}$. Dabei ist \mathbf{t} der Richtungsvektor der gegebenen Raumgeraden. Diese Raumebene nennt man *Blickebene*. Den Zusammenhang zwischen Blickebene und Bildgeraden können wir uns algebraisch mit homogenen Koordinaten noch einfacher und übersichtlicher herleiten. Es sei wie immer $\tilde{\mathbf{u}} = \mathbf{Ax} + \mathbf{a}$ die Abbildungsgleichung und $\mathbf{x} = \mathbf{x}_p + \lambda \mathbf{t}$ sei die Gleichung der Raumgeraden. Weiterhin sei $\mathbf{n} = \mathbf{t} \times (\mathbf{x}_p - \mathbf{x}_c)$ ein Normalenvektor der Blickebene. Dann können wir schreiben:

$$\begin{aligned} \tilde{\mathbf{u}} &= \mathbf{Ax}_p + \lambda \mathbf{At} + \mathbf{a} \\ &= \mathbf{Ax}_p + \mathbf{a} + \lambda \mathbf{At} - \mathbf{Ax}_c - \mathbf{a} \\ &= \mathbf{A}[(\mathbf{x}_p - \mathbf{x}_c) + \lambda \mathbf{t}] \\ &= \mathbf{Ad}. \end{aligned} \quad (14.65)$$

Da der Punkt \mathbf{d} in der Blickebene liegt und $\mathbf{d} = \mathbf{A}^{-1} \tilde{\mathbf{u}}$ ist, folgt $\mathbf{n}^T \cdot \mathbf{A}^{-1} \tilde{\mathbf{u}} = 0$. Wenn nun allgemein $\mathbf{p}^T \cdot \tilde{\mathbf{u}} = 0$ die Gleichung einer Geraden in der Bildebene ist, so ergibt sich der einfache Zusammenhang:

$$\mathbf{A}^{-1} \cdot \mathbf{n} = \mathbf{p} \Leftrightarrow \mathbf{n} = \mathbf{A}^T \cdot \mathbf{p}. \quad (14.66)$$

14.2 Bewegung der Kamera

Die Bewegung einer Kamera hatten wir schon im Zusammenhang mit den Kamerainvarianten behandelt, siehe Abschn. 14.1.11. Wir nehmen an, alle Abbildungsparameter einer Kamera seien bestimmt, also es liegt bereits eine kalibrierte Kamera vor. Wenn wir nun die Kamera bewegen, sind die Kameraparameter nicht mehr richtig, es müsste neu kalibriert werden. Oft kennt man aber die genaue Bewegung im Weltkoordinatensystem, z. B. wenn eine Kamera an einem Roboterarm befestigt ist, dann kann die Bewegung im körpereigenen Koordinatensystem beschrieben werden. Kennt man den Bezug von Roboterkoordinatensystem und Weltkoordinatensystem, dann ist natürlich die Bewegung auch im Weltkoordinatensystem beschreibbar. Eine Bewegung wird beschrieben durch $\mathbf{x} = \mathbf{R}\mathbf{x}' + \mathbf{t}$. Wir kennen also die Rotation \mathbf{R} und die Translation \mathbf{t} . Nun schreiben wir wieder die Abbildungsgleichung $\tilde{\mathbf{u}} = \mathbf{Ax} + \mathbf{a}$ auf und setzen die Bewegung ein:

$$\tilde{\mathbf{u}} = \mathbf{A}\mathbf{Rx}' + \mathbf{At} + \mathbf{a}. \quad (14.67)$$

Damit können wir die neuen Abbildungsparameter sofort ablesen:

$$\mathbf{A}' = \mathbf{A} \cdot \mathbf{R}, \quad \mathbf{a}' = \mathbf{At} + \mathbf{a}. \quad (14.68)$$

Dies nennt man *Rekalibrierung* einer Kamera. Gibt es nun Bewegungen einer Kamera ohne dass sich das Bild ändert, also invariant bleibt? Wir fragen konkret, gibt es 3D-Punkte oder Objekte, die gegen jegliche Bewegung der Kamera im Bild invariant sind, also stets das gleiche Abbild ergeben? Dazu schreiben wir nochmals die Abbildungsgleichungen auf:

$$\tilde{\mathbf{u}} = \mathbf{K} \cdot [\mathbf{R}, \mathbf{t}] \cdot \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \mathbf{K} \cdot [\mathbf{R}, \mathbf{t}] \cdot \tilde{\mathbf{x}}. \quad (14.69)$$

Wenn wir nun uneigentliche Punkte abbilden, ergibt sich:

$$\tilde{\mathbf{u}} = \mathbf{K} \cdot [\mathbf{R}, \mathbf{t}] \cdot \begin{pmatrix} \mathbf{x}_\infty \\ 0 \end{pmatrix} = \mathbf{K}\mathbf{R}\mathbf{x}_\infty. \quad (14.70)$$

Wir sehen, egal welche Translation \mathbf{t} gewählt wird, der 3D-Punkt wird immer in den gleichen Bildpunkt abgebildet, ist invariant gegenüber Translationen. Dies können wir in der Praxis beobachten für „weit entfernte“ Punkte. Fährt man auf der Autobahn immer gerade aus, dann scheint die Sonne regelrecht an einem Punkt „fest zu kleben“. Für die Rotation gelten diese Eigenschaften natürlich nicht, die Rotationsmatrix geht noch in die Abbildung ein. Für diese uneigentlichen Punkte hängt also die Abbildung nur noch ab von den inneren Parametern und der Rotation. Gibt es nun in der Menge aller uneigentlichen Punkte ein Objekt aus uneigentlichen Punkten, wo die Abbildung des Objektes auch nicht mehr von der Rotation abhängt, sondern nur noch von den inneren Parametern, also invariant von

den äußereren Parametern ist. Ein reales Objekt kann das sicher nicht sein, aber vielleicht ein mathematisch abstraktes Objekt. Wenn wir eine Bewegung haben, d. h. Rotation und Translation, dann transformiert sich der \mathbf{x}_∞ -Anteil der uneigentlichen Punkte durch die Bewegung wie oben ersichtlich zu $\mathbf{x}'_\infty = \mathbf{R}\mathbf{x}_\infty$. Wir betrachten nun alle Punkte, die die Gleichung $\mathbf{x}'_\infty^T \mathbf{x}_\infty = 0$ erfüllen, dann folgt:

$$\mathbf{x}'_\infty^T \mathbf{x}'_\infty = \mathbf{x}_\infty^T \mathbf{R}^T \mathbf{R} \mathbf{x}_\infty = \mathbf{x}_\infty^T \mathbf{x}_\infty = 0. \quad (14.71)$$

Alle uneigentlichen Punkte, deren \mathbf{x}_∞ -Anteil die Gleichung $\mathbf{x}_\infty^T \mathbf{x}_\infty = 0$ erfüllen, bleiben bei einer Rotation auf dieser Kurve. Diese Kurve heißt *absolute conic* (AC) und besitzt keine reellen Lösungen, sondern nur komplexe. Wir lösen in der Abbildungsgleichung auf:

$$\tilde{\mathbf{u}} = \mathbf{K}\mathbf{R}\mathbf{x}_\infty \rightarrow \mathbf{x}_\infty = (\mathbf{K}\mathbf{R})^{-1}\tilde{\mathbf{u}} \quad (14.72)$$

und setzen in $\mathbf{x}_\infty^T \mathbf{x}_\infty = 0$ ein:

$$\mathbf{x}_\infty^T \mathbf{x}_\infty = \tilde{\mathbf{u}}^T (\mathbf{K}\mathbf{R})^{-1} (\mathbf{K}\mathbf{R})^{-1} \tilde{\mathbf{u}} = \tilde{\mathbf{u}}^T \mathbf{K}^{-1} \mathbf{R}^{-1} \mathbf{R} \mathbf{K}^{-1} \tilde{\mathbf{u}} = \tilde{\mathbf{u}}^T \mathbf{K}^{-1} \mathbf{K}^{-1} \tilde{\mathbf{u}} = 0. \quad (14.73)$$

Wir sehen, auch das Abbild des AC, beschrieben durch $\tilde{\mathbf{u}}^T \mathbf{K}^{-1} \mathbf{K}^{-1} \tilde{\mathbf{u}} = 0$ ist eine „komplexe“ quadratische Form, genannt *image of the absolute conic* (IAC), die nur noch von den inneren Parametern abhängt. Die Matrix $\omega^{-1} = \mathbf{K}\mathbf{K}^T$ heißt dabei *Kruppa-Matrix*.

Wir betrachten jetzt weiterhin eine spezielle Rotation, wobei sich die Kamera ausschließlich um das Projektionszentrum dreht. Diesen Fall gilt es jetzt zu untersuchen. Da das Projektionszentrum fest bleibt, müssen die 2D-Bilder sich ausschließlich durch eine spezielle 2D-projektive Transformation, eine Homographie, unterscheiden. Eine Rotation um das Projektionszentrum können wir nur beschreiben, wenn wir vorher den Koordinatenursprung in das Projektionszentrum verschieben, also $\mathbf{z} = \mathbf{x} - \mathbf{x}_c$. Dann lautet die Rotation $\mathbf{z}' = \mathbf{R} \cdot \mathbf{z}$. Nun schreiben wir die Abbildungsgleichungen gleich in den neuen Koordinaten auf:

$$\tilde{\mathbf{u}} = \mathbf{A}\mathbf{x} + \mathbf{a} = \mathbf{A}\mathbf{z} \Leftrightarrow \mathbf{z} = \mathbf{A}^{-1}\tilde{\mathbf{u}} \quad (14.74)$$

und damit ist:

$$\tilde{\mathbf{u}}' = \mathbf{A}\mathbf{x}' + \mathbf{a} = \mathbf{A}\mathbf{z}'. \quad (14.75)$$

Nun setzen wir ein und erhalten:

$$\tilde{\mathbf{u}}' = \mathbf{A}\mathbf{R}\mathbf{z} = \mathbf{A}\mathbf{R}\mathbf{A}^{-1}\tilde{\mathbf{u}} = \mathbf{D}\tilde{\mathbf{u}}. \quad (14.76)$$

Die Transformation $\tilde{\mathbf{u}}' = \mathbf{D}\tilde{\mathbf{u}}$ beschreibt eine spezielle (2D \leftrightarrow 2D) Homographie. Dies ist aber nicht mehr der Fall, wenn wir zusätzlich noch eine Translation zulassen würden. Bei

vorgegebener Kamera hängt folglich die Homographie nur noch von den drei Parametern der Rotationsmatrix \mathbf{R} ab. Diese Erkenntnis hat große Bedeutung für die Berechnung von Panoramabildern mit einer sich drehenden Kamera. Diese Einzelbilder sollen harmonisch zu einem Gesamtbild zusammengestzt werden. Zwischen den 2D-Bildern existiert aber nur dann eine $2D \leftrightarrow 2D$ projektive Transformation (Homographie), wenn sich die Kamera um das Projektionszentrum dreht. Das Projektionszentrum wird in diesem Zusammenhang oft als *Nodalpunkt* bezeichnet. Haben wir noch eine Translation enthalten, dann passen die Tiefen, d. h. Vorder- und Hintergrund einfach nicht mehr zusammen. Das Projektionszentrum einer üblichen Kamera liegt oft in der Blendenebene, muss aber empirisch bestimmt werden oder bestimmte Zusatzeinrichtungen von Stativen gewähren dann die gewünschte Rotation. Sind die aufzunehmenden Objekte weit entfernt von der Kamera, dann ist eine kleine Verschiebung nicht so problematisch, die Zusammensetzungsfehler sind dann klein.

Für andere praktische Zwecke ist es durchaus auch sinnvoll ein Bild 2D-projektiv so zu transformieren, dass eine beliebiger Punkt $\tilde{\mathbf{u}}$ in den Hauptpunkt $\tilde{\mathbf{u}}_0$ transformiert wird. Dazu müssen wir die Rotationsmatrix bestimmen. Anschaulich heißt dies aber, wir rotieren den Sehstrahl bezüglich des Punktes $\tilde{\mathbf{u}}$ in die optische Achse. Damit können wir aber Drehachse und Drehwinkel berechnen. Die Drehachse steht senkrecht auf der optischen Achse und dem Sehstrahl, der Drehwinkel ist der Winkel zwischen Sehstrahl und optischer Achse, folglich:

$$\mathbf{s} = \frac{\mathbf{t}_s \times \mathbf{a}_3}{|\mathbf{t}_s \times \mathbf{a}_3|}, \quad \cos \alpha = \frac{\langle \mathbf{t}_s, \mathbf{a}_3 \rangle}{|\mathbf{t}_s| \cdot |\mathbf{a}_3|}. \quad (14.77)$$

Aus Drehachse und Drehwinkel können wir nun (direkt oder mit den Quaternionen) die Rotationsmatrix \mathbf{R} ausrechnen und damit die Homographie. Diese Art der Transformation bezüglich des Hauptpunktes nennt man *Hauptpunkttransformation* nach Kanatani. Diese wird richtig interessant, wenn man „senkrecht“ auf eine Ebene schauen will, dann ist die Abbildung bezüglich dieser Ebene nur eine Ähnlichkeitstransformation. Dazu muss man aber „irgendwie“ den Normalenvektor der uns interessierenden Ebene bestimmen. Der Winkel zwischen Normalenvektor und optischer Achse ist dann der Drehwinkel.

14.3 Invarianten

14.3.1 Punktinvarianten

Als erstes betrachten wir sogenannte Punktinvarianten, d. h. Invarianten die ausschließlich aus Punktanordnungen berechnet werden. Zunächst betrachten wir affine Invarianten in der Ebene. Dazu nehmen wir vier Punkte $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}$, die durch eine affine Transformation in die Bildpunkte $\mathbf{x}'_0, \mathbf{x}'_1, \mathbf{x}'_2, \mathbf{x}'$ überführt wurden. Die drei Punkte $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ spannen ein Koordinatensystem auf, mit \mathbf{x}_0 als Ursprung und $\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0$ als die beiden Basisvektoren.

Folglich können wir \mathbf{x} in diesem Koordinatensystem darstellen:

$$\mathbf{x} = \mathbf{x}_0 + \lambda_1(\mathbf{x}_1 - \mathbf{x}_0) + \lambda_2(\mathbf{x}_2 - \mathbf{x}_0). \quad (14.78)$$

Nun transformieren wir den Vektor der linken Seite und den Vektor der rechten Seite affin und erhalten:

$$\begin{aligned} \mathbf{Ax} + \mathbf{a} &= \mathbf{A}[\mathbf{x}_0 + \lambda_1(\mathbf{x}_1 - \mathbf{x}_0) + \lambda_2(\mathbf{x}_2 - \mathbf{x}_0)] + \mathbf{a} \\ &= \mathbf{Ax}_0 + \mathbf{a} + \lambda_1(\mathbf{Ax}_1 + \mathbf{a} - \mathbf{Ax}_0 - \mathbf{a}) + \lambda_2(\mathbf{Ax}_2 + \mathbf{a} - \mathbf{Ax}_0 - \mathbf{a}). \end{aligned} \quad (14.79)$$

Daraus folgt sofort:

$$\mathbf{x}' = \mathbf{x}'_0 + \lambda_1(\mathbf{x}'_1 - \mathbf{x}'_0) + \lambda_2(\mathbf{x}'_2 - \mathbf{x}'_0). \quad (14.80)$$

Damit sind die beiden Koordinaten λ_1 und λ_2 invariant gegenüber affinen Transformationen. Dies gilt ebenso für höherdimensionale Räume, also auch für den dreidimensionalen Raum. Man erkennt an diesem einfachen Beispiel eine allgemeine Merkregel zur Berechnung von Invarianten, die nicht nur auf Punkt invarianten beschränkt ist:

- Vier Punkte in der Ebene werden durch acht Parameter beschrieben, während die affine Transformation sechs Parameter besitzt. Folglich verfügt man über $8 - 6 = 2$ freie Parameter und kann demzufolge nur zwei Invarianten berechnen.
- Fünf Punkte im Raum werden durch 15 Parameter beschrieben, während die affine Transformation im Raum 12 Parameter besitzt. Folglich kann man $15 - 12 = 3$ drei Invarianten berechnen.
- Allgemein: Anzahl der verwendeten Merkmale *minus* Anzahl der Parameter der verwendeten Transformation ist *gleich* Anzahl der zu berechnenden Invarianten.
- Ein weiteres Beispiel sei die Berechnung der Invarianten von Ellipsen bezüglich affiner Transformationen. Eine Ellipse besitzt 5 Parameter und die affine Transformation 6, folglich ist $5 - 6 = -1$. Folglich existieren für Ellipsen keine affinen Invarianten.

Kommen wir noch einmal auf die Basisdarstellung (14.78) zurück. Diese kann man auch in der Form

$$\mathbf{x} = (1 - \lambda_1 - \lambda_2)\mathbf{x}_0 + \lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 = \mu_1\mathbf{x}_0 + \mu_2\mathbf{x}_1 + \mu_3\mathbf{x}_2, \quad \mu_1 + \mu_2 + \mu_3 = 1 \quad (14.81)$$

schreiben. Dabei nennt man die Punkte $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ eine *affine* Basis der Ebene und μ_1, μ_2, μ_3 mit $\mu_1, \mu_2, \mu_3 = 1$ die *affinen* Koordinaten des Punktes \mathbf{x} . Diese affine Darstellung schreiben wir noch etwas anders:

$$\mu_1\mathbf{x}_0 + \mu_2\mathbf{x}_1 + \mu_3\mathbf{x}_2 - \mathbf{x} = \mathbf{0}, \quad \mu_1 + \mu_2 + \mu_3 - 1 = 0. \quad (14.82)$$

Beide Gleichungen können wir mit einer beliebigen Konstanten multiplizieren, daher schreiben wir:

$$\xi_1 \mathbf{x}_0 + \xi_2 \mathbf{x}_2 + \xi_3 \mathbf{x}_2 + \xi_4 \mathbf{x} = \mathbf{0}, \quad \xi_1 + \xi_2 + \xi_3 + \xi_4 = 0. \quad (14.83)$$

In Matrixschreibweise wollen wir diese beiden Gleichungen zusammenfassen. Mit

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x} \end{pmatrix}, \quad \xi^T = (\xi_1, \xi_2, \xi_3, \xi_4) \quad (14.84)$$

ergibt sich in kompakter Schreibweise $\mathbf{X} \cdot \xi = \mathbf{0}$. Folglich spannen alle Lösungsvektoren ξ den Nullraum $N(\mathbf{X})$ auf. Statt der affinen Koordinaten kann man nun auch als duale Betrachtungsweise den Nullraum benutzen. Transformieren wir nun die vier Punkte in der dualen Darstellung affin, d. h.

$$\xi_1(\mathbf{A}\mathbf{x}_0) + \xi_2(\mathbf{A}\mathbf{x}_1) + \xi_3(\mathbf{A}\mathbf{x}_2) + \xi_4(\mathbf{A}\mathbf{x}) = \mathbf{0}, \quad (14.85)$$

dann ergibt sich:

$$\xi_1 \mathbf{x}'_0 + \xi_2 \mathbf{x}'_1 + \xi_3 \mathbf{x}'_2 + \xi_4 \mathbf{x}' - \mathbf{a}(\xi_1 + \xi_2 + \xi_3 + \xi_4) = \mathbf{0}. \quad (14.86)$$

Daraus kann man nun ablesen: der Nullraum $N(\mathbf{X})$ ist selbst *affin invariant*.

Dies kann man nun auch für m Punkte $\mathbf{x}_0, \dots, \mathbf{x}_{m-1}$ im n -dimensionalen Raum verallgemeinern. Mit

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{m-1} \end{pmatrix} \quad (14.87)$$

ist der Nullraum wieder affin invariant. Er wird in der Literatur als *affine shape* der m Punkte $\mathbf{x}_0, \dots, \mathbf{x}_{m-1}$ bezeichnet. Wenn sich folglich zwei Punktmengen affin ineinander überführen lassen, dann müssen ihre *affine shape* identisch sein. Das Hauptproblem bei der praktischen Nutzung von affinen Koordinaten ist die Wahl der Basis, bezüglich der die anderen Punkte eine invariante Darstellung haben. Daher muss man verschiedene Basen „ausprobieren“. Die numerische Strategie zur effektiven Auswahl aller möglichen Basen wird als *geometric hashing* bezeichnet.

Die „nächsthöhere“ Stufe nach affinen Transformationen sind die projektiven Transformationen. Die projektiven Transformationen werden auch als *Kollineationen* bezeichnet, da stets Geraden wieder in Geraden übergehen. Eine Gerade ist ein 1D-dimensionaler projektiver Raum, ebenso wie ein Geradenbüschel. Zwei Geraden können demnach durch eine $1D \leftrightarrow 1D$ projektive Transformation ineinander überführt werden:

$$x' = \frac{a_{11}x + a_{10}}{a_{21}x + a_{20}}. \quad (14.88)$$

Dasselbe in homogenen Koordinaten:

$$\tilde{\mathbf{x}}' = \begin{pmatrix} a_{11} & a_{10} \\ a_{21} & a_{20} \end{pmatrix} \tilde{\mathbf{x}}. \quad (14.89)$$

Wir wählen zwei Punkte x_1 und x_2 aus, deren Transformierte x'_1 und x'_2 sind. Wir bilden die Differenz:

$$\begin{aligned} x'_1 - x'_2 &= \frac{a_{11}x_1 + a_{10}}{a_{21}x_1 + a_{20}} - \frac{a_{11}x_2 + a_{10}}{a_{21}x_2 + a_{20}} \\ &= \frac{(a_{11}x_1 + a_{10})(a_{21}x_2 + a_{20}) - (a_{11}x_2 + a_{10})(a_{21}x_1 + a_{20})}{(a_{21}x_1 + a_{20})(a_{21}x_2 + a_{20})} \\ &= \frac{(a_{11}a_{20} + a_{10}a_{21})(x_1 - x_2)}{(a_{21}x_1 + a_{20})(a_{21}x_2 + a_{20})}. \end{aligned} \quad (14.90)$$

Ziel ist es, die Transformationsparameter zu eliminieren. Dazu wählen wir einen dritten Punkt x_3 und bilden

$$x'_1 - x'_3 = \frac{(a_{11}a_{20} + a_{10}a_{21})(x_1 - x_3)}{(a_{21}x_1 + a_{20})(a_{21}x_3 + a_{20})}. \quad (14.91)$$

Wir sehen, einen großen Teil der Parameter können wir eliminieren, wenn wir den Quotienten bilden:

$$\frac{x'_1 - x'_2}{x'_1 - x'_3} = \frac{x_1 - x_2}{x_1 - x_3} \cdot \frac{a_{21}x_3 + a_{20}}{a_{21}x_2 + a_{20}}. \quad (14.92)$$

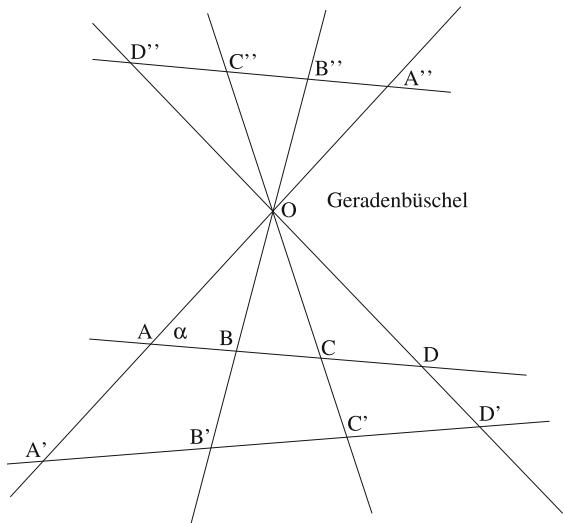
Bei einer affinen Transformation ist $a_{21} = 0$ und $a_{20} = 1$, sodass der Quotient $\frac{x_1 - x_2}{x_1 - x_3}$ nicht mehr von den Transformationsparametern abhängt. Er ist eine grundlegende affine Invariante, das sogenannte *Teilverhältnis*. Bei projektiven Transformationen müssen wir einen Schritt weitergehen. Dazu wählen wir einen 4. Punkt und ersetzen im Teilverhältnis x_1 durch x_4 . Wenn wir den Quotienten dieser beiden Teilverhältnisse bilden, fallen alle Transformationsparameter raus und wir erhalten die erste grundlegende projektive Invariante, das *Doppelverhältnis*. Bezeichnen wir nun die vier Punkte mit A, B, C, D , die alle auf einer Geraden liegen müssen, so ist die Schreibweise üblich:

$$[A, B, C, D] = \frac{\overline{AB}}{\overline{AC}} = \frac{\overline{AB}}{\overline{AC}} \cdot \frac{\overline{DC}}{\overline{DB}}. \quad (14.93)$$

Wir können auch an einem Büschel von Geraden das Doppelverhältnis deuten, siehe Abb. 14.6. Nun wenden wir den allgemein bekannten Sinussatz an:

$$\frac{\sin(AOB)}{\overline{AB}} = \frac{\sin(\alpha)}{\overline{OB}}, \quad \frac{\sin(AOC)}{\overline{AC}} = \frac{\sin(\alpha)}{\overline{OC}}. \quad (14.94)$$

Abb. 14.6 Geradenbüschel und Doppelverhältnis



Wir bilden wieder Quotienten:

$$\frac{\overline{AB}}{\overline{AC}} = \frac{\overline{OB}}{\overline{OC}} \cdot \frac{\sin(AOB)}{\sin(AOC)}. \quad (14.95)$$

Wir ersetzen A durch den vierten Punkt D und erhalten ebenso:

$$\frac{\overline{DB}}{\overline{DC}} = \frac{\overline{OB}}{\overline{OC}} \cdot \frac{\sin(DOB)}{\sin(DOC)}. \quad (14.96)$$

Jetzt bilden wir das Doppelverhältnis aus den vier Punkten A, B, C, D und erhalten:

$$[A, B, C, D] = \frac{\frac{\sin(AOB)}{\sin(AOC)}}{\frac{\sin(DOB)}{\sin(DOC)}}. \quad (14.97)$$

Diese Invariante (14.97) ist ein äquivalenter Ausdruck bezüglich der Winkel als Doppelverhältnis und spielt bei der Epipolartransformation eine Rolle, siehe Abschn. 14.4.2. Wir wissen bereits: Ein Geradenbüschel ist auch ein 1-dimensionaler projektiver Raum. Eine Gerade wird durch $\mathbf{a}^T \mathbf{x} = 0$ beschrieben. Vier Geraden, die durch einen gemeinsamen Schnittpunkt gehen, werden durch vier Vektoren $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$ beschrieben. Diese vier Vektoren sind gleichzeitig interpretierbar als vier Punkte in homogenen Koordinaten auf einer Geraden, also in einem 1-dimensionalen projektiven Raum. Rechnen wir nun diese homogenen Koordinaten der 4 Geraden (die Normalenvektoren) formal in kartesische Koordinaten um, dann können wir von diesen das Doppelverhältnis bilden. Dieses Doppelverhältnis ist dann das Doppelverhältnis der vier Geraden des Geradenbüschels.

14.3.2 Flächeninvarianten

Bei den bisherigen Punkt invarianten hatten wir vorausgesetzt, dass die vier Punkte auf einer Geraden liegen müssen, egal ob die Gerade in der Ebene oder im Raum liegt. Wenn dies nicht erfüllt ist, so nehmen wir im Folgenden an, dass alle Punkte wenigstens in einer Ebene liegen, deshalb werden diese auch *Flächeninvarianten* genannt. Eine bekannte Methode zu deren Herleitung ist die sogenannte *Determinantenmethode*. Drei Punkte liegen immer in einer Ebene, wir bilden die Fläche des durch die drei Punkte aufgespannten Dreiecks:

$$F_{123} = \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}. \quad (14.98)$$

Da wir die projektive Ebene gewählt haben, betrachten wir jetzt wie üblich eine $2D \leftrightarrow 2D$ projektive Transformation. Als Abkürzung sei benutzt:

$$\mathbf{a}_i^T = (a_{i1}, a_{i2}, a_{i0}), \quad \mathbf{x}_k^T = (x_k, y_k, 1). \quad (14.99)$$

Dann können wir die Transformation auch in der Form

$$x'_k = \frac{\mathbf{a}_1^T \mathbf{x}_k}{\mathbf{a}_3^T \mathbf{x}_k}, \quad y'_k = \frac{\mathbf{a}_2^T \mathbf{x}_k}{\mathbf{a}_3^T \mathbf{x}_k} \quad (14.100)$$

schreiben. Nun transformieren wir die Fläche:

$$F'_{123} = \frac{1}{2} \begin{vmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ x'_3 & y'_3 & 1 \end{vmatrix} = \frac{\begin{vmatrix} \mathbf{a}_1^T \mathbf{x}_1 & \mathbf{a}_2^T \mathbf{x}_1 & \mathbf{a}_3^T \mathbf{x}_1 \\ \mathbf{a}_1^T \mathbf{x}_2 & \mathbf{a}_2^T \mathbf{x}_2 & \mathbf{a}_3^T \mathbf{x}_2 \\ \mathbf{a}_1^T \mathbf{x}_3 & \mathbf{a}_2^T \mathbf{x}_3 & \mathbf{a}_3^T \mathbf{x}_3 \end{vmatrix}}{2(\mathbf{a}_3^T \mathbf{x}_1)(\mathbf{a}_3^T \mathbf{x}_2)(\mathbf{a}_3^T \mathbf{x}_3)} = \frac{\left| \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{pmatrix} (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \right|}{2(\mathbf{a}_3^T \mathbf{x}_1)(\mathbf{a}_3^T \mathbf{x}_2)(\mathbf{a}_3^T \mathbf{x}_3)}. \quad (14.101)$$

Mit den Abkürzungen $\beta_i = \mathbf{a}_3^T \mathbf{x}_i$ und $\delta = \det(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ ist dann

$$F'_{123} = \frac{\delta}{\beta_1 \beta_2 \beta_3} F_{123} \quad (14.102)$$

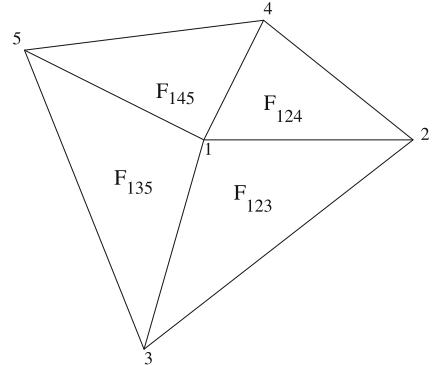
die Transformation der Fläche der drei Punkte. Wir versuchen wieder die Faktoren zu eliminieren, indem wir weitere Punkte hinzuziehen, also einen vierten Punkt und bilden:

$$F'_{124} = \frac{\delta}{\beta_1 \beta_2 \beta_4} F_{124}. \quad (14.103)$$

Wir bilden den Quotienten

$$\frac{F'_{123}}{F'_{124}} = \frac{\beta_4}{\beta_3} \frac{F_{123}}{F_{124}}. \quad (14.104)$$

Abb. 14.7 Fünf-Punkte-Invariante: Der Quotient der Produkte gegenüberliegender Flächen ist projektiv invariant



Bei affinen Transformationen ist $\beta_i = 1, \forall i$, sodass wir mit

$$I_1 = \frac{F_{123}}{F_{124}}, \quad I_2 = \frac{F_{134}}{F_{234}} \quad (14.105)$$

zwei affine vier-Punkte Invarianten erhalten haben. Vier Punkte bestimmen acht Parameter, eine affine Transformation ist durch sechs Parameter bestimmt, also haben wir $8 - 6 = 2$ freie Merkmale und dies sind die beiden Invarianten. Bezuglich projektiver Abbildungen benötigen wir einen weiteren Punkt

$$\frac{F'_{135}}{F'_{145}} = \frac{\beta_4}{\beta_3} \frac{F_{135}}{F_{145}}. \quad (14.106)$$

Durch Quotientenbildung können wir nun die restlichen Parameter eliminieren und erhalten zwei Flächeninvarianten:

$$I_1 = \frac{\frac{F_{123}}{F_{124}}}{\frac{F_{135}}{F_{145}}}, \quad I_2 = \frac{\frac{F_{123}}{F_{124}}}{\frac{F_{235}}{F_{245}}}. \quad (14.107)$$

Eine $2D \leftrightarrow 2D$ projektive Transformation besitzt acht Parameter, fünf Punkte haben zehn Merkmale, deshalb gibt es $10 - 8 = 2$ freie Parameter, also zwei Invarianten. Zur besseren geometrischen Deutung schreiben wir die erste Invariante nochmals in der Form

$$I_1 = \frac{F_{123} F_{145}}{F_{124} F_{135}} \quad (14.108)$$

auf. Wir wählen für die fünf Punkte eine spezielle Anordnung entsprechend Abb. 14.7. Der erste Punkt soll innerhalb des Viereckes liegen. Wir lesen aus (14.108) und der Abb. 14.7 dann ab, dass der Quotient des Produktes der Flächen gegenüberliegender Dreiecke projektiv invariant ist.

14.3.3 Invarianten von Punkten, Geraden, Kurven

Wir betrachten in homogenen Koordinaten eine projektive Transformation $\tilde{\mathbf{x}}' = \mathbf{A}\tilde{\mathbf{x}}$ sowie eine Gerade vor $\mathbf{g}^T\tilde{\mathbf{x}} = 0$ und die Gerade $\mathbf{g}'^T\tilde{\mathbf{x}}' = 0$ nach der projektiven Transformation. Wir setzen ein $\mathbf{g}'^T\tilde{\mathbf{x}}' = \mathbf{g}'^T\mathbf{A}\tilde{\mathbf{x}} = 0$ und erhalten damit $\gamma\mathbf{g} = \mathbf{A}^T\mathbf{g}'$ und damit $\mathbf{g}' = \gamma\mathbf{A}^{-1}\mathbf{g} = \gamma\mathbf{A}^{-1}\mathbf{g}$. Nun betrachten wir normierte homogene Koordinaten, d. h. die dritte Koordinate ist 1 bzw. die ersten beiden sind kartesische Koordinaten, also $\mathbf{x}' = (x', y', 1)^T$ bzw. $\mathbf{x} = (x, y, 1)^T$. Dann lautet die projektive Transformation $\mathbf{x}' = \frac{\mathbf{Ax}}{\mathbf{a}_3^T\mathbf{x}}$ (\mathbf{a}_3^T ist der dritte Zeilenvektor von \mathbf{A}). Wir betrachten nun einmal irgendeinen Punkt \mathbf{x} und dessen transformierten Punkt \mathbf{x}' , die nicht auf der Geraden bzw. der transformierten Geraden liegen müssen. Wir bilden das Produkt:

$$\mathbf{g}'^T\mathbf{x}' = \left(\gamma\mathbf{A}^{-1}\mathbf{g}\right)^T \frac{\mathbf{Ax}}{\mathbf{a}_3^T\mathbf{x}} = \gamma \frac{\mathbf{g}^T\mathbf{x}}{\mathbf{a}_3^T\mathbf{x}}. \quad (14.109)$$

Wir erkennen den Term $\mathbf{g}^T\mathbf{x}$, welcher vor und nach der Transformation vorkommt, aber auf Grund von γ und \mathbf{a}_3^T nicht invariant sein kann. Daher wählen wir zwei Geraden und zwei Punkte und bilden gemischte Produkte der Form:

$$\begin{aligned} \mathbf{g}_1'^T\mathbf{x}_1' &= \gamma_1 \frac{\mathbf{g}_1^T\mathbf{x}_1}{\mathbf{a}_3^T\mathbf{x}_1}, & \mathbf{g}_2'^T\mathbf{x}_2' &= \gamma_2 \frac{\mathbf{g}_2^T\mathbf{x}_2}{\mathbf{a}_3^T\mathbf{x}_2}, \\ \mathbf{g}_1'^T\mathbf{x}_2' &= \gamma_1 \frac{\mathbf{g}_1^T\mathbf{x}_2}{\mathbf{a}_3^T\mathbf{x}_2}, & \mathbf{g}_2'^T\mathbf{x}_1' &= \gamma_2 \frac{\mathbf{g}_2^T\mathbf{x}_1}{\mathbf{a}_3^T\mathbf{x}_1}. \end{aligned} \quad (14.110)$$

Wir können alle Faktoren eliminieren, wenn wir den folgenden Quotienten bilden:

$$I = \frac{\mathbf{g}_1'^T\mathbf{x}_1' \cdot \mathbf{g}_2'^T\mathbf{x}_2'}{\mathbf{g}_1'^T\mathbf{x}_2' \cdot \mathbf{g}_2'^T\mathbf{x}_1'} = \frac{\mathbf{g}_1^T\mathbf{x}_1 \cdot \mathbf{g}_2^T\mathbf{x}_2}{\mathbf{g}_1^T\mathbf{x}_2 \cdot \mathbf{g}_2^T\mathbf{x}_1}. \quad (14.111)$$

Folglich haben wir eine Invariante von zwei Geraden und zwei Punkten berechnet. Zwei Geraden und zwei Punkte besitzen acht Merkmale, die projektive Transformation besitzt ebenso acht Parameter, also kann es doch gar keine 2D-projektive Invarianten von zwei Geraden und zwei Punkten geben? Wir haben aber doch zweifelsfrei eine hergeleitet. Betrachten wir einmal in Abb. 14.8 die Konstellation. Mit den beiden Schnittpunkten haben wir vier Punkte auf einer Geraden und wir haben folglich weiter nichts als das Doppelverhältnis dieser vier Punkte berechnet, also eigentlich eine 1D-projektive Invariante.

Wir betrachten im Folgenden Kurven zweiter Ordnung in der Form:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0. \quad (14.112)$$

Wir führen die Matrix

$$\mathbf{E} = \begin{pmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{pmatrix} \quad (14.113)$$

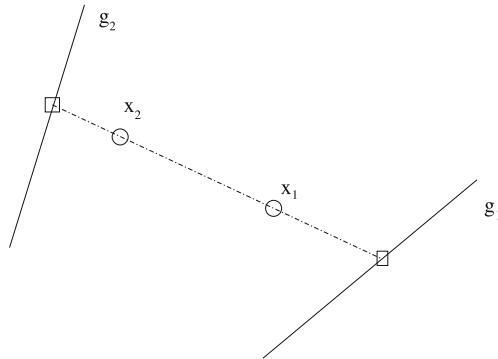


Abb. 14.8 Invariante: Zwei Geraden und zwei Punkte

ein und schreiben für (14.112) in Matrixnotation $\mathbf{x}^T \mathbf{E} \mathbf{x} = 0$ mit $\mathbf{x} = (x, y, 1)^T$. Eine Ellipse besitzt keine projektiven Invarianten, weil $5 - 8 = -3 < 0$ ist. Notiert man die projektive Transformation in der Matrixform $\mathbf{x}' = \frac{\mathbf{A}\mathbf{x}}{\mathbf{a}_3^T \mathbf{x}}$, dann ist:

$$\mathbf{x}'^T \mathbf{E}' \mathbf{x}' = \frac{\mathbf{x}^T \mathbf{A}^T \mathbf{E}' \mathbf{A} \mathbf{x}}{(\mathbf{a}_3^T \mathbf{x})^2} = 0 \quad \forall \mathbf{x}. \quad (14.114)$$

Das Gleiche mit allgemeinen homogenen Koordinaten $\tilde{\mathbf{x}}' = \mathbf{A}\tilde{\mathbf{x}}$ ergibt:

$$\tilde{\mathbf{x}}'^T \mathbf{E}' \tilde{\mathbf{x}}' = \tilde{\mathbf{x}}^T \mathbf{A}^T \mathbf{E}'^T \mathbf{A} \tilde{\mathbf{x}} = 0. \quad (14.115)$$

Folglich ist

$$\alpha \mathbf{E} = \mathbf{A}^T \mathbf{E}' \mathbf{A} \rightarrow \mathbf{E}' = \alpha (\mathbf{A}^T)^{-1} \mathbf{E} \mathbf{A}^{-1} \quad (14.116)$$

die projektive Transformation von \mathbf{E} . Wir multiplizieren nun \mathbf{E}' entsprechend (14.114) von links und rechts mit zwei Punkten $\mathbf{x}'_1, \mathbf{x}'_2$, die nicht auf der Ellipse liegen sollten und nutzen gleichzeitig die Transformation von \mathbf{E} entsprechend (14.116):

$$z_{1,2} = \mathbf{x}'_1^T \mathbf{E}' \mathbf{x}'_2 = \frac{\alpha \mathbf{x}'_1^T \mathbf{A}^T (\mathbf{A}^T)^{-1} \mathbf{E} \mathbf{A}^{-1} \mathbf{A} \mathbf{x}'_2}{(\mathbf{a}_3^T \mathbf{x}'_1)(\mathbf{a}_3^T \mathbf{x}'_2)} = \frac{\alpha \mathbf{x}'_1^T \mathbf{E} \mathbf{x}'_2}{(\mathbf{a}_3^T \mathbf{x}'_1)(\mathbf{a}_3^T \mathbf{x}'_2)}. \quad (14.117)$$

$z_{1,2}$ ist bis auf einen Faktor α schon eine affine Invariante, aber keine projektive. Daher bilden wir zusätzlich die Ausdrücke $z_{1,1} = \mathbf{x}'_1^T \mathbf{E}' \mathbf{x}'_1$ und $z_{2,2} = \mathbf{x}'_2^T \mathbf{E}' \mathbf{x}'_2$. Wir kombinieren diese drei Ausdrücke durch:

$$I = \frac{z_{1,2}^2}{z_{1,1} z_{2,2}} = \frac{(\mathbf{x}'_1^T \mathbf{E} \mathbf{x}'_2)^2}{(\mathbf{x}'_1^T \mathbf{E} \mathbf{x}'_1)(\mathbf{x}'_2^T \mathbf{E} \mathbf{x}'_2)}. \quad (14.118)$$

Eine Ellipse und zwei Punkte werden durch neun Merkmale beschrieben, daher ist $9 - 8 = 1$, d. h. wir erhalten tatsächlich eine „echte“ 2D-Invariante. Wir betrachten nun nochmals eine Ellipse und einen Punkt \mathbf{x}' , der nicht auf der Ellipse liegen braucht, dann ist

$$\mathbf{x}'^T \mathbf{E}' \mathbf{x}' = \alpha \frac{\mathbf{x}^T \mathbf{E} \mathbf{x}}{(\mathbf{a}_3^T \mathbf{x})^2} = d \quad (\text{Defekt}) \quad (14.119)$$

der Defekt oder die algebraische Distanz. Für affine Transformationen erhalten wir:

$$\mathbf{x}'^T \mathbf{E}' \mathbf{x}' = \alpha \mathbf{x}^T \mathbf{E} \mathbf{x} = d. \quad (14.120)$$

Damit ist diese algebraische Distanz d bis auf einen Faktor *affin invariant*. Von einer Ellipse allein können wir keine Invarianten bilden, wohl aber von einem Paar von Ellipsen, nämlich $10 - 8 = 2$ Invarianten. Die Ellipsenparameter transformieren sich entsprechend (14.116):

$$\mathbf{E}'_1 = \alpha_1 (\mathbf{A}^T)^{-1} \mathbf{E}_1 \mathbf{A}^{-1}, \quad \mathbf{E}'_2 = \alpha_2 (\mathbf{A}^T)^{-1} \mathbf{E}_2 \mathbf{A}^{-1}. \quad (14.121)$$

Wir bilden:

$$\mathbf{E}'_1 \cdot (\mathbf{E}'_2)^{-1} = \frac{\alpha_1}{\alpha_2} (\mathbf{A}^T)^{-1} \mathbf{E}_1 \mathbf{E}_2^{-1} \mathbf{A}^T. \quad (14.122)$$

Bis auf einen Faktor liegen jetzt sogenannte ähnliche Matrizen vor. Zwei Matrizen \mathbf{A}, \mathbf{B} heißen *ähnlich*, wenn es eine reguläre Matrix \mathbf{Q} gibt, sodass gilt:

$$\mathbf{A} = \mathbf{Q}^{-1} \mathbf{B} \mathbf{Q}. \quad (14.123)$$

Ähnliche Matrizen haben die gleichen charakteristischen Polynome, denn es gilt:

$$|\mathbf{A} - \lambda \mathbf{I}| = |\mathbf{Q}^{-1} \mathbf{B} \mathbf{Q} - \lambda \mathbf{I}| = |\mathbf{Q}| |\mathbf{Q}^{-1} \mathbf{B} \mathbf{Q} - \lambda \mathbf{I}| |\mathbf{Q}^{-1}| = |\mathbf{B} - \lambda \mathbf{I}|. \quad (14.124)$$

Für die (3,3)-Matrix \mathbf{E} der Ellipse ist z. B.:

$$|\mathbf{E} - \lambda \mathbf{I}| = c_0 - c_1 \lambda + c_2 \lambda^2 - \lambda^3 \quad (14.125)$$

mit

$$c_0 = \sum_i |\mathbf{E}_i^3| = |\mathbf{E}|, \quad (14.126)$$

$$c_1 = \sum_i |\mathbf{E}_i^2| = \det \begin{pmatrix} a & \frac{b}{2} \\ \frac{b}{2} & c \end{pmatrix} + \det \begin{pmatrix} c & \frac{e}{2} \\ \frac{e}{2} & f \end{pmatrix} + \det \begin{pmatrix} a & \frac{d}{2} \\ \frac{d}{2} & f \end{pmatrix}, \quad (14.127)$$

$$c_2 = \sum_i |\mathbf{E}_i^1| = a + c + f = \text{Spur}(\mathbf{E}). \quad (14.128)$$

Die Koeffizienten des charakteristischen Polynoms werden durch die *Hauptminoren* gebildet. Dies nutzen wir nun für unsere Aufgabe zu:

$$c_0(\mathbf{E}'_1 \mathbf{E}'^{-1}_2) = |\mathbf{E}'_1 \mathbf{E}'^{-1}_2| = \left(\frac{\alpha_1}{\alpha_2} \right)^3 |\mathbf{E}_1 \mathbf{E}_2^{-1}|, \quad (14.129)$$

$$c_1(\mathbf{E}'_1 \mathbf{E}'^{-1}_2) = \left(\frac{\alpha_1}{\alpha_2} \right)^2 c_1(\mathbf{E}_1 \mathbf{E}_2^{-1}), \quad (14.130)$$

$$c_2(\mathbf{E}'_1 \mathbf{E}'^{-1}_2) = \left(\frac{\alpha_1}{\alpha_2} \right) \cdot \text{Spur}(\mathbf{E}_1 \mathbf{E}_2^{-1}). \quad (14.131)$$

Damit haben wir „fast schon“ Invarianten, die unbekannten Faktoren stören, diese müssen wir beseitigen und bilden die beiden Ausdrücke:

$$I_1 = \frac{\text{Spur}(\mathbf{E}_1 \mathbf{E}_2^{-1})}{\sqrt[3]{|\mathbf{E}_1 \mathbf{E}_2^{-1}|}}, \quad I_2 = \frac{\text{Spur}(\mathbf{E}_2 \mathbf{E}_1^{-1})}{\sqrt[3]{|\mathbf{E}_2 \mathbf{E}_1^{-1}|}}. \quad (14.132)$$

Natürlich hätten wir auch andere Kombinationen aufschreiben können, diese sind aber alle abhängig von den beiden Invarianten (14.132), denn es kann nur zwei unabhängige Invarianten geben.

14.4 Epipolargeometrie

Wir betrachten mehrere Kameras, in der Regel zwei Kameras, die als Lochkameras modellierbar seien. Verzeichnungen werden hier nicht berücksichtigt. Zwei Kameras nennen wir ein Stereopaar. Dazu sind einige Grundbegriffe einzuführen und zu klären.

In der Abb. 14.9 ist eine typische Stereo-Anordnung zu sehen. Wenn wir die beiden Projektionszentren durch eine Gerade verbinden, durchstößt diese Gerade die Bilder in zwei Punkten, den sogenannten *Epipolen*. Die Verbindungslinie der beiden Projektionszentren wird Basislinie (*baseline* genannt). Wenn wir durch die drei Punkte (Raumpunkt, Projektionszentrum 1, Projektionszentrum 2) eine Ebene legen (die Blickebene), so schneidet diese die Bildebenen in den sogenannten *Epipolarlinien* (Blickebene → siehe Abschn. 14.1.12). Man sieht, dass korrespondierende Punkte im Bild immer auf den zugehörigen Epipolarlinien liegen müssen, siehe Abb. 14.10. Drehen wir die Blickebene um die Achse, die durch beide Projektionszentren geht, dann entstehen wieder zwei zugeordnete Epipolarlinien. So entstehen bei weiterem Drehen zwei Bündel von Epipolarlinien, die durch eine 1D projektive Transformation und die beiden Epipole ineinander überführbar sind, dies nennen wir *Epipolar-Transformation*. Im sogenannten Stereo-Normalfall, d. h. die optischen Achsen sind parallel (siehe Abschn. 14.1.9), sind die Epipolarlinien Parallelen zur u -Achse mit gleicher v -Koordinate. Daher sind in diesem Falle korrespondierende Punkte (Parallaxe) recht einfach zu suchen. Den Abstand zweier korrespondierender Punkte (u -Parallaxe) nennt man in diesem Falle *Disparität*.

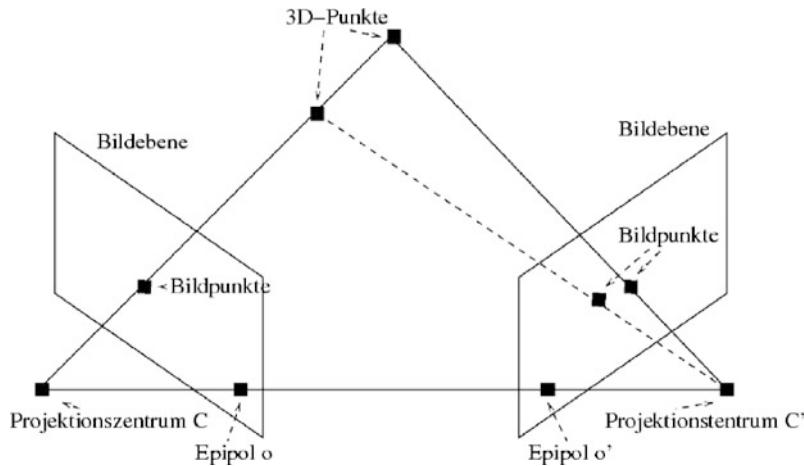


Abb. 14.9 Aufbau eines Stereosystems mit Bezeichnungen der Epipolargeometrie

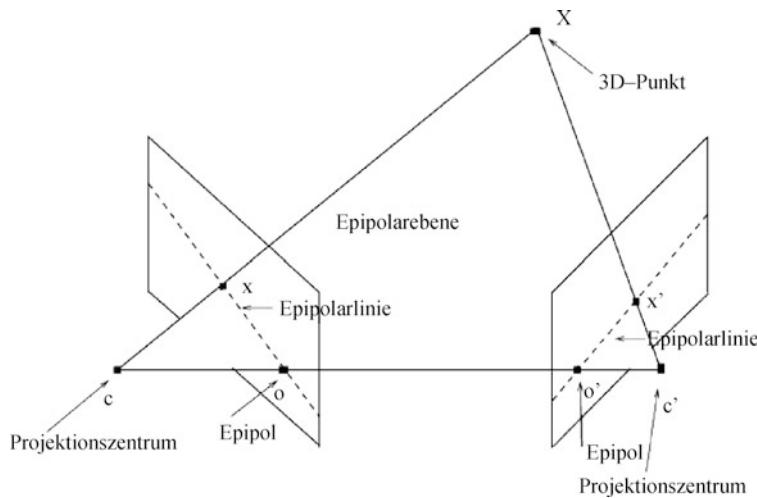


Abb. 14.10 Epipolarlinien

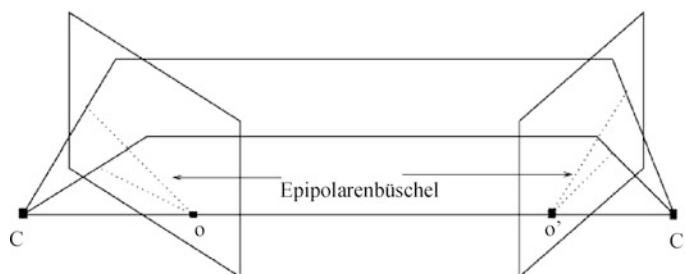


Abb. 14.11 Epipolaren-Büschele

Eine andere einfache geometrische Beziehung gibt es für den Bildhauptpunkt u_0, v_0 . Geraden im 3D-Raum die zueinander parallel sind, sind dies nicht mehr im Bild, schneiden sich aber alle in einem einzigen Punkt, dem Fluchtpunkt (*vanishing point*). Hat man jetzt im 3D-Raum 3 Mengen von je parallelen Geraden, so haben wir im Bild drei Fluchtpunkte. Sind diese drei Mengen von parallelen Geraden im 3D-Raum paarweise zueinander orthogonal, dann können wir mit Hilfe der drei Fluchtpunkte den Bildhauptpunkt bestimmen. Die drei Fluchtpunkte bilden ein Dreieck, dessen Orthozentrum (Schnittpunkt der drei Höhen) gerade den Bildhauptpunkt bildet. Diese Eigenschaft wird oft auch **Ortho-Theorem** genannt. Diese Eigenschaft kann man sich geometrisch relativ leicht überlegen. Wenn man einen orthogonalen 3D-Kalibrierkörper besitzt, so könnte man aus diesen Geraden den Bildhauptpunkt berechnen.

14.4.1 Zusammenhang über die Projektionsmatrizen

Nun soll der Zusammenhang zwischen zwei Bildern über die beiden kalibrierten Kameras untersucht werden. Es seien zwei Lochkameras gegeben:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}, \quad \tilde{\mathbf{y}}' = \tilde{\mathbf{A}}' \cdot \tilde{\mathbf{x}}. \quad (14.133)$$

Die beiden Projektionszentren berechnen sich zu:

$$\tilde{\mathbf{A}} \tilde{\mathbf{x}}_c = 0, \quad \tilde{\mathbf{A}}' \tilde{\mathbf{x}}'_c = 0. \quad (14.134)$$

Mit den Abkürzungen

$$\tilde{\mathbf{A}} = (\mathbf{A}_l, \mathbf{a}), \quad \tilde{\mathbf{A}}' = (\mathbf{A}'_l, \mathbf{a}'), \quad (14.135)$$

kann man die beiden Projektionszentren ausrechnen zu:

$$\tilde{\mathbf{x}}_c = \begin{pmatrix} \mathbf{A}_l^{-1} \mathbf{a} \\ -1 \end{pmatrix}, \quad \tilde{\mathbf{x}}'_c = \begin{pmatrix} \mathbf{A}'_l^{-1} \mathbf{a}' \\ -1 \end{pmatrix}. \quad (14.136)$$

Die Epipole können dann berechnet werden zu:

$$\tilde{\mathbf{o}}' = \tilde{\mathbf{A}}' \tilde{\mathbf{x}}_c, \quad \tilde{\mathbf{o}} = \tilde{\mathbf{A}} \tilde{\mathbf{x}}'_c, \quad (14.137)$$

$$\tilde{\mathbf{o}}' = \tilde{\mathbf{A}}' \begin{pmatrix} \mathbf{A}_l^{-1} \mathbf{a} \\ -1 \end{pmatrix} = \mathbf{A}'_l \mathbf{A}_l^{-1} \mathbf{a} - \mathbf{a}', \quad \tilde{\mathbf{o}} = \tilde{\mathbf{A}} \begin{pmatrix} \mathbf{A}'_l^{-1} \mathbf{a}' \\ -1 \end{pmatrix} = \mathbf{A}_l \mathbf{A}'_l^{-1} \mathbf{a}' - \mathbf{a}. \quad (14.138)$$

Es sei nun $\tilde{\mathbf{x}}_\infty$ der Schnittpunkt des Sehstrahles $\langle \tilde{\mathbf{y}}, \tilde{\mathbf{x}}_c \rangle$ mit der uneigentlichen Ebene, d. h. es gilt:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{A}} \tilde{\mathbf{x}}_\infty, \quad \tilde{\mathbf{y}} = \mathbf{A}_l \mathbf{x}_\infty, \quad \tilde{\mathbf{x}}_\infty = \begin{pmatrix} \mathbf{x}_\infty \\ 0 \end{pmatrix}, \quad \mathbf{x}_\infty = \mathbf{A}_l^{-1} \tilde{\mathbf{y}}. \quad (14.139)$$

Wir bilden nun diesen Schnittpunkt mit der zweiten Kamera ab:

$$\tilde{\mathbf{y}}'_{\infty} = \tilde{\mathbf{A}}' \tilde{\mathbf{x}}_{\infty} = \tilde{\mathbf{A}}' \begin{pmatrix} \mathbf{A}_l^{-1} \tilde{\mathbf{y}} \\ 0 \end{pmatrix} = \mathbf{A}'_l \mathbf{A}_l^{-1} \tilde{\mathbf{y}}. \quad (14.140)$$

Die dem Punkt $\tilde{\mathbf{y}}$ im zweiten Bild zugeordnete Epipolarlinie geht durch die Punkte $\tilde{\mathbf{o}}'$ und $\tilde{\mathbf{y}}'_{\infty}$. Jede Gerade kann durch das Kreuzprodukt zweier Punkte eindeutig beschrieben werden (falls diese in homogenen Koordinaten vorliegen), daher auch diese Epipolarlinie:

$$ep_{\tilde{\mathbf{y}}}' = \tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}'_{\infty}. \quad (14.141)$$

Der linearen Operation \times können wir eine Matrix \mathbf{O}' zuordnen, für die gilt:

$$\mathbf{O}' \cdot \mathbf{x} = \tilde{\mathbf{o}}' \times \mathbf{x} \quad \forall \mathbf{x}. \quad (14.142)$$

Es ist trivial, dass die antisymmetrische Matrix die Form

$$\mathbf{O}' = \begin{pmatrix} 0 & -o'_3 & +o'_2 \\ +o'_3 & 0 & -o'_1 \\ -o'_2 & +o'_1 & 0 \end{pmatrix} \quad (14.143)$$

hat. Daher gilt:

$$ep_{\tilde{\mathbf{y}}}' = \tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}'_{\infty} = \left(\mathbf{A}_l \mathbf{A}_l'^{-1} \mathbf{a}' - \mathbf{a} \right) \times \tilde{\mathbf{y}}'_{\infty} = \mathbf{O}' \tilde{\mathbf{y}}'_{\infty} = \mathbf{O}' \mathbf{A}'_l \mathbf{A}_l^{-1} \tilde{\mathbf{y}} = \mathbf{F} \tilde{\mathbf{y}}. \quad (14.144)$$

Die Matrix \mathbf{F} heißt *Fundamentalmatrix*. Für jeden Punkt $\tilde{\mathbf{y}}'_{ep}$ auf der Epipolarlinie $ep_{\tilde{\mathbf{y}}}'$ gilt dann (laut Definition einer Geraden mit dem Kreuzprodukt):

$$\tilde{\mathbf{y}}'^T \mathbf{F} \tilde{\mathbf{y}} = 0. \quad (14.145)$$

Folglich gilt dies für den Referenzpunkt $\tilde{\mathbf{y}}'$ von $\tilde{\mathbf{y}}$ erst recht. Da in die Fundamentalmatrix \mathbf{F} die beiden Referenzpunkte gar nicht eingehen, gilt diese Beziehung für alle Referenzpunktpaare $\tilde{\mathbf{y}} \leftrightarrow \tilde{\mathbf{y}}'$ und wir können aufschreiben:

$$\tilde{\mathbf{y}}'^T \mathbf{F} \tilde{\mathbf{y}} = 0 \quad \forall \text{Referenzpunktpaare } \tilde{\mathbf{y}}, \tilde{\mathbf{y}}'. \quad (14.146)$$

Dies ist die fundamentale Gleichung, da sie auch zur Berechnung der Fundamentalmatrix aus verfügbaren Referenzpunktpaaren benutzt wird. Zuvor sollen noch genauer Eigenschaften dieser Matrix untersucht werden. Das Kreuzprodukt $ep_{\tilde{\mathbf{y}}}^T = \mathbf{F} \tilde{\mathbf{y}}$ charakterisierte die Epipolarlinie. Wenn dieser Vektor zum Nullvektor entartet, dann ist diese Epipolarlinie unbestimmt, dies geht aber nur, wenn $\tilde{\mathbf{y}}$ zum Epipol $\tilde{\mathbf{o}}$ entartet, daher muss gelten:

$$\mathbf{F} \tilde{\mathbf{o}} = 0. \quad (14.147)$$

Dies kann als Bestimmungsgleichung zur Berechnung des Epipols $\tilde{\mathbf{o}}$ genutzt werden. Andererseits gilt auch:

$$(\tilde{\mathbf{y}}'^T \mathbf{F} \tilde{\mathbf{y}})^T = \tilde{\mathbf{y}}^T \mathbf{F}^T \tilde{\mathbf{y}}' = 0. \quad (14.148)$$

Daher gilt für den anderen Epipol:

$$\mathbf{F}^T \tilde{\mathbf{o}}' = 0. \quad (14.149)$$

Da die Epipole verschieden sind, kann also \mathbf{F} keine symmetrische Matrix sein. Da die beiden Gleichungssysteme zur Bestimmung der Epipole lineare homogene Gleichungssysteme sind und nichttriviale Lösungen besitzen müssen, muss für den Rang der Fundamentalmatrix gelten:

$$\text{Rang}(\mathbf{F}) \leq 2. \quad (14.150)$$

Es gilt sogar:

$$\text{Rang}(\mathbf{F}) = 2, \quad (14.151)$$

da $\text{Rang}(\mathbf{F}) = 1$ nur möglich ist, wenn die Epipole mehrdeutig sind. In unserer Herleitung haben wir als „Trick“ die uneigentliche Ebene benutzt. Damit können wir auch eine Homographie von einem Bild zum anderen beschreiben, wenn wir Punkte aus dieser uneigentlichen Ebene abbilden. Die Abbildungsgleichungen lauten:

$$\tilde{\mathbf{u}} = \mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{t}) = \mathbf{K}\mathbf{R}\mathbf{x} + \mathbf{K}\mathbf{t} = \mathbf{A}\mathbf{x} + \mathbf{a}. \quad (14.152)$$

Nun legen wir einmal das Weltkoordinatensystem in das Kamerakoordinatensystem der 1. Kamera, daher lauten die Abbildungsgleichungen $\tilde{\mathbf{u}} = \mathbf{K}\mathbf{x}$ und die der 2. Kamera $\tilde{\mathbf{u}}' = \mathbf{K}'\mathbf{R}\mathbf{x} + \mathbf{K}'\mathbf{t}$, wobei \mathbf{R}, \mathbf{t} die Bewegung der 2. Kamera gegenüber der 1. Kamera beschreibt. Bilden wir nun Punkte der uneigentlichen Ebene ab, so lauten die Gleichungen $\tilde{\mathbf{u}} = \mathbf{K}\mathbf{x}_\infty$, $\tilde{\mathbf{u}}' = \mathbf{K}'\mathbf{R}\mathbf{x}_\infty$. Daraus ergibt sich $\tilde{\mathbf{u}}' = \mathbf{K}'\mathbf{R}\mathbf{K}^{-1}\tilde{\mathbf{u}}$. Daraus können wir die Homographie-Matrix ablesen:

$$\mathbf{H}_\infty = \mathbf{K}'\mathbf{R}\mathbf{K}^{-1}. \quad (14.153)$$

14.4.2 Zusammenhang über die Epipolartransformation

Wir können den Zusammenhang zweier Bilder auch über die Epipolartransformation erklären. Wenn eine sogenannte Blickebene durch einen Raumpunkt und beide Projektionszentren geht, so schneidet diese Ebene beide Bildebenen in je einer Geraden, der Epipolarlinie. Wählen wir nun eine andere Blickebene, so entstehen wieder zwei Geraden usw.,

d. h. in jedem Bild entsteht ein Geradenbüschel, das sogenannte Epipolarenbüschel. Beide Epipolarenbüschel werden also über die Blickebenen ineinander überführt, was i. Allg. *Epipolartransformation* genannt wird. Nun soll der Zusammenhang der Fundamentalmatrix zur Epipolartransformation hergestellt werden. Für beide Epipolarenbüschel ist trivialerweise das Doppelverhältnis bezüglich die Büschel schneidender Geraden invariant, da eine $2D \leftrightarrow 2D$ projektive Transformation existiert, die beide Büschel ineinander überführt. Also können wir für das Doppelverhältnis die „Sinus“ der Winkel entsprechend (14.97) benutzen. Man kann nun leicht zeigen (mittels trigonometrischer Additionstheoreme anschließend durch „cos“ teilen, damit der „tangens“ erzeugt wird), dass

$$[A, B, C, D] = [\tau_A, \tau_B, \tau_C, \tau_D] \quad (14.154)$$

gilt. Dabei sind die τ die Anstiege ($\tan \alpha$) der Epipolarlinien. Damit ist (14.154) eine weitere Modifikation des Doppelverhältnisses als Invariante. Folglich muss eine $1D \leftrightarrow 1D$ projektive Transformation bezüglich der Anstiege korrespondierender Epipolarlinien existieren:

$$\tau' = \frac{a \cdot \tau + b}{c \cdot \tau + d}. \quad (14.155)$$

Wir können dies auch noch einfacher erklären. Zusätzlich zur Blickebene stellen wir uns eine Ebene vor, die senkrecht auf der Verbindung der beiden Projektionszentren steht und in der Mitte zwischen diesen beiden Projektionszentren plaziert wird. Die Blickebene schneidet diese Ebene ebenfalls in einer Geraden und es entsteht wieder ein Geradenbüschel. Das eine Epipolarenbüschel wird also mittels dieses „Hilfsbüschels“ in das zweite Epipolarenbüschel abgebildet. Damit haben wir aber eine $2D \leftrightarrow 2D$ projektive Transformation konstruiert, wobei jede Epipolarlinie in die entsprechende Epipolarlinie des anderen Bildes übergeht. Da ein Geradenbüschel einen $1D$ projektiven Raum bildet, muss zwischen den Geraden der Büschel eine $1D$ projektive Transformation existieren. Die Geraden der Büschel müssen durch $1D$ -Werte bez. der Epipole beschrieben werden. Man könnte einfach die Winkel zur u -Achse nehmen, die bilden aber keinen $1D$ -projektiven Raum (bei 2π hört der Raum auf). Nehmen wir z. B. die Anstiege, also $\tan \alpha$, dann haben wir eine projektiven Raum, wobei 90 Grad Anstieg ein uneigentlicher Punkt ist, der aber zum projektiven Raum dazugehört.

Daher kann die Epipolartransformation beschrieben werden durch die vier Koordinaten der beiden Epipole $\tilde{\mathbf{o}}, \tilde{\mathbf{o}'}$ und die drei Parameter der $1D \leftrightarrow 1D$ projektiven Transformation, also insgesamt durch sieben Parameter. Wir benutzen nun die beiden Epipole $\tilde{\mathbf{o}}, \tilde{\mathbf{o}'}$ und Referenzpunktpaare $\tilde{\mathbf{y}}, \tilde{\mathbf{y}'}$ und setzen diese in die $1D \leftrightarrow 1D$ projektive Transformation ein:

$$\frac{\frac{\tilde{\mathbf{y}}'_2 - \tilde{\mathbf{o}}'_2}{\tilde{\mathbf{y}}'_3 - \tilde{\mathbf{o}}'_3}}{\frac{\tilde{\mathbf{y}}'_1 - \tilde{\mathbf{o}}'_1}{\tilde{\mathbf{y}}'_3 - \tilde{\mathbf{o}}'_3}} = \frac{a \cdot \frac{\frac{\tilde{\mathbf{y}}_2 - \tilde{\mathbf{o}}_2}{\tilde{\mathbf{y}}_3 - \tilde{\mathbf{o}}_3}}{\frac{\tilde{\mathbf{y}}_1 - \tilde{\mathbf{o}}_1}{\tilde{\mathbf{y}}_3 - \tilde{\mathbf{o}}_3}} + b}{c \cdot \frac{\frac{\tilde{\mathbf{y}}_2 - \tilde{\mathbf{o}}_2}{\tilde{\mathbf{y}}_3 - \tilde{\mathbf{o}}_3}}{\frac{\tilde{\mathbf{y}}_1 - \tilde{\mathbf{o}}_1}{\tilde{\mathbf{y}}_3 - \tilde{\mathbf{o}}_3}} + d}. \quad (14.156)$$

Durch Ausmultiplizieren und Koeffizientenvergleich mit der Beziehung $\tilde{\mathbf{y}}'^T \mathbf{F} \tilde{\mathbf{y}} = 0$ ergeben sich für die Elemente der Fundamentalmatrix die Ausdrücke:

$$\begin{aligned}
 f_{11} &= -bo'_3 o_3 \\
 f_{12} &= -ao'_3 o_3 \\
 f_{13} &= ao'_3 o_2 + bo'_3 o_1 \\
 f_{21} &= do'_3 o_3 \\
 f_{22} &= co'_3 o_3 \\
 f_{23} &= -co'_3 o_2 - do'_3 o_1 \\
 f_{31} &= -do_3 o'_2 + bo_3 o'_1 \\
 f_{32} &= -co_3 o'_2 + ao_3 o'_1 \\
 f_{33} &= do'_2 o_1 + co_2 o'_2 - ao_2 o'_1 - bo'_1 o_1.
 \end{aligned} \tag{14.157}$$

Daher hängt die Fundamentalmatrix direkt von den sieben Parametern der Epipolartransformation ab und ist nur eine andere Beschreibungsform der Epipolartransformation. Wir können nun die Epipolattransformation ausschließlich aus Punktreferenzen berechnen. Dazu berechnen wir als erstes die Fundamentalmatrix aus den Punktreferenzen. Dann berechnen wir die beiden Epipole aus der Fundamentalmatrix durch:

$$\mathbf{F}\tilde{\mathbf{o}} = \mathbf{0}, \quad \mathbf{F}^T \tilde{\mathbf{o}}' = \mathbf{0}. \tag{14.158}$$

Und schließlich, wie wir aus der Beziehung (14.157) sehen, können wir auch die $1D \leftrightarrow 1D$ projektive Transformation aus der Fundamentalmatrix ablesen:

$$a = -f_{12}, \quad b = -f_{11}, \quad c = f_{22}, \quad d = f_{21}. \tag{14.159}$$

Dies ist möglich, da die Größen $f_{11}, f_{12}, f_{21}, f_{22}$ in (14.157) einen gemeinsamen Faktor besitzen.

14.4.3 Zusammenhang über die *Essential Matrix*

Ziel aller Ausführungen mit der *Essential Matrix* (*E*-Matrix) ist es, ausschließlich über Punktkorrespondenzen die Rotation und Translation zweier Kameras, bzw. bei einer Kamera die Rotation und Translation eines 3D-Objektes zu bestimmen. Wir betrachten das 3D-Kamerakoordinatensystem der ersten Kamera, das dann rotiert und translatiert wird. Die Punkte seien zunächst ideal in diesem System gegeben (und nicht als 2D-Bildkoordinaten). Wir können uns auch vorstellen, das Weltkoordinatensystem ist mit dem Kamerakoordinatensystem der ersten Kamera identisch. Dann gilt wie üblich

$$\mathbf{x}_{K_2} = \mathbf{R}\mathbf{x}_{K_1} + \mathbf{t} \tag{14.160}$$

oder einfach als

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t} \quad (14.161)$$

geschrieben. Dies beschreibt die Bewegung der Kamera bzw. was dasselbe ist, die Bewegung des Objektes, wenn nur eine Kamera gegeben ist. Wenn Punktekorrespondenzen gegeben sind, dann können wir sofort daraus die Transformationsparameter bestimmen. Nun ist es aber fast ausgeschlossen, dass uns die 3D-Koordinaten bekannt sind, eine Rekonstruktion wollen wir ja gerade vermeiden. Daher leiten wir aus obiger Beziehung eine andere ab, die uns nützlicher erscheinen wird. Es ist anschaulich trivial, dass $\mathbf{x}', \mathbf{t}, \mathbf{x}' - \mathbf{t} = \mathbf{R}\mathbf{x}$ in einer Ebene liegen müssen, daher muss das Spatprodukt verschwinden:

$$\mathbf{x}'^T [\mathbf{t} \times \mathbf{R}\mathbf{x}] = 0. \quad (14.162)$$

Der binären Operation \times (Kreuzprodukt) können wir wie üblich eine antisymmetrische Matrix zuordnen, daher ordnen wir dem Vektor $\mathbf{t}^T = (t_1, t_2, t_3)$ die Matrix

$$\mathbf{T} = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix} \quad (14.163)$$

zu. Folglich können wir schreiben:

$$\mathbf{x}'^T [\mathbf{t} \times \mathbf{R}\mathbf{x}] = \mathbf{x}'^T \mathbf{T} \mathbf{R} \mathbf{x} = 0. \quad (14.164)$$

Die Matrix $\mathbf{E} = \mathbf{T} \cdot \mathbf{R}$ nennt man *Essential Matrix* oder kurz *E-Matrix*. Woher bekommen wir nun 3D Punktkorrespondenzen. Wir denken jetzt an die normalisierten Koordinaten einer idealen Kamera, siehe Abschn. 14.1.9. Dazu bestimmen wir die inneren Parameter der Kamera und rechnen die homogenen Pixelkoordinaten durch Multiplikation mit der inversen inneren Projektionsmatrix in die normalisierten Koordinaten um $\tilde{\mathbf{y}}_N = \mathbf{K}^{-1}\tilde{\mathbf{y}}$. Diese normalisierten homogenen Pixelkoordinaten (homogene Pixelkoordinaten sind dreidimensional) sind nun bis auf einen unbekannten Faktor tatsächlich identisch mit den 3D-Kamerakoordinaten. Die unbekannten Faktoren stören aber (14.164) nicht, so dass wir (14.164) sofort mit normalisierten homogenen Pixelkoordinaten $\tilde{\mathbf{y}}'_N, \tilde{\mathbf{y}}_N$ aufschreiben dürfen:

$$\tilde{\mathbf{y}}'^T_N \mathbf{T} \mathbf{R} \tilde{\mathbf{y}}_N = \tilde{\mathbf{y}}'^T_N \mathbf{E} \tilde{\mathbf{y}}_N = 0. \quad (14.165)$$

Diese Gleichung nennt man *Longuet-Higgins-Gleichung* (1981). Für beide Kameras gilt für die Normalisierung $\tilde{\mathbf{y}}_N = \mathbf{K}^{-1}\tilde{\mathbf{y}}, \tilde{\mathbf{y}}'_N = \mathbf{K}'^{-1}\tilde{\mathbf{y}}'$, diese setzen wir in (14.165) ein und erhalten die Beziehung für die Fundamentalmatrix

$$\tilde{\mathbf{y}}'^T (\mathbf{K}'^{-1})^T \mathbf{E} \mathbf{K}^{-1} \tilde{\mathbf{y}} = 0 \Rightarrow \mathbf{F} = (\mathbf{K}'^{-1})^T \mathbf{E} \mathbf{K}^{-1} = (\mathbf{K}'^{-1})^T \mathbf{T} \mathbf{R} \mathbf{K}^{-1}. \quad (14.166)$$

Wenn das zweite Bild so entsteht, indem die erste Kamera bewegt wird und die inneren Parameter gleich bleiben, dann erhalten wir:

$$\mathbf{F} = (\mathbf{K}^{-1})^T \mathbf{E} \mathbf{K}^{-1} = (\mathbf{K}^{-1})^T \mathbf{T} \mathbf{R} \mathbf{K}^{-1}. \quad (14.167)$$

In diesem Fall können wir die F-Matrix noch etwas anders schreiben. Dazu betrachten wir den Term $\mathbf{t}' = \mathbf{K}\mathbf{t}$, wobei \mathbf{t} wieder die Translation zwischen den beiden Kameras bedeutet. Mit diesem Term formulieren wir folgendes

Lemma 14.1 *Mit $\mathbf{t}' = \mathbf{K}\mathbf{t}$, wobei \mathbf{K} sogar eine beliebige reguläre Matrix sein darf und der Bedingung $\det(\mathbf{K}) = +1$ genügt, gilt:*

$$\mathbf{T} = \mathbf{K}^T \mathbf{T}' \mathbf{K} \quad \text{bzw.} \quad \mathbf{T}' = \mathbf{K}^{T^{-1}} \mathbf{T} \mathbf{K}^{-1}. \quad (14.168)$$

(\mathbf{T}, \mathbf{T}' sind die zugeordneten, schiefsymmetrischen Matrizen der Translationsvektoren \mathbf{t} und \mathbf{t}' .)

Beweis Der Einfachheit halber führen wir den Beweis in umgekehrter Reihenfolge, man braucht ihn nur umdrehen. Dazu bezeichnen wir

$$\mathbf{K} = \begin{pmatrix} \mathbf{k}_1^T \\ \mathbf{k}_2^T \\ \mathbf{k}_3^T \end{pmatrix}, \quad \mathbf{t}' = \mathbf{K}\mathbf{t} = \begin{pmatrix} \mathbf{k}_1^T \mathbf{t} \\ \mathbf{k}_2^T \mathbf{t} \\ \mathbf{k}_3^T \mathbf{t} \end{pmatrix}. \quad (14.169)$$

Aus $\mathbf{T} = \mathbf{K}^T \mathbf{T}' \mathbf{K}$ folgt $\mathbf{T}\mathbf{x} = \mathbf{K}^T \mathbf{T}' \mathbf{K}\mathbf{x}$ für alle \mathbf{x} . Daher auch $\mathbf{K}^{T^{-1}} \mathbf{T}\mathbf{x} = \mathbf{T}' \mathbf{K}\mathbf{x}$. Dies schreiben wir wieder mit dem Kreuzprodukt

$$\mathbf{K}^{T^{-1}} (\mathbf{t} \times \mathbf{x}) = (\mathbf{K}\mathbf{t}) \times (\mathbf{K}\mathbf{x}). \quad (14.170)$$

Wir stellen auch die Inverse auf der linken Seite mit den Kreuzprodukten dar:

$$\frac{1}{\det(\mathbf{K})} \begin{pmatrix} (\mathbf{k}_2 \times \mathbf{k}_3)^T \\ (\mathbf{k}_3 \times \mathbf{k}_1)^T \\ (\mathbf{k}_1 \times \mathbf{k}_2)^T \end{pmatrix} (\mathbf{t} \times \mathbf{x}) = (\mathbf{K}\mathbf{t}) \times (\mathbf{K}\mathbf{x}). \quad (14.171)$$

Da laut Voraussetzung die Determinante gleich eins ist, verschwindet dieser Faktor. Wir schreiben einmal die erste Zeile dieser drei Gleichungen auf:

$$(\mathbf{k}_2 \times \mathbf{k}_3)^T (\mathbf{t} \times \mathbf{x}) = (\mathbf{k}_2^T \mathbf{t})(\mathbf{k}_3^T \mathbf{x}) - (\mathbf{k}_3^T \mathbf{t})(\mathbf{k}_2^T \mathbf{x}). \quad (14.172)$$

Auf der linken Seite haben wir das Skalarprodukt zweier Vektorprodukte, für solche Produkte gibt es die Lagrangesche Identität. Vergleicht man mit dieser die rechte Seite, so finden wir die vollständige Übereinstimmung. q.e.d.

Mit Beziehung (14.168) finden wir nun eine andere Darstellung der Fundamentalmatrix:

$$\mathbf{F} = (\mathbf{K}^{-1})^T \mathbf{E} \mathbf{K}^{-1} = (\mathbf{K}^{-1})^T \mathbf{T} \mathbf{K}^{-1} \mathbf{K} \mathbf{R} \mathbf{K}^{-1} = \mathbf{T}' \mathbf{R} \mathbf{K}^{-1}. \quad (14.173)$$

Die Beschränkung der Determinante mit $\det(\mathbf{K}) = +1$ ist keine echte Beschränkung, da die Fundamentalmatrix ohnehin nur bis auf einen Faktor angebar ist. \square

Eigenschaften der Essential Matrix \mathbf{E} Bevor wir wieder zur Fundamentalmatrix zurückkommen, einige interessante Eigenschaften der E-Matrix:

- Für die E-Matrix muss, da $\text{Rang}(\mathbf{F}) = 2$ ist, ebenfalls $\text{Rang}(\mathbf{E}) = 2$ gelten.
- Entsprechend der Herleitung lässt sie sich zerlegen in ein Produkt einer antisymmetrischen Matrix \mathbf{T} und einer Rotationsmatrix \mathbf{R} , folglich $\mathbf{E} = \mathbf{T} \cdot \mathbf{R}$. Trivialerweise ist auch $\det(\mathbf{T}) = 0$ ersichtlich. Da wir als Kamerakoordinaten nicht die absoluten Koordinaten, sondern nur die normalisierten Pixelkoordinaten zur Verfügung haben, also nur die Sehstrahlen, so ist natürlich die absolute Verschiebung nicht bestimmbar, sondern nur die Richtung der Verschiebung bis auf einen Faktor. Daher wird die E-Matrix nur durch fünf Parameter bestimmt, drei für die Rotationsmatrix \mathbf{R} und zwei für die Translation (Richtungsvektor, aber unbekannte Länge). Da die Beziehung (14.165) eine Gleichung für eine Punktreferenz liefert, benötigen wir minimal fünf Punktreferenzen, um die Matrix \mathbf{E} bestimmen zu können.
- Es gilt:

$$\mathbf{E}^T \mathbf{t} = 0. \quad (14.174)$$

Damit ist \mathbf{t} Eigenvektor von \mathbf{E}^T zum Eigenwert 0.

Beweis:

$$\mathbf{E}^T \mathbf{t} = (\mathbf{T} \mathbf{R})^T \mathbf{t} = -\mathbf{R}^T \mathbf{T} \mathbf{t} = -\mathbf{R}^T [\mathbf{t} \times \mathbf{t}] = \mathbf{0}. \quad (14.175)$$

Damit ist die Translationsrichtung aus (14.174) bestimbar.

- Die E-Matrix ist nur bis auf einen Faktor bestimbar, das ist darin begründet, dass man nicht die absolute Translation bestimmen kann, d. h. der Translationsvektor \mathbf{t} ist nur bis auf seine Länge bestimbar. Deshalb setzt man auch oft der Einfachheit halber $\|\mathbf{t}\| = 1$. Dann ist $\text{svd}(\mathbf{E}) = (1, 1, 0)^T$, d. h. die Singulärwerte sind 1, 1, 0 oder was auch dasselbe ist, die Eigenwerte von $\mathbf{E} \mathbf{E}^T$ und $\mathbf{E}^T \mathbf{E}$ sind 1, 1, 0. Zum Beweis siehe Faugeras [22].
- Unter der Voraussetzung $\|\mathbf{t}\| = 1$ gilt $\|\mathbf{e}\|^2 = \text{Spur}(\mathbf{E}^T \mathbf{E}) = 2$. Dazu fassen wir die Elemente der E-Matrix in einem Vektor zusammen:

$$\mathbf{e} = (e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}). \quad (14.176)$$

Nun sollen noch einige Bemerkungen zur Berechnung der E-Matrix gemacht werden, da die Probleme ähnlich derer bei der Berechnung der Fundamentalmatrix sind. Zunächst betrachtet man immer lineare Ansätze. Dazu bestimmen wir mittels Ausgleichsrechnung und unserer Grundgleichung die neun Koeffizienten der E-Matrix. Eigentlich sind es nur acht, da wir ein homogenes System zur Verfügung haben. Daher benötigen wir mindestens acht Punktereferenzen. Wir schreiben dies noch einmal auf. Die normalisierten homogenen Koordinaten schreiben wir der Einfachheit halber zu $\tilde{\mathbf{y}} = (x, y, 1)^T$. Sind nun n Punktereferenzen gegeben, so können wir unsere Grundgleichung (14.165) auch als $\mathbf{Be} = \mathbf{0}$ schreiben, wobei

$$\mathbf{B} = \begin{pmatrix} x_1 x'_1 & x_1 y'_1 & x_1 & y_1 x'_1 & y_1 y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots \\ x_n x'_n & x_n y'_n & x_n & y_n x'_n & y_n y'_n & y_n & x'_n & y'_n & 1 \end{pmatrix} \quad (14.177)$$

ist. Nun lösen wir

$$\min_{\mathbf{e}} \|\mathbf{Be}\|^2 \quad \text{bei } \|\mathbf{e}\|^2 = 2. \quad (14.178)$$

Die Lösung ist der Eigenvektor von $\mathbf{B}^T \mathbf{B}$ zum kleinsten Eigenwert. Wenn wir \mathbf{E} berechnet haben, dann bestimmen wir \mathbf{t} als Eigenvektor von \mathbf{E}^T zum Eigenwert 0, damit haben wir \mathbf{T} berechnet. Bezuglich \mathbf{R} lösen wir anschließend:

$$\min_{\mathbf{R}} \|\mathbf{TR} - \mathbf{E}\|^2 \quad \text{bei } \mathbf{R} \text{ ist eine Rotationsmatrix.} \quad (14.179)$$

Wir formulieren das Problem etwas um:

$$\mathbf{E} - \mathbf{TR} = (\mathbf{ER}^T - \mathbf{T})\mathbf{R}. \quad (14.180)$$

Für die Frobenius-Matrixnorm gilt dann

$$\|\mathbf{E} - \mathbf{TR}\|_F^2 = \|\mathbf{ER}^T - \mathbf{T}\|_F^2. \quad (14.181)$$

Es bezeichnen $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ die Spaltenvektoren von \mathbf{E}^T , dann können wir das äquivalente Problem

$$\min_{\mathbf{R}} \sum_{i=1}^3 \|\mathbf{Re}_i - \mathbf{t}_i\|^2 \rightarrow \text{Minimum} \quad \text{mit } \mathbf{R}^T \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = +1 \quad (14.182)$$

formulieren (\mathbf{t}_i sind die Spalten von \mathbf{T}^T bzw. die Zeilen von \mathbf{T}). Dies ist formal dasselbe Problem, wenn wir aus 3D-Referenzpunkten die Rotation mittels der kleinsten Quadratmethode bestimmen würden. Im Abschn. 13.2.11 haben wir dieses Problem mit Quaternionen gelöst, so dass wir nur noch ein simples Eigenwertproblem zu lösen haben. Die Zerlegung

in die Translation und die Rotation ist nicht eindeutig. Es gibt vier Lösungen. Wir kennen das Vorzeichen $+, -$ der Translation nicht und auch nicht das Vorzeichen der E -Matrix. Diese beiden Kombinationen ergeben vier mögliche Lösungen. Eine von diesen vier Lösungen ist durch Plausibilitätsbetrachtungen auszuwählen. Bei diesem Algorithmus haben wir überhaupt nicht zwei wesentliche Eigenschaften der E -matrix berücksichtigt: Der Rang ist zwei und zwei Singulärwerte sind gleich. Dies führt dazu, dass dieser Algorithmus sehr empfindlich gegenüber Rauschen der Punktkorrespondenzen ist und oft zu unbrauchbaren Resultaten führt. Deshalb wird oft direkt unsere Grundgleichung (14.165) in die Lösung einbezogen:

$$\sum_{i=1}^n \alpha_i (\tilde{\mathbf{y}}_i'^T \mathbf{T} \mathbf{R} \tilde{\mathbf{y}}_i)^2 \rightarrow \text{Minimum} \quad \text{bei } \mathbf{R} \text{ ist eine Rotationsmatrix.} \quad (14.183)$$

Dies ist aber eine mehrparametrische nichtlineare Optimierungsaufgabe, die es numerisch zu lösen gilt. Noch bessere Ergebnisse werden erreicht, wenn die Summanden gewichtet werden. Die Summanden selbst stellen Defekte dar, und zwar ob der korrespondierende Punkt auf der Epipolarlinie liegt oder nicht. Also sind diese Defekte als Abstände zur Epipolarlinie deutbar, diese „Abstände“ müssen wir aber noch normieren, z. B. mit dem Betrag des charakteristischen Vektors der Epipolarline. Daher wollen wir als Spezialfall (für die Fundamentalmatrix hatten wir dies schon getan) diese Vektoren nochmals darstellen. Wir betrachten wieder unsere Rotation $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$, diese schreiben wir jetzt in homogenen Koordinaten zu:

$$\tilde{\mathbf{x}}' = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tilde{\mathbf{x}}. \quad (14.184)$$

Wir betrachten wieder ideale Kameras, daher erhalten wir die beiden Abbildungsgleichungen:

$$\tilde{\mathbf{y}} = (\mathbf{I}, \mathbf{0}) \tilde{\mathbf{x}}; \quad \tilde{\mathbf{y}}' = (\mathbf{I}, \mathbf{0}) \tilde{\mathbf{x}}'. \quad (14.185)$$

Wir beziehen uns jetzt auf das Kamerakoordinatensystem der ersten Kamera, daher ergibt sich:

$$\tilde{\mathbf{y}}' = (\mathbf{I}, \mathbf{0}) \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tilde{\mathbf{x}} = (\mathbf{R}, \mathbf{t}) \tilde{\mathbf{x}}. \quad (14.186)$$

Nun berechnen wie wieder die Epipole:

$$\tilde{\mathbf{o}} = \mathbf{A}_l \mathbf{A}_l'^{-1} \mathbf{a}' - \mathbf{a} = \mathbf{R}\mathbf{t}. \quad (14.187)$$

$$\tilde{\mathbf{o}}' = \mathbf{A}_l' \mathbf{A}_l^{-1} \mathbf{a} - \mathbf{a}' = -\mathbf{t}. \quad (14.188)$$

Mit Hilfe der Epipole können wir die Epipolarlinien ausdrücken:

$$ep'_{\tilde{y}} = \tilde{o}' \times \tilde{y}'_\infty = \tilde{o}' \times (\mathbf{A}_l' \mathbf{A}_l^{-1} \tilde{y}) = -\mathbf{t} \times (\mathbf{R}\tilde{y}). \quad (14.189)$$

$$ep_{\tilde{y}'} = \tilde{o} \times \tilde{y}'_\infty = \tilde{o} \times (\mathbf{A}_l \mathbf{A}_l'^{-1} \tilde{y}') = \mathbf{R}^T \mathbf{t} \times \mathbf{R}^T \tilde{y}' = \mathbf{R}^T (\mathbf{t} \times \tilde{y}'). \quad (14.190)$$

Daher wählen wir als Gewichte:

$$\alpha_i = \frac{1}{\|\mathbf{R}^T(\mathbf{t} \times \tilde{y}'_i)\| + \|\mathbf{t} \times (\mathbf{R}\tilde{y}_i)\|}. \quad (14.191)$$

Auf Grund der Hesseschen Normalform der Geraden bezüglich des Abstandes eines Punktes von einer Geraden sind in der Norm nur die beiden ersten Komponenten der homogenen Koordinaten zu benutzen.

Es gibt auch direkte, nichtlineare Verfahren zur Berechnung der E-Matrix. Da die E-Matrix fünf Parameter besitzt, gibt es einen Fünf-Punkte-Algorithmus, siehe Nister [17]. Dies kann man noch abschwächen zu einem Sechs-Punkte- oder Sieben-Punkte-Algorithmus. Bei einem Acht-Punkte-Algorithmus sind wir wieder bei den obigen Betrachtungen angelangt.

14.4.4 Spezialfall: 3D-Punkte liegen in einer Ebene

Wir benutzen jetzt nur Referenzen von 3D-Punkten, die in einer Ebene liegen. Für diese Punkte wissen wir, dass eine $2D \leftrightarrow 2D$ projektive Transformation \mathbf{H} zwischen den Referenzpunkten existiert, also

$$\tilde{y}' = \mathbf{H} \cdot \tilde{y}. \quad (14.192)$$

Daraus folgt für unsere Grundgleichung (14.146)

$$\tilde{y}'^T \mathbf{F} \tilde{y} = \tilde{y}^T \mathbf{H}^T \mathbf{F} \tilde{y} = \tilde{y}^T \mathbf{F}^T \mathbf{H} \tilde{y} = 0. \quad (14.193)$$

Da diese Gleichung für beliebige \tilde{y} erfüllt ist, muss $\mathbf{H}^T \mathbf{F}$ antisymmetrisch sein, d. h. es gilt

$$\mathbf{H}^T \mathbf{F} + \mathbf{F}^T \mathbf{H} = 0. \quad (14.194)$$

Lemma 14.2 *Gleichung (14.194) ist genau dann erfüllt, wenn*

$$\mathbf{F} = \mathbf{O}' \cdot \mathbf{H} \quad (14.195)$$

gilt.

Beweis Folgende Identität gilt:

$$\mathbf{H}^T \cdot \mathbf{O}' \cdot \mathbf{H} + \mathbf{H}^T \cdot \mathbf{O}'^T \cdot \mathbf{H} = \mathbf{H}^T (\mathbf{O}'\mathbf{H} - \mathbf{O}'^T\mathbf{H}) = 0. \quad (14.196)$$

Es ist

$$\begin{aligned} 0 &= \tilde{\mathbf{y}}^T \mathbf{F}^T \mathbf{H} \tilde{\mathbf{y}} = (\mathbf{F}\tilde{\mathbf{y}})^T \mathbf{H} \tilde{\mathbf{y}} = (e p_{\tilde{\mathbf{y}}}^T)^T \mathbf{H} \tilde{\mathbf{y}} = (\tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}')^T \mathbf{H} \tilde{\mathbf{y}} \\ &= \tilde{\mathbf{y}}'^T \mathbf{O}'^T \mathbf{H} \tilde{\mathbf{y}} = -\tilde{\mathbf{y}}'^T \mathbf{O}' \mathbf{H} \tilde{\mathbf{y}} = \tilde{\mathbf{y}}'^T \mathbf{O}' \mathbf{H} \tilde{\mathbf{y}} = 0. \end{aligned} \quad (14.197)$$

Daraus folgt $\mathbf{F} = \mathbf{O}' \cdot \mathbf{H}$. \square

Natürlich können wir dieses Lemma auch mit $\mathbf{F} = \mathbf{H}^{-1} \cdot \mathbf{O}$ formulieren. Weiter betrachten wir

$$\mathbf{H}^T e p_{\tilde{\mathbf{y}}}^T = \mathbf{H}^T \mathbf{F} \tilde{\mathbf{y}} = -\mathbf{F}^T \mathbf{H} \tilde{\mathbf{y}} = -\mathbf{F}^T \tilde{\mathbf{y}}' = -e p_{\tilde{\mathbf{y}}}', \quad (14.198)$$

d. h. die Matrix \mathbf{H}^T überführt direkt die Epipolarlinien ineinander. Wir nehmen einmal an, die Matrix \mathbf{H} sei durch genau vier Punktereferenzen bestimmt worden (die vier 3D-Punkte müssen natürlich in einer Ebene liegen) und dies sei nun bekannt. Nun nehmen wir einen weiteren Punkt aus dieser Ebene, also einen fünften Punkt, für den dann trivialerweise $\tilde{\mathbf{y}}'_5 = \mathbf{H}\tilde{\mathbf{y}}_5$ gelten muss. Aus

$$\tilde{\mathbf{y}}_5^T (\mathbf{H}^T \mathbf{F} + \mathbf{F}^T \mathbf{H}) \tilde{\mathbf{y}}_5 = 0 \quad (14.199)$$

folgt dann

$$\tilde{\mathbf{y}}_5'^T \mathbf{F} \tilde{\mathbf{y}}_5 + \tilde{\mathbf{y}}_5^T \mathbf{F}^T \tilde{\mathbf{y}}_5' = 2\tilde{\mathbf{y}}_5'^T \mathbf{F} \tilde{\mathbf{y}}_5 = 0, \quad (14.200)$$

d. h. wenn \mathbf{H} aus vier Punkten einer Ebene bekannt ist, dann impliziert

$$\mathbf{H}^T \mathbf{F} + \mathbf{F}^T \mathbf{H} = 0, \quad (14.201)$$

dass für jeden weiteren Punkt dieser Ebene automatisch unsere Beziehung $\tilde{\mathbf{y}}'^T \mathbf{F} \tilde{\mathbf{y}} = 0$ erfüllt ist. Nun andersherum, nutzen wir Punktereferenzen und die Beziehung $\tilde{\mathbf{y}}'^T \mathbf{F} \tilde{\mathbf{y}} = 0$ (die 3D-Punkte liegen alle in einer Ebene) zur Bestimmung der Fundamentalmatrix, so sind nur vier Punktereferenzen wirksam und die anderen überflüssig. Aus $\mathbf{H}^T \mathbf{F} + \mathbf{F}^T \mathbf{H} = 0$ erhalten wir sechs Gleichungen (bei bekanntem \mathbf{H}) zur Bestimmung der Fundamentalmatrix, da diese aber sieben Parameter enthält, erhalten wir eine einparametrische Schar von Lösungen.

14.4.5 Berechnung der Fundamentalmatrix

In den Bezeichnungen lehnen wir uns an den Abschn. 14.4.3 an. Daher führen wir ähnlich einen Vektor \mathbf{f} ein, der alle Koeffizienten von \mathbf{F} enthält und wieder die Matrix \mathbf{B} , die

die entsprechenden Ausdrücke der n Punktkorrespondenzen enthält. Daher können wir wieder ein „lineares“ Kriterium mit mindestens acht Punktkorrespondenzen formulieren:

$$\min_{\mathbf{f}} \|\mathbf{B}\mathbf{f}\|^2 \quad \text{bei } \|\mathbf{f}\|^2 = 1. \quad (14.202)$$

Die Lösung ist wieder der Eigenvektor von $\mathbf{B}^T \mathbf{B}$ zum kleinsten Eigenwert. Das „lineare“ Kriterium hat wieder den entscheidenden Nachteil, dass die Bedingung $\text{Rang}(\mathbf{F}) = 2$ unberücksichtigt bleibt. Es sei nun $\tilde{\mathbf{y}} = (u, v, 1)^T$, $\tilde{\mathbf{o}} = (o_1, o_2, 1)^T$, dann können wir setzen:

$$u = o_1 - x, \quad v = o_2 - y, \quad (14.203)$$

wobei (x, y) der Abstandsvektor von $\tilde{\mathbf{y}}$ zu $\tilde{\mathbf{o}}$ sei. Folglich gilt:

$$ep'_{\tilde{\mathbf{y}}} = \mathbf{F}\tilde{\mathbf{y}} = \mathbf{F} \begin{pmatrix} o_1 - x \\ o_2 - y \\ 1 \end{pmatrix} = \mathbf{F}\tilde{\mathbf{o}} - \mathbf{F} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}. \quad (14.204)$$

Ist nun \mathbf{F} numerisch mit dem „linearen“ Kriterium berechnet worden und ist $\text{Rang}(\mathbf{F}) = 3$, so gilt $\mathbf{r} = \mathbf{F}\tilde{\mathbf{o}} \neq 0$. Weiter erinnern wir daran, dass ein Punkt $\mathbf{p} = (x_0, y_0, 1)^T$ zur Geraden \mathbf{l} mit $l_1 x + l_2 y + l_3 = 0$ den Abstand

$$d(\mathbf{p}, \mathbf{l}) = \frac{|l_1 x_0 + l_2 y_0 + l_3|}{\sqrt{l_1^2 + l_2^2}} \quad (14.205)$$

hat. Folglich bilden wir jetzt den Abstand von $ep'_{\tilde{\mathbf{y}}}$ zum Epipol $\tilde{\mathbf{o}}' = (o'_1, o'_2, 1)$:

$$d(\tilde{\mathbf{o}}', ep'_{\tilde{\mathbf{y}}}) = \frac{|r_1 o'_1 + r_2 o'_2 + r_3 - (f_{11} o'_1 + f_{21} o'_2 + f_{31})x - (f_{12} o'_1 + f_{22} o'_2 + f_{32})y|}{\sqrt{(r_1 - f_{11}x - f_{12}y)^2 + (r_2 - f_{21}x - f_{22}y)^2}}. \quad (14.206)$$

Nähern wir uns nun im ersten Bild dem Epipol, d. h. $(x, y) \rightarrow (0, 0)$, so folgt

$$d(\tilde{\mathbf{o}}', ep'_{\tilde{\mathbf{y}}}) = \frac{|r_1 o'_1 + r_2 o'_2 + r_3|}{\sqrt{r_1^2 + r_2^2}}. \quad (14.207)$$

Obwohl die r_i sehr klein sein können, kann dieser Abstand und damit der Fehler groß werden. Wählen wir also Originalpunkte „weit weg vom Epipol“, so werden die zugehörigen Epipolarlinien annähernd durch den Epipol im zweiten Bild gehen. Sind diese aber nahe am Epipol gewählt, so werden die Epipolarlinien im zweiten Bild im Widerspruch zur Epipolargeometrie stehen. Deshalb liefert das „lineare“ Kriterium numerisch instabile Lösungen und insbesondere Lösungen, die mit der Epipolargeometrie inkonsistent sind. Deshalb werden „nichtlineare“ Kriterien verwendet.

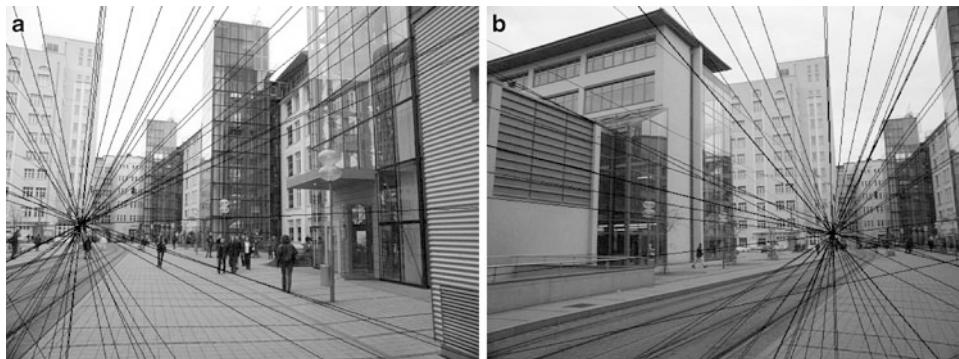


Abb. 14.12 **a** Erstes Epipolarenbüschel bezüglich des Stereopaars: Campus der Friedrich-Schiller-Universität Jena, **b** zweites Epipolarenbüschel bezüglich des Stereopaars: Campus der Friedrich-Schiller-Universität Jena

Eine weitere „lineare“ Methode besteht darin, aus N Ebenen im Bild jeweils die $2D \leftrightarrow 2D$ projektive Transformation H_i zu bestimmen und dann die Gleichungen

$$\begin{aligned} \mathbf{H}_1^T \mathbf{F} + \mathbf{F}^T \mathbf{H}_1 &= 0 \\ \mathbf{H}_2^T \mathbf{F} + \mathbf{F}^T \mathbf{H}_2 &= 0 \\ &\vdots \\ \mathbf{H}_N^T \mathbf{F} + \mathbf{F}^T \mathbf{H}_N &= 0 \end{aligned} \quad (14.208)$$

im Sinne der kleinsten Quadrate nach \mathbf{F} zu lösen. Hier benutzen wir eigentlich schon eine Zusatzinformation, wir müssen natürlich wissen, welche Referenzpunkte zu einer Ebene gehören.

Als „nichtlineares“ Kriterium werden oft die Abstände der Referenzpunkte zur zugehörigen Epipolarlinie minimiert, dies muss aber symmetrisch für beide Bilder erfolgen, also:

$$\sum_i d^2(\tilde{\mathbf{y}}'_i, \mathbf{F}\tilde{\mathbf{y}}_i) + d^2(\tilde{\mathbf{y}}_i, \mathbf{F}^T\tilde{\mathbf{y}}'_i) \rightarrow \text{Minimum.} \quad (14.209)$$

Dies können wir wieder ausführlich aufschreiben zu:

$$\sum_i \left(\frac{1}{(\mathbf{F}\tilde{\mathbf{y}}_i)_1^2 + (\mathbf{F}\tilde{\mathbf{y}}_i)_2^2} + \frac{1}{(\mathbf{F}^T\tilde{\mathbf{y}}'_i)_1^2 + (\mathbf{F}^T\tilde{\mathbf{y}}'_i)_2^2} \right) (\tilde{\mathbf{y}}'^T_i \mathbf{F} \tilde{\mathbf{y}}_i)^2 \rightarrow \text{Minimum.} \quad (14.210)$$

Ähnlich dem Fünf-Punkte Algorithmus zur Bestimmung der E-matrix gibt es auch einen Sieben-Punkte Algorithmus zur Bestimmung der F-matrix. In Abb. 14.12 sind korrespondierende Epipolarenbüschel zu sehen, wobei die Fundamentalmatrix mit der „linearen“ Methode berechnet wurde. Einige Ausreißer wurden durch Wahl von Bildpunkten in der Nähe der Epipole erzeugt.

Als Kamerakalibrierung bezeichnet man allgemein die Bestimmung der Abbildungsparameter einer Kamera, dabei ist das gewählte Modell völlig egal. Auch Verzeichnungsparameter müssen kalibriert werden, falls sie im gewählten Modell enthalten sind. Die klassische Methode ist die Kalibrierung einer Kamera mit einem 3D-Kalibrierkörper. Diese hat den Nachteil, dass man die Kalibrierkörper hoch genau herstellen muss und daher teuer sind. Weiterhin sind die Kalibrierkörper nicht flexibel einsetzbar. Daher wurden Methoden entwickelt, die auch ohne Kalibrierkörper funktionieren. Diese nutzen die Aufnahmen natürlicher Szenen und benötigen in der Regel mehrere Aufnahmen oder ungedreht mehrere Kameras. Aus mehreren Bildern werden dann Referenzpunkte bestimmt und zur Kalibrierung genutzt. Mit dieser Art Kalibrierung entstand der Begriff *Selbstkalibrierung*. Probleme der Kamerakalibrierung gibt es eigentlich nur noch bezüglich der Robustheit und Fehlerabschätzung. Moderne Forschungen beschäftigen sich außerdem mit dem Gebiet der „automatischen Kalibrierung in Multi-Kamerasystemen“. In den nächsten Abschnitten soll nur auf die elementaren Prinzipien der Kamerakalibrierung eingegangen werden.

15.1 Direkte Kalibrierung

Eine Kamera wird durch ihr Modell beschrieben. Wir nehmen einmal die Lochkamera, Modell (14.34) mit 11 unabhängigen Parametern. Dies entspricht formal der allgemein bekannten zentralprojektiven Abbildung in homogenen Koordinaten, also mit $\tilde{\mathbf{K}} = (\mathbf{K}, \mathbf{0})$ nach (14.33) und wieder $\tilde{\mathbf{u}} = \tilde{\mathbf{K}} \cdot \tilde{\mathbf{R}}\mathbf{T} \cdot \tilde{\mathbf{x}}_W = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_W$, folgt allgemein:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{10} \\ a_{21} & a_{22} & a_{23} & a_{20} \\ a_{31} & a_{32} & a_{33} & a_{30} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}. \quad (15.1)$$

Im Folgenden beziehen wir uns bei der Kalibrierung auf dieses Modell, d. h. ein zentralprojektives Modell ohne Verzeichnungen. Eine Kalibrierung mit diesem Modell nennt man in der Photogrammetrie direkte lineare Transformation (DLT). Da man einen der 12 Koeffizienten normieren kann, haben wir 11 unabhängige Parameter, dies entspricht einem Modell mit 6 äußeren und 5 inneren Kameraparametern. *Kamerakalibrierung* nennt man nun entsprechend dem verwendeten Modell die Bestimmung der Abbildungsparameter, hier folglich die Projektionsmatrix $\tilde{\mathbf{A}}$. Die 11 Parameter dieses Modells sind konkret für eine Kamera zu bestimmen. Dies geschieht im einfachsten Fall so, indem man sich 3D-Punkte mit bekannten Koordinaten in einem Weltkoordinatensystem vorgibt, die 2D-Koordinaten dieser Punkte im Bild ermittelt und aus diesen Beziehungen die Parameter ausrechnet. Setzen wir einen Punkt in die Abbildungsgleichung ein, so erhalten wir für die kartesischen Koordinaten zwei Gleichungen. Folglich brauchen wir mindestens 6 Punkte, um die 11 Unbekannten ausrechnen zu können. Aus der Projektionsmatrix können dann getrennt die inneren (*intrinsic*) und äußeren (*extrinsic*) Parameter berechnet werden. Dazu benutzt man in der Regel sogenannte *Kalibrierkörper*, die maßgerecht und genau gebaut werden müssen. Diese verkörpern das Weltkoordinatensystem mit entsprechend vielen markierten 3D-Punkten, deren Koordinaten im Kalibrierkörper bekannt sein müssen. Außerdem müssen die Punkte so markiert werden, dass diese im 2D-Bild leicht und genau gefunden werden können. Dabei ist zu beachten, dass die 3D-Punkte auch tatsächlich auf die detektierten 2D-Punkte abgebildet werden. Ein einfaches Gegenbeispiel sind Kreise, die in Ellipsen abgebildet werden. Die Kreismittelpunkte werden aber mit dem Modell (15.1) nicht in die Ellipsenmittelpunkte abgebildet. Die Abbildung dieser Mittelpunkte ist somit keine kovariante Abbildung.

Schreiben wir einmal die Abbildungsgleichungen (15.1) einer Lochkamera in der Form:

$$\tilde{\mathbf{u}} = \mathbf{Ax}_W + \mathbf{a}. \quad (15.2)$$

Aus $\tilde{\mathbf{u}} = \mathbf{Kx}_K, \mathbf{x}_K = \mathbf{Rx}_W + \mathbf{t}$ folgt $\mathbf{A} = \mathbf{KR}, \mathbf{a} = \mathbf{Kt}$. Damit folgt aber, dass $\mathbf{a}_3^T = \mathbf{r}_3^T$ ist (dies ist schon aus (14.35) bekannt), was wir als eine Kalibrierbedingung nutzen können. Nun müssen wir aber zunächst ein Ausgleichsproblem für die Kalibrierung formulieren. Wir geben n 3D-Punkte vor, deren 2D-Bildkoordinaten zu ermitteln sind. Wir schreiben zunächst die Abbildungsgleichungen mit kartesischen Koordinaten auf:

$$u_i = \frac{\mathbf{a}_1^T \mathbf{x}_i + \mathbf{a}_{10}}{\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}}, \quad v_i = \frac{\mathbf{a}_2^T \mathbf{x}_i + \mathbf{a}_{20}}{\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}}. \quad (15.3)$$

Wir schreiben diese jetzt in einer linearen Form auf:

$$u_i \cdot (\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}) - \mathbf{a}_1^T \mathbf{x}_i - \mathbf{a}_{10} = 0, \quad v_i \cdot (\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}) - \mathbf{a}_2^T \mathbf{x}_i - \mathbf{a}_{20} = 0. \quad (15.4)$$

Diese nutzen wir nun zur linearen Ausgleichsrechnung:

$$\sum_{i=1}^n (u_i * (\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}) - \mathbf{a}_1^T \mathbf{x}_i - \mathbf{a}_{10})^2 + (v_i * (\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}) - \mathbf{a}_2^T \mathbf{x}_i - \mathbf{a}_{20})^2 \rightarrow \text{Min.} \quad (15.5)$$

Der Preis der Linearisierung ist das nicht mehr ganz korrekte Maß, welches wir minimieren. Der Vorteil ist, dass wir ein lineares Ausgleichsproblem erhalten, d. h. wir können die Gaußschen Normalengleichungen aufstellen, diese lösen ohne eine Startlösung kennen zu müssen. Weiterhin sehen wir aber, dass es ein homogenes Ausgleichsproblem ist, d. h. die triviale Lösung können wir sofort als Lösung ablesen, ist aber praktisch sinnlos. Daher müssen wir wie üblich sinnvolle Restriktionen stellen. Eine mögliche wäre z. B. $a_{30} = 1$. Da diese Bedingung selbst linear ist, bleibt auch unser Ausgleichsproblem nach Einsetzen derselben linear und wir können es mit den Gaußschen Normalengleichungen lösen. Der Schwachpunkt bleibt aber: Ist es eine sinnvolle Restriktion, d. h. können wir aus praktischen Gründen immer garantieren, dass $a_{30} \neq 0$ gilt, oder fast Null ist? In den meisten Fällen funktioniert es, wir können es aber nicht garantieren, so dass durchaus numerische Instabilitäten auftreten können. Dazu betrachten wir einmal unsere Abbildungsgleichungen $\tilde{\mathbf{u}} = \mathbf{Ax} + \mathbf{a}$. Wir wissen, dass $\mathbf{0} = \mathbf{Ax} + \mathbf{a}$ die Bestimmungsgleichung des Projektionszentrums ist. Ist der Koordinatenursprung des Weltkoordinatensystems selbst im Projektionszentrum, dann ist $\mathbf{0} = \mathbf{A}\mathbf{0} + \mathbf{a}$ und damit $\mathbf{a} = \mathbf{0}$, womit $a_{30} = 0$ ist. Weiterhin wissen wir aus (14.35), dass $a_{30} = t_3$ ist, d. h. die Translation des Kamerakoordinatensystems gegenüber dem Weltkoordinatensystem in der z -Richtung. Die Kamera müsste also unbedingt anteilig in z -Richtung translatiert sein. Es gibt also Fälle, wo wir die Kalibrierbedingung $a_{30} = 1$ nicht einhalten können.

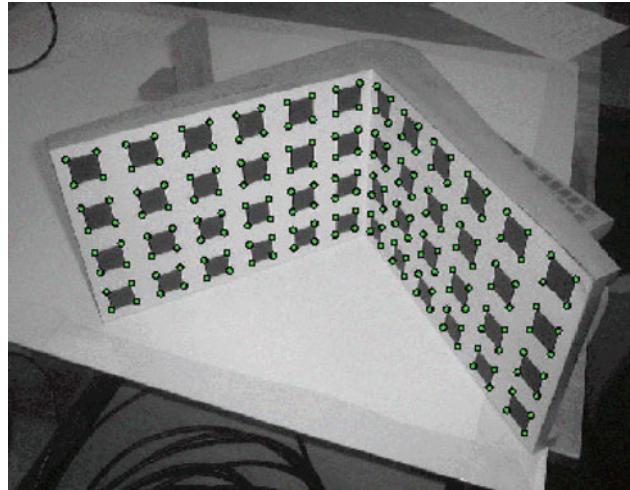
Sehen wir uns nach anderen Restriktionen um. Wir hatten eben gezeigt bzw. wissen aus (14.35), dass $\mathbf{a}_3^T = \mathbf{r}_3^T$ gilt und demnach $\|\mathbf{a}_3\| = 1$ als Kalibrierbedingung benutzt werden kann. Hier gibt es keinerlei Schwierigkeiten in der Anordnung der Kamera. Es tritt aber eine andere Schwierigkeit auf, nämlich dass wir die erste Nichtlinearität haben. Diese Art Bedingungen können wir aber mit Eigenwertproblemen lösen, und ist damit noch voll beherrschbar.

Bisher sind wir von einem Modell mit 11 Parametern ausgegangen. Was ist aber, wenn wir z. B. genau wissen, dass der Scherungswinkel des Sensorfeldes 90 Grad ist, d. h. wir eigentlich nur vier innere Parameter haben, wie kann dieses Wissen in die Kalibrierung einfließen? Dazu erinnern wir: der Vektor \mathbf{a}_3 steht senkrecht auf der u, v -Ebene (also der Bildebene). Die Abbildung $\mathbf{a}_1^T(\mathbf{x} - \mathbf{x}_c) = 0$ sagt aus, dass \mathbf{a}_1 senkrecht auf der v -Achse steht, entsprechend steht \mathbf{a}_2 auf der u -Achse, damit bildet $\mathbf{a}_2 \times \mathbf{a}_3$ die u -Achse und $\mathbf{a}_1 \times \mathbf{a}_3$ die v -Achse. Wenn folglich das Skalarprodukt aus beiden Null ist, dann ist der Scherungswinkel gleich 90 Grad. Daher verwenden wir für ein Modell mit 10 Parametern die zwei Restriktionen

$$\|\mathbf{a}_3\| = 1, (\mathbf{a}_1 \times \mathbf{a}_3)^T \cdot (\mathbf{a}_2 \times \mathbf{a}_3) = 0. \quad (15.6)$$

Wenn wir die zweite Bedingung entsprechend der Definition des Kreuzproduktes auflösen, sehen wir, dass eine nichtlineare, komplizierte Bedingung entstanden ist und wir damit ein nichtlineares Ausgleichsproblem zu lösen haben, welches nicht mehr durch ein Eigenwertproblem beschreibbar ist. Eine Kalibrierung mit diesen beiden Bedingungen (15.6) nennt man Kalibrierung nach *Faugeras*.

Abb. 15.1 Kalibrierkörper mit detektierten Ecken des Kalibriermusters



Wenn wir für ein Modell mit 11 Parametern ein nichtlineares Ausgleichsproblem mit einer Restriktion ansetzen, dann können wir auch unser Kalibrierproblem direkt nichtlinear formulieren und brauchen an das Modell mit 11 Parametern keine Restriktion zu konstruieren. Für das Modell mit 11 Parametern hätten wir folglich zu lösen:

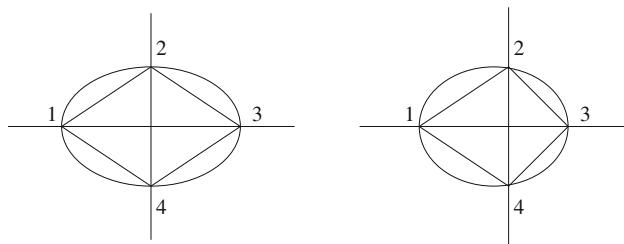
$$\sum_{i=1}^n \left(u_i - \frac{\mathbf{a}_1^T \mathbf{x}_i + \mathbf{a}_{10}}{\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}} \right)^2 + \left(v_i - \frac{\mathbf{a}_2^T \mathbf{x}_i + \mathbf{a}_{20}}{\mathbf{a}_3^T \mathbf{x}_i + \mathbf{a}_{30}} \right)^2 \rightarrow \text{Minimum.} \quad (15.7)$$

Mit einer geeigneten Startlösung (z. B. aus dem linearen Problem (15.5)) und dem *Levenberg-Marquardt-Verfahren* (siehe Abschn. 21.2.2) ist die Lösung stabil zu berechnen. Wollen wir z. B. bei der Kalibrierung direkt die inneren und äußeren Kameraparameter berechnen, so nehmen wir Modell (14.26) oder Modell (14.32). Eine nichtlineare Kalibrierung mit Modell (14.26) ergibt:

$$\begin{aligned} & \sum_{i=1}^n \left(u_i - \frac{(\alpha r_{11} + u_0 r_{31})x + (\alpha r_{12} + u_0 r_{32})y + (\alpha r_{13} + u_0 r_{33})z + \alpha t_1 + u_0 t_3}{r_{31}x + r_{32}y + r_{33}z + t_3} \right)^2 \\ & + \left(v_i - \frac{(\alpha r_{21} + v_0 r_{31})x + (\alpha r_{22} + v_0 r_{32})y + (\alpha r_{23} + v_0 r_{33})z + \alpha t_2 + v_0 t_3}{r_{31}x + r_{32}y + r_{33}z + t_3} \right)^2 \\ & \rightarrow \text{Minimum.} \end{aligned} \quad (15.8)$$

Bei dieser Formulierung müssen wir aber noch Restriktionen an die r_{ij} stellen, da diese Elemente einer Rotationsmatrix sind. Eine andere Möglichkeit ist auch, ausgehend von den Rotationsmatrizen mit den Eulerschen Winkeln, sukzessive die Abbildung aufzuschreiben und die Eulerschen Winkel direkt zu bestimmen. Eine gute Anfangsnäherung muss aber zur Verfügung gestellt werden. In der Abb. 15.1 ist ein Kalibrierkörper zu sehen. Ein übliches Problem ist: welche Marker benutzt man, damit die Abbilder der Marker im

Abb. 15.2 Darstellung der projektiven Transformation einer Ellipse



2D-Bild sicher und mit hoher Genauigkeit detektieren werden können. Oft werden Quadrate verwendet und die vier Eckpunkte des Quadrates im Bild bestimmt. Man berechnet dann den Schnittpunkt der Diagonalen und dieser Schnittpunkt mit seinen Koordinaten ist der eigentliche Punkt. Dabei haben wir folgende wichtige Beziehung benutzt: im Bild ist das Quadrat kein Quadrat mehr, sondern irgendein Viereck und wir berechnen nun den Schnittpunkt dieser Diagonalen. Der Schnittpunkt der Diagonalen des Quadrates muss aber in den Schnittpunkt der Diagonalen des Vierecks übergehen, da Geraden nun einmal in Geraden übergehen und der Schnittpunkt auf beiden Geraden liegt. Man muss sich also bei der Konstruktion der Marker genau überlegen, ob das Referenzproblem auch stimmt. Dazu ein einfaches Gegenbeispiel:

Es ist naheliegend auf dem Kalibrierkörper Kreise aufzutragen und den Kreismittelpunkt als Marker festzulegen. Im Bild geht nun ein Kreis in eine Ellipse über, man detektiert diese und bestimmt den Ellipsenmittelpunkt als Referenzpunkt im Bild. Dies wäre bildanalytisch viel einfacher zu bewerkstelligen als die Quadrate. Man irrt sich hier aber gewaltig:

Der Kreismittelpunkt geht im Allgemeinen nicht in den Ellipsenmittelpunkt über!

In der Abb. 15.2 ist dies deutlich zu sehen. Auf der linken Seite ist eine Ellipse mit den vier Scheitelpunkten zu sehen, wobei die Scheitelpunkte als Quellpunkte dienen sollen. Vier Quellpunkte und vier Zielpunkte bestimmen eindeutig eine $2D \leftrightarrow 2D$ projektive Transformation. Nun geben wir uns auf der rechten Seite vier Zielpunkte vor, die auch auf einer Ellipse liegen sollen und bestimmen die projektive Transformation. Eine Ellipse geht in eine Ellipse über, die vier Punkte bleiben also auf der Ellipse. Folglich geht der Mittelpunkt nicht mehr in den Mittelpunkt über. Der Mittelpunkt einer Ellipse ist gleichzeitig ihr Schwerpunkt, daher geht bei einer projektiven Abbildung der Schwerpunkt nicht mehr in den Schwerpunkt über (bei affinen Transformationen ist das noch richtig). Wir können folglich auch nicht von anderen Objekten die Schwerpunkte als Referenzpunkte nutzen.

15.2 Selbstkalibrierung

15.2.1 Allgemeines

Da die Kalibrierkörper umständlich zu handhaben sind, steht die Frage, ob es auch ohne solche Körper möglich ist, Kameras zu kalibrieren. Dazu ein Gedankenexperiment: wenn

die Koordinaten der Raumpunkte nicht bekannt sind, dann nehmen wir sie einfach als zusätzliche Variable in die Rechnung mit auf. Für eine Kamera haben wir demnach 11 unbekannte Kameraparameter und für einen 3D-Punkt drei Unbekannte. Für eine Kamera erhöht also jeder weitere Punkt die Anzahl der Unbekannten um drei. Mit einer Kamera scheint dies also nicht möglich zu sein. Nun nehmen wir k Kameras gleichzeitig, die alle augenblicklich zusammen kalibriert werden sollen. Die Anzahl der gedachten 3D-Punkte sei p . Also liefern die 3D-Punkte $3p$ Unbekannte. Jede Kamera liefert 11 Unbekannte, also haben wir $11k$ Kameraparameter zu bestimmen, damit insgesamt $11k + 3p$ Unbekannte. Die Abbildungsgleichungen liefern für jede Kamera und jeden Punkt 2 Gleichungen, also insgesamt $2pk$ Gleichungen. Damit das Problem überhaupt lösbar sein kann, brauchen wir mindestens soviel Gleichungen wie Unbekannte. Daher muss gelten:

$$11 \cdot k + 3 \cdot p \leq 2 \cdot p \cdot k. \quad (15.9)$$

Dies gilt natürlich nur, wenn die Gleichungen keine Abhängigkeiten enthalten. Wir schauen, ob es für zwei Kameras Lösungen gibt. Dies ist mit $p \geq 22$ formal der Fall, d. h. wenn wir zwei Kameras haben und mindestens 22 Punkte in jedem Bild finden (Referenzpunkte, also Abbilder der gleichen 3D-Punkte, deren 3D-Koordinaten wir aber nicht kennen). Dann können wir alle Parameter beider Kameras berechnen und erhalten gleichzeitig die eigentlich nicht benötigten 3D-Koordinaten der Kalibrierpunkte mit. Wir müssen nun schauen, ob in diesem Falle Abhängigkeiten in den Gleichungen enthalten sein können. Da die Lage des Weltkoordinatensystems völlig egal ist, könnte man dies auch in die erste Kamera legen, d. h. die sechs äußeren Parameter dieser Kamera sind frei wählbar. Weiterhin ist die Skalierung völlig offen, dies ist ein freier Parameter. Damit sind sieben Parameter wählbar, ohne dass inhaltlich sich etwas ändern würde. Folglich sind nicht 22 Kameraparameter berechenbar, sondern eigentlich nur 15. Demzufolge brauchen wir auch nur 15 Punktreferenzen, um die restlichen Parameter und die 3D-Punkte auszurechnen, denn die sieben frei wählbaren Parameter geben wir fest vor. Damit wären nur 15 Punktreferenzen als Minimalinformation nötig und nicht 22 Punktreferenzen, wie wir formal mit (15.9) bestimmt haben. Durch diese Überlegung haben wir gezeigt, dass die Gleichungen Abhängigkeiten enthalten müssen. Ein Nachteil dieser Art Selbstkalibrierung ist offensichtlich der steigende numerische Rechenaufwand, wir haben unter Berücksichtigung der Minimalinformation mit nur 15 Punktreferenzen und 15 Kameraparametern ein System mit 60 Gleichungen und 60 Unbekannten ($15 = 22 - 7$ Kameraparameter, $45 = 3 \cdot 15$ Unbekannte für die 3D-Punkte) zu lösen. Diese Art Kalibrierung nennt man *Selbstkalibrierung* oder auch *Bündelausgleichung*. Da wir nun die Kameras kalibriert haben und in einem 3D-Koordinatensystem sogar die Kalibrierpunkte berechnet haben, können wir auch aus Bildpunkten 3D-Rekonstruktionen berechnen, d. h. weitere 3D-Koordinaten. Da wir aber keinen Kalibrierkörper hatten, wissen wir auch nicht, wenn wir nun einen benutzen würden, welche metrische Länge einer Einheit entspricht, d. h. ist das 1 cm oder 1 m? Daher

können wir nicht die absolute Skalierung des 3D-Körpers berechnen. Kalibrieren wir dagegen mit einem Kalibrierkörper, dann ist dieser das Weltkoordinatensystem und die absolute Skalierung ist festgelegt. Für die Bündelausgleichung gibt es professionelle Software zu kaufen. Das Schwierige dabei ist die numerische Lösung von großen nichtlinearen Gleichungssystemen.

Andere Methoden der Selbstkalibrierung beruhen oft darauf, ausschließlich die inneren Kameraparameter zu berechnen. Nehmen wir einmal unsere fundamentale Beziehung (14.173) zwischen E-Matrix und F-Matrix $F = (\mathbf{K}^{-1})^T \mathbf{E} \mathbf{K}^{-1}$ für eine rotierte und translatierte Kamera deren innere Parameter sich nicht geändert haben. Die F-Matrix bestimmen wir aus Punktkorrespondenzen und sei nun bekannt. Dann suchen wir eine obere Dreiecksmatrix, so dass die F-matrix transformiert die E-Matrix ergibt. Die E-Matrix haben wir allerdings nicht, wir nutzen aber die Tatsache, dass die beiden ersten singulären Werte gleich Eins sein müssen.

Eine weitere Idee besteht in den sogenannten Kruppa-Gleichungen. Wir erinnern uns, dass die Punkte in homogenen Koordinaten $\tilde{\mathbf{x}}_\infty^T = (\mathbf{x}_\infty^T, 0)$ Punkte der uneigentlichen Ebene im Raum sind. Damit gilt für die Abbildungsgleichung $\tilde{\mathbf{u}} = \mathbf{K} \mathbf{R} \mathbf{x}_\infty$, siehe auch (14.70). Also ist $\mathbf{A} = \mathbf{K} \mathbf{R}$ eine $2D \leftrightarrow 2D$ projektive Transformation (Homographie) der uneigentlichen Ebene π_∞ in die Bildebene. In der uneigentlichen Ebene befindet sich der sogenannte *absolute conic* Ω_∞ (AC) mit der Bedingung

$$\begin{aligned}\tilde{\mathbf{x}}^T &= (\tilde{x}, \tilde{y}, \tilde{z}, \tilde{t}) \quad \text{mit } \tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2 = 0, \quad \tilde{t} = 0, \quad \text{also } \tilde{\mathbf{x}}^T = (\mathbf{x}_\infty, 0)^T, \\ \|\mathbf{x}_\infty\|^2 &= 0,\end{aligned}\tag{15.10}$$

siehe auch Abschn. 14.2. Der AC enthält keine reellen Punkte, sondern nur komplexe und kann im Bild nicht direkt beobachtet werden. Seine theoretischen Gleichungen im Bild können aber angegeben werden:

$$0 = \mathbf{x}_\infty^T \mathbf{I} \mathbf{x}_\infty = \tilde{\mathbf{u}}^T (\mathbf{K} \mathbf{R})^{-1} \mathbf{I} (\mathbf{K} \mathbf{R})^{-1} \tilde{\mathbf{u}} = \tilde{\mathbf{u}}^T \mathbf{K}^{-1} \mathbf{I} \mathbf{R}^{-1} \mathbf{K}^{-1} \tilde{\mathbf{u}}.\tag{15.11}$$

Also ist $\omega = \mathbf{K}^{-1} \mathbf{I} \mathbf{R}^{-1} \mathbf{K}^{-1}$ das Abbild des AC, genannt *image of the absolute conic* (IAC). Dieser IAC ist also unabhängig von der äußeren Orientierung der Kamera und hängt nur noch von den inneren Kameraparametern ab und natürlich von einem Faktor. Hat man auf irgendeine Weise diese quadratische Form berechnet, dann man durch Cholesky-Zerlegung (Ziehen der Wurzel aus einer Matrix) die innere Projektionsmatrix \mathbf{K} berechnen. Wir betrachten jetzt eine Ebene π mit $\tilde{z} = 0$, welche die uneigentliche Ebene schneidet und zwar in einer uneigentlichen Geraden. Als muss für diese uneigentliche Gerade $\tilde{z} = 0, \tilde{t} = 0$ gelten, während \tilde{x}, \tilde{y} beliebig sind. Diese uneigentliche Gerade schneidet den AC in genau zwei Punkten $(1, \pm i, 0, 0)^T$, wie man leicht überprüfen kann. Diese beiden Punkte werden *circular points* der Ebene π genannt. Wenn wir nun einmal das Koordinatensystem rotieren

und translatieren sowie skalieren und damit die Punkte der uneigentlichen Ebene transformieren, so ergibt sich

$$\tilde{\mathbf{x}}'_{\infty} = s \tilde{\mathbf{R}} \tilde{\mathbf{T}} \tilde{\mathbf{x}}_{\infty} \quad (15.12)$$

und damit:

$$\mathbf{x}'_{\infty} = s \mathbf{R} \mathbf{x}_{\infty}. \quad (15.13)$$

Nun bilden wir die Gleichung des AC:

$$\|\mathbf{x}'_{\infty}\|^2 = s^2 \|\mathbf{R} \mathbf{x}_{\infty}\|^2 = s^2 \|\mathbf{x}_{\infty}\|^2 = 0. \quad (15.14)$$

Damit ist die Beschreibung des AC unabhängig von der Bewegung und Skalierung des Koordinatensystems. Rotieren und translatieren wir ein Koordinatensystem und schneiden den AC mit der Ebene (wie oben beschrieben) $\tilde{z} = 0$, so müssen die beiden Schnittpunkte auf dem gleichen AC wie im vorherigen Koordinatensystem sein, allerdings sind es zwei andere Schnittpunkte. Also könnten wir in verschiedene Kalibrierebenen je ein neues Weltkoordinatensystem legen und die Schnittpunkte mit dem AC berechnen, die alle auf dem einzigen AC liegen müssen. Daraus kann man nun eine einfache Kalibrierung aufbauen:

Wir bauen so etwas wie einen Kalibrierkörper. Er besteht aus drei Ebenen, die nicht notwendig orthogonal zueinander sein müssen, aber drei paarweise verschiedene Ebenen müssen es sein. In jede Ebene tragen wir einen Marker in Form eines Quadrates auf, dies ist alles, es werden keine Koordinaten benötigt. Für jede Ebene, d. h. für jedes Quadrat bestimmen wir nun eine $2D \leftrightarrow 2D$ projektive Transformation H (Homographie), sodass die Punkte $(0, 0)^T, (0, 1)^T, (1, 0)^T, (1, 1)^T$ in die vier Bildpunkte des Quadrates (die jetzt im Bild irgendein Viereck bilden) überführt werden. Wir haben damit in jede Ebene nacheinander ein Weltkoordinatensystem mit $\tilde{z} = 0$ gelegt. Da die Ebene mit $\tilde{z} = 0$, in der das Quadrat liegt, als uneigentliche Punkte die *circular points* haben muss (sie schneidet ja die uneigentliche Ebene in einer Geraden, auf der diese Punkte liegen), bilden wir sie nun ins Bild ab durch $\mathbf{H} \cdot (1, \pm i, 0)^T$. Diese berechneten Punkte sind natürlich auch komplex. Dies tun wir für alle drei Quadrate. Dadurch erhalten wir sechs verschiedene Punkte im Bild, die (theoretisch) auf dem IAC liegen müssen. Daher fitten wir an diese sechs komplexen Punkte eine „Ellipse“ und erhalten die Matrix ω . Diese bzw. ω^{-1} zerlegen wir mit der Cholesky-Zerlegung in die interne Projektionsmatrix \mathbf{K} und erhalten die inneren Kameraparameter. Natürlich baut man praktisch nicht solch einen Kalibrierkörper, sondern geht anders vor. In einer Ebene bringt man ein Kalibriermuster an und bestimmt die Homographie ins Bild. Nun bewegt man die Kamera und bestimmt eine zweite, bzw. dritte Aufnahme. Um so mehr Aufnahmen, umso besser ist das Fitten der quadratischen Form. So kann man also aus einer Ebene mit mehreren Aufnahmen die inneren Kameraparameter bestimmen.

Diese Kalibriertechnik mittels Ebenen zur Bestimmung der inneren Parameter kann man aber noch einfacher herleiten.

15.2.2 Planare Kalibrierung nach Zhang

Dazu legen wir das Weltkoordinatensystem in eine Ebene und bilden die x, y -Ebene mit $z = 0$ ab und erhalten damit eine $2D \leftrightarrow 2D$ projektive Transformation \mathbf{H} :

$$\begin{aligned}\tilde{\mathbf{u}} &= \mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{t}) = \mathbf{K} \cdot \left((\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \mathbf{t} \right) \\ &= \mathbf{K} \cdot \left((\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \cdot \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \mathbf{t} \right) \\ &= \mathbf{K} \cdot (\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}) \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{H} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \quad (15.15)\end{aligned}$$

Wenn wir die Spaltenvektoren von \mathbf{H} mit $(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3)$ bezeichnen, dann ist folglich:

$$\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3) = \mathbf{K} \cdot (\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}). \quad (15.16)$$

Also ist:

$$\mathbf{h}_1 = \mathbf{K}\mathbf{r}_1, \mathbf{h}_2 = \mathbf{K}\mathbf{r}_2. \quad (15.17)$$

Damit folgt:

$$\mathbf{r}_1 = \mathbf{K}^{-1}\mathbf{h}_1, \mathbf{r}_2 = \mathbf{K}^{-1}\mathbf{h}_2. \quad (15.18)$$

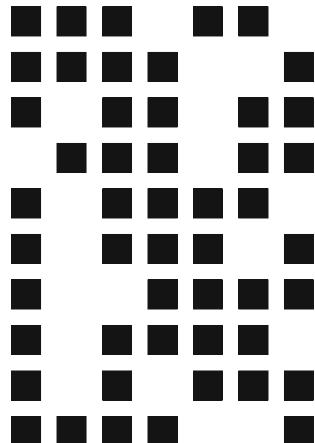
Da die beiden Vektoren $\mathbf{r}_1, \mathbf{r}_2$ die ersten beiden Spaltenvektoren der Rotationsmatrix \mathbf{R} sind, haben sie die gleiche Länge und sind orthogonal zueinander (die Bedingung $\|\mathbf{r}_i\| = 1$ können wir nicht nutzen, da \mathbf{H} mit einer Konstanten multipliziert werden darf). Also folgen die beiden Gleichungen:

$$\mathbf{h}_1^T \mathbf{K}^{-1} \mathbf{K}^{-1} \mathbf{h}_2 = 0 \quad (15.19)$$

$$\mathbf{h}_1^T \mathbf{K}^{-1} \mathbf{K}^{-1} \mathbf{h}_1 - \mathbf{h}_2^T \mathbf{K}^{-1} \mathbf{K}^{-1} \mathbf{h}_2 = 0. \quad (15.20)$$

Man sieht, die Unbekannte ist wieder die Matrix $\omega = \mathbf{K}^{-1} \mathbf{K}^{-1}$ des IAC. Die Matrix ω hat mit einem Faktor sechs Unbekannte, ohne Faktor die fünf inneren Parameter. Für eine Homographie \mathbf{H} erhalten wir zwei Gleichungen, so dass wir wenigstens drei Homographien benötigen, d. h. drei Kalibrierpositionen der Kamera benötigen, die ein Muster in einer 2D-Ebene aufnimmt um die Homographie zu bestimmen. Wollen wir z. B. das Produkt von Skalierungsfaktor und Kammerkonstante nicht in Pixeln sondern z. B. in m oder

Abb. 15.3 Planares Kalibriermuster ohne Landmarken



mm, so müssen wir das Kalibriermuster ausmessen und die Koordinaten entsprechend dieser Maßeinheiten angeben. Wenn wir $\omega = \mathbf{K}^{-1} \mathbf{K}^{-1}$ bis auf einen Faktor berechnet haben, berechnen wir \mathbf{K} von ω^{-1} mittels der Cholesky-Zerlegung eine obere Dreiecksmatrix, die auch bis auf einen Faktor bestimmt ist. Da aber $k_{33} = 1$ sein muss, können wir diesen Faktor eliminieren. Gedanklich sitzt unser Weltkoordinatensystem auf dem planaren Kalibrierkörper. Daher können wir auch die äußeren Kameraparameter ausrechnen mit:

$$\mathbf{r}_1 = \mathbf{K}^{-1} \mathbf{h}_1, \quad \mathbf{r}_2 = \mathbf{K}^{-1} \mathbf{h}_2, \quad \mathbf{t} = \mathbf{K}^{-1} \mathbf{h}_3. \quad (15.21)$$

Da der Kalibrierkörper planar ist, ergänzen wir einfach \mathbf{r}_3 orthogonal zu den beiden bereits berechneten Vektoren $\mathbf{r}_1, \mathbf{r}_2$, wobei alle drei Rechtssysteme bilden sollen. Da wir zur Berechnung der inneren Kameraparameter mindestens drei Kamerapositionen benötigen, können wir auch für jede Kameraposition nach dieser Formel die äußeren Kameraparameter berechnen. Folglich können wir auch daraus (über die Dreiecksbeziehung) die Bewegung der Kamera gegeneinander ausrechnen. In Abb. 15.3 ist ein Beispiel eines planaren Kalibrierkörpers zu sehen, welcher keine Landmarken besitzt. Weiterhin ist durch die Anordnung der Quadrate die Symmetrie gebrochen, damit die Lösung des Korrespondenzproblems eindeutig ist. Um die Homographie zum aufgenommenen Muster zu bestimmen, müssen die Punktkorrespondenzen ermittelt werden. Dazu kann man z. B. die Abbilder der schwarzen Quadrate im Bild bestimmen, die theoretisch beliebige Vierecke bilden. Von den Vierecken bestimmt man den Schnittpunkt der Diagonalen als den repräsentativen Punkt des Viereckes. Der Schwerpunkt ist problematisch, da er nicht projektiv kovariant ist. Mit Hilfe eines geeigneten Verfahrens (z. B. der DAPPM-Algorithmus, siehe Abschn. 19.8.4) werden die Punktkorrespondenzen zwischen dem theoretischen Muster aus Abb. 15.3 und dem aufgenommenen Bild vom Muster bestimmt, woraus man dann mittels Ausgleichsrechnung die Homographie bestimmt.

Das Verfahren zur Bestimmung der Punktkorrespondenzen muss so robust sein, dass auch trotz Verdeckungen die richtige Homographie bestimmt wird, siehe Algorithmus

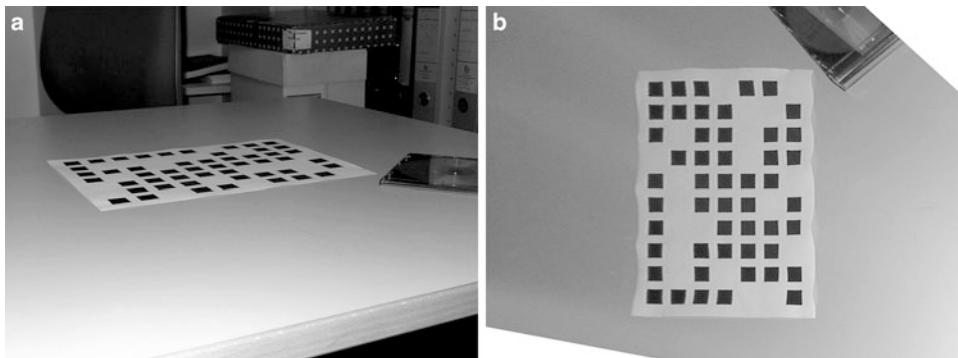


Abb. 15.4 Kalibrierung nach Zhang: **a** Extrem projektiv verzerrtes Bild vom planaren Kalibriermuster, **b** mittels Punktcorrespondenzen und DAPPM-Algorithmus (Abschn. 19.8.4) projektiv entzerrtes Bild

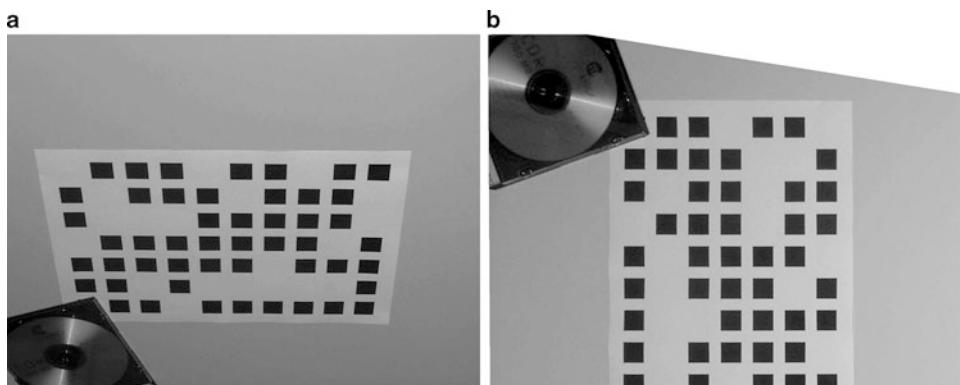


Abb. 15.5 Kalibrierung nach Zhang: **a** Bild vom Kalibriermuster mit Verdeckung, **b** trotz Verdeckung richtig bestimmte projektive Entzerrung

DAPPM (Abschn. 19.8.4) und Abb. 15.5. Bei den drei Aufnahmen des planaren Kalibriermusters ist weiterhin darauf zu achten, dass die Kamera in den drei Positionen nicht die gleichen Blickwinkel zum Kalibriermuster hat, sondern drei „echt“ verschiedene Blickwinkelpositionen gewählt werden.

Nehmen wir einmal an, wir hätten „im Labor“ durch planare Kalibrierung die inneren Kameraparameter ausgerechnet (wenn nötig auch die Verzeichnung), damit ist also \mathbf{K} bekannt. Wenn wir jetzt an einem anderen Ort mit der Kamera aus zwei Positionen heraus ein Objekt vermessen wollten (durch Triangulation), brauchen wir vor Ort nur die äußeren Kameraparameter bestimmen. Dies können wir wieder durch planare Kalibrierung lösen, wir brauchen jetzt nur die Rollen zu vertauschen. Wenn wir pro Kameraposition die Homographie H bestimmt haben, können wir wieder entsprechend (15.21) die äußeren Kameraparameter bestimmen. Bei Kenntnis der inneren Parameter wissen wir, dass nicht

jede Homographie zugelassen ist, es gibt die zwei Restriktionen wie (15.20), nämlich:

$$\mathbf{h}_1^T \mathbf{K}^{-1} {}^T \mathbf{K}^{-1} \mathbf{h}_2 = 0 \quad (15.22)$$

$$\mathbf{h}_1^T \mathbf{K}^{-1} {}^T \mathbf{K}^{-1} \mathbf{h}_1 - \mathbf{h}_2^T \mathbf{K}^{-1} {}^T \mathbf{K}^{-1} \mathbf{h}_2 = 0, \quad (15.23)$$

wobei \mathbf{K} als bekannt vorausgesetzt wird. Damit ist die minimale Anzahl von Referenzpunkten zur Bestimmung der Homographie nicht vier, sondern nur drei (allerdings nicht ganz eindeutig, sondern drei + plus eine gewisse Zusatzinformation, aber weniger Information als vier Punkte). Wenn wir jetzt mehr als 3 Referenzpunkte zur Verfügung haben, dann berechnen wir die Homographie als Ausgleichsproblem, allerdings mit den beiden Restriktionen (15.22), (15.23). Diese Idee können wir für ein einfaches, praktisches Vermessungsproblem mit einer „handelsüblichen“ Kamera nutzen, siehe Abschn. 16.5.

Manchmal betrachtet man statt des IAC ω die inverse Matrix ω^{-1} , genannt dualer IAC (DIAC) oder auch *Kruppa-Matrix*. Es sei wieder $\tilde{\mathbf{o}}$ der Epipol und $\tilde{\mathbf{y}}$ ein gedachter Punkt im Bild. Die Gerade durch beide Punkte ist eine Epipolarlinie. Diese Epipolarlinie ist genau dann tangent zum IAC, wenn der Normalenvektor der Epipolarlinie auf dem DIAC liegt, d. h. wenn

$$(\tilde{\mathbf{o}} \times \tilde{\mathbf{y}})^T \omega^{-1} (\tilde{\mathbf{o}} \times \tilde{\mathbf{y}}) = 0 \quad (15.24)$$

gilt, siehe auch (13.38). Da dies eine quadratische Form darstellt, haben wir nur fünf eigentliche Parameter. Die Matrix ist symmetrisch, dadurch haben wir formal sechs Matrixparameter, in denen aber ein Skalierungsfaktor enthalten ist. Dies ist nun als Bestimmungsgleichung für den Punkt $\tilde{\mathbf{y}}$ anzusehen. Wir notieren einmal die Elemente in einer Form nach Kruppa (ist ungewöhnlich, aber eben nur eine Bezeichnung):

$$\omega^{-1} = \begin{pmatrix} -\delta_{23} & \delta_3 & \delta_2 \\ \delta_3 & -\delta_{13} & \delta_1 \\ \delta_2 & \delta_1 & -\delta_{12} \end{pmatrix}. \quad (15.25)$$

Da uns nur diejenige Epipolarlinie interessiert, die tangent zum IAC ist, nehmen wir als zweiten Punkt nicht irgendeinen Punkt auf der Epipolarlinie, sondern den Schnittpunkt dieser Epipolarlinie mit der uneigentlichen Geraden $\tilde{y}_3 = 0$, so dass nur \tilde{y}_1, \tilde{y}_2 zu bestimmen sind. Wenn wir dies in (15.24) einsetzen zur Bestimmung dieses Punktes, so erhalten wir die Gleichung:

$$A_{11}\tilde{y}_1^2 + 2A_{12}\tilde{y}_1\tilde{y}_2 + A_{22}\tilde{y}_2^2 = 0 \quad (15.26)$$

mit den Koeffizienten:

$$\begin{aligned} A_{11} &= -\delta_{13}\delta_3^2 - \delta_{12}\delta_2^2 - 2\delta_1\delta_2\delta_3 \\ A_{12} &= \delta_{12}\delta_1\delta_2 - \delta_3\delta_3^2 + \delta_2\delta_2\delta_3 + \delta_1\delta_1\delta_3 \\ A_{22} &= -\delta_{23}\delta_3^2 - \delta_{12}\delta_1^2 - 2\delta_2\delta_1\delta_3. \end{aligned} \quad (15.27)$$

Wenn wir dies nun für eine zweite Kamera aufschreiben würden, so brauchen wir überall nur einen „Strich“ an die Variablen zu machen. Ist die zweite Kamera aber nur eine Bewegung der ersten, d. h. die inneren Parameter bleiben konstant, so ändert sich der IAC (die Koeffizienten von ω bzw. ω^{-1}) nicht, denn dieser hängt nicht von der Orientierung der Kamera ab. Bei zwei Positionen einer Kamera können wir nun die Epipolartransformation aufschreiben als eindimensionale projektive Transformation, die von drei Parametern abhängt, siehe auch (14.155) und (14.156). Dazu nehmen wir an, diese sei bekannt:

$$\tau' = \frac{a\tau + b}{c\tau + d}. \quad (15.28)$$

Dabei ist τ der Tangens, d.h der Anstieg der Epipolarlinien. Da der Punkt $\tilde{\mathbf{y}}$ aber auf der uneigentlichen Geraden liegt, können wir den Tangens einfach zu $\tau = \tilde{y}_2/\tilde{y}_1$ berechnen (analog auch τ'). Dies sieht man leicht an folgender Beziehung:

$$\tau = \tan \alpha = \lim_{\beta \rightarrow 0} \frac{\frac{\tilde{o}_2}{\tilde{o}_3} - \frac{\tilde{y}_2}{\beta}}{\frac{\tilde{o}_1}{\tilde{o}_3} - \frac{\tilde{y}_1}{\beta}} = \lim_{\beta \rightarrow 0} \frac{\beta \frac{\tilde{o}_2}{\tilde{o}_3} - \frac{\tilde{y}_2}{1}}{\beta \frac{\tilde{o}_1}{\tilde{o}_3} - \frac{\tilde{y}_1}{1}} = \frac{\tilde{y}_2}{\tilde{y}_1}. \quad (15.29)$$

Mit diesem Tangens setzen wir die Transformation in obige zwei Gleichungen ein und erhalten:

$$A_{11} + 2A_{12}\tau + A_{22}\tau^2 = 0 \quad (15.30)$$

$$A'_{11}(c\tau + d)^2 + 2A'_{12}(a\tau + b)(c\tau + d) + A'_{22}(a\tau + b)^2 = 0. \quad (15.31)$$

Beide Gleichungen, die jeweils quadratisch sind, müssen die gleichen Lösungen in τ besitzen. Daher quadrieren wir aus und vergleichen die Koeffizienten, die bis auf einen gemeinsamen Faktor gleich sein müssen. Daher haben wir drei Gleichungen mit einem unbekannten Faktor. Wir dividieren, um den Faktor zu eliminieren und erhalten folglich nur noch zwei Gleichungen:

$$A_{11}(A'_{11}c^2 + 2A'_{12}ac + A'_{22}a^2) - A_{22}(A'_{11}d^2 + 2A'_{12}bd + A'_{22}b^2) = 0 \quad (15.32)$$

$$A_{11}(A'_{11}cd + A'_{12}ad + A'_{12}bc + A'_{22}ab) - A_{12}(A'_{11}d^2 + 2A'_{12}bd + A'_{22}b^2) = 0. \quad (15.33)$$

Das sind die sogenannten Gleichungen nach **Kruppa**. Gehen wir nochmals von der Beziehung

$$(\tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}')^T \omega^{-1} (\tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}') = 0 \quad (15.34)$$

in der zweiten Kamera aus, wobei die inneren Parameter wieder gleich sein sollen, d. h. wir haben den gleichen DIAC oder die gleiche Kruppa-Matrix. Nun schreiben wir das Kreuzprodukt wieder mit den schiefsymmetrischen Matrizen auf, d. h.

$$(\tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}')^T \omega^{-1} (\tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}') = \tilde{\mathbf{y}}'^T \mathbf{O}' \omega^{-1} \mathbf{O}' \tilde{\mathbf{y}}' = 0. \quad (15.35)$$

Die dem Punkt $\tilde{\mathbf{y}}'$ zugeordnete Epipolarlinie in der ersten Kamera lautet $\mathbf{l} = \mathbf{F}^T \tilde{\mathbf{y}}'$ und damit die Gleichung des DIAC: $\mathbf{l}^T \omega^{-1} \mathbf{l} = \tilde{\mathbf{y}}'^T \mathbf{F} \omega^{-1} \mathbf{F}^T \tilde{\mathbf{y}}' = 0$. Aus beiden Beziehungen folgt $\mathbf{F} \omega^{-1} \mathbf{F}^T = \lambda \mathbf{O}' \omega^{-1} \mathbf{O}'$. Die gleiche Beziehung kann auch noch anders hergeleitet werden. Wenn wir die uneigentliche Ebene in beiden Kameras abbilden, dann gibt es zwischen beiden Kamerabildern eine Homographie $\mathbf{H}_\infty = \mathbf{K}' \mathbf{R} \mathbf{K}^{-1}$, wobei \mathbf{R} die Rotation zwischen den beiden Kamerakoordinatensystemen bedeutet. Da wir annehmen, dass die inneren Parameter gleich bleiben, gilt $\mathbf{H}_\infty = \mathbf{R} \mathbf{K}^{-1}$. Wir bilden einfach $\mathbf{H}_\infty \omega^{-1} \mathbf{H}_\infty^T = \mathbf{H}_\infty \mathbf{K} \mathbf{K}^T \mathbf{H}_\infty^T = \mathbf{K} \mathbf{K}^T = \omega^{-1}$. Daher bilden wir das Produkt $\mathbf{O}' \omega^{-1} \mathbf{O}' = \mathbf{O}' \mathbf{H}_\infty \omega^{-1} \mathbf{H}_\infty^T \mathbf{O}'$. Mit (14.195) hatten wir die Beziehung $\mathbf{F} = \mathbf{O}' \mathbf{H}$ für eine $2D \leftrightarrow 2D$ projektive Transformation (Homographie) der beiden Bildebenden bezüglich einer abgebildeten Ebene im Raum hergeleitet. Dies gilt genauso für die uneigentliche Ebene, also $\mathbf{F} = \mathbf{O}' \mathbf{H}_\infty$. Setzen wir diese Beziehung oben ein, so erhalten wir auch die Kruppa-Gleichungen in Matrixform. Da innerhalb dieser Matrixgleichung Abhängigkeiten auftauchen, bleiben eigentlich nur die oben abgeleiteten zwei Elementargleichungen übrig. Erinnern wir uns mit (14.173) an eine weitere Darstellung der Fundamentalmatrix $\mathbf{F} = \mathbf{T}' \mathbf{R} \mathbf{K}^{-1}$ mit $\mathbf{t}' = \mathbf{K} \mathbf{t}$, wobei \mathbf{t} wieder die Translation und \mathbf{R} die Rotation zwischen den beiden Kamerakoordinatensystemen sind und \mathbf{T}, \mathbf{T}' die bezüglich des Kreuzproduktes zugeordneten schiefsymmetrischen Matrizen der Vektoren $\mathbf{t}, \mathbf{t}' = \mathbf{K} \mathbf{t}$ sind. Damit haben wir zwei verschiedene Darstellungsformen der Kruppa-Gleichungen in Matrixdarstellung:

$$\mathbf{F} \omega^{-1} \mathbf{F}^T = \lambda \mathbf{O}' \omega^{-1} \mathbf{O}' \quad \text{und} \quad \mathbf{F} \omega^{-1} \mathbf{F}^T = \mu \mathbf{T} \omega^{-1} \mathbf{T}'^T. \quad (15.36)$$

Jede der beiden Matrix-Gleichungen repräsentiert aber nur zwei skalare, nichtlineare Gleichungen.

15.2.3 Tomasi und Kanade: Schwache Perspektive

Bei der allgemeinen Selbstkalibrierung (Bündelausgleichung) zur simultanen Bestimmung der Kameraparameter und der 3D-Punkte muss ein nichtlineares Gleichungssystem gelöst werden. Kann man dieses unter Einschränkungen auf eine Folge von (fast) linearen Problemen zurückführen? Dazu nehmen wir an, die inneren Parameter (Matrix \mathbf{K}) seien bekannt. Die Bildkoordinaten rechnen wir somit um in normalisierte Koordinaten, wir tun so, als hätten wir eine ideale Lochkamera mit der Kamerabrennweite $f = 1$. Weiterhin nehmen wir jetzt die Schwache Perspektive an, d. h. die skalierte orthografische Projektion als Approximation der perspektivischen Projektion, siehe auch (14.7) und (14.8). Gegeben seien nun K Kameras $k = 1, \dots, K$ und P 3D-Punkte $\mathbf{x}_p, p = 1, \dots, P$. Es bezeichne $\bar{\mathbf{x}}$ den Schwerpunkt der P Punkte. Pro Kamera denken wir uns eine Ebene E_k durch den Schwerpunkt der P Punkte, die parallel zu der jeweiligen Bildebene liegen möge. Die äußeren Parameter der k -ten Kamera seien \mathbf{R}_k und \mathbf{t}_k . Es bezeichne $\mathbf{R}_k^T = (\mathbf{r}_{k,1}, \mathbf{r}_{k,2}, \mathbf{r}_{k,3})$, d. h. $\mathbf{r}_{k,1}^T, \mathbf{r}_{k,2}^T, \mathbf{r}_{k,3}^T$ sind die Zeilenvektoren von \mathbf{R}_k . Bei der Schwachen Perspektive nimmt man

an, alle 3D-Punkte liegen in der Ebene E_k und haben damit die gleiche Tiefe s_k . Dann ist:

$$s_k = \mathbf{r}_{k,3}^T \cdot (\bar{\mathbf{x}} - \mathbf{t}_k). \quad (15.37)$$

Die Abbildungsgleichungen sind dann:

$$u_{k,p} = \frac{1}{s_k} \mathbf{r}_{k,1}^T (\mathbf{x}_p - \mathbf{t}_k) = \mathbf{m}_k^T \mathbf{x}_p + a_k, \quad (15.38)$$

$$v_{k,p} = \frac{1}{s_k} \mathbf{r}_{k,2}^T (\mathbf{x}_p - \mathbf{t}_k) = \mathbf{n}_k^T \mathbf{x}_p + b_k. \quad (15.39)$$

Die Abbildungsgleichungen wollen wir in Matrixnotation $\mathbf{W} = \mathbf{MX} + \mathbf{T}$ aufschreiben, dann ist:

$$\begin{pmatrix} u_{1,1} & \cdots & u_{1,P} \\ \vdots & & \vdots \\ u_{K,1} & \cdots & u_{K,P} \\ \vdots & & \vdots \\ v_{1,1} & \cdots & v_{1,P} \\ \vdots & & \vdots \\ v_{K,1} & \cdots & v_{K,P} \end{pmatrix} = \begin{pmatrix} \mathbf{m}_1^T \\ \vdots \\ \mathbf{m}_K^T \\ \mathbf{n}_1^T \\ \vdots \\ \mathbf{n}_K^T \end{pmatrix} \cdot (\mathbf{x}_1 \quad \cdots \quad \mathbf{x}_P) + \begin{pmatrix} a_1 & \cdots & a_1 \\ \vdots & & \vdots \\ a_K & \cdots & a_K \\ b_1 & \cdots & b_1 \\ \vdots & & \vdots \\ b_K & \cdots & b_K \end{pmatrix}. \quad (15.40)$$

Auf der linken Seite steht die Meßmatrix, die für alle Kameras und alle Punkte bekannt ist. Auf der rechten Seite stehen nur Unbekannte, die Matrix \mathbf{M} ist eine skalierte Rotationsmatrix, die Matrix \mathbf{X} besteht aus den unbekannten Koordinaten der 3D-Punkte und in \mathbf{T} geht die Translation ein. Legen wir nun den Koordinatenursprung des Weltkoordinatensystems in den Schwerpunkt $\bar{\mathbf{x}}$, dann können wir tatsächlich die Matrix \mathbf{T} allein aus der bekannten Meßmatrix \mathbf{W} berechnen, denn dann folgt aus $\sum_k \mathbf{x}_k = 0$ sofort:

$$\sum_p u_{k,p} = \sum_p (\mathbf{m}_k^T \mathbf{x}_p + a_k) = \sum_p a_k = P \cdot a_k \rightarrow a_k = \frac{1}{P} \sum_p u_{k,p}. \quad (15.41)$$

Damit brauchen wir \mathbf{T} nicht mehr als Unbekanntes auffassen, wir setzen $\mathbf{W}^* = \mathbf{W} - \mathbf{T}$ und betrachten nur noch das Problem $\mathbf{W}^* = \mathbf{M} \cdot \mathbf{X}$. Da die Matrix \mathbf{M} drei Spalten und die Matrix \mathbf{X} drei Zeilen hat, kann der Rang von \mathbf{W}^* höchstens drei sein. Die Matrizen \mathbf{M} und \mathbf{X} berechnen wir nun mit der Singulärwertzerlegung, wir erinnern an die SVD im Abschn. 22.3:

$$\mathbf{W}^* = \mathbf{U} \Lambda \mathbf{V}^T. \quad (15.42)$$

Damit wäre $\mathbf{M} = \mathbf{U} \cdot \sqrt{\Lambda}$ und $\mathbf{X} = \sqrt{\Lambda} \cdot \mathbf{V}^T$. Sollte in der Praxis auf Grund von Rauschen, Fehlern usw. der Rang von \mathbf{W}^* größer als drei sein, so werden bei der SVD nur die ersten drei Singulärwerte berücksichtigt, dies ist dann die beste Approximation im Sinne der

Frobenius-Norm durch alle Matrizen vom Rang drei. Es wäre zu schön, wenn dies nun schon das Endergebnis wäre, man sieht sofort, es stimmt etwas nicht: Die Spalten von \mathbf{U} und \mathbf{V} sind orthogonal, die Spalten von \mathbf{M} bzw. die Zeilen von \mathbf{X} sind in der Regel nicht orthogonal. Dazu betrachten wir eine beliebige reguläre Matrix \mathbf{A} mit drei Spalten und drei Zeilen, dann sind $\mathbf{M}' = \mathbf{M} \cdot \mathbf{A}$ und $\mathbf{X}' = \mathbf{A}^{-1} \cdot \mathbf{X}$ auch gültige Faktorisierungen. Wir müssen folglich die „richtige“ Matrix \mathbf{A} finden. Wir nutzen dazu das Wissen, dass die Matrix \mathbf{M}' eine skalierte Rotationsmatrix ist, daher können wir als Bedingungen

$$|\mathbf{m}'_k|^2 = (\mathbf{m}'_k^T \mathbf{A})(\mathbf{m}'_k^T \mathbf{A})^T = \mathbf{m}'_k^T \mathbf{A} \mathbf{A}^T \mathbf{m}_k = \mathbf{n}'_k^T \mathbf{A} \mathbf{A}^T \mathbf{n}_k = |\mathbf{n}'_k|^2 \quad (15.43)$$

$$(\mathbf{m}'_k)^T \mathbf{n}'_k = \mathbf{m}'_k^T \mathbf{A} \mathbf{A}^T \mathbf{n}_k = 0 \quad (15.44)$$

formulieren. Wir sehen, dass $\mathbf{A} = \mathbf{0}$ auch Lösung ist, daher müssen wir noch eine Normierung festlegen, z. B.:

$$(\mathbf{m}'_1)^T \mathbf{m}'_1 = \mathbf{m}'_1^T \mathbf{A} \mathbf{A}^T \mathbf{m}_k = 1. \quad (15.45)$$

Mit dieser Normierung ergibt sich sofort die relative Tiefe der Punkte bezüglich der ersten Kamera zu $s_1 = 1$, alle anderen Tiefen bezüglich der anderen Kameras beziehen sich dann auf diese Normierung. Zur Bestimmung von \mathbf{A} berechnen wir zunächst die symmetrische Matrix $\mathbf{Q} = \mathbf{A} \mathbf{A}^T$ mit 6 Unbekannten. Da \mathbf{Q} linear in die Restriktionen eingeht, haben wir ein einfaches lineares Ausgleichsproblem zu lösen, anschließend ziehen wir die Wurzel aus \mathbf{Q} und haben das konkrete \mathbf{A} berechnet. Allerdings bleibt selbst dann noch eine letzte Mehrdeutigkeit für \mathbf{A} , wenn \mathbf{A} Lösung ist, dann auch $-\mathbf{A}$, eine von den beiden ist geeignet auszuwählen.

- Restriktionen haben wir nun $2P \cdot K - 2K = 2(P - 1)K$. Durch die Lage des Koordinatenursprungs mussten wir $2K$ Restriktionen wieder abziehen.
- Unbekannte haben wir $3P - 3 + 3K + K = 3P + 4K - 3$.
- Forderung: $2(P - 1)K \geq 3P + 4K - 3 \rightarrow P_{\min} = 4, K_{\min} = 5$.
Folgerung: Es müssen alle 3D-Punkte von mindestens 5 Kameras aus vollständig sichtbar sein und das Referenzproblem muss gelöst sein.
- Wenn $\text{Rang}(\mathbf{W}^*) < 3$, dann ist die Faktorisierung in zwei Matrizen mit dem Rang 3 nicht möglich. Spezialfälle sind dann:
 - $\text{Rang}(\mathbf{M}') < 3$, z. B. sind die Kameras nur translatiert und nicht zueinander rotiert.
 - $\text{Rang}(\mathbf{X}') < 3$, Punkte sind kollinear oder komplanar.
 - Wenn folglich zufällig alle Punkte in einer Ebene liegen und die schwache Perspektive fällt mit der Perspektive zusammen, dann versagt die Faktorisierung.

Es gibt auch eine Faktorisierung für die paraperspektivische Abbildung, formal anders sind lediglich die Restriktionen zur Bestimmung der Matrix \mathbf{A} .

15.2.4 Tomasi und Kanade: Perspektive

Nun soll die Idee für die Schwache Perspektive iterativ genutzt werden, um den Fall der echten Perspektive zu behandeln. Wir nehmen wieder an, dass die inneren Parameter jeder Kamera bekannt sind und benutzen daher normalisierte Bildkoordinaten. Somit lauten die Abbildungsgleichungen:

$$u_{k,p} = \frac{\mathbf{r}_{k,1}^T \mathbf{x}_p + t_{k,1}}{\mathbf{r}_{k,3}^T \mathbf{x}_p + t_{k,3}}, \quad v_{k,p} = \frac{\mathbf{r}_{k,2}^T \mathbf{x}_p + t_{k,2}}{\mathbf{r}_{k,3}^T \mathbf{x}_p + t_{k,3}}. \quad (15.46)$$

Wenn wir $t_{k,3} \neq 0$ annehmen, dann können wir auch schreiben:

$$u_{k,p} = \frac{\frac{\mathbf{r}_{k,1}^T \mathbf{x}_p}{t_{k,3}} + \frac{t_{k,1}}{t_{k,3}}}{1 + \varepsilon_{k,p}}, \quad v_{k,p} = \frac{\frac{\mathbf{r}_{k,2}^T \mathbf{x}_p}{t_{k,3}} + \frac{t_{k,2}}{t_{k,3}}}{1 + \varepsilon_{k,p}} \quad \text{mit } \varepsilon_{k,p} = \frac{\mathbf{r}_{k,3}^T \mathbf{x}_p}{t_{k,3}}. \quad (15.47)$$

Durch diese Schreibweise sieht man deutlich, dass die Zähler genau der Schwachen Perspektive entsprechen. Ist das Objekt „weit genug entfernt“, dann gilt $\varepsilon_{k,p} \approx 0$ und die Schwache Perspektive ist eine gute Näherung. Wir setzen nun $\lambda_{k,p} = 1 + \varepsilon_{k,p}$, multiplizieren damit die linke und rechte Seite und erhalten folglich eine skalierte Messmatrix W_s :

$$\mathbf{W}_s = \begin{pmatrix} \lambda_{1,1} u_{1,1} & \cdots & \lambda_{1,P} u_{1,P} \\ \vdots & & \vdots \\ \lambda_{K,1} u_{K,1} & \cdots & \lambda_{K,P} u_{K,P} \\ \lambda_{1,1} v_{1,1} & \cdots & \lambda_{1,P} v_{1,P} \\ \vdots & & \vdots \\ \lambda_{K,1} v_{K,1} & \cdots & \lambda_{K,P} v_{K,P} \end{pmatrix}. \quad (15.48)$$

Die eigentliche Messmatrix \mathbf{W} nennen wir jetzt projektive Messmatrix, diese liegt vor und wurde aus den K Bildern der K Kameras durch Lösen des Referenzproblems der P Punkte ermittelt. Nun setzt man zu Beginn einfach $\varepsilon_{k,p} = 0$ und führt mit der Schwachen Perspektive die Faktorisierung durch. Dabei werden die äußeren Parameter und die Koordinaten aller 3D-Punkte berechnet. Diese sind zunächst nicht richtig, da die Schwache Perspektive nur eine Approximation der Perspektive darstellt. Bei der Faktorisierung berechnen wir direkt nur $\mathbf{r}_{k,1}$ und $\mathbf{r}_{k,2}$ der Rotationsmatrix. Wir wissen aber, dass $\mathbf{r}_{k,3}$ senkrecht auf diesen beiden steht und ergänzen diesen Vektor. Wir erhalten aber generell bei der Faktorisierung zwei gespiegelte Lösungen. Da wir nun alle 3D-Punkte und die Kameraparameter haben, bilden wir nun mit den perspektivischen Abbildungsgleichungen alle 3D-Punkte mit allen K Kameras ab und erhalten eine „noch etwas falsche“ projektive Messmatrix \mathbf{W}' . Dies ist das allgemeine Prinzip der Rückprojektion. Nun können wir den Grad der „Falschheit“ bestimmen, wir bilden mit der Frobeniusnorm den Fehler $\|\mathbf{W} - \mathbf{W}'\|$. Ist dieser klein genug, sind wir fertig, wobei wir jetzt entscheiden können, welche von den beiden

spiegelsymmetrischen Lösungen wir nehmen. Ansonsten verfahren wir wie folgt: Mit den perspektivischen Abbildungsgleichungen haben wir jetzt konkrete $\varepsilon_{k,p}$ berechnet. Mit diesen können wir jetzt aus \mathbf{W}' die skalierte, projektive Messmatrix \mathbf{W}'_s berechnen. Mit dieser führen wir wieder die Faktorisierung durch und wiederholen den gesamten Prozess.

15.3 Kalibrierung mit der E-Matrix

Wir wollen nun die Eigenschaften der E-Matrix, siehe Abschn. 14.4.3, zur Kalibrierung der äußeren Kameraparameter nutzen.

Annahme Wir haben zwei Kameras mit bereits bestimmten, inneren Kameraparametern. Wir wollen durch Triangulation die Oberfläche eines Objektes bestimmen, wobei die Größe und Lage des 3D-Objektes in einem Weltkoordinatensystem völlig egal ist. Dann ist die Bestimmung der äußeren Parameter recht einfach: Wir legen in die erste Kamera das Weltkoordinatensystem, damit sind die 6 äußeren Parameter dieser Kamera bekannt. Wir bestimmen aus den zwei 2D-Bildern mindestens 5 Punktereferenzen, bestimmen daraus die E-Matrix und durch Zerlegung der E-Matrix die Rotationsmatrix und den Translationsvektor. Diese beschreiben die Bewegung der zweiten Kamera gegenüber der ersten Kamera, in der das Welkoordinatensystem liegt. Durch Normierung des Translationsvektors auf die Länge Eins (die Länge können wir nicht bestimmen) haben wir die Skalierung festgelegt, die ja absolut nicht bestimmbar ist. Damit sind wir mit der Kalibrierung schon fertig.

15.4 Kameraparameter aus der Projektionsmatrix

Manchmal besteht der Wunsch, aus den Kalibrierdaten explizit die inneren und/oder äußeren Kameraparameter zu berechnen. Denken wir bloß einmal an die E-Matrix, wenn wir mit einer Kamera die Bewegung eines Objektes bestimmen wollen. Dann brauchen wir die Referenzpunkte mit normalisierten Koordinaten, d. h. wir brauchen die innneren Kameraparameter, also die Matrix K . Wir erinnern, die Abbildungsgleichungen sind:

$$\hat{\mathbf{u}} = \mathbf{Ax} + \mathbf{a}, \quad \mathbf{A} = \mathbf{KR}, \quad \mathbf{a} = \mathbf{Kt} \rightarrow \mathbf{R} = \mathbf{K}^{-1}\mathbf{A}, \quad \mathbf{t} = \mathbf{K}^{-1}\mathbf{a}. \quad (15.49)$$

Wenn wir also \mathbf{A} bestimmt haben, wird \mathbf{A} mittels der sogenannten RQ-Dekomposition zerlegt. Oft steht in Softwaresystemen aber nur die QR-Zerlegung zur Verfügung. Ein kleiner Trick hilft. Wir invertieren die Matrix \mathbf{A} , zerlegen sie mittels der QR-Dekomposition, und invertieren dann wieder \mathbf{K} , bzw. transponieren die Orthogonalmatrix \mathbf{R} . Eine obere Dreiecksmatrix invertiert ergibt wieder eine obere Dreiecksmatrix. (\mathbf{K} ist eine obere Dreiecksmatrix und \mathbf{R} eine Rotationsmatrix). Mit bekanntem \mathbf{a} wird dann die Translation zu $\mathbf{t} = \mathbf{K}^{-1}\mathbf{a}$ bestimmt.

Wir können auch völlig anders vorgehen. Wir bestimmen die inneren Parameter direkt aus der Projektionsmatrix \mathbf{A} mit Hilfe der Kamerainvarianten, siehe Abschn. 14.1.11. Nach (14.51) bestimmen wir den Hauptpunkt (u_0, v_0) . Nach (14.53) die beiden Skalierungsfaktoren α_u, α_v und schließlich mit Hilfe von (14.52) den Parameter $\alpha_s = \alpha_v \cdot \cot \varphi$. Wenn wir nun noch zusätzlich die äußeren Kameraparameter berechnen wollen, dann benutzen wir aus (15.49) nicht die Beziehung $\mathbf{R} = \mathbf{K}^{-1}\mathbf{A}$, da wir auf Grund von Fehlern nicht garantieren können, dass \mathbf{R} auch wirklich eine Rotationsmatrix ist. Wir benutzen stattdessen die Beziehung $\mathbf{A} = \mathbf{KR}$ und rechnen analog der Idee bei der E-Matrix nach (14.182) die Rotationsmatrix \mathbf{R} mit einer LSE-Schätzung aus. Zur Lösung dieses Ausgleichsproblems nutzen wir wieder Quaternionen entsprechend Abschn. 13.2.11.

15.5 Verzeichnungen

Oft gibt es Weitwinkelobjektive, die nicht ideal dem zentralperspektivischen Modell einer Lochkamera genügen. Man sieht am Rande des Bildes „krumme“ Geraden, d. h. Geraden gehen eigentlich zentralprojektiv in Geraden über, sind aber im Bild keine Geraden, sondern gekrümmte Linien. Auffällig ist weiterhin, dass im Bildinnern die Verzeichnungen nicht so stark wie am Bildrand sind. In der Nähe, wo die optische Achse das Bild schneidet, dürften so gut wie keine Verzeichnungen zu sehen sein. Man unterscheidet „tonnenförmige“ oder „kissenförmige“ Verzeichnungen der Objektive, siehe Abb. 15.7. In der Abb. 15.6a ist eine tonnenförmige Verzeichnung zu sehen, in der Abb. 15.6b ist diese korrigiert worden. Es gibt auch Bilder von einem extremen Objektiv, dem sogenannten Fischauge-Objektiv. Diese erlauben Sichtwinkel bis 190 Grad und erzeugen extreme Verzeichnungen, die aber von vornherein gewollt sind. Normale Weitwinkelobjektive haben natürlich nicht solch extreme Verzeichnungen. Insbesondere wenn man Bilder zu Messaufgaben benutzen will, muss man diese entzerren, d. h. die Verzeichnungen wieder korrigieren. Man kann dazu das ganze Bild transformieren (damit es auch „schön“ aussieht), oder nur die Punkte in einem Rechenprozess entzerren, die man benötigt. Dazu dient in der Regel ein

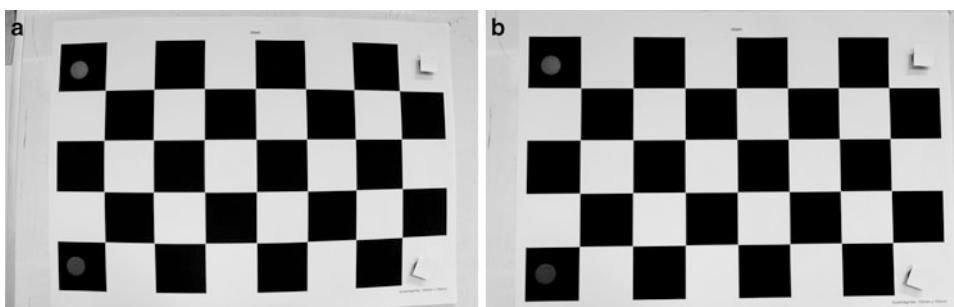
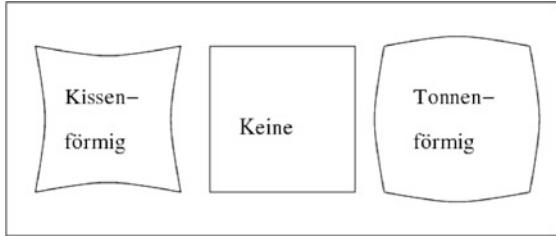


Abb. 15.6 **a** Verzeichnungen, **b** Verzeichnungen korrigiert

Abb. 15.7 Verzeichnungsarten

Modell für radialsymmetrische Verzeichnungen, das oft ausreicht für eine korrekte Entzerrung. Erweiterungen dieses Modells sind dann die tangentialsymmetrischen Verzeichnungen. Wir gehen also von einem Symmetriezentrum (X, Y) aus, das in der Nähe des Bildhauptpunktes liegen wird, aber nicht notwendig mit diesem übereinstimmen braucht. Bei „unbearbeiteten“ Bildern ist der Bildmittelpunkt eine brauchbare erste Näherung. In dieses Symmetriezentrum setzen wir nun den Koordinatenursprung und gehen zu Polarkoordinaten über. Dann soll unabhängig vom Winkel φ die Verzeichnung modellierbar sein, also radialsymmetrisch sein. Wir bezeichnen nun mit r' den verzeichneten Punkt und mit r den unverzeichneten Punkt. Dann ist folglich unabhängig vom Winkel φ die Verzeichnungsfunktion $r' = f(r)$ zu finden. Diese können wir parametrisch vorgeben und mit einem LSE-Schätzer die Parameter bestimmen. Wie man sich leicht überlegen kann, ist das aber nicht so einfach, wir benötigen dazu eine Liste von Referenzpunkten (r', r) . Woher sollen wir aber wissen, welcher unverzeichnete Punkt r zum gewählten verzeichneten Punkt r' gehört? Zunächst müssen wir klären, welche Modelfunktionen $r' = f(r)$ können wir verwenden? Üblich sind die vier Modelle:

$$\begin{aligned} r' &= r \cdot (1 + a_2 r^2 + a_4 r^4) \\ r' &= r / (1 + a_2 r^2 + a_4 r^4) \\ r &= r' \cdot (1 + a_2 r'^2 + a_4 r'^4) \\ r &= r' / (1 + a_2 r'^2 + a_4 r'^4). \end{aligned} \tag{15.50}$$

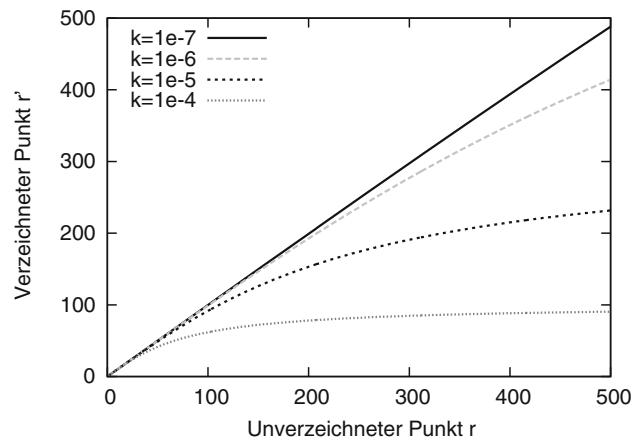
Natürlich könnten noch weitere Reihenkoeffizienten aufgenommen werden. Die Funktion $f(r)$ bzw. $f^{-1}(r')$ sollte monoton steigend sein und im Symmetriezentrum die Ableitung $\lim_{r \rightarrow 0} \frac{df(r)}{dr} = 1$ haben. Betrachten wir der Einfachheit halber ein Modell mit nur einem Parameter, da wir in diesem Falle (Lösen einer quadratischen Gleichung) auch die inverse Abbildung in analytischer Form angeben können:

$$r = \frac{r'}{1 + kr'^2} \leftrightarrow r' = \frac{1}{2kr} \left(1 - \sqrt{1 - 4kr^2} \right). \tag{15.51}$$

Da in diesem Modell für eine typische, tonnenförmige radiale Verzeichnung $k < 0$ ist ($k > 0$ für radiale, kissenförmige Verzeichnungen), verwenden wir formal:

$$r = \frac{r'}{1 - kr'^2} \leftrightarrow r' = \frac{1}{2kr} \left(\sqrt{1 + 4kr^2} - 1 \right), \quad k > 0. \tag{15.52}$$

Abb. 15.8 Verzeichnungsfunktion $r' = f(r)$ nach (15.52)



In der Abb. 15.8 ist der Graph dieser Funktion (in der Form $r' = f(r)$) zu sehen. Punkte weit außen werden also stärker ans Symmetriezentrum „herangerückt“ als Punkte im Inneren. Damit werden Geraden in Richtung Symmetriezentrum gekrümmmt. Lassen wir einmal in diesem Modell $r \rightarrow \infty$ gehen, so sehen wir, dass dann $r' = \frac{1}{\sqrt{k}}$ ist. Alle unendlich fernen Punkte liegen auf einem endlichen Kreis mit dem angegebenen Radius und der Mittelpunkt ist das Symmetriezentrum. Da r und r' beginnend bei Null stetig größer werden, ist der Begriff der Umkehrfunktion nur in den Intervallen

$$0 \leq r \leq \infty, \quad 0 \leq r' < \frac{1}{\sqrt{k}} \quad (15.53)$$

überhaupt sinnvoll. Wir lassen also die entsprechende Variable von Null beginnend so lange monoton wachsen, bis eine Polstelle auftritt.

Unabhängig von unserem Modell transformieren sich generell die Koordinaten nach dem Strahlensatz, d. h.

$$\frac{x - X}{r} = \frac{x' - X}{r'}, \quad \frac{y - Y}{r} = \frac{y' - Y}{r'} \quad (15.54)$$

In dieser Gleichung kann man nach den gewünschten Variablen umstellen. Wir setzen jetzt einmal gedanklich den Koordinatenursprung in das Symmetriezentrum X, Y und stellen nach x, y um:

$$x = \frac{x'}{r'} r, \quad y = \frac{y'}{r'} r. \quad (15.55)$$

Wir betrachten nun im Bild eine beliebige Gerade der Form $ax + by + c = 0$. Wir substituieren für x und y in der Geradengleichung die Beziehungen (15.55) und benutzen für r die Transformation (15.52). Wir erhalten

$$c(1 - kr'^2) + ax' + by' = 0 \quad (15.56)$$

und sehen sofort, dass dies die Gleichung eines Kreises in den verzeichneten Koordinaten ist, wobei der Mittelpunkt im allgemeinen nicht mehr das Symmetriezentrum ist. Obiges spezielles Modell bewirkt also, dass Geraden im Bild in Kreise übergehen. Man sieht nun leicht, dass beim zweiparametrischen Modell vierte Potenzen eingehen und damit keine Kreise entstehen können. Das Modell, dass also Geraden in Kreise übergehen, ist somit das simpelste Verzeichnungsmodell. Was würde sich generell ändern, wenn wir in Modell (15.52) r und r' vertauschen? Dann ist:

$$r' = \frac{r}{1 + kr^2} \leftrightarrow r = \frac{1}{2kr'} \left(1 - \sqrt{1 - 4kr'^2} \right), \quad k > 0. \quad (15.57)$$

Gehen wir also von diesem „Bruch“ aus, können wir die inverse Funktion auch analytisch aufschreiben. Aber: Geraden gehen nicht mehr in Kreise über. Nehmen wir nun aber ein anderes einparametrisches Modell, z. B.

$$r' = r(1 - kr^2), \quad k > 0 \quad (15.58)$$

oder

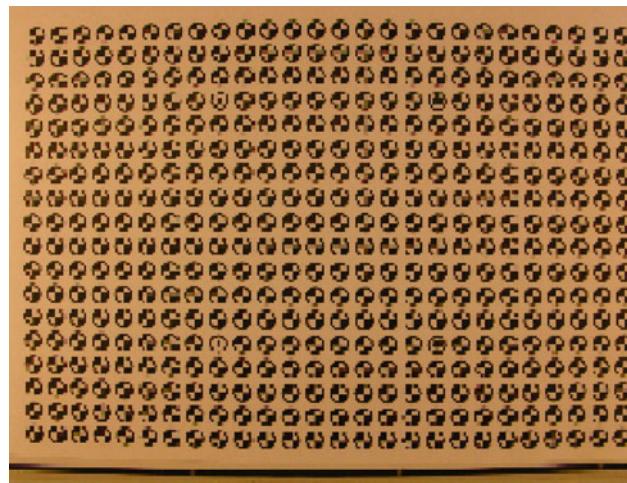
$$r = r'(1 + kr'^2), \quad k > 0, \quad (15.59)$$

so hat die Verzeichnungsfunktion einen ähnlichen Graph wie die obigen, aber die Umkehrfunktion können wir nicht mehr so leicht aufschreiben, da dies ein Polynom 3. Grades darstellt. Numerisch stellt dies aber kein Problem dar. Die Umkehrfunktion bezieht sich wieder auf das „erste Monotoniesegment“. Der wesentliche Unterschied zu den anderen Modellen ist auch der Definitionsbereich, d. h. beide Variablen laufen von Null bis Unendlich. Wie man leicht sieht, gehen Geraden auch nicht in Kreise über, sondern in ein Polynom 3. Grades.

Wie kann man nun die beiden Parameter bestimmen? Natürlich durch Kalibrierung. Wenn wir eine Kamera kalibrieren, nehmen wir diese Verzeichnungen mit ins Modell auf und bestimmen diese. Oft braucht man aber nicht die Kameraparameter, sondern will nur das Bild entzerren, dann ist der $3D \rightarrow 2D$ Kalibrierungsprozess zu aufwendig. Wir kalibrieren dann in der Ebene. Wir konstruieren uns einen „ebenen Kalibrierkörper“, d. h. ein Blatt Papier, auf dem ein Gitter oder ähnliches aufgedruckt ist. Wir kennen die Abstände der Gitterpunkte und legen damit die Koordinaten fest. Dieses Papier nehmen wir mit einer Kamera auf und erhalten das verzeichnete Gitter. Nun bestimmen wir die verzeichneten Gitterpunkte im Bild und ordnen diese den „eigentlichen“ Gitterpunkten zu, wir müssen also das Referenzproblem geschickt lösen (bei der 3D-Kalibrierung müssen wir dies ja auch tun). In der Abb. 15.9 ist die Verzeichnung eines 2D-Kalibriermusters deutlich zu sehen. Die „körpereigenen“ Punkte der Objekte können automatisch zugeordnet werden. Wenn man das Referenzproblem gelöst hat, haben wir eine Liste von Punktreferenzen

$$\mathbf{x}_i \leftrightarrow \tilde{\mathbf{x}}_i, \quad i = 1, \dots, n \quad (15.60)$$

Abb. 15.9 Aufgenommenes 2D-Kalibermuster



mit Punkten \mathbf{x}_i im planaren Kalibermuster und den zugeordneten Punkten $\tilde{\mathbf{x}}_i$ im verzeichneten Bild. Nun formulieren wir die planaren Abbildungsgleichungen. Zunächst werden die Punkte \mathbf{x}_i durch eine Homographie in Punkte \mathbf{x}'_i abgebildet. Anschließend werden die Punkte \mathbf{x}'_i durch ein Verzeichnungsmodell in die Punkte \mathbf{x}''_i abgebildet. Nun bestimmen wir aus

$$\sum_i |\mathbf{x}''_i - \tilde{\mathbf{x}}_i|^2 \rightarrow \text{Minimum} \quad (15.61)$$

die Abbildungsparameter, d. h. die Homographie und die Verzeichnungsparameter. Zur Entzeichnung irgendeines aufgenommenen Bildes benötigen wir aber nur noch die Verzeichnungsparameter.

Wenn wir für alle Brennweiten einer Kamera jeweils die Verzeichnungsparameter kalibriert haben und uns in einer Datei merken, können wir später jedes aufgenommene Bild auch entzeichnen, vorausgesetzt, wir wissen mit welcher Brennweite das Bild aufgenommen wurde.

Wenn man nachträglich Bilder entzeichnen möchte ohne die Kamera oder die Verzeichnungsparameter zu kennen, könnte man Methoden der *blind dedistortion* einsetzen. Dazu benötigen wir aber einiges A-priori-Wissen über die Szene im Bild. Wenn z. B. Geraden aufgenommen wurde, sind diese durch die Verzeichnungen als „krumme“ Linien im Bild zu sehen. Man muss nun alle „krummen“ Linien im Bild detektieren, die eigentlich Geraden sein müssten. Nun fitten wir zunächst an die „krummen“ Linien Kreise. Im nächsten Schritt setzen wir ein Verzeichnungsmodell so an, dass die Kreise zu Geraden entzeichnet werden und bestimmen mit einem LSE-Schätzer die Verzeichnungsparameter.

Oft liegen von 3D-Objekten nur die Projektionen vor, d. h. bez. eines bestimmten Abbildungsmodells (z. B. orthografische Projektion, Zentralprojektion usw.) haben wir ein 2D-Bild zur Verfügung. Das kann ein Bild von einer Kamera sein, dann sprechen wir von monokularer Rekonstruktion. Das können Bilder von zwei Kameras und mehreren Kameras sein, dann haben wir ein Stereo-Rekonstruktionsproblem zu lösen. Weiterhin gibt es viele Rekonstruktionsziele: Man kann die absolute Lage des 3D-Objektes in einem Weltkoordinatensystem berechnen oder man berechnet nur die Oberfläche des 3D-Objektes, unabhängig von der Raumlage. Manchmal werden sogar nur einige Maße des 3D-Objektes benötigt und nicht seine ganze Oberfläche. In der Medizin werden häufig Tomographiebilder benutzt, dies sind in der Regel vollständige 3D-Grauwert- oder sogar Farbbilder. Diese werden wiederum aus 2D-Grauwertbildern berechnet, wobei die 2D-Grauwertbilder als Projektionen der 3D-Bilder aufzufassen sind.

Unabhängig von der konkreten Aufgabe ist auf jeden Fall ein inverses Problem zu lösen und damit i. Allg. ein schwieriges Problem.

16.1 Rekonstruktion aus Projektionen

Eine bekannte Projektionseigenschaft der Fouriertransformation (FT) kann in der medizinischen Bildverarbeitung, insbesondere bei der CT (*Computer Tomography*) ausgenutzt werden. Bei der CT werden senkrecht zu einer Bildebene parallele Röntgenstrahlen durch das Gewebe geschickt und deren Absorption gemessen. Das theoretische Modell dazu nennt man Radontransformation, siehe Abschn. 10.8.5. Aus vielen solchen zueinander rotierten Bildebenen soll das eigentliche 3D-Bild des Organs erstellt werden. Leider besteht ein solches 2D-Bild aber nur aus Projektionen der Grauwerte des 3D-Bildes und zwar senkrecht zur Bildebene. Unter Projektion verstehen wir hier die Aufsummierung

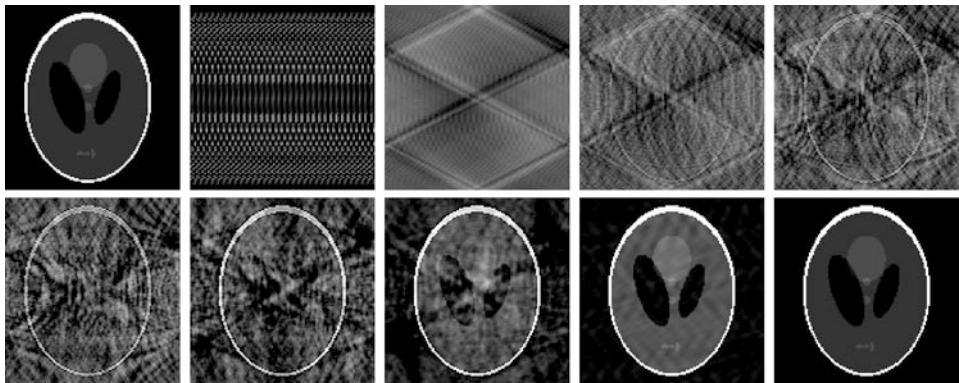


Abb. 16.1 Rekonstruktion eines 2D-Bildes (oben links) aus einzelnen 1D-Projektionen. Die Ergebnisbilder wurden anhand von 1, 4, 16, 32, 64, 128, 256, 512 und 1024 Projektionen erstellt. Das Eingabebild ist das *head phantom* von [66]

der Grauwerte längs eines Röntgenstrahles. Nun soll aber aus diesen Projektionen auf die eigentliche 3D-Information zurückgerechnet werden und zwar auf die Grauwerte eines 3D-Bildes, wie ist das möglich? Exakt mathematisch ist das natürlich nicht möglich, aber näherungsweise und für praktische Anwendungen ausreichend ist es doch möglich.

Der Einfachheit halber reduzieren wir zunächst die Dimension, d. h. der zu rekonstruierende 3D-Raum wird zum 2D-Raum, die Projektionsebenen werden zu Projektionsgeraden.

Rekonstruktion nach dem Fourier-Slice-Theorem Dazu nehmen wir einmal als simpelste Projektionsgerade die x -Achse selbst, dann lautet die Projektionsfunktion bezüglich der zu rekonstruierenden 2D-Funktion $f(x, y)$:

$$g(x) = \int_{-\infty}^{+\infty} f(x, y) dy. \quad (16.1)$$

Falls $f(x, y)$ wie in der Stochastik eine Dichtefunktion wäre, dann ist $g(x)$ die sogenannte Randdichte. Diese Funktion $g(x)$ kann also durch die Röntgenstrahlen gemessen werden. Die Funktion $f(x, y)$ soll aber aus vielen solchen Funktionen $g(x)$ bezüglich der rotierten x -Achse berechnet werden. Dazu betrachten wir einmal theoretisch das 2D-Spektrum der Funktion $f(x, y)$ nach (3.59):

$$\alpha_f(v_1, v_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-2\pi i(xv_1 + yv_2)} dx dy. \quad (16.2)$$

Nun betrachten wir auch die v_1 -Achse, aber nicht die Projektionen auf diese Achse, sondern wir „schneiden“ aus dem 2D-Spektrum diese v_1 -Achse mit den darüberliegenden Spektralwerten als „gedachte Scheibe (*slice*)“ aus und erhalten damit das eigentliche *Fourier-Slice-Theorem*:

$$\begin{aligned} slice(v_1) &= \alpha_f(v_1, 0) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-2\pi i x v_1} dx dy \\ &= \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} f(x, y) dy \right] e^{-2\pi i x v_1} dx = \int_{-\infty}^{+\infty} g(x) e^{-2\pi i x v_1} dx = \alpha_g(v_1). \end{aligned} \quad (16.3)$$

Wir können also tatsächlich aus dem 1D-Spektrum $\alpha_g(v_1)$ der Projektionsfunktion $g(x)$ eine „echte“ Scheibe des tatsächlichen 2D-Spektrums von $f(x, y)$ berechnen. Um das vollständige 2D-Spektrum zu berechnen brauchen wir aber viele solcher Scheiben. Dazu rotieren wir einmal die Funktion $f(x, y)$, dann wissen wir von (4.203), dass sich das 2D-Spektrum genauso rotiert. Statt die Funktion $f(x, y)$ zu rotieren, ist es aber gedanklich dasselbe, wir rotieren die x -Achse und betrachten die Projektionen senkrecht zu dieser Geraden. Da sich das Spektrum mitrotiert hatte, können wir genauso die Scheibe der entsprechend rotierten v_1 -Achse ausrechnen. Diesen Sachverhalt drückt man am besten in Polarkoordinaten aus. Im Abschn. 10.8.5 wurde die Projektionsfunktion mit $I(p, \theta)$ bezeichnet. Wenn wir betonen wollen, dass wir diese für einen festen Winkel betrachten, schreiben wir $I_\theta(p)$. Daher bezeichnen wir das Spektrum mit $\alpha_\theta(r)$. Wenn wir auch das 2D-Spektrum von f in Polarkoordinaten r, θ ausdrücken, dann gilt nach dem Fourier-Slice-Theorem $\alpha^{polar}(r, \theta) = \alpha_\theta(r)$.

Zusammenfassung

- Man bilde die Projektionsfunktion $I(p, \theta)$ senkrecht zu einer beliebig orientierten Projektionsgeraden, d. h. senkrecht zur Projektionsgeraden werden alle Grauwerte längs der senkrechten Projektionsstrahlen addiert und der Projektionsgeraden an diesem Punkte zugeordnet, siehe Abschn. 10.8.5.
- Man bildet das 1D-Spektrum dieser Projektionsfunktion. Dieses 1D-Spektrum ist gleichzeitig der Ausschnitt (*slice*) aus dem 2D-Spektrum längs einer Geraden, die die gleiche Orientierung hat wie die Projektionsgerade, allerdings muss diese durch den Koordinatenursprung gehen.
- Man erzeuge nun „viele“ *slices* des 2D-Spektrums, interpoliere und transformiere zurück mit der inversen 2D-FT. Damit ist die Funktion $f(x, y)$ näherungsweise berechnet.

Man versucht folglich, das 2D-Spektrum durch eine Vielzahl von 1D-Spektren zu approximieren um es dann zurückzutransformieren. Eine andere Idee hängt auch direkt mit dem Fourier-Slice-Theorem zusammen und nennt sich *gefilterte Rückprojektion*.

Rekonstruktion mit der gefilterten Rückprojektion Wir gehen einmal vom 2D Spektrum $\alpha(v_1, v_2)$ der Funktion $f(x, y)$ aus (obwohl es nicht gegeben ist):

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \alpha(v_1, v_2) e^{+2\pi i(v_1 x + v_2 y)} dv_1 dv_2. \quad (16.4)$$

Wir gehen zu Polarkoordinaten mit $v_1 = r \cos \theta$, $v_2 = r \sin \theta$, $dv_1 dv_2 = rdr d\theta$ über, folglich wird (16.4) zu:

$$\begin{aligned} f(x, y) &= \int_0^{2\pi} \int_0^{\infty} \alpha^{\text{polar}}(r, \theta) e^{+2\pi i r(x \cos \theta + y \sin \theta)} rdr d\theta \\ &= \int_0^{\pi} \int_{-\infty}^{\infty} \alpha^{\text{polar}}(r, \theta) e^{+2\pi i r(x \cos \theta + y \sin \theta)} |r| dr d\theta. \end{aligned} \quad (16.5)$$

Wir setzen $p = x \cos \theta + y \sin \theta$ und mit

$$g_{\theta}(p) = \int_{-\infty}^{\infty} \alpha^{\text{polar}}(r, \theta) \cdot |r| e^{+2\pi i rp} dr = \int_{-\infty}^{\infty} \alpha_{\theta}(r) \cdot |r| e^{+2\pi i rp} dr \quad (16.6)$$

ist:

$$f(x, y) = \int_0^{\pi} g_{\theta}(p) d\theta = \int_0^{\pi} g_{\theta}(x \cos \theta + y \sin \theta) d\theta. \quad (16.7)$$

In (16.6) wird das Spektrum $\alpha_{\theta}(r)$ der 1D-Projektionsfunktion mit $|r|$ multipliziert. Dies bedeutet nach dem Faltungstheorem eine LSI-Filterung der Projektionsfunktion $I_{\theta}(p)$ mit der Funktion $\alpha_{|r|}^{-1}(v)$. Mit dieser Funktion werden tiefe Frequenzen abgeschwächt und hohe verstärkt. Falls die Projektionen verrauscht sind, wird damit das Rauschen verstärkt. Folglich haben einige Autoren vorgeschlagen, Filter zu benutzen, bei denen die hohen Frequenzen etwas abgeschwächt werden. Diese werden benannt nach Shepp und Logan bzw. Ramachandran und Lakshminarayanan. Formel (16.7) bedeutet praktisch für die Rekonstruktion:

Man wählt ein Pixel (x, y) , dann wird der Grauwert $f(x, y)$ durch Summation aller gefilterten Projektionen $g_{\theta}(p)$ über alle Richtungen θ berechnet.

Die Methode der gefilterten Rückprojektionen lässt auch eine anschauliche Interpretation zu. Wir gehen einmal von den ungefilterten Projektionen $I(p, \theta)$ aus. In der Ebene, die mit dem Grauwert Null initialisiert wurde, wird der Wert I für alle (x, y) längs der Geraden $p = x \cos \theta + y \sin \theta$ akkumuliert, siehe auch Abschn. 10.8. Wenn wir dies für alle p und θ getan haben, schauen wir uns die Ebene als „Bild“ an und erkennen die Originalfunktion $f(x, y)$, allerdings sehr „verschmiert“. Diese Verschmierung bewirkt eine

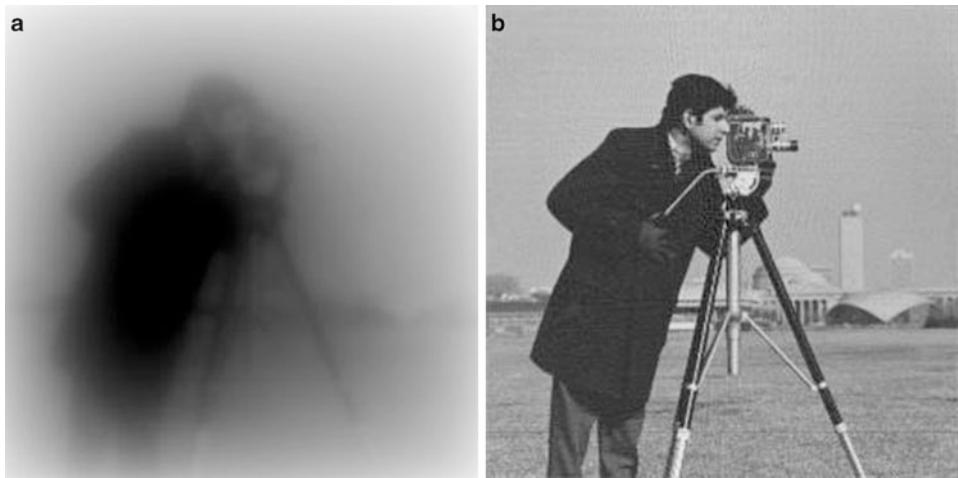


Abb. 16.2 a Ungefilterte, b Gefilterte Rückprojektion

Faltung von $f(x, y)$ mit einer Funktion $g(x, y)$. Wir müssen folglich anschließend mit einer inversen Faltung die „Verschmierung“ aufheben. Dies können wir aber bereits schon mit den Projektionen tun, und nicht erst zum Schluss. Die Filterung mit den Projektionen stellt gewissermaßen eine inverse Faltung dar. Die Abb. 16.2 bezieht sich auf das Originalbild 10.21a und dessen Radontransformierte in Abb. 10.21b. In Abb. 16.2a ist die ungefilterte Rückprojektion zu sehen, d. h. aus den Projektionen wird das Original ausschließlich durch Akkumulation erzeugt. In Abb. 16.2b ist die gefilterte Rückprojektion zu sehen, die eine nahezu perfekte Restaurierung des Originalbildes 10.21a darstellt.

16.2 Rekonstruktion der 3D-Struktur von polyedrischen Objekten

Ohne Kalibrierung von einer oder mehreren Kameras ist natürlich ein Objekt in einem Weltkoordinatensystem nicht zu rekonstruieren. Man kann aber die 3D-Struktur eines Objektes ermitteln, d. h. wir bestimmen 3D-Koordinaten des Objektes, diese sind aber nur eindeutig bis auf eine $3D \leftrightarrow 3D$ Abbildung. Diese Abbildung ergibt sich aus dem Kameramodell. Alle Kameramodelle setzen sich aus $3D \leftrightarrow 3D$ Modellen mit einer anschließenden orthografischen Projektion zusammen.

16.2.1 Weak perspective camera

Die $3D \leftrightarrow 3D$ Transformation besteht jetzt aus einer Ähnlichkeitstransformation. Wir nehmen einmal an, das 3D-Objekt bestehe nur aus vier nichtkomplanaren Punkten

$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Wir eliminieren die Translation, indem wir den Koordinatenursprung in den Punkt \mathbf{x}_0 legen. Es bezeichne \mathbf{B} die *Gramsche Matrix*:

$$\mathbf{B} = \begin{pmatrix} \mathbf{x}_1^T \mathbf{x}_1 & \mathbf{x}_1^T \mathbf{x}_2 & \mathbf{x}_1^T \mathbf{x}_3 \\ \mathbf{x}_2^T \mathbf{x}_1 & \mathbf{x}_2^T \mathbf{x}_2 & \mathbf{x}_2^T \mathbf{x}_3 \\ \mathbf{x}_3^T \mathbf{x}_1 & \mathbf{x}_3^T \mathbf{x}_2 & \mathbf{x}_3^T \mathbf{x}_3 \end{pmatrix}. \quad (16.8)$$

Diese Matrix ist trivialerweise *invariant* gegenüber Euklidischen Bewegungen, da sich Längen und Winkel nicht ändern. Es seien $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ die Abbilder der vier 3D-Punkte in der 2D-Bildebene. Wir setzen ebenso den Koordinatenursprung in den Punkt \mathbf{y}_0 und betrachten nur noch die restlichen drei Punkte. Es sei nun $\mathbf{u}^T = (y_{11}, y_{21}, y_{31})$ der Vektor der x -Koordinaten der drei 2D-Punkte und $\mathbf{v}^T = (y_{12}, y_{22}, y_{32})$ der Vektor der y -Koordinaten der drei 2D-Punkte. Nun müssen wir die Abbildung beschreiben, dazu bezeichne $\mathbf{R} = (r_{ij})$ die 3D-Rotationsmatrix und s den Skalierungsfaktor. Weiterhin bezeichne:

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \rightarrow \mathbf{B} = \mathbf{X}^T \mathbf{X}, \mathbf{r}_i^T = (r_{i1}, r_{i2}, r_{i3}), \quad i = 1, 2, 3. \quad (16.9)$$

Dann können wir die beiden Abbildungsgleichungen auch in der Form:

$$\mathbf{u}^T = s \cdot \mathbf{r}_1^T \mathbf{X}, \quad \mathbf{v}^T = s \cdot \mathbf{r}_2^T \mathbf{X} \quad (16.10)$$

schreiben. Daher gilt:

$$\mathbf{r}_1^T = \frac{1}{s} \mathbf{u}^T \mathbf{X}^{-1}, \quad \mathbf{r}_2^T = \frac{1}{s} \mathbf{v}^T \mathbf{X}^{-1}. \quad (16.11)$$

Jetzt nutzen wir die beiden Orthonormalitätsrelationen $\mathbf{r}_1^T \mathbf{r}_2 = 0$, $\mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2$ aus und erhalten mit der trivialen Hilfsbeziehung $\mathbf{X}^{-1} (\mathbf{X}^{-1})^T = (\mathbf{X}^T \mathbf{X})^{-1}$ die beiden Gleichungen:

$$\begin{aligned} \mathbf{u}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{v} &= 0 \\ \mathbf{u}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{u} - \mathbf{v}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{v} &= 0. \end{aligned} \quad (16.12)$$

Man sieht, in beide Gleichungen geht die inverse Gramsche Matrix ein. Die zwei Gleichungen aus (16.12) bestimmen die sechs Unbekannten der Gramschen Matrix (eigentlich nur fünf, da wir ein homogenes System vorliegen haben). Daher benötigen wir drei Bilder, dann hätten wir sechs Gleichungen mit den sechs Unbekannten und können die Gramsche Matrix \mathbf{B} bis auf einen Skalierungsfaktor eindeutig berechnen. Dies bedeutet natürlich, dass wir bei mehreren Bildern genau die Referenzbeziehungen zwischen den Punkten kennen müssen. Ist es nun möglich aus der berechneten Gramschen Matrix \mathbf{B} die drei 3D-Originalpunkte $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ zu berechnen? Da die Gramsche Matrix \mathbf{B} invariant gegenüber Euklidischen Bewegungen ist, kann die Lösung nicht eindeutig sein, sondern wir berechnen irgendeine, die aber zu jeder anderen, die aus dieser mittels Euklidischer Bewegungen

hervorgeht, äquivalent ist. Als „irgendeine“ Lösung (wir müssen sozusagen die „Wurzel“ aus einer Matrix $\mathbf{B} = \mathbf{X}^T \mathbf{X}$ ziehen, gegeben ist \mathbf{B} , gesucht ist \mathbf{X}) wählen wir diejenige aus, die wir mit dem Cholesky-Verfahren zur Lösung linearer Gleichungssysteme erhalten. Diese Lösung ist eindeutig und besitzt die Gestalt einer oberen Dreiecksmatrix. Damit ist nun jede Lösung gleichberechtigt, die aus einer Ähnlichkeitstransformation des rekonstruierten Dreibeins hervorgeht, denn die Gramsche Matrix selbst war nur eindeutig bis auf einen Skalierungsfaktor berechenbar. Mit drei Aufnahmen haben wir folglich (bis auf Skalierungen) die Euklidische Struktur des 3D-Objektes berechnet, daher auch der Name *structure of motion*. Mit der Berechnung der Matrix \mathbf{X} haben folglich ein Dreibein (eine Basis) berechnet. Jeden weiteren Punkt \mathbf{x}_i des 3D-Objektes können wir nun leicht berechnen. Wir stellen diese Punkte bezüglich der errechneten Basis \mathbf{X} dar, also:

$$\mathbf{x}_i = (\mathbf{b}_i)_1 \mathbf{x}_1 + (\mathbf{b}_i)_2 \mathbf{x}_2 + (\mathbf{b}_i)_3 \mathbf{x}_3 = \mathbf{X} \mathbf{b}_i. \quad (16.13)$$

Bilden wir nun den Punkt \mathbf{x}_i ab, so erhalten wir die gleichen Beziehungen für die 2D-Punkte:

$$\mathbf{y}_i = (\mathbf{b}_i)_1 \mathbf{y}_1 + (\mathbf{b}_i)_2 \mathbf{y}_2 + (\mathbf{b}_i)_3 \mathbf{y}_3. \quad (16.14)$$

Allerdings benötigen wir jetzt zwei Bilder für den Punkt \mathbf{x}_i , da obige Beziehung nur zwei Gleichungen für drei Unbekannte darstellt. Damit können die Koordinaten eines jeden Punktes des 3D-Objektes bezüglich der Basis errechnet werden. Wir erinnern aber nochmals daran, zur Basisbestimmung waren drei Bilder notwendig. Damit ist die 3D-Struktur des Objektes bestimmt und diese ist bis auf $3D \leftrightarrow 3D$ Ähnlichkeitstransformationen eindeutig bestimmt.

16.2.2 Affine Kamera

Die $3D \leftrightarrow 3D$ Abbildung besteht jetzt aus einer affinen Transformation. Die Grundidee geht auf J.J. Koenderink und A.J. van Doorn zurück. Wir nutzen dazu die Kenntnis, dass die Koordinaten bezüglich einer Basis affin invariant sind, siehe Abschn. 14.3.1. Im 3D-Raum sei ein Dreibein, also vier Punkte als Basis gegeben. Damit lässt sich jeder weitere Punkt mit dieser Basis darstellen:

$$\mathbf{x}_4 = \mathbf{x}_0 + \lambda_1(\mathbf{x}_1 - \mathbf{x}_0) + \lambda_2(\mathbf{x}_2 - \mathbf{x}_0) + \lambda_3(\mathbf{x}_3 - \mathbf{x}_0). \quad (16.15)$$

Unterwerfen wir diese Punkte einer $3D \leftrightarrow 3D$ affinen Transformation, so gilt:

$$\mathbf{x}'_4 = \mathbf{x}'_0 + \lambda_1(\mathbf{x}'_1 - \mathbf{x}'_0) + \lambda_2(\mathbf{x}'_2 - \mathbf{x}'_0) + \lambda_3(\mathbf{x}'_3 - \mathbf{x}'_0). \quad (16.16)$$

Daher gilt diese Beziehung trivialerweise auch für die Bildpunkte in der Bildebene:

$$\mathbf{y}_4 = \mathbf{y}_0 + \lambda_1(\mathbf{y}_1 - \mathbf{y}_0) + \lambda_2(\mathbf{y}_2 - \mathbf{y}_0) + \lambda_3(\mathbf{y}_3 - \mathbf{y}_0). \quad (16.17)$$

Die Beziehung (16.17) stellt zwei Gleichungen mit drei Unbekannten dar, daher brauchen wir zwei Bilder mit den zuhörigen fünf Referenzpunkten und können die Koordinaten eines jeden Punktes bezüglich der Basis ausrechnen. Der wesentliche Unterschied zum vorigen Modell besteht aber darin, dass wir die 3D-Basis überhaupt nicht mehr bestimmen können, denn vier Punkte im Raum und deren Referenzpunkte bestimmen eindeutig eine affine Transformation. Daher geben wir uns ein beliebiges Dreibein des uns unbekannten 3D-Objektes vor (vier Punkte) und können dann die nach (16.17) berechneten Koordinaten bezüglich dieses Dreibeins nutzen. Damit haben wir ein spezielles Objekt aus einer affinen Äquivalenzklasse gewählt, folglich die affine Struktur rekonstruiert.

16.2.3 Pinhole camera

Die $3D \leftrightarrow 3D$ Abbildung besteht jetzt aus einer projektiven Transformation. Da eine projektive Transformation im Raum durch fünf Quell- und Zielpunkte eindeutig bestimmt ist, müssen wir uns fünf Punkte vorgeben um dann einen Repräsentanten aus der projektiven Äquivalenzklasse zu rekonstruieren, um damit die projektive Struktur zu beschreiben. Die Idee dazu geht auf Faugeras (1992) zurück. Dazu gehen wir im Folgenden zu homogenen Koordinaten über und beschreiben das Modell 7 durch die Abbildung:

$$\tilde{\mathbf{u}} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}. \quad (16.18)$$

Die $\tilde{\mathbf{u}}$ sind die Bildpunkte, besitzen also drei Koordinaten, die Projektionsmatrix $\tilde{\mathbf{A}}$ ist eine (3×4) -Matrix und die 3D-Punkte $\tilde{\mathbf{x}}$ besitzen vier Koordinaten. Im 3D-Raum ist eine $3D \leftrightarrow 3D$ projektive Transformation eindeutig durch fünf Quell- und Zielpunkte (wovon keine vier komplanar sein dürfen) beschrieben. Daher seien im 3D-Raum jetzt fünf (Quell-)Punkte $\tilde{\mathbf{x}}_i$, $i = 1, 2, 3, 4, 5$ gegeben. Die Zielpunkte sind leider nur in der Bildebene gegeben, dafür nehmen wir aber an, dass zwei Bilder vorliegen und die Referenzen zwischen den Bildpunkten bekannt sind, diese Bildpunkte bezeichnen wir mit $\tilde{\mathbf{u}}_i, \tilde{\mathbf{u}}'_i$, $i = 1, 2, 3, 4, 5$. Die Idee besteht nun darin, die Projektionsmatrizen beider Kameras durch Kalibrierung zu bestimmen, um dann jeden Punkt durch Triangulation rekonstruieren zu können. Da die Kalibrierung nur die projektive 3D-Struktur festlegt, können wir beliebige fünf Punkte festlegen. Wir wählen dazu die sogenannte projektive Standardbasis, also $\tilde{\mathbf{x}}_1 = (1, 0, 0, 0)^T, \tilde{\mathbf{x}}_2 = (0, 1, 0, 0)^T, \tilde{\mathbf{x}}_3 = (0, 0, 1, 0)^T, \tilde{\mathbf{x}}_4 = (0, 0, 0, 1)^T, \tilde{\mathbf{x}}_5 = (1, 1, 1, 1)^T$. In den beiden Bildebenen benutzen wir ebenfalls die projektiven Standardbasen, diese bestehen aber jetzt nur aus den jeweils ersten vier Punkten, da wir eine Dimension weniger haben. Aus den fünf Punkten gilt es nun die beiden Projektionmatrizen zu berechnen, was wir formal nur für eine tun.

Kalibrierung durch fünf Punkte und die Epipole In homogenen Koordinaten gilt nun für die ersten vier Punkte:

$$\rho_i \tilde{\mathbf{u}}_i = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_i, \quad i = 1, 2, 3, 4. \quad (16.19)$$

Durch die Wahl der projektiven Standardbasis für die Quell- und Zielpunkte lässt sich sofort die Struktur von $\tilde{\mathbf{A}}$ durch die ersten vier Punkte ablesen:

$$\tilde{\mathbf{A}} = \begin{pmatrix} \rho_1 & 0 & 0 & \rho_4 \\ 0 & \rho_2 & 0 & \rho_4 \\ 0 & 0 & \rho_3 & \rho_4 \end{pmatrix}. \quad (16.20)$$

Nun müssen wir die Abbildung des fünften Punktes mit einbeziehen. Die Koordinaten des fünften Punktes in der Bildebene sind nicht frei wählbar, sie sind durch die Basis eindeutig bestimmt, wir bezeichnen sie mit $\tilde{\mathbf{u}}_5 = (\alpha, \beta, \gamma)^T$. Die Abbildung:

$$\rho_5 \tilde{\mathbf{u}}_5 = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_5 \quad (16.21)$$

ergibt die Beziehungen

$$\rho_5 \alpha = \rho_1 + \rho_4, \quad \rho_5 \beta = \rho_2 + \rho_4, \quad \rho_5 \gamma = \rho_3 + \rho_4. \quad (16.22)$$

Wir ersetzen formal $\mu = \rho_5$, $\nu = \rho_4$ und erhalten die triviale Beziehung:

$$\tilde{\mathbf{A}} = \mu \mathbf{B} + \nu \mathbf{C} \quad (16.23)$$

mit

$$\mathbf{B} = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \quad (16.24)$$

Da wir homogene Koordinaten haben, ist die Projektionsmatrix bis auf einen Parameter (und nicht zwei) bekannt. Dies folgt auch daraus, da wir 5 Punkte zur Kalibrierung benutzen, zur vollständigen Kalibrierung aber 6 Punkte benötigt werden. Analog gilt das gleiche für die zweite Kamera, also:

$$\tilde{\mathbf{A}}' = \mu' \mathbf{B}' + \nu' \mathbf{C}. \quad (16.25)$$

Das gegebene Stereopaar ist somit bis auf zwei unbekannte Parameter kalibriert, den Parameter pro Kamera drücken wir durch die Verhältnisse

$$\lambda = \frac{\mu}{\nu}, \quad \lambda' = \frac{\mu'}{\nu'} \quad (16.26)$$

aus. Als nächstes berechnen wir die beiden Projektionszentren der beiden Kameras. Vom Kameramodell 7 wissen wir, dass sich die Zentren durch

$$\tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}_c = 0 \quad (16.27)$$

berechnen lassen. Durch einsetzen der oben hergeleiteten Form für $\tilde{\mathbf{A}}$ ergibt sich:

$$\tilde{\mathbf{x}}_c = \left(\frac{1}{v - \alpha\mu}, \frac{1}{v - \beta\mu}, \frac{1}{v - \gamma\mu}, \frac{1}{v} \right)^T \equiv \left(\frac{1}{1 - \alpha\lambda}, \frac{1}{1 - \beta\lambda}, \frac{1}{1 - \gamma\lambda}, 1 \right)^T. \quad (16.28)$$

Identische Ausdrücke erhalten wir für die zweite Kamera:

$$\begin{aligned} \tilde{\mathbf{x}}'_c &= \left(\frac{1}{v' - \alpha'\mu'}, \frac{1}{v' - \beta'\mu'}, \frac{1}{v' - \gamma'\mu'}, \frac{1}{v'} \right)^T \\ &\equiv \left(\frac{1}{1 - \alpha'\lambda'}, \frac{1}{1 - \beta'\lambda'}, \frac{1}{1 - \gamma'\lambda'}, 1 \right)^T, \end{aligned} \quad (16.29)$$

die nur noch von den Verhältnissen abhängen. Verbinden wir die beiden Projektionszentren durch eine Gerade, dann heißen die Durchstoßpunkte dieser Geraden durch beiden Bildebenen *Epipole* $\tilde{\mathbf{o}}$ und $\tilde{\mathbf{o}'}$, deren Koordinaten wir angeben wollen. Wir benutzen dazu die bisherige Struktur der Projektionematrizen und berechnen die Koordinaten von $\tilde{\mathbf{o}'}$ durch $\tilde{\mathbf{o}'} = \tilde{\mathbf{A}}'\tilde{\mathbf{x}}_c$:

$$\begin{aligned} \tilde{\mathbf{o}'} &= \left(\frac{\mu'\alpha' - v'}{v - \mu\alpha} + \frac{v'}{v}, \frac{\mu'\beta' - v'}{v - \mu\beta} + \frac{v'}{v}, \frac{\mu'\gamma' - v'}{v - \mu\gamma} + \frac{v'}{v} \right)^T \\ &\equiv \left(\frac{\lambda'\alpha' - \lambda\alpha}{1 - \lambda\alpha}, \frac{\lambda'\beta' - \lambda\beta}{1 - \lambda\beta}, \frac{\lambda'\gamma' - \lambda\gamma}{1 - \lambda\gamma} \right)^T. \end{aligned} \quad (16.30)$$

Analog ergibt sich mit $\tilde{\mathbf{o}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}'_c$:

$$\begin{aligned} \tilde{\mathbf{o}} &= \left(\frac{\mu\alpha - v}{v' - \mu'\alpha'} + \frac{v}{v'}, \frac{\mu\beta - v}{v' - \mu'\beta'} + \frac{v}{v'}, \frac{\mu\gamma - v}{v' - \mu'\gamma'} + \frac{v}{v'} \right)^T \\ &\equiv \left(\frac{\lambda\alpha - \lambda'\alpha'}{1 - \lambda'\alpha'}, \frac{\lambda\beta - \lambda'\beta'}{1 - \lambda'\beta'}, \frac{\lambda\gamma - \lambda'\gamma'}{1 - \lambda'\gamma'} \right)^T. \end{aligned} \quad (16.31)$$

Die Koordinaten der Epipole unterliegen gewissen Restriktionen aus der Epipolargeometrie. So gehen alle Epipolarlinien durch den Epipol, folglich haben wir zwei *Büsche* von Epipolarlinien. Es gibt eine $2D \rightarrow 2D$ projektive Transformation, so dass beide Büschel ineinander überführbar sind. Daher können wir das Doppelverhältnis auf die Geradenbüschel anwenden, so muss gelten:

$$[\langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_1 \rangle, \langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_2 \rangle, \langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_3 \rangle, \langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_4 \rangle] = [\langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_1 \rangle, \langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_2 \rangle, \langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_3 \rangle, \langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_4 \rangle] \quad (16.32)$$

$$[\langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_1 \rangle, \langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_2 \rangle, \langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_3 \rangle, \langle \tilde{\mathbf{o}}, \tilde{\mathbf{u}}_5 \rangle] = [\langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_1 \rangle, \langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_2 \rangle, \langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_3 \rangle, \langle \tilde{\mathbf{o}}', \tilde{\mathbf{u}}'_5 \rangle]. \quad (16.33)$$

(Die spitzen Klammern sollen nur die Gerade durch zwei Punkte bezeichnen, kein Skalarprodukt.) Wir nehmen jetzt einmal an, die Koordinaten der beiden Epipole $\tilde{\mathbf{o}} = [\tilde{o}_1, \tilde{o}_2, \tilde{o}_3]^T$

und $\tilde{\mathbf{o}}' = [\tilde{o}_1', \tilde{o}_2', \tilde{o}_3']^T$ seien bekannt, dann können wir die beiden restlichen Parameter λ und λ' zur eindeutigen Bestimmung der beiden Projektionsmatrizen daraus ausrechnen. Auf Grund der Doppelverhältnisse erhalten wir nur zwei unabhängige Gleichungen anstatt eigentlich vier, aus den Koordinaten für $\tilde{\mathbf{o}}'$ erhalten wir z. B.:

$$\frac{\tilde{o}_1'}{\tilde{o}_3'} = \frac{\lambda\alpha - \lambda'\alpha'}{\lambda\gamma - \lambda'\gamma'} \cdot \frac{\lambda\gamma - 1}{\lambda\alpha - 1}, \quad \frac{\tilde{o}_2'}{\tilde{o}_3'} = \frac{\lambda\beta - \lambda'\beta'}{\lambda\gamma - \lambda'\gamma'} \cdot \frac{\lambda\gamma - 1}{\lambda\beta - 1}. \quad (16.34)$$

Diese beiden Gleichungen stellen je für sich Kurven zweiter Ordnung dar und die gesuchte Lösung sind die Schnittpunkte dieser beider Kurven. Es gibt aber nur eine Lösung, die mit der Epipolargeometrie verträglich ist:

$$\begin{aligned} \lambda &= \frac{\tilde{\mathbf{o}}'^T \cdot (\tilde{\mathbf{u}}_5 \times \tilde{\mathbf{u}}'_5)}{[\alpha\tilde{o}'_1, \beta\tilde{o}'_2, \gamma\tilde{o}'_3] \cdot (\tilde{\mathbf{u}}_5 \times \tilde{\mathbf{u}}'_5)}, \\ \lambda' &= \lambda \cdot \frac{y\tilde{o}'_1\tilde{o}'_2(\beta - \alpha) + \alpha\tilde{o}'_2\tilde{o}'_3(\gamma - \beta) + \beta\tilde{o}'_3\tilde{o}'_2(\alpha - \gamma)}{y'\tilde{o}'_1\tilde{o}'_2(\beta - \alpha) + \alpha'\tilde{o}'_2\tilde{o}'_3(\gamma - \beta) + \beta'\tilde{o}'_3\tilde{o}'_2(\alpha - \gamma)}. \end{aligned} \quad (16.35)$$

Wir konnten jetzt die beiden Projektionsmatrizen vollständig berechnen, so dass wir eigentlich aus jedem weiteren Referenzpaar den 3D-Punkt rekonstruieren können. Dazu waren bisher 5 Referenzen notwendig und wir nahmen an, die Koordinaten der Epipole seien bekannt, worauf wir noch zu sprechen kommen. Zunächst klären wir, wie wir aus zwei Referenzen $(\tilde{\mathbf{y}}, \tilde{\mathbf{y}}')$ den 3D-Punkt in unserem projektiven Koordinatensystem rekonstruieren können. Dazu bezeichne $\tilde{\mathbf{x}}_\infty$ den Schnittpunkt des Sehstrahles $\langle \tilde{\mathbf{y}}, \tilde{\mathbf{x}}_c \rangle$ mit der uneigentlichen Ebene, daher muss gelten:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_\infty \quad \text{bzw.} \quad \tilde{\mathbf{y}} = \mathbf{A}\mathbf{x}_\infty, \quad \mathbf{x}_\infty = (\mathbf{x}_\infty, 0)^T. \quad (16.36)$$

(\mathbf{A} ist die (3×3) linke Untermatrix von $\tilde{\mathbf{A}}$.) Der zu rekonstruierende 3D-Punkt $\tilde{\mathbf{x}}$ erfüllt dann

$$\tilde{\mathbf{x}} = \xi \cdot \tilde{\mathbf{x}}_c + \eta \cdot \tilde{\mathbf{x}}_\infty, \quad (16.37)$$

woraus η und ξ zu bestimmen sind. Es gilt $\tilde{\mathbf{y}}' = \tilde{\mathbf{A}}' \cdot \tilde{\mathbf{x}}$ und daher:

$$\tilde{\mathbf{y}}' = \xi \cdot \tilde{\mathbf{o}}' + \eta \mathbf{A}' \mathbf{A}^{-1} \tilde{\mathbf{y}}. \quad (16.38)$$

Aus der Struktur der Projektionsmatrizen sieht man sofort, dass

$$\mathbf{A}' \mathbf{A}^{-1} = \begin{pmatrix} \frac{\alpha'\lambda'-1}{\alpha\lambda-1} & 0 & 0 \\ 0 & \frac{\beta'\lambda'-1}{\beta\lambda-1} & 0 \\ 0 & 0 & \frac{\gamma'\lambda'-1}{\gamma\lambda-1} \end{pmatrix}. \quad (16.39)$$

Damit haben wir zur Bestimmung von η und ξ 3 Gleichungen mit 2 Unbekannten gegeben und können diese ausrechnen. Nun können wir den rekonstruierten Punkt darstellen zu:

$$\tilde{\mathbf{x}} = \xi \left(\frac{1}{\lambda\alpha - 1}, \frac{1}{\lambda\beta - 1}, \frac{1}{\lambda\gamma - 1}, -1 \right)^T + \eta \left(\frac{\tilde{y}_1}{\lambda\alpha - 1}, \frac{\tilde{y}_2}{\lambda\beta - 1}, \frac{\tilde{y}_3}{\lambda\gamma - 1}, 0 \right)^T. \quad (16.40)$$

Bestimmung der Epipole Im Folgenden soll erläutert werden, wie man die Koordinaten der Epipole bestimmen kann. Mit den Bezeichnungen für die Projektionsmatrizen:

$$\tilde{\mathbf{A}} = (\mathbf{A}, \mathbf{a}), \quad \tilde{\mathbf{A}}' = (\mathbf{A}', \mathbf{a}') \quad (16.41)$$

ergeben sich die Projektionszentren zu:

$$\tilde{\mathbf{x}}_c = \begin{pmatrix} \mathbf{A}^{-1}\mathbf{a} \\ -1 \end{pmatrix}, \quad \tilde{\mathbf{x}}'_c = \begin{pmatrix} \mathbf{A}'^{-1}\mathbf{a}' \\ -1 \end{pmatrix}. \quad (16.42)$$

Damit ergibt sich der Epipol $\tilde{\mathbf{o}}'$ zu:

$$\tilde{\mathbf{o}}' = \tilde{\mathbf{A}}'\tilde{\mathbf{x}}_c = \mathbf{A}' \cdot \begin{pmatrix} \mathbf{A}^{-1}\mathbf{a} \\ -1 \end{pmatrix} = \mathbf{A}'\mathbf{A}^{-1}\mathbf{a} - \mathbf{a}'. \quad (16.43)$$

Mit $\tilde{\mathbf{y}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_\infty$ bzw. $\tilde{\mathbf{y}} = \mathbf{A}\mathbf{x}_\infty$ ist $\tilde{\mathbf{y}}'_\infty = \tilde{\mathbf{A}}'\tilde{\mathbf{x}}_\infty = \mathbf{A}'\mathbf{A}^{-1}\tilde{\mathbf{y}}$. Die Epipolarlinie bez. $\tilde{\mathbf{y}}$ geht also durch die zwei Punkte $\tilde{\mathbf{o}}'$ und $\tilde{\mathbf{y}}'_\infty$. In homogenen Koordinaten ist eine Gerade durch zwei Punkte charakterisiert durch das Kreuzprodukt, also $ep_{\tilde{\mathbf{y}}} = \tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}'_\infty$. Der linearen Operation \times können wir einen linearen Operator, also eine Matrix \mathbf{O}' , zuordnen, für die gilt:

$$\mathbf{O}' \cdot \mathbf{x} = \tilde{\mathbf{o}}' \times \mathbf{x} \quad \forall \mathbf{x}. \quad (16.44)$$

Es ist trivial, dass die antisymmetrische Matrix die Form:

$$\mathbf{O}' = \begin{pmatrix} 0 & -o'_3 & +o'_2 \\ +o'_3 & 0 & -o'_1 \\ -o'_2 & +o'_1 & 0 \end{pmatrix} \quad (16.45)$$

hat. Folglich gilt:

$$ep_{\tilde{\mathbf{y}}} = \tilde{\mathbf{o}}' \times \tilde{\mathbf{y}}'_\infty = \mathbf{O}' \cdot \tilde{\mathbf{y}}'_\infty = \mathbf{O}'\mathbf{A}'\mathbf{A}^{-1}\tilde{\mathbf{y}} = \mathbf{F}\tilde{\mathbf{y}}. \quad (16.46)$$

Die Matrix \mathbf{F} heißt *Fundamentalmatrix*. Für jeden Punkt $\tilde{\mathbf{y}}_{ep}$ auf der Epipolarlinie $ep_{\tilde{\mathbf{y}}}$ gilt dann:

$$\tilde{\mathbf{y}}_{ep}^T \mathbf{F} \tilde{\mathbf{y}} = 0, \quad (16.47)$$

also auch für den Referenzpunkt \tilde{y}' selbst. Diese Gleichung entspricht der sogenannten *Longuet-Higgins* Gleichung aus der „motion analysis“. Da die Matrix nur von den Projektionsmatrizen und den Epipolen abhängt, gilt diese Beziehung für alle Referenzpunktpaare und ist unabhängig von der Wahl der Referenzen. Die Matrix besteht aus neun Unbekannten, also eigentlich nur aus acht, daher benötigen wir wenigstens acht Referenzen, um aus diesen die Matrix F auszurechnen. Die Epipolarlinie $ep_{\tilde{y}}'$ entartet genau dann, wenn $ep_{\tilde{y}}'$ der Nullvektor ist, dies ist aber nur möglich, wenn für den Originalpunkt $\tilde{y} = \tilde{o}$ gilt. Daher können wir nun den Epipol \tilde{o} aus der Beziehung $F \cdot \tilde{o} = \mathbf{0}$ (Nullvektor) berechnen. Ähnlich berechen wir nun die Koordinaten des Epipols \tilde{o}' .

Zusammenfassung Mit wenigstens 8 Punktereferenzen können wir alle Informationen berechnen, die wir für die projektive Rekonstruktion benötigen. Fünf von den acht wählen wir als projektive Basis aus (es muss aber garantiert werden, dass keine 4 von diesen 5 im Raum komplanar sind), dann können die restlichen drei Punkte und jeder weitere Punkt, für den eine Korrespondenz vorliegt, bezüglich der projektiven Basis rekonstruiert werden.

Bemerkung Die acht Referenzen sind aus praktischen Gründen nötig, eigentlich benötigen wir nur sieben Referenzen, denn:

Aus den fünf gegebenen Punkten könnten wir die beiden Projektionsmatrizen berechnen bis auf die beiden Parameter λ und λ' . Da wir auch die Epipole in Abhängigkeit von diesen beiden Parametern ausgedrückt haben, können wir nun letztendlich die Fundamentalmatrix F von diesen beiden Parametern ausdrücken, es gilt dann:

$$F = O' A' A^{-1} = \begin{pmatrix} 0 & -\frac{\beta' \lambda' - 1}{\beta \lambda - 1} \cdot \frac{\gamma' \lambda' - \gamma \lambda}{\gamma \lambda - 1} & \frac{\gamma' \lambda' - 1}{\gamma \lambda - 1} \cdot \frac{\beta' \lambda' - \beta \lambda}{\beta \lambda - 1} \\ \frac{\alpha' \lambda' - 1}{\alpha \lambda - 1} \cdot \frac{\gamma' \lambda' - \gamma \lambda}{\gamma \lambda - 1} & 0 & -\frac{\gamma' \lambda' - 1}{\gamma \lambda - 1} \cdot \frac{\alpha' \lambda' - \alpha \lambda}{\alpha \lambda - 1} \\ -\frac{\alpha' \lambda' - 1}{\alpha \lambda - 1} \cdot \frac{\beta' \lambda' - \beta \lambda}{\beta \lambda - 1} & \frac{\beta' \lambda' - 1}{\beta \lambda - 1} \cdot \frac{\alpha' \lambda' - \alpha \lambda}{\alpha \lambda - 1} & 0 \end{pmatrix}. \quad (16.48)$$

Benutzen wir nun direkt diese Struktur der Matrix, so haben wir nur zwei Unbekannte und benötigen daher nur zwei Referenzen zur Bestimmung dieser beiden Parameter, dies ist wie bei der Bestimmung aus den Epipolen ein nichtlineares Gleichungssystem mit zwei Unbekannten. Allerdings dürfen wir nicht zwei von den fünf gegebenen Punkten benutzen, da wir diese ja schon zur Bestimmung der Struktur dieser Matrix benutzt haben, daher müssen wir uns zusätzlich zwei Referenzen vorgeben, also insgesamt sieben Referenzen, dies ist die eigentliche minimale Information.

Was ist nun eigentlich zu tun?

- Berechnung bzw. Festlegung aller n Punktereferenzen in beiden Bildern, markieren von 4 bzw. 5 ausgezeichneten Punktereferenzen. Diese 5 ausgezeichneten Punkte können nicht trianguliert werden.

- b) Die ausgezeichneten 4 Punkte in jedem Bild werden auf die projektive Standardbasis abgebildet und daraus eine $2D \rightarrow 2D$ projektive Transformation berechnet. Mit dieser werden alle Punkte transformiert und die neuen Koordinaten berechnet. Dies geschieht für jedes Bild getrennt.
- c) Berechnung der Fundamentalmatrix \mathbf{F} mit wenigstens 8 Punktereferenzen, wobei die ausgezeichneten 5 Punkte verwendet werden dürfen!
- d) Berechnung wenigstens eines Epipoles durch $\mathbf{F}\mathbf{\tilde{o}} = \mathbf{0}$.
- e) Berechnung der beiden Unbekannten λ und λ' . Damit sind die beiden Projektionsmatrizen bestimmt.
- f) Triangulation aller Punktpaare (bis auf die 5...) bezüglich der 3D-projektiven Standardbasis.
- g) Durch A-priori-Wissen über das 3D-Objekt: Bestimmung einer $3D \rightarrow 3D$ projektiven Transformation zur maßstabsgerechten Darstellung des 3D-Objektes.

16.3 Triangulation von Raumpunkten

Gegeben seien $k = 1, 2, \dots, K$ Kameras. Die Abbildungsgleichungen waren:

$$\mathbf{\tilde{u}}_k = \mathbf{A}_k \mathbf{x} + \mathbf{a} \quad (16.49)$$

und damit sind die Sehstrahlen durch $\mathbf{t}_k = \mathbf{A}_k^{-1} \mathbf{\tilde{u}}_k$ bestimmt. Aus den K Bildpunkten $\mathbf{\tilde{u}}_k$ der K Kameras ist durch Triangulation der Raumpunkt \mathbf{p} zu bestimmen, dessen Abbilder $\mathbf{\tilde{u}}_k$ in den K Kameras zu sehen sind. Die Sehstrahlen werden sich im Allgemeinen nicht in einem Punkt schneiden, da stets Fehler in einer realen Kamera auftreten. Wir betrachten dazu das entsprechende Problem in der Ebene, wo sich im Gegensatz dazu Sehstrahlen immer schneiden, siehe Abb. 16.3.

Daher muss ein „ausgeglichener“ Raumpunkt ermittelt werden. Dazu sind zwei Strategien üblich. Zunächst bestimmen wir den Abstandsvektor \mathbf{d}_k eines Punktes \mathbf{p} von einem Sehstrahl $\mathbf{x}_c + \lambda_k \mathbf{s}_k$ mit einem normierten Richtungsvektor $\mathbf{s}_k = \frac{\mathbf{t}_k}{\|\mathbf{t}_k\|}$ zu:

$$\mathbf{d}_k = (\mathbf{p} - \mathbf{x}_c) \times \mathbf{s}_k. \quad (16.50)$$

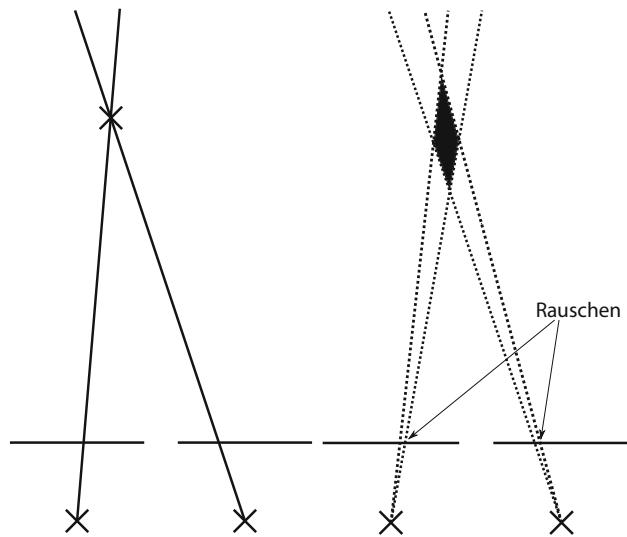
Dieser Vektor hat die Komponenten

$$\begin{aligned} d_{kx} &= p_y s_{kz} - p_z s_{ky} - (x_c)_{ky} s_{kz} + (x_c)_{kz} s_{ky} \\ d_{ky} &= p_z s_{kx} - p_x s_{kz} - (x_c)_{kz} s_{kx} + (x_c)_{kx} s_{kz} \\ d_{kz} &= p_x s_{ky} - p_y s_{kx} - (x_c)_{kx} s_{ky} + (x_c)_{ky} s_{kx}. \end{aligned} \quad (16.51)$$

Nun bestimmen wir den Raumpunkt \mathbf{p} derart, dass

$$\sum_{k=1}^K (d_k)^2_x + (d_k)^2_y + (d_k)^2_z \rightarrow \text{Minimum} \quad (16.52)$$

Abb. 16.3 a Normale Triangulation, b Triangulation mit Fehlern



erfüllt ist. Dies ist ein lineares Ausgleichsproblem, wir können daher einfach die Gaußschen Normalengleichungen aufstellen und den Lösungsvektor ausrechnen. Diese Methode nennt man in der Photogrammetrie *Vorwärtsschnitt*.

Relativ einfach wird der Vorwärtsschnitt, wenn wir nur zwei Kameras zur Verfügung haben, dann berechnen wir den Raumpunkt

$$\mathbf{p} = \frac{(\mathbf{x}_c)_1 + \lambda_1 \mathbf{t}_1 + (\mathbf{x}_c)_2 + \lambda_2 \mathbf{t}_2}{2}, \quad (16.53)$$

wobei die beiden Parameter λ_1 und λ_2 aus der Beziehung

$$d = (\mathbf{r}_1 - \mathbf{r}_2)^2 = ((\mathbf{x}_c)_1 + \lambda_1 \mathbf{t}_1 - (\mathbf{x}_c)_2 - \lambda_2 \mathbf{t}_2)^2 \rightarrow \text{Minimum} \quad (16.54)$$

bestimmt werden. Die Gaußschen Normalengleichungen sind recht simpel:

$$\begin{pmatrix} \mathbf{t}_1^T \mathbf{t}_1 & -\mathbf{t}_1^T \mathbf{t}_2 \\ -\mathbf{t}_1^T \mathbf{t}_2 & \mathbf{t}_2^T \mathbf{t}_2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} -((\mathbf{x}_c)_1 - (\mathbf{x}_c)_2)^T \mathbf{t}_1 \\ ((\mathbf{x}_c)_1 - (\mathbf{x}_c)_2)^T \mathbf{t}_2 \end{pmatrix}. \quad (16.55)$$

Ein völlig anderes Fehlermaß zur Bestimmung eines „ausgeglichenen“ Raumpunktes ist der sogenannte *Rückwärtsschnitt*. Dabei suchen wir denjenigen Raumpunkt \mathbf{p} , dessen Bildpunkte

$$\tilde{\mathbf{u}}_k^{(p)} = \mathbf{A}_k \mathbf{p} + \mathbf{a}_k \quad (16.56)$$

mit den Beobachtungen $\tilde{\mathbf{u}}_k$ „möglichst gut“ übereinstimmen, d.h in homogenen Koordinaten

$$S(\mathbf{p}) = \sum_{k=1}^K (\tilde{\mathbf{u}}_k^{(p)} - \tilde{\mathbf{u}}_k)^2 \rightarrow \text{Minimum.} \quad (16.57)$$

Nun, dies ist ein schönes Beispiel dafür, wann homogene Koordinaten nicht verwendet werden dürfen. Beide Vektoren sind nur eindeutig bis auf Faktoren, daher können wir beliebige Differenzenvektoren erzeugen und die Aufgabe ist nicht korrekt gestellt. Daher sind wir hier gezwungen zu kartesischen Koordinaten überzugehen. Es bezeichne $\xi_k^{(p)} = \frac{(\tilde{\mathbf{u}}_k)_1^{(p)}}{(\tilde{\mathbf{u}}_k)_3^{(p)}}$, $\xi_k = \frac{(\tilde{\mathbf{u}}_k)_1}{(\tilde{\mathbf{u}}_k)_3}$, $\eta_k^{(p)} = \frac{(\tilde{\mathbf{u}}_k)_2^{(p)}}{(\tilde{\mathbf{u}}_k)_3^{(p)}}$, $\eta_k = \frac{(\tilde{\mathbf{u}}_k)_2}{(\tilde{\mathbf{u}}_k)_3}$, dann lösen wir das Ausgleichsproblem

$$S(\mathbf{p}) = \sum_{k=1}^K (\xi_k^{(p)} - \xi_k)^2 + (\eta_k^{(p)} - \eta_k)^2 \rightarrow \text{Minimum.} \quad (16.58)$$

Da die Abhängigkeit von \mathbf{p} in Form von Quotienten gegeben sind, haben wir für den *Rückwärtsschnitt* ein nichtlineares Ausgleichsproblem zu lösen. Wir können das System aber wie bei der Kamerakalibrierung linearisieren:

$$\begin{aligned} S(\mathbf{p}) &= \sum_{k=1}^K \left(\xi_k (\mathbf{a}_3^T \mathbf{p} + a_{30}) - (\mathbf{a}_1^T \mathbf{p} + a_{10}) \right)^2 \\ &\quad + \sum_{k=1}^K \left(\eta_k (\mathbf{a}_3^T \mathbf{p} + a_{30}) - (\mathbf{a}_2^T \mathbf{p} + a_{20}) \right)^2 \rightarrow \text{Minimum.} \end{aligned} \quad (16.59)$$

Dies ist nun wieder ein lineares Ausgleichsproblem, welches mit den Gaußschen Normalengleichungen gelöst werden kann.

16.4 Bestimmung der Punktkorrespondenzen

Bisher sind wir immer von der Voraussetzung ausgegangen, dass die Punktkorrespondenzen vorliegen. Diese müssen aber erst bestimmt werden und zwar stabil, mit geringen Fehlern und möglichst kleinen Fehlzuordnungen. Dies kann mit oder ohne Hilfsmitteln erfolgen. Hilfsmittel können z. B. Marker oder Projektoren sein, die statistische Muster, Texturen oder ähnliches auf die Oberfläche projizieren. Mit diesen projizierten Mustern ist es dann einfacher, die Referenzpunkte zu finden. Nicht immer sind Hilfsmittel einsetzbar, dann müssen die korrespondierenden Punkte, Linien oder Objekte direkt in den beiden Bildern gefunden werden. Es ist offensichtlich, dass die Punkte in der Umgebung eine „gewisse“ Struktur haben müssen, ansonsten können wir keine Merkmale berechnen, die der korrespondierende Punkt auch besitzt. Die nächste Frage schließt sich gleich an: welche Umgebung des Punktes wählt man zur Berechnung der Merkmale? In Abb. 16.4 sind die

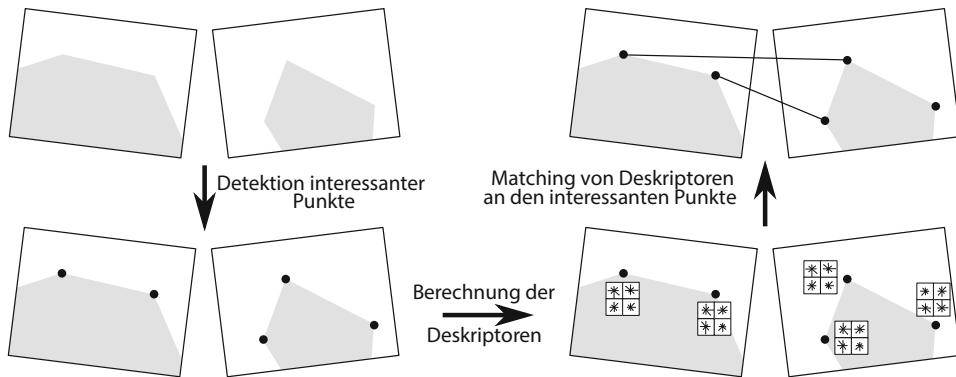


Abb. 16.4 Punktkorrespondenzen – Prinzipskizze

interessanten Punkte die Eckpunkte von Polygonen. Von diesen Eckpunkten müssen nun in beiden Bildern geeignete Merkmale, Deskriptoren genannt, bestimmt werden. Wie bestimmt man nun die interessanten Punkte (Detektor) und welche Merkmale (Deskriptor) berechnet man? Einige Möglichkeiten werden im Abschn. 19.9 ausführlich besprochen, als typische Vertreter seien z. B. die SIFT-Merkmale genannt. Bei den Merkmalen ist natürlich zu beachten, dass sie und die Umgebung kovariant bezüglich einer geeigneten geometrischen Transformation sein sollten. Näherungsweise reichen dazu Ähnlichkeitstransformationen aus. Diese Kovarianz bezüglich Ähnlichkeitstransformationen und der Umgebung garantieren z. B. die SIFT-Merkmale. Liegen z. B. Punkte auf einer Geraden im Raum, dann kann man auf den Geraden im Bild die Invariante *Doppelverhältnis* für auf der Geraden detektierte Punkte nutzen. Liegen dagegen Punkte in einer 3D-Ebene, dann existiert eine Homographie zwischen beiden Bildern im Bereich der abgebildeten Ebene. Diese Eigenschaft kann man natürlich geschickt ausnutzen und zur Korrespondenzfindung *projektive Flächeninvarianten* nutzen. Bezuglich der nutzbaren Invarianten siehe die Ausführungen im Abschn. 14.3.1.

Wenn für die Korrespondenzfindung die Kameras kalibriert sind, dann haben wir immer eine Grundinformation zur Verfügung: ein Punkt bestimmt in dem anderen Bild eine *Epipolarlinie*. Der korrespondierende Punkt muss immer auf der *Epipolarlinie* liegen, siehe dazu die Epipolareometrie im Abschn. 14.4. Damit können wir aber den Suchraum drastisch beschränken.

Nach der Anwendung von Detektoren und Deskriptoren erhalten wir in der Regel für jedes Bild eine Liste mit detektierten Punkten und deren Deskriptoren. Nun gilt es, Punktkorrespondenzen aus beiden Listen zu bestimmen, d. h. die eigentliche Zuordnung der Punkte vorzunehmen. Dazu werden von den beiden Listen Matchingmaße berechnet, siehe Abschn. 19.8.1. Damit beschreibt irgendein Matchingmaß $d_{i,j}$ die Ähnlichkeit des Punktes i der ersten Liste mit dem Punkt j der zweiten Liste. Mit diesen Matchingmaßen wird nun durch einen Algorithmus die Zuordnung durchgeführt. Das kann ein einfacher nächs-

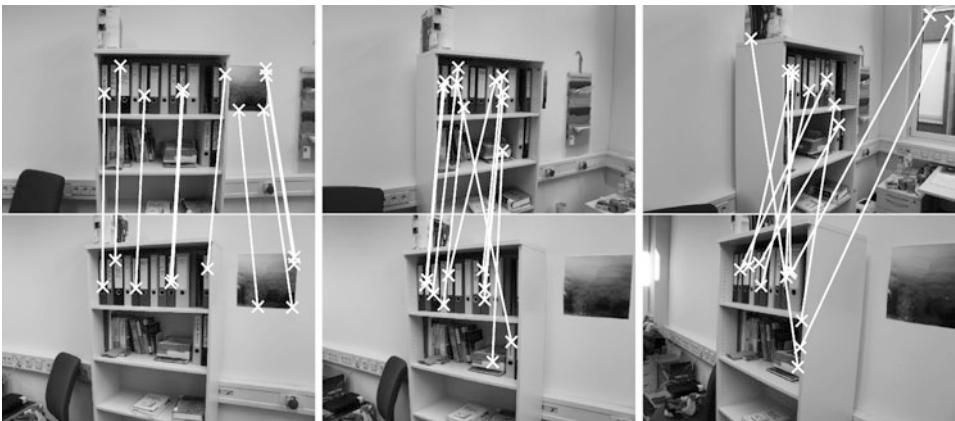


Abb. 16.5 Punktcorrespondenzen-Ausreißer: *small baseline* → *wide baseline*

ter Nachbaralgorithmus sein (z. B. ICP-Algorithmus) oder ein aufwendigerer Algorithmus, der die Zuordnungskosten minimiert (z. B. Ungarische Methode). Die folgenden praktischen Schwierigkeiten treten dabei auf:

- Punkte aus der Liste 1 treten in der Liste 2 womöglich überhaupt nicht auf, siehe dazu Abb. 16.4.
- Punkte aus der Liste 2 treten womöglich auch nicht in der Liste 1 auf.
- Die Deskriptoren sind auf Grund von Rauschen, Störungen oder Aufnahmen aus geometrisch „sensitiven Positionen“ falsch berechnet worden. Dies führt auf das Problem der Ausreißer, siehe dazu Abb. 16.5. Insbesondere wenn der Abstand der beiden Projektionszentren sehr groß ist, erhöht sich die Wahrscheinlichkeit von Fehlzuordnungen.

Oft besteht auch ein wechselseitiger Zusammenhang zwischen Punktcorrespondenzen und Kalibrierung der Kameras. Zu Beginn ermittelt man Punktcorrespondenzen und kalibriert damit die Kameras. Auf Grund der Epipolareometrie kann man dann „besser“ neue Correspondenzen bestimmen. Mit diesen kann man wiederum die Kalibrierung verbessern usw.

16.5 Eine einfache, praktische 3D-Vermessungsaufgabe

Im Zusammenhang mit Punktcorrespondenzen und der Triangulation kann man eine einfache, praktische Aufgabe sehr elegant lösen.

Planare Vermessung Zunächst ist naheliegend, ein planares Muster im Bild zu haben, deren metrische Größen man kennt. Die Mindestpunktzahl des Muster beträgt vier. Da

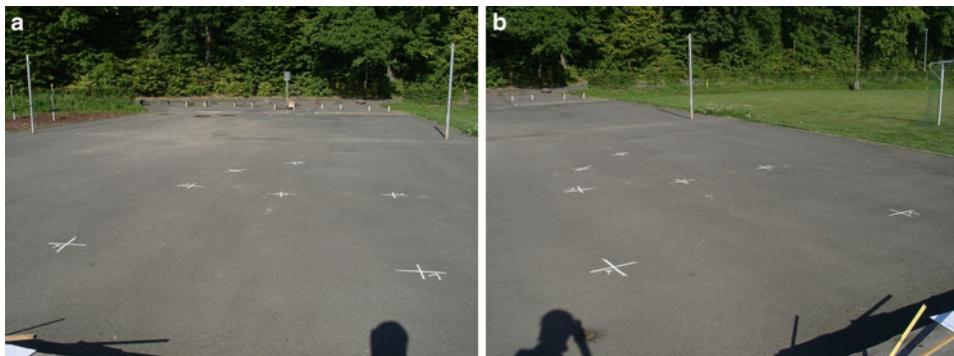


Abb. 16.6 Planares Kalibriermuster: metrisch vermessene Kreidekreuze. **a** Erste Aufnahme, **b** zweite Aufnahme

man dieses Kalibermuster einschließlich seiner metrischen Größen kennt, kann man die Messpunkte des Kalibermusters in ein Koordinatensystem eintragen. Bezuglich der Messpunkte im Koordinatensystem und den Bildkoordinaten der Messpunkte berechnet man eine Homographie. Mit dieser Homographie kann man das Bild entzerrn. Dies ist aber gar nicht nötig, es werden nur die interessierenden Meßpunkte mit der Homographie transformiert. Nun kann man alle weiteren Punkte, die in der Ebene des Kalibermusters liegen, vermessen. Man braucht sie nur mit der Homographie transformieren und die Euklidischen Abstände der transformierten Punkte ausrechnen. Dazu wird nur ein einziges Bild von der Szene benötigt. In Abb. 16.6 sind in zwei verschiedenen Aufnahmen ein Punkt-muster in einer Ebene zu sehen. Die „Ebene“ stellt der asphaltierte Platz dar. Wenn man folglich die metrischen Abstände der „Kreidekreuze“ kennt, kann man alle weiteren Punkte in der Ebene vermessen. Die Ebene bedeutet: Man kann nur innerhalb des asphaltierten Platzes vermessen und nichts weiter. Allerdings kann man die planaren Muster kaskadierten und somit durch mehrere Aufnahmen „lange planare Szenen“ vermessen. Es muss nur von Aufnahme zu Aufnahme ein genügend großer Überlappungsbereich garantiert werden. Mit dieser Methode wurden z. B. Autounfälle durch die Polizei vermessen und registriert. Die Ebene übernimmt dabei die Straße. Abstände können dann allerdings nur in der Ebene der Straße vermessen werden.

Einfache 3D-Vermessung Deshalb entsteht nun die Frage, kann man mit diesem planaren Kalibermuster auch eine Vermessung von 3D-Punkten durchführen, die nicht in einer Ebene liegen?

Die Aufgabe ist tatsächlich lösbar. Wir formulieren sie allgemein:

Mit einer einzigen einfachen, handelsüblichen Kamera sollen simple 3D-Vermessungsaufgaben durchgeführt werden, z. B. die Bestimmung von Abständen im Gelände, die Bestimmung der Größe von Personen, die Bestimmung der Höhe von Bäumen oder Gebäuden, die Vermessung von archäologischen Fundstellen, die Vermessung von Straßenszenen und vieles mehr. Dabei brauchen die zu vermessenden Punkte (Messpunkte) nicht in einer Ebene liegen.

Die Aufgabe wird in folgenden Schritten gelöst:

- Die handelsübliche Kamera sollte keine oder kaum Verzeichnungen erzeugen. Sollte dies dennoch der Fall sein, so wird im „Labor“ mit einem Kalibriergitter (siehe Abschn. 15.5) das Bild entzeichnet. Hat die Kamera mehrere Brennweiten, so wird eine Tabelle erstellt, so dass pro Brennweite die Verzeichnungsparameter eingetragen sind. Bei einer praktischen Bildaufnahme sollte aus dem Bildfile die Brennweite auslesbar sein, so dass das Bild genau genug laut Tabelle entzeichnet werden kann. Für dieses Problem wird im Internet bereits Software angeboten.
- Im „Labor“ wird nun die Kamera mit der Methode nach Zhang (siehe Abschn. 15.2.2) kalibriert. Es werden aber nur die inneren Parameter bestimmt, d. h. die Kalibriermatrix K . Hat die Kamera mehrere Brennweiten, so wird wie bei der Verzeichnung verfahren, d. h. es wird pro Brennweite kalibriert und in eine Tabelle eingetragen.
- Mit der so kalibrierten Kamera gehen wir an den Ort der Vermessung. Wir brauchen von der zu vermessenden Szene zwei Aufnahmen, um mittels korrespondierender Punkte und Triangulation die räumliche Entfernung zweier 3D-Messpunkte auszurechnen.
- Dazu müssen wir aber vor Ort die äußeren Kameraparameter kalibrieren. Dazu sind laut Abschn. 15.2.2 nur vier Punkte eines planaren Kalibermusters nötig. Wir können dazu natürliche Objekte nutzen, z. B. ein Rechteck eines Gebäudeteiles, von dem wir allerdings genau die Seitenlängen kennen müssen oder eine Bodenplatte, deren Maße wir vor Ort bestimmen müssen, ähnlich zu Abb. 16.6. Ist nichts dergleichen vorhanden, bauen wir uns vorher ein Quadrat oder Rechteck aus einfachen Materialien mit bekannten Maßen und legen dieses in die Szene.
- Von der zu vermessenden Szene mit diesem planaren Kalibermuster machen wir zwei Aufnahmen mit einer geeignet zu wählenden *baseline*.
- Nun können wir wieder ins Büro gehen und in Ruhe am Computer die Vermessung durchführen.
- Zunächst kalibrieren wir die äußeren Kameraparameter indem wir in beiden Bildern die (mindestens) vier korrespondierenden Referenzpunkte des planaren Kalibermusters bestimmen. Dies kann interaktiv geschehen mit einer gewissen Nachbehandlung im Sinne einer Genauigkeitsverbesserung.
- Jetzt kann die Kamera aus beiden Aufnahmepositionen heraus vollständig kalibriert werden. Da wir die metrischen Größen des planaren Kalibermusters kennen, so ist die Maßeinheit im Weltkoordinatensystem eindeutig festgelegt.
- Der letzte Schritt ist nun die eigentliche Vermessung. Wir bestimmen von zwei zu vermessenden 3D-Punkten interaktiv (vielleicht mit einer automatischen Nachbesserung) die korrespondierenden Punkte in beiden Bildern und können durch Triangulation die Entfernung berechnen.
- Wenn man einige Punkte vermessen hat, dann kann man das Wissen über diese korrespondierenden Punkte nutzen, um die äußere Kalibrierung weiter zu verbessern im Sinne einer höheren Genauigkeit.

- Der entscheidende Vorteil dieses simplen 3D-Vermessungsverfahrens ist, dass man vor Ort nur zwei Aufnahmen mit einem einfachen, in die Szene zu integrierenden, planaren Kalibriermuster braucht. Der „Rest“ kann alles im Büro oder nachträglich erfolgen. Man muss allerdings beachten, dass man nur diejenigen 3D-Punkte vermessen kann, die auch als korrespondierende Punkte in beiden Bildern zu sehen sind, siehe z. B. Abb. 16.6.

16.6 Aktives Sehen

Aktives Sehen bedeutet, dass man nicht nur mit zwei oder mehreren Kameras durch Triangulation die 3D-Oberfläche berechnet, sondern man hat zusätzlich noch Hilfsmittel zur Verfügung. Diese Hilfsmittel können Projektoren oder andere Beleuchtungshilfsmittel sein.

16.6.1 Projektion von Lichtstrahlen

Die einfachste Art der Triangulation mit einem Hilfsmittel geschieht dadurch, dass ein Lichtprojektor einen Lichtstrahl in die Szene projiziert. Der daraus resultierende Lichtfleck auf der Oberfläche wird von einer Kamera aufgenommen und als Bildpunkt im Bild identifiziert. Bei kalibrierter Kamera kann der Sehstrahl berechnet werden. Ist die Lage des Projektors (die Koordinaten der punktförmigen Lichtquelle) bekannt und der Richtungsvektor des Lichtstrahls auch bekannt, so kann durch Triangulation der 3D-Punkt im Weltkoordinatensystem berechnet werden. Als Lichtquelle wird häufig eine Laserdiode mit Kollimatoroptik verwendet. Pro Lichtpunkt muss ein Bild aufgenommen werden und der Lichtstrahl muss zweidimensional ablenkbar sein, was einen enormen Aufwand bedeutet.

16.6.2 Projektion mit Lichtebenen

Grundidee ist es zunächst eine „ideale“ Lichtebe zu projizieren. Im Bild „sieht“ man dann eine Linie der Lichtebe. Vom Projektor her muss nun im Weltkoordinatensystem wieder die Lage des Projektors und der Normalenvektor der Lichtebe bekannt sein. Dann kann folgendermaßen trianguliert werden:

Man sucht einen Bildpunkt auf der beleuchteten Linie im Bild und berechnet den Schnittpunkt des von diesem Punkte ausgehenden Sehstrahles und der Lichtebe. Der Schnittpunkt muss „zumindestens theoretisch“ ein Oberflächenpunkt des zu vermessenden 3D-Objektes sein. Nun verfolgen wir die beleuchtete Linie im Bild und suchen auf der Linie den nächsten Punkt, triangulieren wieder usw. bis alle Bildpunkte dieser Linie verarbeitet wurden. Somit haben wir mit einem Bild alle 3D-Punkte rekonstruiert, die mit dieser Lichtebe beleuchtet wurden. Dies tun wir nun für alle weiteren Lichtebenen

des Projektors. Wir brauchen also genauso viele Bilder wie Lichtebenen. Dieses Verfahren wird oft als Profil oder Linienmessung bezeichnet, oft auch als *Lichtschnittverfahren*.

16.6.3 Codierter Lichtansatz (Coded Light Approach)

Die Grundidee zur Beschleunigung des Lichtebenen-Ansatzes geht auf Altschuler (1979) zurück und wurde von Wahl (1984) [87] wesentlich erweitert und verbessert.

Wir wollen alle Lichtebenen gleichzeitig in das Bild projizieren und pro Bildpunkt die richtige Ebene identifizieren. Bei einer sehr geringen Anzahl von Lichtebenen gelingt das auch, dann ist aber die Auflösung zu gering. Vielleicht gelingt uns ein Kompromiss zwischen „alle Ebenen einzeln nacheinander projizieren“ und „alle Ebenen gleichzeitig projizieren“.

Da es aber technisch schwierig ist, viele Lichtebenen gleichzeitig zu projizieren und im Bild zu detektieren, gehen wir „dual“ vor: Wir projizieren helle und dunkle Streifen und die idealen Lichtebenen sind die Hell/Dunkel-Übergänge. Wir stellen uns einmal vor, ein Projektor hat ein Streifengitter, und jeder Streifen ist elektronisch ansteuerbar, kann folglich geöffnet oder geschlossen werden. Durch die Beleuchtung des Projektors (Zentralprojektion) entstehen helle oder dunkle Streifen auf der Oberfläche des 3D-Objektes, je nachdem ob ein Streifen geöffnet oder geschlossen ist, siehe Abb. 16.7. Dort sieht man im schraffierten Bereich, wie dadurch die Oberfläche hell angestrahlt wird oder dunkel eingestellt wird. Jeder Beleuchtungssektor hat damit einen binären Zustand, er ist hell oder dunkel. Wir können mit $\log_2(n)$ Mustern, d. h. mit $\log_2(n)$ Bildern für jedes einzelne Pixel entscheiden, welches von den n verschiedenen Beleuchtungssektoren (entsprechen den Gitterstreifen des Projektors) das Pixel „getroffen“ hat. Zur Einfachheit demonstrieren wir dies am Beispiel eines Projektors mit nur acht Gitterstreifen. Dazu schreiben wir die acht Binärzahlen als Spalten in eine Matrix:

$$\begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix} . \quad (16.60)$$

Da die Spalten die Zustände der Gitterstreifen des Projektors darstellen, sehen wir, dass die Dualzahlen der Spalten genau die Gitterstreifen durchnummernieren. Wir beleuchten also das Objekt dreimal, wobei jeder Gitterstreifen entsprechend der Zeile unserer Matrix mit 1 oder 0 belegt ist, ob er geöffnet oder geschlossen ist. Wir brauchen also nur drei Bilder und können eindeutig jedem Pixel einen der acht Kodes (Spalten der Matrix) zuordnen. Da der Kode der direkten Dualzahl entspricht, können wir die Nummer des Gitterstreifens ablesen. Bei 256 Gitterstreifen des Projektors brauchen wir also nur acht Muster zu projizieren.

Probleme gibt es nun zu entscheiden, ob ein Pixel beleuchtet gewesen ist. Hier müssen sinnvolle lokale Binarisierungsstrategien eingesetzt werden. Außerdem können Pixel im

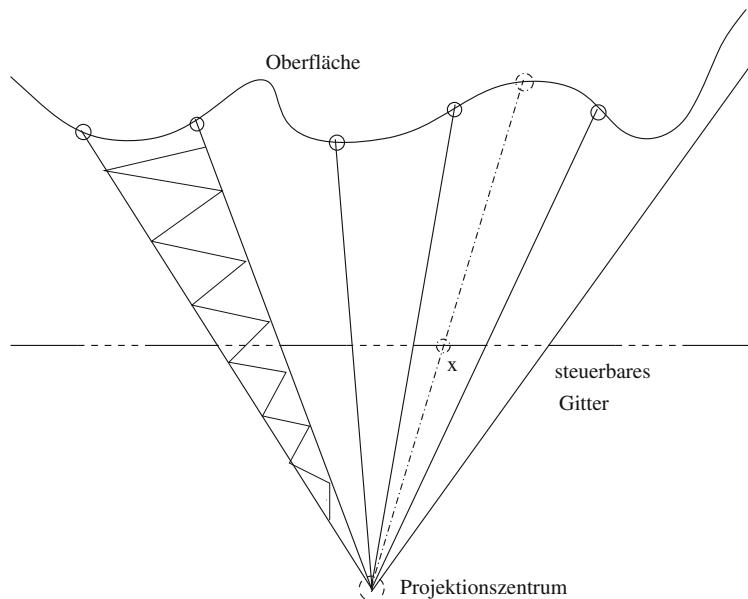


Abb. 16.7 Projektionsanordnung beim codierten Lichtansatz

„Schatten“ liegen, diese Pixel müssen a priori durch Beleuchtungsversuche „markiert“ bzw. ausgeschlossen werden.

Wird nun einmal bei einem Pixel eine Hell/dunkel-Entscheidung falsch getroffen, so wird bei der Zuordnung zur Lichtebene ein grober Fehler begangen. Befindet sich das Pixel in einem homogen beleuchteten Gebiet des Bildes, z. B. in der Umgebung des Pixels ist alles dunkel oder alles hell, so ist es zumindestens sehr unwahrscheinlich, dass wir eine falsche Entscheidung treffen. Befindet sich das Pixel aber in einem Übergangsbereich zwischen hell und dunkel, d. h. ein Beleuchtungssektor geht in einen anderen über, dann werden wir oft die falsche Entscheidung treffen. Genau diese Übergänge bestimmen aber gerade die Lichtebenen, die wir zur Rekonstruktion benötigen. Daher ist es günstiger, die Spalten unserer Matrix mit dem *Gray-Kode* zu kodieren. Der Gray-Kode benachbarter Kodewörter unterscheidet sich stets genau um ein Bit. Ein 3-Bit Gray-Kode sieht z. B. so aus

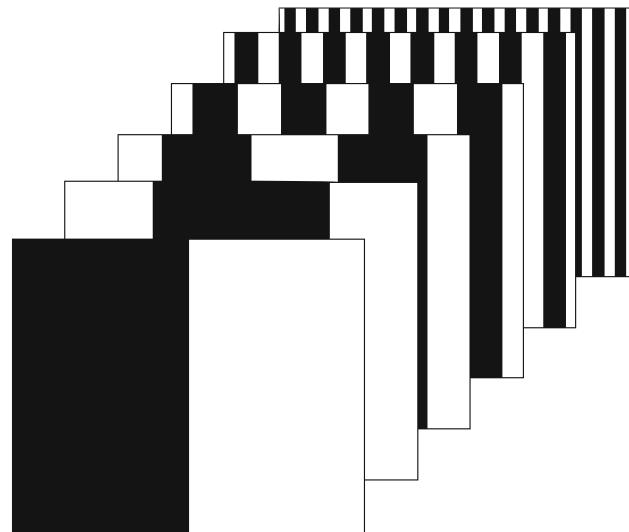
$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array} . \quad (16.61)$$

Ein Kode-Wort der Länge n des Binärkodes bestehe aus den Bits $b_1 b_2 \dots b_n$ und das zugehörige Kodewort des Gray-Kodes aus den Bits $g_1 g_2 \dots g_n$. Dann gilt die Bijektion:

$$b_i = g_1 + g_2 + \dots + g_i \pmod{2} \quad (16.62)$$

$$g_i = b_{i-1} + b_i \pmod{2}. \quad (16.63)$$

Abb. 16.8 Projektion nach dem Gray-Kode



Wir sehen, die Gitterstreifen werden anders belegt als die binär-kodierten Gitterstreifen. Wird bei einem Bit im Übergangsbereich ein Detektions-Fehler begangen, ändert sich die Nummer des Gitterstreifens nur um eine Einheit, der Rekonstruktions-Fehler wird dadurch minimiert. Wird dagegen ein Detektions-Fehler begangen, der nicht im Übergangsbereich liegt, dann erhalten wir eine völlig falsche Streifennummer. Aber diese Art Detektions-Fehler sind sehr unwahrscheinlich und für die folgende Bildverarbeitung auch unwesentlich. Wie wird nun die 3D-Oberfläche rekonstruiert? Wir nehmen nun an, für alle Pixel im 2D-Bild haben wir die Entscheidung getroffen, zu welcher Lichtebene es gehört. Für Punkte innerhalb eines solchen Sektors (siehe Abb. 16.7, die gestrichelte Linie) kennen wir den Punkt x nicht und können demnach nicht die mathematische Ebene bestimmen, um den Schnittpunkt des Sehstrahles bezüglich einer Kamera mit dieser Ebene zu berechnen. Wir können aber im 2D-Bild die Übergänge zum nächsten Sektor berechnen, wenn Pixel von unbeleuchtet zu beleuchtet oder umgekehrt wechseln. Wenn wir z. B. einen Projektor mit acht Gitterstreifen haben, dann würde jedes Pixel eine Nummer von eins bis acht erhalten. Dieses nummerierte Bild segmentieren wir nun, wir müssen folglich die Segmentgrenzen finden. Diese Segmentgrenzen müssen wir sogar durch Interpolation im Subpixelbereich bestimmen. Einem berechneten Segment-(Sub)Pixel können wir aber auf der Gittergeraden denjenigen Punkt zuordnen, der das Ende eines Gitterelementes markiert bzw. den Anfang des nächsten Elementes markiert (durch die nichtgestrichelten Linien dargestellt). Die Genauigkeit (besser die Auflösung) der 3D-Rekonstruktion hängt damit von der Anzahl der Gitterstreifen des Projektors ab, die „Dicke“ des Beleuchtungssektors bestimmt die Auflösung, siehe Abb. 16.7. Da der Projektor aus technischen Gründen nicht zu viele Streifen projizieren kann, benötigt man eine weitere Idee, um innerhalb der „Dicke“ eines Beleuchtungssektors weitere 3D-Punkte berechnen zu können. Wie in Abb. 16.7 zu sehen ist, benötigt man den Schnittpunkt x der gestrichelten Linie

mit der Projektor-Gitterlinie. Dazu dient das sogenannte *Phasenshift-Verfahren*, welches in Abschn. 16.6.5 näher vorgestellt wird.

16.6.4 Moire-Technik

Die Moire-Technik ist eine Rekonstruktionsmethode, die der Optik und Messtechnik entnommen wurde. Mittels strukturierter Licht werden Tiefeninformationen gewonnen. In der Abb. 16.9 ist eine Prinzipskizze zu sehen. Eine Objektoberfläche wird durch eine punktförmige Lichtquelle beleuchtet und durch eine Kamera auf eine Bildebene abgebildet. Dabei ist das Koordinatensystem so orientiert, dass die optische Achse der Kamera mit der z -Achse übereinstimmt und Kamera bzw. Lichtquelle die gleiche (negative) z -Koordinate $z_K = z_L = l$ besitzen. Es ist $K = (0, 0, l)$ der Ort des Brennpunktes der Kamera und $L = (d, 0, l)$ der Ort der Lichtquelle. Ein Oberflächenpunkt $S = (x, y, z)$ wird in den Punkt $B = (x_B, y_B)$ der Bildebene abgebildet, wobei folgende Beziehungen gelten:

$$\frac{x_B}{x} = \frac{f}{z - l}, \quad \frac{y_B}{y} = \frac{f}{z - l}. \quad (16.64)$$

Hier ist f die Brennweite der Kamera. Die entscheidende Idee ist nun, in die x - y -Ebene ein periodisches Streifengitter zu legen, dessen Streifen parallel zur y -Achse verlaufen sollen. Wieviel Licht der jeweilige Oberflächenpunkt S von der Lichtquelle erhält, hängt ab von der Lage des Punktes $P = (x_P, y_P, 0)$ in der Gitterebene. Es gilt:

$$\frac{d - x_P}{-l} = \frac{d - x}{z - l}, \quad \frac{y_P}{-l} = \frac{y}{z - l}. \quad (16.65)$$

Für die Koordinaten des anderen Punktes gilt Analoges:

$$\frac{x_Q}{-l} = \frac{x}{z - l}, \quad \frac{y_Q}{-l} = \frac{y}{z - l}. \quad (16.66)$$

Es sei nun $h(x)$ die „Durchlässigkeit“ des Gitters an der Stelle x . Dann wird die Helligkeit $g(x_B, y_B)$ des Bildpunktes durch das Produkt:

$$\begin{aligned} h(x_Q) \cdot h(x_P) &= h\left(\frac{-lx}{z - l}\right) \cdot h\left(d + \frac{l(d - x)}{z - l}\right) \\ &= h\left(\frac{-l\frac{x_B(z-l)}{f}}{z - l}\right) \cdot h\left(d + \frac{l\left(d - \frac{x_B(z-l)}{f}\right)}{z - l}\right) \end{aligned} \quad (16.67)$$

charakterisiert. Diese Formel lässt sich weiter zu:

$$h\left(\frac{-l \cdot x_B}{f}\right) \cdot h\left(\frac{d \cdot z}{z - l} - \frac{l \cdot x_B}{f}\right) \quad (16.68)$$

vereinfachen. Daraus folgt, dass $g(x_B, y_B)$ bei fester Abbildungsgeometrie lediglich eine Funktion der „Tiefe“ z des in B abgebildeten Oberflächenpunktes S ist.

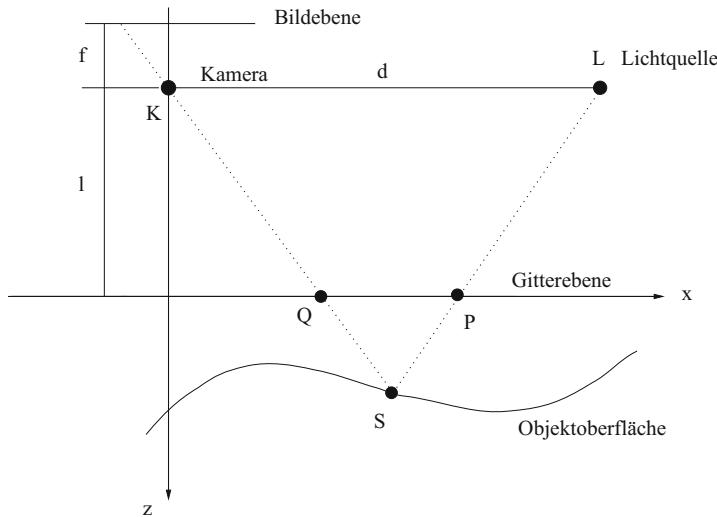


Abb. 16.9 Moire – Prinzipskizze

Nun wollen wir die Gitterfunktion weiter untersuchen. Das Gitter ist eine periodische Funktion mit der gegebenen Periode X . Dabei spielt es aber für den Abbildungsprozess eine wichtige Rolle, welche Verschiebung eine periodische Funktion bezüglich des Koordinatenursprungs aufweist. Der Lichtdurchgang wird sicher ein anderer sein, wenn man dasselbe Gitter verschiebt. Daher modellieren wir unsere Funktion $h(x)$ als Verschiebung der eigentlich periodischen Funktion $f(x)$:

$$h(x) = f(x + \nu), \quad \text{Verschiebung } \nu, \text{ Periode } X. \quad (16.69)$$

Wir bezeichnen im Folgenden:

$$x_P - x_Q = \Delta x, \quad x_P = x_Q + \Delta x. \quad (16.70)$$

Es ist (bis auf einen konstanten Faktor):

$$\begin{aligned} g(x_b, y_B) &\propto h(x_Q) \cdot h(x_P) = h(x_Q) \cdot h(x_Q + \Delta x) \\ &= f(x_Q + \nu) \cdot f(x_Q + \nu + \Delta x) \end{aligned} \quad (16.71)$$

Nun integrieren wir diesen Ausdruck bezüglich der Verschiebungen ν über eine Periode und erhalten:

$$\begin{aligned} I &= \int_0^X f(x_Q + \nu) \cdot f(x_Q + \nu + \Delta x) d\nu = \int_0^X f(\Delta x + \nu) f(\nu) d\nu \\ &= (f \circ f)(\Delta x) = I(\Delta x). \end{aligned} \quad (16.72)$$

Das Symbol \circ ist eine Notation für die Kreuzkorrelation zweier Funktionen f, g :

$$(f \circ g)(t) = \int_0^X f(\xi + t) \overline{g(\xi)} d\xi, \quad (16.73)$$

siehe auch Abschn. 2.1.2. Damit ist obiger Ausdruck $(f \circ f)(\Delta x)$ die Autokorrelationsfunktion von f bzw. h an der Stelle Δx . Die Autokorrelationsfunktion ist selbst wieder periodisch mit der Periode X und hat ihr Maximum bei $t = 0$. Suchen wir, z. B. Stellen maximaler Helligkeit im Bild, dann können wir setzen:

$$\Delta x = 0 + j \cdot X, \quad j \in \mathbb{Z}. \quad (16.74)$$

Da aber

$$x_P - x_Q = \Delta x = \frac{d \cdot z}{z - l} \quad (16.75)$$

ist, ergibt sich:

$$\frac{d \cdot z}{z - l} = j \cdot X. \quad (16.76)$$

Praktisch heißt dies: Wir haben bei der Bildermittlung eine Integration (Summation) über Verschiebungen des Gitters vorzunehmen. Im Moire-Bild suchen wir dann „Streifen“. Alle Punkte mit höchster Intensität auf einem „Streifen“ besitzen dann die gleiche Tiefe, sind folglich Isolinien. Die „Nachbarstreifen“ sind dann der nächst „höheren“ oder „niederen“ Periode j zuzuordnen. Diese Zuordnung funktioniert aber nur auf der mathematischen Annahme einer stetigen Abbildung und dass „benachbarte Streifen“ eine benachbarte Nummer aufweisen. Das wird sicher bei schönen glatten „Freiformobjekten“ funktionieren, aber bei Körpern mit abrupten Übergängen führt das zu Problemen. Dies nennt man allgemein *phase-unwrapping*. Außerdem kann man nur eine relative Tiefenkarte erstellen, für absolute Werte müssten wir wenigstens die absolute Tiefe eines Streifens kennen. Ein weiterer Nachteil ist der Abstand der Isolinien, so dass wir nur eine geringe Auflösung der 3D-Rekonstruktion erhalten. Dies führt wieder zu aufwendigen Interpolationsmethoden zwischen den Isolinien.

Es gibt eine Reihe von modifizierten Moire-Verfahren in der Literatur. Zum Beispiel ist es einfach möglich, für die Gitter von Kamera und Beleuchtung zwei verschiedene Gitter mit der gleichen Periode zu benutzen. Dann geht unsere Autokorrelationsfunktion einfach in die Kreuzkorrelation über und alle wesentlichen Eigenschaften bleiben erhalten. Um die Struktur der Streifen besser zu verstehen, benutzen wir die Fouriertransformation von $h(x)$:

$$h(x) = \sum_{n=-\infty}^{\infty} \alpha_n e^{2\pi i n \frac{x-v}{X}}. \quad (16.77)$$

Dann ist:

$$\begin{aligned} h(x_P) \cdot h(x_Q) &= \sum_n \sum_m \alpha_n \overline{\alpha_m} e^{2\pi i n \frac{x_P - v}{X}} \cdot e^{-2\pi i m \frac{x_Q - v}{X}} \\ &= \sum_n \sum_m \alpha_n \overline{\alpha_m} e^{2\pi i (n \frac{x_P}{X} - m \frac{x_Q}{X})} \cdot e^{2\pi i (m \frac{v}{X} - n \frac{v}{X})}. \end{aligned} \quad (16.78)$$

Der zweite e -Term ist eins für $m = n$. Integrieren wir diesen Ausdruck bezüglich v über eine Periode X , dann ist dieser Ausdruck identisch Null für $m \neq n$. Für $m = n$ ist dies eine Konstante, sodass wir für $I(\Delta x)$ den Ausdruck:

$$I(\Delta x) = \sum_{n=-\infty}^{\infty} \alpha_n^2 e^{\frac{2\pi i n}{X} \Delta x} \quad (16.79)$$

als Fourierdarstellung erhalten. Das Ergebnis ist eigentlich trivial: Wir wissen doch ohnehin, dass das Leistungsspektrum von f identisch mit dem Spektrum der Autokorrelationsfunktion ist.

16.6.5 Phasenshift-Verfahren

Die Idee des Phasenshift-Verfahrens ist lange bekannt. Forschungs- und Entwicklungsarbeiten sowie die praxisreife Einführung in der industriellen Inspektion gehen vor allem auf eine Gruppe am Fraunhofer-Institut in Jena zurück, stellvertretend sei [44] genannt.

Wir stellen uns ein ideales Liniengitter vor (alle Linien haben eine gleiche „Breite“) und schicken Licht mit einem Projektor (Zentralprojektion) durch dieses Gitter. Wenn wir das Intensitätsprofil bezogen auf diese Gitterlinien ansehen, dann sehen wir im Idealfall eine Rechteckfunktion: dort wo das Gitter undurchlässig ist, ist es dunkel und dort, wo es durchlässig ist, ist es hell. Wir legen also eine x -Achse senkrecht zu den Gitterlinien auf das Liniengitter und betrachten die Intensitätsfunktion (Helligkeitsfunktion) über dieser x -Achse. Bezüglich der Intensität bedeute I_{\min} die minimale (z. B. dunkel) und I_{\max} die maximale (z. B. hell) Intensität bezogen auf die ideale Rechteckfunktion. Diese Rechteckfunktion hat nun die Periode X , dies ist die „Breite“ von zwei Linien. Nun wird das Gitter mittels eines (Zentralprojektion) Projektors auf die Oberfläche eines Objektes projiziert und mit einer (*pinhole*) Kamera aufgenommen. Von der Kamera aus beobachten wir ein Pixel, dessen Sehstrahl wir infolge der Kamerakalibrierung im Weltkoordinatensystem berechnen können. Nun müssen wir nur noch für das Pixel berechnen, von welcher idealen Lichtebeine dieser beleuchtete Punkt stammt. Unter einer idealen Lichtebeine verstehen wir eine Ebene, die die x -Achse des Liniengitters schneidet und senkrecht auf dieser x -Achse steht. Wenn der Projektor einschließlich dieser x -Achse im Weltkoordinatensystem bezüglich der Kamera kalibriert ist, dann kann man durch Triangulation den 3D-Punkt berechnen (Schnittpunkt einer Geraden mit einer Ebene).

Betrachten wir wieder die ideale Rechteckfunktion. Diese können wir in eine Fourierreihe entwickeln, siehe (4.92). Durch Beugung und andere optische Effekte ist diese Funktion

bandbegrenzt, d. h. die Fourierreihe enthält nur die ersten n Glieder. Durch bewusstes „Unscharfmachen“ mit dem Projektor (z. B. Defokussieren, Schlitzblende usw.) kann man die Bandbegrenzung soweit treiben, dass die Reihe nur noch die beiden ersten Summanden enthält. Moderne Projektoren können elektronisch angesteuert werden, sodass man beliebige Gitter direkt erzeugen kann. Dadurch erzeugen wir bewusst einen idealen Tiefpass oder erzeugen direkt eine „ $(1 + \cos x)$ “-Funktion:

$$f(x) = \frac{1}{2}(I_{\max} + I_{\min}) + \frac{2}{\pi}(I_{\max} - I_{\min}) \cos\left(2\pi\frac{x}{X}\right). \quad (16.80)$$

Mit den Abkürzungen

$$\varphi = 2\pi\frac{x}{X}, \quad M = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}, \quad H = \frac{I_{\max} + I_{\min}}{2} \quad (16.81)$$

ergibt sich (16.80) zu:

$$g(\varphi) = f\left(\frac{\varphi X}{2\pi}\right) = H \cdot \left(1 + \frac{4}{\pi}M \cos \varphi\right). \quad (16.82)$$

Wir nehmen einmal an, in einem Pixel könnten wir die tatsächliche Lichtintensität g eines Lichtstrahles messen, der vom Projektor durch dieses Liniengitter geht, siehe Abb. 16.7 die gestrichelte Linie. Dann wollen wir in einem Pixel von der Intensität $g(x)$ auf die sogenannte Phase x schließen, d. h. den Ort der Lichtebene auf der x -Achse im Liniengitter. Dies ist so aber nicht möglich, da wir eine Gleichung und die drei Unbekannten φ, M, H haben. Wir haben allgemein ein mathematisches Problem der Form

$$g(x) = a + b \cdot f(x) \quad (16.83)$$

gegeben, wobei die Funktion f und der Funktionswert $g(x)$ gegeben sind. Vom Funktionswert $g(x)$ ausgehend suchen wir folglich das Argument x und müssen daher ein inverses Problem lösen. Zusätzlich wird das inverse Problem noch etwas schwieriger, weil die Größen a und b unbekannt sind. Aus einer Gleichung können wir aber nicht drei Unbekannte ausrechnen, wir benötigen noch mindestens zwei Gleichungen. Daher messen wir g noch zusätzlich, wenn wir f verschieben, dabei sind die Verschiebungen x_1 und x_2 bekannt:

$$\begin{aligned} g(x) &= a + b \cdot f(x) \\ g(x + x_1) &= a + b \cdot f(x + x_1) \\ g(x + x_2) &= a + b \cdot f(x + x_2). \end{aligned} \quad (16.84)$$

Damit haben wir ein nichtlineares Gleichungssystem mit drei Gleichungen und drei Unbekannten x, a, b . Wir nehmen an, dieses sei eindeutig lösbar. Die eindeutige Lösbarkeit hängt sicher vom Funktionstyp f ab. Wählen wir z. B. eine lineare Funktion $f(x) = x$,

dann ist die eindeutige Lösbarkeit nicht gegeben. Wählen wir z. B. $f(x) = \cos x$ oder $f(x) = \sin x$, dann lässt sich dieses Gleichungssystem eindeutig lösen. Nun kehren wir wieder zur Notation (16.82) zurück. Wir benötigen folglich mindestens drei verschiedene Aufnahmen mit einem verschobenen Liniengitter, folglich wird (16.82) zu:

$$g_\alpha = g(\varphi + \alpha) = H \left(1 + \frac{4}{\pi} M \cos(\varphi + \alpha) \right). \quad (16.85)$$

Aus Stabilitätsgründen wählen wir für α eine äquidistante Unterteilung der Periode 2π , die drei Werte bezeichnen wir mit $\alpha_0, \alpha_1, \alpha_2$. Aus (16.85) erhalten wir dann drei Gleichungen, die wir nach den Unbekannten auflösen:

$$\begin{aligned} H &= \frac{1}{3}(g_{\alpha_1} + g_{\alpha_2} + g_{\alpha_3}), \\ M &= \frac{2\sqrt{g_{\alpha_0}^2 + g_{\alpha_1}^2 + g_{\alpha_2}^2 - g_{\alpha_0}(g_{\alpha_1} + g_{\alpha_2}) - g_{\alpha_1}g_{\alpha_2}}}{2g_{\alpha_0} - g_{\alpha_1} + g_{\alpha_2}} \\ \varphi &= \arctan \left(\sqrt{3} \frac{g_{\alpha_2} - g_{\alpha_1}}{2g_{\alpha_0} - g_{\alpha_1} - g_{\alpha_2}} \right). \end{aligned} \quad (16.86)$$

Die nächste und wichtigste Frage ist nun, woher bekommen wir die Helligkeitswerte g_α ? Der Grauwert G_α in einem Pixel unterscheidet sich doch von diesem Helligkeitswert durch Reflexionen, Absorptionen und anderen Beleuchtungseinflüssen. Wir können aber als einfaches Modell $G_\alpha = c \cdot g_\alpha$ mit einem unbekannten Faktor c annehmen. Da uns in (16.86) nur die Phase φ interessiert, sehen wir, dass sich dieser unbekannte Faktor c kürzt. Damit ist die Phase φ invariant gegenüber c und damit können wir auch tatsächlich den Grauwert G_α anstatt dem unbekannten Helligkeitswert g_α benutzen.

Da wir ein inverses Problem bezüglich einer periodischen Funktion $\cos x$ gelöst haben, kann die Lösung natürlich nur eindeutig innerhalb der Periode X sein. Da die Amplitude und der Gleicheanteil (beide Größen M und H) unabhängig von der Wahl der Periode X sind, wird g_α immer „flacher“, je größer X wird, d. h. die Kosinusfunktion bekommt immer kleiner werdende Anstiege. Da wir aber ein inverses Problem zu lösen haben, wird bei Variation des Funktionswertes die Variation des Argumentes immer größer, damit steigt der Fehler der Phase drastisch an, wenn die Grauwerte mit leichten Fehlern bestimmt werden. Folglich müssen wir die Periode möglichst klein halten, andererseits ist die Lösung aber nur eindeutig in dieser Periode und wir können nicht die Phase absolut auf der x -Achse des Projektors bestimmen. Daher kombiniert man das Phasenshift-Verfahren mit dem kodierten Lichtansatz. In Abb. 16.7 erkennt man, wie die Periode zu wählen ist. Der konkrete periodische Abschnitt (Anfang und Ende dieses Abschnittes) kann mit dem kodierten Lichtansatz absolut auf der x -Achse bestimmt werden. In der Praxis werden häufig statt drei Projektionen vier Projektionen benutzt. In Abb. 16.10 sind die ersten beiden Aufnahmen mit einem phasenverschobenem sinusförmigen Gitter zu sehen.

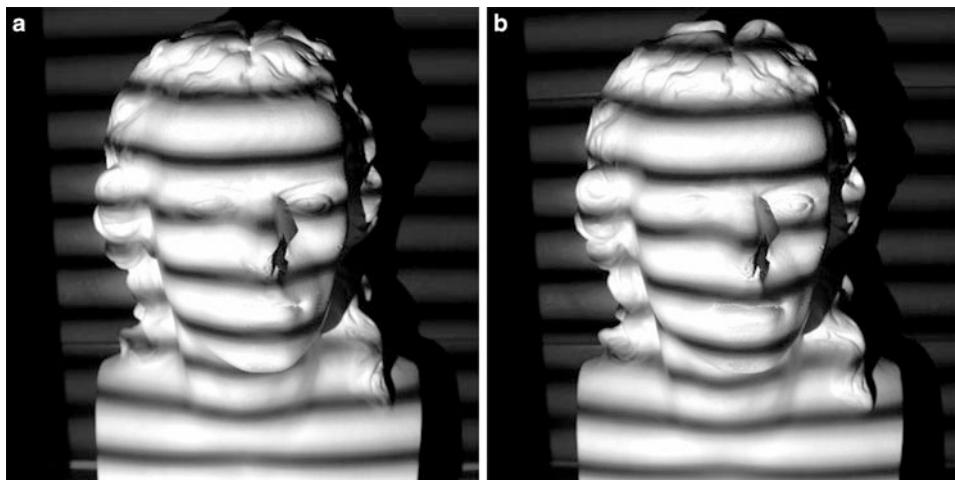


Abb. 16.10 Phasenshiftverfahren: **a** Projektion sinusförmiges Muster, **b** sinusförmiges Muster ist gegenüber dem Muster in **a** phasenverschoben

16.6.6 Profilometrie

Das Vermessen von 3D-Oberflächen mit gewissen Glattheitsforderungen wird oft ganz allgemein als Profilometrie bezeichnet. Je nach Art der verwendeten Technik gibt es verschiedene Bezeichnungen, z. B. Moire-Profilometrie, optische Profilometrie, Phasen-Profilometrie und Profilometrie basierend auf der Fouriertransformation. Im Folgenden soll kurz die Idee der Profilometrie basierend auf der Fouriertransformation (FTP – Fourier transform profilometry) erläutert werden. Beim Phasenshift-Verfahren wird ein sinusförmiges Gitter auf die Oberfläche projiziert. Man benötigt aber mindestens drei Aufnahmen, kann dann aber sehr genau rekonstruieren. Für bewegte Objekte ist dies aber ungeeignet. Es entsteht dann automatisch die Fragestellung, ob man nicht mit einem Bild eine 3D-Rekonstruktion durchführen kann. Dies ist tatsächlich möglich, d. h. allein aus dem Bild in Abb. 16.10a kann eine 3D-Rekonstruktion erfolgen. Natürlich wird es einige drastische Einschränkungen und Bedingungen geben.

Wir nehmen folglich an, ein Objekt werde wie in Abb. 16.10a durch ein sinusförmiges Gitter beleuchtet. Die „ $(1 + \cos x)$ “-Funktion aus (16.80) wird dann als Grauwertbild $f(x, y)$ ortsabhängig beschrieben:

$$\begin{aligned} f(x, y) &= a(x, y) + b(x, y) \cos(\Delta\varphi(x, y)) = a(x, y) + c(x, y) + \overline{c(x, y)} \\ c(x, y) &= \frac{1}{2}b(x, y)e^{i\Delta\varphi(x, y)}. \end{aligned} \quad (16.87)$$

In $a(x, y)$ und $b(x, y)$ sind Beleuchtungsänderungen und Reflexionsverhalten enthalten während $\Delta\varphi(x, y)$ als Phase die Deformation des sinusförmigen Gitters beschreibt.

Ausschließlich in dieser Deformation und damit in dieser Phase $\Delta\varphi(x, y)$ ist die 3D-Tiefeninformation enthalten. Man muss demzufolge aus der Grauwertfunktion $f(x, y)$ die Phase $\Delta\varphi(x, y)$ berechnen können. Dazu bilden wir das Fourier-Integralspektrum der Grauwertfunktion $f(x, y)$:

$$\alpha_f(u, v) = \alpha_a(u, v) + \alpha_c(u, v) + \overline{\alpha_c(u, v)}. \quad (16.88)$$

Wenn wir aus dem Spektrum $\alpha_f(u, v)$ das Spektrum $\alpha_c(u, v)$ selektieren könnten, dann würden wir dieses mit der inversen Fouriertransformation zurücktransformieren und könnten aus dem komplexen Signal $c(x, y)$ die Phase ermitteln. Da das Spektrum $\alpha_f(u, v)$ in (16.88) additiv aus den drei Komponenten zusammengesetzt ist, könnten wir $\alpha_c(u, v)$ bzw. $\overline{\alpha_c(u, v)}$ selektieren, wenn sich die Spektren $\alpha_c(u, v)$ und $\alpha_a(u, v)$ nicht zu sehr bzw. noch besser, „überhaupt nicht“, überlappen. Wenn dies gegeben ist, dann wird das Spektrum $\alpha_a(u, v)$ einen Peak einer gewissen Ausdehnung um den Koordinatenursprung bilden, während die Spektren $\alpha_c(u, v)$ und $\overline{\alpha_c(u, v)}$ zwei bezüglich des Koordinatenur sprungs symmetrische Peaks einer gewissen Ausdehnung bilden. Wir müssen nun den Peak in der „Nähe“ zum Koordinatenursprung einer gewissen Ausdehnung für $\alpha_c(u, v)$ finden, d. h. diesen Peak belassen und alle anderen Frequenzen auf Null setzen. Dies ist ein spezieller Bandpass, der aber nur einseitig und nicht symmetrisch durchgeführt werden darf. Nur dann wird das zurücktransformierte Signal komplex und wir können die Phase bestimmen. In der berechneten Phase ist die 3D-Information enthalten. Um die Tiefendaten zu berechnen, bedarf es einer bestimmten Versuchsanordnung von Projektor, Kamera und einer gedachten Referenzebene. Da die Phase periodisch ist, muss man die Tiefe entsprechend fortsetzen, dies macht aber Probleme bei Unstetigkeiten der Oberfläche, dies nennt man wie wir schon wissen *phase-unwrapping*. Zusätzlich führt die Umrechnung der Phasenwerte in Tiefenwerte zu Beschränkungen in der maximal möglichen Tiefenberechnung. Die wesentliche Beschränkung des Verfahrens besteht aber in der Voraussetzung, dass sich das Spektrum separieren lassen muss. Dies heißt aber, die Oberfläche muss glatt sein und darf keine großen Sprünge enthalten. Außerdem muss die Beleuchtung auch einigermaßen homogen sein. Alle diese Voraussetzungen schränken den Anwendungsbereich auf glatte Oberflächen (keine abrupten Änderungen) mit nur geringer Tiefenänderung ein. Zum Versuchsaufbau und zur konkreten Berechnung der Tiefe aus den Phasenwerten siehe die Originalarbeit von Takeda et al. [75].

16.6.7 Statistische Muster

Eine völlig andere und sehr simple Idee basiert auf der statistischen Projektion bestimmter Muster. Wir stellen uns vor, dass wir zwei kalibrierte Kameras und einen Projektor zur Verfügung haben, wobei der Projektor nicht kalibriert sein muss. Der Projektor projiziert auch keine sinusförmigen Muster, sondern statistische Punktmuster. In einfachsten Falle eines Binärmusters wird ein Pixel „angeleuchtet“ oder nicht. Dies geschieht zufällig mit

einer bestimmten Wahrscheinlichkeit. Ein von der Kamera aufgenommenes Bild liefert folglich im Idealfall ein Binärbild. Nun erzeugen wir mit dem Projektor eine Folge von n statistischen Punktmustern. Die Kameras müssen synchronisiert sein und „sehr schnell“ diese Folge aufnehmen können. Wir erhalten dann eine Serie von n Binärbildern für die erste Kamera und eine weitere Serie für die zweite Kamera. Pro Pixel haben wir nun in beiden Binärbildern je einen n -dimensionalen Vektor, im einfachsten Falle bestehend nur aus Nullen und Einsen. Da wir für jedes Pixel im ersten Bild den korrespondierenden Punkt im zweiten Bild suchen, müssen wir im zweiten Bild den „passenden“ Vektor finden. Dazu benötigen wir ein geeignetes Matchingmaß, siehe Abschn. 19.8.1. Eine Möglichkeit bietet der normierte Korrelationskoeffizient. Für einen industriellen Einsatz sind dabei noch viele praktische Probleme zu lösen, z. B.: wieviele Bilder benötigt man mindestens, um mit einer gewissen Wahrscheinlichkeit zu garantieren, dass man den richtigen korrespondierenden Punkt gefunden hat. Wie sicher ist überhaupt die Entscheidung, ob ein Pixel „beleuchtet“ ist oder nicht. Wie geht man praktisch mit Verdeckungen um und viele andere Details. Die Gruppe um R. Kowarschik aus Jena hat diese praktischen Probleme eingehend untersucht, siehe z. B. [44].

16.6.8 Shape from focus/defocus

Shape from focus Die Idee aus scharfen Bildern die Tiefe zu berechnen ist sehr einfach, siehe auch [49]. Dazu benutzen wir das Modell einer dünnen Linse, siehe Abb. 14.2 und (14.12). Es sei f_L die Linsenbrennweite, f die Kamerabrennweite, d. h. der Abstand des Projektionszentrums oder der Linse zur Bildebene, und d_{Objekt} der Abstand der Objektebene zur Linse, dann gilt:

$$\frac{1}{f_L} = \frac{1}{f} + \frac{1}{d_{\text{Objekt}}}. \quad (16.89)$$

Wird wie in der Abb. 14.2 ein 3D-Punkt scharf abgebildet, dann kann man mit bekannten f und f_L die Tiefe d_{Objekt} ausrechnen. Wir müssen folglich für jedes Pixel feststellen, ob es scharf abgebildet ist oder nicht. Dazu benötigen wir Maße, diese sind im Abschn. 7.6 untersucht worden. Es ergeben sich nun eine Reihe praktischer Probleme:

- Es muss systematisch entweder das Objekt verschoben, die Kamerabrennweite geändert werden, oder die Linse verschoben werden. Eigentlich muss diese Veränderung kontinuierlich erfolgen. Dies ist aber praktisch nicht möglich, so dass in diskreten Schritten die Veränderungen erfolgen müssen. Es sind so ca. 10 bis 15 Aufnahmen zu erstellen.
- Für jedes Pixel ist diejenige Aufnahme zu ermitteln, bei der das Pixel am schärfsten abgebildet wird. Dazu sind die Maße aus Abschn. 7.6 zu benutzen, wobei für jedes Pixel nur eine Fensterumgebung zu verwenden ist.
- Dabei ist zusätzlich zu beachten, dass durch Größenänderungen sich ein 3D-Punkt als abgebildetes Pixel im Bild verschieben kann.

- Durch die endliche Anzahl von Bildern leidet die Genauigkeit, deshalb muss zwischen den Aufnahmen interpoliert werden.
- Das Schärfemass wird auf homogene Flächen nicht reagieren, deshalb muss die Oberfläche gut texturiert sein.
- Das Verfahren bedingt folglich einen hohen technischen und zeitlichen Aufwand.
- Ein großer Vorteil ist: Es gewinnt die 3D-Information ausschließlich mit einer Kamera, es sind keinerlei Punktreferenzen notwendig und die Tiefe wird sogar absolut berechnet.
- Das Anwendungsspektrum ist folglich beschränkt, oft findet man im Mikroskopiebereich Anwendungen.

Shape from defocus Natürlich sind die Begriffe focus und defocus „dual“ zueinander. Im folgenden soll wieder die Grundidee gezeigt werden, ausführliche Darlegungen sind z. B. in [14] zu finden. Wenn wir in Abb. 14.2 die Bildebene verschieben, aber die Objektebene belassen, dann wird der Objektpunkt „verschmiert“ abgebildet. Die Größe des „verschmierten“ Bereiches kann eindeutig der Entfernung des Objektpunktes zugeordnet werden. Dual zur Idee „shape from focus“, wo wir die schärfste Abbildung suchen, wollen wir die gegebene Unschärfe nutzen, um daraus direkt die Tiefe des Objektpunktes zu berechnen. Wir benötigen dazu eine der Formel (16.89) entsprechende Beziehung. Diese Beziehung muss den Zusammenhang zwischen Kameraparametern, Objektpunktentfernung, Blendengröße und Größe des Verschmierungsbereiches (PSF) herstellen. Dazu nehmen wir einmal an, die Blende sei kreisförmig. Dann kann man mit der geometrischen Optik zeigen, dass der Verschmierungsbereich (PSF) auch eine Kreisfläche mit einem Helligkeitswert darstellt, oft als „pillbox“ (Pillendose) bezeichnet. In der Praxis ist dies aber viel komplizierter. Eine bessere Näherung ist eine rotationssymmetrische Gaußfunktion, siehe (2.83), wobei σ die Rolle des Kreisradius der pillbox übernimmt. Es sei r der Radius der kreisförmigen Apertur, f wieder der Abstand der diesmal nicht notwendig fokussierten Bildebene zum Projektionszentrum, und r^{PSF} der Radius des „Verschmierungskreises“. dann gilt:

$$r^{\text{PSF}} = r \cdot f \left(\frac{1}{f_L} - \frac{1}{f} - \frac{1}{d_{\text{Objekt}}} \right). \quad (16.90)$$

Benutzen wir die Gaußfunktion (2.83), dann ist $\sigma = \rho \cdot r^{\text{PSF}}$, wobei ρ ein kameraabhängiger Faktor ist, der von der Optik und der Auflösung der Sensoren abhängt. Dieser Faktor ρ sei a priori durch eine geeignete Kalibrierung bekannt. Dann ist

$$\sigma = \rho \cdot r \cdot f \left(\frac{1}{f_L} - \frac{1}{f} - \frac{1}{d_{\text{Objekt}}} \right). \quad (16.91)$$

Wenn man außer den Kameraparametern auch σ für jeden einzelnen Objektpunkt bestimmen kann, dann kann man aus (16.91) für jeden Objektpunkt die Tiefe d_{Objekt} berechnen. Wie kann man nun für jeden Objektpunkt σ berechnen?

Dazu benötigen wir mindestens zwei Bilder mit unterschiedlichen Schärfeeinstellungen, damit auch unterschiedlichen Kameraparametern, daher schreiben wir (16.91) für zwei

Bilder auf:

$$\sigma_i = \rho \cdot r_i \cdot f_i \left(\frac{1}{f_{L_i}} - \frac{1}{f_i} - \frac{1}{d_{\text{Objekt}}} \right), \quad i = 1, 2. \quad (16.92)$$

Wir müssen zunächst auf jeden Fall die Unbekannte d_{Objekt} eliminieren, dies gelingt durch simple Differenzbildung, wobei wir eine Gleichung erhalten:

$$\sigma_1 = a \cdot \sigma_2 + b \quad (16.93)$$

mit den Größen:

$$a = \frac{r_1 f_1}{r_2 f_2}, \quad b = \rho \cdot r_1 \cdot f_1 \left(\frac{1}{f_{L_1}} - \frac{1}{f_1} - \frac{1}{f_{L_2}} + \frac{1}{f_2} \right). \quad (16.94)$$

Eine Gleichung drückt nun das Verhältnis der beiden „Verschmierungskreise“ aus. Wir benötigen folglich noch eine zweite Beziehung, um die Größen σ_1 und σ_2 berechnen zu können. Da wir einem Objektpunkt eine PSF zuordnen, ist die PSF nicht verschiebungsinvariant, wir haben einen klassischen Fall von linearen Systemen, die nicht verschiebungsinvariant sind. Deshalb weichen wir unsere Annahmen etwas auf und nehmen an, in einer gewissen Fensterumgebung des Objektpunktes sei die Tiefe des Objekts näherungsweise konstant. Daher können wir in dieser Fensterumgebung des Punktes (x, y) Verschiebungsinvarianz annehmen und die Bildaufnahme als Faltung formulieren:

$$g_i(x, y) = h_i(x, y) * f(x, y), \quad i = 1, 2. \quad (16.95)$$

Die beiden PSF $h_i(x, y)$, $i = 1, 2$ sind die beiden modellierten Gaußfunktionen nach (2.83). Da in beide Gleichungen die gleiche Originalfunktion $f(x, y)$ eingeht, müssen wir diese wieder eliminieren, allerdings geht dies nicht durch simple Differenzbildung. Daher wenden wir auf (16.95) das Faltungstheorem (4.23) an und bilden die Quotienten aus den jeweils linken bzw. rechten Seiten der Fouriertransformierten (siehe auch (4.95)):

$$\frac{\alpha_{g_1}(\omega_1, \omega_2)}{\alpha_{g_2}(\omega_1, \omega_2)} = \frac{\alpha_{h_1}(\omega_1, \omega_2)}{\alpha_{h_2}(\omega_1, \omega_2)} = e^{-\frac{1}{2}(\omega_1^2 + \omega_2^2)(\sigma_1^2 - \sigma_2^2)}. \quad (16.96)$$

Wir lösen nach $\sigma_1^2 - \sigma_2^2$ auf:

$$\sigma_1^2 - \sigma_2^2 = \frac{-2}{\omega_1^2 + \omega_2^2} \log \frac{\alpha_{g_1}(\omega_1, \omega_2)}{\alpha_{g_2}(\omega_1, \omega_2)}. \quad (16.97)$$

Da die rechte Seite für jedes (ω_1, ω_2) konstant sein muss, dies aber in der Praxis nicht der Fall sein wird, mitteln wir die rechte Seite durch:

$$C = \frac{1}{A} \iint_R \frac{-2}{\omega_1^2 + \omega_2^2} \log \frac{\alpha_{g_1}(\omega_1, \omega_2)}{\alpha_{g_2}(\omega_1, \omega_2)}, \quad (16.98)$$

wobei R ein geeigneter Mittelungsbereich ist und A die Fläche dieses Bereiches R ist. Folglich erhalten wir zu (16.93) eine zweite Gleichung:

$$\sigma_1^2 - \sigma_2^2 = C. \quad (16.99)$$

Beide Gleichungen lösen wir und erhalten eine quadratische Gleichung

$$(a^2 - 1)\sigma_2^2 + 2ab\sigma_2 + b^2 = C, \quad (16.100)$$

welche wir leicht nach σ_2 auflösen können. Nun können wir die Beziehung (16.91) nach der Objektentfernung d_{Objekt} auflösen.

Diese Berechnung müssen wir nun für jedes Pixel (x, y) durchführen und erhalten aus lediglich zwei Bildern, die mit verschiedenen Kameraparametern aufgenommen wurden, eine vollständige Tiefenkarte. Die Probleme, die bei der praktischen Realisierung auftreten, sind die folgenden:

- Ein Vorteil ist, dass die Methode sehr simpel ist, nur zwei Aufnahmen verlangt und keinerlei Punktereferenzen benötigt. Der Preis dafür ist, dass sie in der Genauigkeit etwas schlechter als die leistungsfähigen Stereo-Methoden ist. Der durchschnittliche Tiefenfehler liegt bei ca. 5–8 Prozent.
- Da wir die PSF durch eine Gauß-Funktion modelliert haben, entstehen Modellierungsfehler. Andere Modelle könnten besser sein.
- Die Kameraparameter müssen für die beiden Aufnahmen sehr genau bestimmt werden.
- Die größte Fehlerquelle ist das Problem der lokalen Fensterwahl. Die Annahme, in einer lokalen Fensterumgebung sei die Tiefe des Objektes annähernd konstant, d. h. wir können lokal die Faltung mit der PSF nutzen, ist sehr schwer zu überprüfen. Was passiert, wenn sich die Tiefe z. B. abrupt ändert.
- Wenn die Grauwertfunktion $f(x, y)$ über einem lokalen Bereich konstant ist, dann bedeutet eine Faltung lediglich eine Multiplikation mit einem Faktor, man kann folglich nicht die relative Änderung der „Verschmierung“ zwischen zwei Aufnahmen bestimmen und das Verfahren funktioniert nicht. Die Oberfläche muss folglich wie bei *shape from focus* gut texturiert sein.

16.7 Photometrische Methoden der Rekonstruktion

Wenn eine Lichtquelle eine Oberfläche beleuchtet, dann gibt es Schatten in Abhängigkeit von der Tiefe. Diese Art Schattenwurf soll für eine Tiefenrekonstruktion genutzt werden.

16.7.1 Beleuchtung und Reflexion

Die Beleuchtungsstärke ist die Lichtmenge, die auf eine Oberfläche fällt, gemessen in W/m^2 . Diese wird *irradiance* genannt. Die Lichtmenge, die wieder abgestrahlt wird, nennt

man *radiance*. Wir stellen uns eine Oberfläche vor mit einer Normalen in einem Oberflächenpunkt. Weiterhin sei die Richtung einer punktförmigen Lichtquelle gegeben und die Richtung des Betrachters. Den Winkel zwischen Lichtquelle und Normale bezeichnen wir mit α (*incident*), den Winkel zwischen Normale und Betrachter mit β (*emergent*) und den Winkel zwischen Lichtquelle und Betrachter mit γ (*phase*). Alle drei Richtungsvektoren müssen nicht in einer Ebene liegen. Dadurch kann man eine Reflexionsfunktion $r(\alpha, \beta, \gamma)$ definieren. Eine ideale Reflexionsfunktion (Indikatorfunktion) liegt vor, wenn nur reflektiert wird, wenn alle drei Vektoren in einer Ebene liegen und Einfallsinkel gleich Ausfallwinkel ist:

$$r(\alpha, \beta, \gamma) = \begin{cases} 1, & \text{wenn } \alpha = \beta \text{ und } \alpha + \beta = \gamma \\ 0, & \text{sonst.} \end{cases} \quad (16.101)$$

Dies ist die Reflexionsfunktion einer ideal spiegelnden Oberfläche, d. h. man sieht viele Oberflächenpunkte gar nicht (schwarz), bzw. einige (die eben spiegeln) ganz hell. Mit solch einer Funktion können wir sicher nicht die 3D-Form rekonstruieren. Dazu benötigen wir eine diffus strahlende Oberfläche, genannt *Lambert*-Oberfläche. Dann lautet die Reflexionsfunktion:

$$r_L(\alpha, \beta, \gamma) = I_0 \rho \cos(\alpha) = I_0 \rho \cos(\mathbf{l}, \mathbf{n}). \quad (16.102)$$

Diese Funktion ist völlig unabhängig vom Standpunkt des Betrachters. Egal ob wir einen Punkt von der Seite, von vorn usw. anschauen, er ist immer gleich dunkel/hell. Das bedeutet doch, das Licht muss „gleichmäßig“ nach allen Richtungen reflektiert werden. Dies nennt man *diffus* reflektierend. Von solche Oberflächen werden wir im Folgenden ausgehen. Dabei ist

- I_0 – die Lichtintensität der Lichtquelle,
- ρ – die *Albedo*, oder auch Reflexionsfaktor genannt, der in erster Linie vom Material abhängt, aber auch vom Ort abhängen könnte.

Zum Schluss dieses Abschnittes soll noch im Zusammenhang mit der Reflexion eine häufig gestellte, „amüsante“ Frage beantwortet werden:

- Frage: Wenn man in einen Spiegel schaut, warum wird dann links und rechts vertauscht, aber nicht oben und unten?
- Antwort: Die Frage ist total unsinnig gestellt. Ein Spiegel vertauscht niemals links und rechts. Wenn man in den Spiegel schaut, ist der rechte Arm nach wie vor rechts von der Nase, natürlich vom Betrachter aus gesehen. Links, rechts, oben und unten sind immer normierte Begriffe und müssen sich auf einen Betrachter beziehen. Ein Spiegel vertauscht aber vom Betrachter aus gesehen immer vorn und hinten. Ist der Spiegel an der Decke oder auf dem Boden, dann bedeutet der Tausch von vorn und hinten tatsächlich den Tausch von oben und unten. Wenn man in einen See schaut, stehen die Bäume

im Spiegel der Wasseroberfläche auf dem Kopf. Damit ist die Aussage, dass ein Spiegel nicht oben und unten vertauscht, auch nicht richtig.

16.7.2 Shape from Shading

Für das weitere nehmen wir an:

- Die Oberfläche ist diffus, also eine Lambert-Oberfläche.
- Wir haben paralleles Licht, d. h. für alle Oberflächenpunkte haben wir den gleichen Richtungsvektor der Lichtquelle. Bei einer weit entfernten Punktlichtquelle kann man dies näherungsweise annehmen.
- Das Reflexionsbild $I(\xi, \eta)$ ist durch Rauschen wenig gestört. Es ist folglich $I(\xi, \eta) = r(\alpha, \beta, \gamma) = R(\mathbf{n})$, d.h alle Intensitäten im Bild hängen nur von der Richtung (nicht Länge) der Oberflächennormalen ab.

Wir wollen nun für jedes Pixel im Bild die Oberflächennormale bestimmen. Durch „übliche“ Integration über die Oberflächennormale können wir dann die Oberfläche selbst berechnen. Mit den Bezeichnungen

$$z = f(x, y), \quad \mathbf{n} = (f_x, f_y, -1)^T, \quad \mathbf{l} = (l_x, l_y, l_z)^T \quad (16.103)$$

ist dann:

$$I(\xi, \eta) = I_0 \rho \frac{f_x l_x + f_y l_y - l_z}{|\mathbf{n}| \cdot |\mathbf{l}|}. \quad (16.104)$$

Wir betrachten nun nur Einheitsvektoren, d. h. $|\mathbf{n}| = |\mathbf{l}| = 1$. Dann ist:

$$I(\xi, \eta) = I_0 \rho \langle \mathbf{n}, \mathbf{l} \rangle \rightarrow I_r = \frac{I(\xi, \eta)}{I_0 \rho} = \langle \mathbf{n}, \mathbf{l} \rangle. \quad (16.105)$$

Daher ist:

$$I_r \cdot \langle \mathbf{l}, \mathbf{l} \rangle = \langle \mathbf{n}, \mathbf{l} \rangle \rightarrow \langle \mathbf{n} - I_r \cdot \mathbf{l}, \mathbf{l} \rangle = 0. \quad (16.106)$$

Weiter ist:

$$|\mathbf{n} - I_r \mathbf{l}|^2 = |\mathbf{n}|^2 + I_r^2 \cdot |\mathbf{l}|^2 - 2I_r \langle \mathbf{n}, \mathbf{l} \rangle = 1 - I_r^2. \quad (16.107)$$

Bezeichnen wir den Vektor

$$r(x, y, z)^T = \mathbf{n} - I_r \mathbf{l}, \quad (16.108)$$

dann haben wir folglich zwei Bestimmungsgleichungen für \mathbf{r} :

$$\langle \mathbf{r}, \mathbf{l} \rangle = 0, \quad |\mathbf{r}|^2 = 1 - I_r^2. \quad (16.109)$$

Diese zwei Gleichungen mit den drei Unbekannten x, y, z schreiben wir nochmals auf:

$$l_x x + l_y y + l_z z = 0, \quad x^2 + y^2 + z^2 = 1 - I_r^2. \quad (16.110)$$

Diese beschreiben eine Kurve zweiter Ordnung, das System ist also unterbestimmt. Wir können dies auch anders notieren:

$$I = I_0 \rho \mathbf{n}^T \mathbf{l}, \quad \|\mathbf{n}\|^2 = 1 \quad (16.111)$$

Es sind zwei Gleichungen, die aber drei Unbekannte enthalten. Dabei wurde \mathbf{l} a priori auf Eins normiert. Alle Normalenvektoren, die bezüglich \mathbf{l} einen Kegel aufspannen, folglich mit \mathbf{l} den gleichen Winkel haben, sind mögliche Lösungen. Dabei seien die Längen auf eins normiert. I_r hängt dabei von I_0 und der Albedo ρ ab. Dieses Produkt muss bekannt sein, und ist nicht etwa ein Faktor auf den es uns nicht ankommt. Angenommen, wir kennen dieses Produkt nicht, sei also eine Konstante, dann ist auch I_r eine unbekannte Größe, allerdings für jedes Pixel bis auf diesen gemeinsamen Faktor bekannt. Dadurch ist der absolute Grauwert im Bild unbekannt. Lösen wir obige zwei Gleichungen für eine Konstante, die nicht dem tatsächlichen Produkt entspricht, so ist das Skalarprodukt ein anderes. Die Länge des Normalenvektors bleibt aber gleich eins, daher muss sich der Winkel zwischen Normalenvektor und \mathbf{l} ändern. Damit bestimmen wir einen völlig falschen Normalenvektor. Wir müssen also die beiden Parameter kennen.

Für jedes Pixel im Bild könnten wir nun als Lösung einen Punkt dieser Kurve zweiter Ordnung auswählen, wir wollen aber eindeutig jedem Pixel einen Normalenvektor zuordnen. Wir führen deshalb eine sinnvolle Zusatzbedingung ein: Die Oberfläche soll glatt sein, d. h. die Änderung der Oberflächennormale soll nicht abrupt erfolgen. Deshalb minimieren wir folgendes Funktional:

$$Z = \iint \left\{ (I_r(x, y) - \mathbf{n}^T \mathbf{l})^2 + \lambda (\|\mathbf{n}_x\|^2 + \|\mathbf{n}_y\|^2) + \mu (\|\mathbf{n}\|^2 - 1) \right\} dx dy \\ \rightarrow \text{Minimum.} \quad (16.112)$$

Der erste Term ist der eigentliche Term bezüglich des Bildes, der zweite soll die Änderungen der Normalen minimieren (Glattheitsforderung) und der dritte ist lediglich die Normierung auf Eins. Um dieses Variationsproblem zu lösen, müssen wir die *Eulerschen Gleichungen* aufschreiben:

$$(I_r - \mathbf{n}^T \mathbf{l}) \cdot \mathbf{l} + \lambda \nabla^2 \mathbf{n} - \mu \mathbf{n} = 0. \quad (16.113)$$

Diese müssen wir nun diskretisieren und in einer iterativen Prozedur lösen. Was uns an dem Verfahren stört sind sicher die unbekannte Beleuchtungsstärke, die Albedo sowie die

komplizierte Glattheitsbedingung. Daher betrachten wir die Möglichkeit, ob es uns mit mehreren Beleuchtungsquellen besser gelingt. Die Rekonstruktion einer Oberfläche mit mehreren Beleuchtungsquellen nennt man **photometrisches Stereo**. Nehmen wir einmal zwei Beleuchtungsquellen mit gleichen Beleuchtungsstärken an. Wir justieren eine Kamera und die zwei Leuchten fest und nehmen wieder eine Lambert-Oberfläche an. Wir knipsen die erste Lampe an und nehmen ein Bild auf, $I_1(x, y)$. Nun schalten wir Lampe eins aus und schalten Lampe zwei an und erhalten das zweite Bild $I_2(x, y)$. Da wir Kamera und Objekt nicht bewegt haben, sind alle Pixel in beiden Bildern synchron. Nun bilden wir den Ausdruck:

$$\text{Quot}(x, y) = \frac{I_0 \rho \mathbf{n}^T \mathbf{l}_1}{I_0 \rho \mathbf{n}^T \mathbf{l}_2} = \frac{I_1(x, y)}{I_2(x, y)} = \tilde{I}(x, y), \quad (16.114)$$

der nicht mehr von der Beleuchtungsstärke und der Albedo abhängt. Nun berechnen wir wieder unser Integral zu:

$$Z = \iint \left\{ (\tilde{I}(x, y) - \text{Quot}(x, y))^2 + \lambda(\|\mathbf{n}_x\|^2 + \|\mathbf{n}_y\|^2) + \mu(\|\mathbf{n}\|^2 - 1) \right\} dx dy \\ \rightarrow \text{Minimum} \quad (16.115)$$

Die Invarianz gegenüber Beleuchtungsstärke und Albedo haben wir mit einem noch komplizierterem Zielfunktional und zwei Beleuchtungsquellen erkaufte.

Nun nehmen wir noch eine dritte Lichtquelle hinzu und erhalten drei Bilder:

$$\begin{aligned} I_1(x, y) &= I_0 \rho \cdot \mathbf{n}^T \mathbf{l}_1 \\ I_2(x, y) &= I_0 \rho \cdot \mathbf{n}^T \mathbf{l}_2 \\ I_3(x, y) &= I_0 \rho \cdot \mathbf{n}^T \mathbf{l}_3. \end{aligned} \quad (16.116)$$

Wenn wir die Konstanten I_0 und ρ kennen, haben wir ein lineares Gleichungssystem mit drei Gleichungen und drei Unbekannten. Die Längen der \mathbf{l}_i , $i = 1, 2, 3$ sind auf eins normiert gegeben. Für den unbekannten Vektor \mathbf{n} haben wir auch noch die Restriktion $\|\mathbf{n}\| = 1$ gegeben, folglich eine vierte Gleichung. Wenn unser Modell stimmt und die beiden Konstanten bekannt sind, dann muss diese vierte Bedingung überflüssig sein. Ist das Produkt der beiden Konstanten nun nicht bekannt, so legen wir dieses auf Eins fest und lösen das lineare Gleichungssystem. Den errechneten Normalenvektor normieren wir nun auf Eins, der Faktor für die Normierung muss dann theoretisch das Produkt aus I_0 und ρ sein. Das Kuriose ist nun, mit drei Lichtquellen brauchen wir die Konstanten nicht mehr zu kennen, die Lösung des linearen Gleichungssystems, also der Normalenvektor, hängt multiplikativ (nur die Länge des Vektors wird beeinflusst, nicht der Winkel) von diesem Konstantenprodukt ab. Mit drei Lichtquellen scheint also ein Idealfall gegeben zu sein: Ein lineares Gleichungssystem pro Pixel und die Konstanten braucht man nicht kennen.

16.8 Monokulare Rekonstruktion

Ohne A-priori-Wissen kann man in der Regel mit nur einer Kamera und keinen weiteren technischen Hilfsmitteln kein Objekt rekonstruieren. Es ist Wissen über das zu rekonstruierende Objekt nötig. Das können mathematische Beschreibungen sein, Kenntnisse der Form oder Größe usw. So ist bei dem A-priori-Wissen: „das zu rekonstruierende Objekt ist eine Kugel“ nur noch die Kenntnis des Radius der Kugel nötig. Die Grundidee der Rekonstruktion ist immer dieselbe:

Die Grundvoraussetzung ist natürlich, dass die Kamera kalibriert ist. Von den beobachteten Punkten, Kanten usw. im Bild berechnet man den Sichtkegel, die Sichtpyramide usw. Nun „legen“ wir das zu rekonstruierende Objekt so in den Kegel, dass sich gerade das beobachtete Bild ergibt. Dies muss nicht immer eindeutig sein. Bei der Rekonstruktion einer Kugel ist dies sehr anschaulich: Die abgebildete Kugel ergibt im Bild eine Ellipse. Von dieser Ellipse berechnen wir den elliptischen Sichtkegel, der auf beiden Seiten des Projektionszentrums aussieht wie eine „Kaffeefiltertüte“, die auf der Bildseite abgeschnitten wurde. Die Bildseite liege „unten“. Im 3D-Raum ist sie eigentlich unendlich ausgedehnt, wir „schneiden“ sie aber mal in großer Höhe „oben“ ab. Nun nehmen wir die Kugel und werfen sie in die Filtertüte „oben“ hinein. Dort wo sie zum liegen kommt, ist die rekonstruierte Position der Kugel im Raum.

Gibt es tatsächlich keinerlei monokulare Rekonstruktion ohne A-priori-Wissen? Wenn man nur bestimmte Maße des 3D-Objektes wissen möchte und diese auch nur in einem statistischen Sinne, dann ist dies überraschenderweise bei bestimmten Objekten möglich. Dazu werden wir im ersten Abschnitt etwas zur *Siebanalyse* ausführen. In den folgenden Abschnitten werden dann bestimmte Objekte monokular rekonstruiert, natürlich mit A-priori-Wissen.

16.8.1 Siebanalyse

Bei der Siebanalyse möchte man die sogenannte Korngrößenverteilung bestimmter Stoffe (z. B. Sand, Kies, Granulate usw.) ermitteln. Dies geschah in der Vergangenheit rein mechanisch. Durch eine Folge von Sieben mit verschiedenen Maschenweiten wurde der Stoff gesiebt und der „Siebrest“ in jedem Sieb durch Wägen bestimmt. Dies ist natürlich sehr aufwendig, so dass in der Neuzeit Geräte (sogenannte *CAMSIZER*) entwickelt wurden, die dieses Problem mit einer Kamera und Methoden der Bildverarbeitung lösen. Eine generelle Lösung für alle Kornformen und nur eine Kamera gibt es aber immer noch nicht, allerdings für bestimmte Kornformen schon. Wir stellen uns solch ein Gerät folgendermaßen vor: Auf einem kleinen Förderband wird das Stoffgemisch transportiert und fällt am Ende wie ein Wasserfall in einen Behälter. Gegenüber dem „Wasserfall“ ist orthogonal dazu eine Kamera angebracht, die in einer Folge von Bildern die Körner aufnimmt. Es entsteht somit eine Folge von 2D-Bildern dieser 3D-Objekte. Als Abbildungsmodell kann man die orthografische Projektion näherungsweise annehmen. Aus den orthografischen Projektionen dieser Körner soll nun deren „echte“ 3D-Korngrößenverteilung berechnet werden. Für

beliebige Objekte ist dies sehr schwierig, der triviale Fall ist sicher der, dass alle Körner fast Kugeln sind. Dann brauchen wir im Bild nur den Kreisdurchmesser bestimmen und sind fertig. Infolgedessen wird häufig versucht, von den 2D-Objekten auf eine Kugel zu schließen und deren Durchmesser zu nutzen. Je weiter sich aber ein 3D-Objekt von einer Kugel unterscheidet, umso größer werden natürlich die Fehler sein. Dazu werden von den 2D-Objekten Maße bestimmt, z. B.:

- Pro 3D-Objekt erhalten wir im Bild ein 2D-Objekt mit einer geschlossenen Kontur.
- Man legt im 2D-Bild eine Messrichtung fest.
- MARTIN-Durchmesser: In Messrichtung wird die Länge eines Geradensegmentes bestimmt, das das Objekt in zwei Teile zerlegt, die beide flächengleich sind.
- FERET-Durchmesser: In Messrichtung legen wir zwei Tangenten an das Objekt. Der Abstand dieser Tangenten ist der FERET-Durchmesser.
- Maximale Sehne: Ist ähnlich dem FERET-Durchmesser, nur senkrecht zur Messrichtung.
- Durchmesser: Ist der Durchmesser eines flächengleichen Kreises.

Gibt es denn nun außer dem Trivialfall der Kugeln noch andere nichttriviale Fälle, bei denen man die Verteilungen bestimmter Maße der 3D-Objekte exakt berechnen kann? Die gibt es tatsächlich, dazu ein Beispiel.

Segmentlängenverteilungen Wir stellen uns vor, die Körner haben alle eine Nadelform, dann ist ausschließlich die Nadellängenverteilung von Interesse. Das gleiche gilt auch für andere dünne, rotationssymmetrische 3D-Objekte. Dann fassen wir die 3D-Objekte mathematisch als Geradensegmente auf, deren Längen zu bestimmen sind. Im 2D-Bild haben wir dann auch nur Geradensegmente, es ist aber unmöglich, von der Länge eines 2D-Geradensegmentes auf die Länge des 3D-Geradensegmentes zu schließen. Im Extremfall entartet das 2D-Geradensegment sogar zu einem Punkt. Wir nehmen an, beim Fallen der Nadeln vom Förderband sind alle Raumlagen der Nadeln gleichberechtigt. Da wir im 2D-Bild nur die Segmentlänge messen, ist diese unabhängig von einer Translation und Rotation bezüglich der Bildebene. Einzig der Winkel β zur z -Achse (Sichtachse) beeinflusst die Länge im 2D-Bild. Die „echte“ Länge eines Segmentes sei l und die Länge der Projektion sei s . Eine „zufällige räumliche Orientierung“ der Segmente bedeutet aber noch lange nicht, dass die Winkel β gleichverteilt sind. Dazu betrachten wir die Einheitskugel und Breitenkreise auf der Kugel. Wieviele Lagen von Segmenten mit einem Winkel β gibt es nun? Die Anzahl wird durch die Anzahl der Punkte (Bogenlänge) auf einem Breitenkreis bestimmt, der den Winkelabstand β zum Nordpol besitzt. Dieser Breitenkreis hat den Radius $\sin \beta$ und daher die Bogenlänge (Umfang) $2\pi \sin \beta$. Die „Summe“ aller Bogenlängen aller Breitenkreise der Nordhalbkugel ist

$$Z_{\text{ges}} = \int_0^{\frac{\pi}{2}} 2\pi \sin \beta d\beta = 2\pi, \quad (16.117)$$

was weiter nichts als die Oberfläche der Nordhalbkugel ist. Alle Segmente von 0 bis β (d. h. diese Teiloberfläche der Kugel) müssen wir relativ auf die gesamte Oberfläche der Nordhalbkugel beziehen, dann ist:

$$Z(\beta) = \frac{1}{Z_{\text{ges}}} \int_0^\beta 2\pi \sin \beta' d\beta' = 1 - \cos \beta. \quad (16.118)$$

Für ein Segment der Länge l und gegebenen Winkel β ist die Projektionslänge $s = l \cdot \sin \beta$, damit ist $\beta = \arcsin \frac{s}{l}$. Daher erhalten wir als Wahrscheinlichkeit, dass die Projektionslänge (bei festem l) unterhalb von s liegt:

$$P(l, S < s) = Z(\beta) = 1 - \cos \arcsin \frac{s}{l} = 1 - \sqrt{1 - \left(\frac{s}{l}\right)^2}. \quad (16.119)$$

Für $s > l$ gilt natürlich $P(l, S < s) = 1$. Damit haben wir die Verteilungsfunktion $F(l, s) = P(l, S < s)$ für ein Segment einer bestimmten, aber festen Länge l hergeleitet. Wir wollen aber eigentlich die Verteilung der Längen l ausrechnen, wenn viele Segmente mit unterschiedlichen Längen l gegeben sind. Nehmen wir einmal an, wir hätten 2 Segmente mit den Längen l_1 und l_2 , die mit den Wahrscheinlichkeiten h_1 und h_2 ($h_1 + h_2 = 1$) vorkommen, dann ist die Verteilungsfunktion einfach das gewichtete Mittel:

$$F(s) = h_1 \cdot F(l_1, s) + h_2 \cdot F(l_2, s). \quad (16.120)$$

Dies kann man nun auf eine beliebige diskrete Anzahl ausdehnen. Da wir aber die diskrete Anzahl nicht kennen, gehen wir nicht von Wahrscheinlichkeiten h_i aus, sondern von einer Dichtefunktion $h(l)$ und verallgemeinern die obige Beziehung zu:

$$F(s) = \int_0^\infty F(l, s) \cdot h(l) dl = \int_0^s h(l) dl + \int_s^\infty \left(1 - \sqrt{1 - \left(\frac{s}{l}\right)^2}\right) \cdot h(l) dl. \quad (16.121)$$

Da das erste Integral die Verteilungsfunktion $H(s)$ bezüglich der Dichte $h(l)$ ist, kann man die Gleichung auch in der Form

$$F(s) = H(s) + \int_s^\infty \left(1 - \sqrt{1 - \left(\frac{s}{l}\right)^2}\right) \cdot h(l) dl \quad (16.122)$$

schreiben. In dieser Gleichung ist die Verteilung $F(s)$ der projizierten Segmentlängen gegeben, diese wird statistisch aus dem 2D-Bild ermittelt. Gesucht ist dann die Dichtefunktion $h(l)$ bzw. die Verteilungsfunktion $H(l)$. Mathematisch stellt (16.122) eine *Integralgleichung* dar, die mit numerischen Methoden zu lösen ist.

Rotationsellipsoide Können wir dieses schöne Ergebnis noch verallgemeinern? Unsere praktischen Objekte müssen sehr dünn sein, damit man sie noch mit Gerdensegmenten modellieren kann. Wie ist es aber z. B. mit Bleistiften, Zigarren usw. Dazu modellieren wir theoretisch diese Objekte als Rotationsellipsoide mit der kleinen Achse B und der großen Achse A ($A \geq B$). Bei Parallelprojektion gehen Ellipsoide immer in Ellipsen über, daher ist die große Achse der Ellipse bei Parallelprojektion:

$$s = \sqrt{B^2 \cos^2 \beta + A^2 \sin^2 \beta}. \quad (16.123)$$

Nun müssen wir analog zu den Gerdensegmenten (16.123) nach β auflösen

$$\cos \beta = \sqrt{\frac{B^2 - s^2}{A^2 - B^2}} \quad (16.124)$$

und verfahren analog zu den Gerdensegmenten. In der analogen Integralgleichung ist nun β zu ersetzen. Weiterhin ersetzen wir jetzt l durch A und beachten in der Integralgleichung, dass immer $A \geq B$ sein muss. Wir erhalten dann:

$$\begin{aligned} F(s) &= \int_B^\infty F(A, s) \cdot h(A) dA \\ &= \int_B^s h(A) dA + \int_s^\infty \left(1 - \sqrt{\frac{A^2 - s^2}{A^2 - B^2}}\right) \cdot h(A) dA \\ s &\geq B. \end{aligned} \quad (16.125)$$

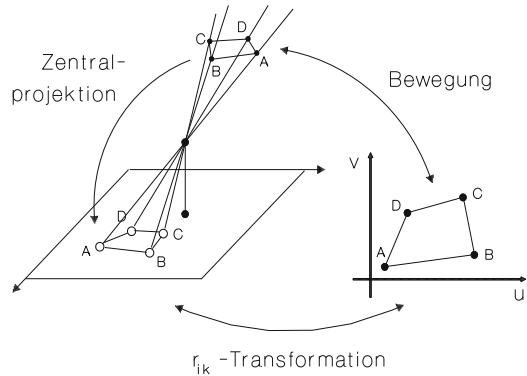
Für ein festes, gegebenes B müssen wir die Integralgleichung nach $h(A)$ auflösen. Wir könnten jetzt auch noch andere längliche Objekte betrachten, wenn es uns gelingt, die Projektionsfunktion s aufzustellen. Dem ganzen sind natürlich Grenzen gesetzt, wenn wir z. B. längliche (Prolaten) und breite Ellipsoide (Oblaten) vermischen, dann können wir die Projektionsfunktion nicht mehr berechnen.

16.8.2 Planare Vierecke

Wir wollen als erstes Beispiel planare Vierecke rekonstruieren. Monokular ist dies nur möglich, wenn wir genau die Form und Größe des Viereckes kennen. Dieses A-priori-Wissen legen wir durch ein CAD-Koordinatensystem fest, in dem alle vier Punkte bekannt sind $\Xi_k = (\xi_k, \eta_k, 1)^T$, $k = 1, 2, 3, 4$. Nehmen wir nun das reale Viereck mit einer kalibrierten Kamera auf, so werden den vier Eckpunkten im Raum die entsprechenden vier Bildpunkte zugeordnet:

$$\tilde{\mathbf{u}}_k = \mathbf{A}(\mathbf{x}_k - \mathbf{x}_c), \quad k = 1, 2, 3, 4. \quad (16.126)$$

Abb. 16.11 Rekonstruktion von planaren Vierpunkte-Konfigurationen



Nun ordnen wir die vier Punkte im CAD-System den vier Bildpunkten zu, dies bestimmt eindeutig eine projektive Transformation, beschrieben durch:

$$\tilde{\mathbf{u}}_k = \mathbf{B}\Xi_k, \quad k = 1, 2, 3, 4. \quad (16.127)$$

Man beachte, nicht die Vierecke werden zugrordnet, sondern die vier Punkte, d. h. man muss genau wissen, welcher Bildpunkt welchem CAD-Punkt entspricht. Die Sehstrahlen der vier Bildpunkte bestimmen eine vielseitige (unendliche) Sichtpyramide. Diese Sichtpyramide schneiden wir nun mit einer Ebene, die das CAD-Viereck enthält. Wir müssen also so schneiden, dass sich gerade die vier Bildpunkte ergeben. Dazu müssen wir natürlich das CAD-Koordinatensystem in die Raumbene legen, siehe Abb. 16.11. Die Eckpunkte des Viereckes beschreiben sich dann zu:

$$\mathbf{x}_k = \mathbf{x}_c + \mathbf{d} + \xi_k \mathbf{e}_1 + \eta_k \mathbf{e}_2 \quad (16.128)$$

mit der Bedingung $|\mathbf{e}_1| = |\mathbf{e}_2| = 1, \mathbf{e}_1^T \mathbf{e}_2 = 0$. Diese setzen wir nun in die Abbildungsgleichungen ein und erhalten:

$$\begin{aligned} \tilde{\mathbf{u}}_k &= \mathbf{A}(\mathbf{d} + \xi_k \mathbf{e}_1 + \eta_k \mathbf{e}_2), \quad k = 1, 2, 3, 4 \\ \tilde{\mathbf{u}}_k &= \mathbf{Ad} + \xi_k \mathbf{Ae}_1 + \eta_k \mathbf{Ae}_2, \quad k = 1, 2, 3, 4 \end{aligned} \quad (16.129)$$

Mit $\tilde{\mathbf{u}}_k = \mathbf{B}\Xi_k, \mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ ist dann auch:

$$\tilde{\mathbf{u}}_k = \xi_k \mathbf{b}_1 + \eta_k \mathbf{b}_2 + \mathbf{b}_3. \quad (16.130)$$

Folglich können wir ablesen:

$$\mathbf{Ad} = \mathbf{b}_3, \mathbf{Ae}_1 = \mathbf{b}_1, \mathbf{Ae}_2 = \mathbf{b}_2. \quad (16.131)$$

Die Gleichheit gilt aber nur bis auf einen gemeinsamen Faktor. Wir haben also simpel drei lineare Gleichungssysteme zu lösen. Nun müssen wir noch den Faktor eliminieren, dazu können die Zusatzbedingungen genutzt werden, z. B. $|\mathbf{e}_1| = 1$.

Bemerkung Stellt man sich ein Viereck als Schnittfigur mit der Sichtpyramide vor, so kann man sich leicht vorstellen, dass durch „verkippen“ und „verdrehen“ des Vierecks sich das gleiche Bild ergibt, d. h. man erhält eine weitere Rekonstruktionslösung. Dies ist aber nicht möglich, da wir vom CAD-System zum Bild Referenzpaare festgelegt haben, wir rekonstruieren also eine spezielle Vierpunkte-Konfiguration und nicht allgemein ein Viereck, dies gilt es zu beachten. Durch andere Referenzbeziehungen erhalten wir auch andere Lösungen oder es existiert keine.

16.8.3 Rechtecke

Das Rekonstruktionsproblem ist in Abb. 16.12 deutlich dargestellt. Ein Rechteck ist eine spezielles Viereck, deshalb könnten wir die Methode des vorigen Abschnittes nehmen. Dabei bekommen wir aber immer mehr Schwierigkeiten, die richtigen Referenzen zwischen Bild und CAD-System auszuwählen. Daher nutzen wir die zusätzlichen Eigenschaften eines Rechteckes, z. B. halbieren sich die Diagonalen bei deren Schnitt. Es seien $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ die vier Punkte des Rechteckes im Raum. Daher gilt für den Schnittpunkt:

$$\mathbf{x}_D = \frac{\mathbf{x}_1 + \mathbf{x}_3}{2} = \frac{\mathbf{x}_2 + \mathbf{x}_4}{2}. \quad (16.132)$$

Betrachten wir die Abbildungsgleichung und einen Sehstrahlvektor

$$\tilde{\mathbf{u}}_k = \mathbf{A}(\mathbf{x}_k - \mathbf{x}_c), \mathbf{x}_k = \mathbf{x}_c + \lambda \mathbf{t}_k, \quad (16.133)$$

so wissen wir bereits, dass wir für den Sehstrahlvektor $\mathbf{t}_k = \mathbf{A}^{-1}\tilde{\mathbf{u}}_k$ erhalten. Da die vier Punkte auf den Sehstrahlen liegen müssen, können wir auch schreiben:

$$\mathbf{x}_k = \mathbf{x}_c + \lambda_k \mathbf{A}^{-1} \tilde{\mathbf{u}}_k. \quad (16.134)$$

Wir müssen nun die vier λ_k bestimmen, d. h. wir müssen ausrechnen, für welche λ_k sich die Diagonalen der vier \mathbf{x}_k halbieren:

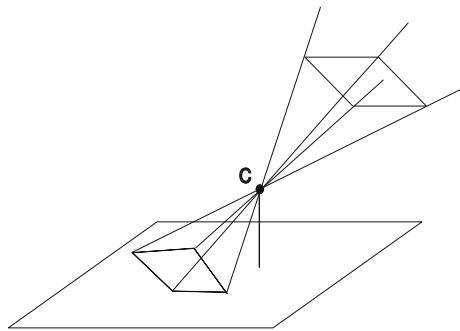
$$\begin{aligned} \mathbf{x}_D &= \frac{\mathbf{x}_c + \lambda_1 \mathbf{A}^{-1} \tilde{\mathbf{u}}_1 + \mathbf{x}_c + \lambda_3 \mathbf{A}^{-1} \tilde{\mathbf{u}}_3}{2} \\ &= \frac{\mathbf{x}_c + \lambda_2 \mathbf{A}^{-1} \tilde{\mathbf{u}}_2 + \mathbf{x}_c + \lambda_4 \mathbf{A}^{-1} \tilde{\mathbf{u}}_4}{2}. \end{aligned} \quad (16.135)$$

Es muss also gelten:

$$\lambda_1 \mathbf{A}^{-1} \tilde{\mathbf{u}}_1 + \lambda_3 \mathbf{A}^{-1} \tilde{\mathbf{u}}_3 - \lambda_2 \mathbf{A}^{-1} \tilde{\mathbf{u}}_2 - \lambda_4 \mathbf{A}^{-1} \tilde{\mathbf{u}}_4 = 0. \quad (16.136)$$

Abb. 16.12 Rekonstruktion

von Rechtecken



Wir haben jetzt ein unterbestimmtes lineares Gleichungssystem, d. h. 3 Gleichungen und 4 Unbekannte $\lambda_k, k = 1, 2, 3, 4$. Da es ein homogenes Gleichungssystem ist, sind alle Lösungen über einen gemeinsamen Faktor verbunden. Wir können z. B. $\lambda_4 = 1$ setzen und $\lambda_1, \lambda_2, \lambda_3$ ausrechnen. Wir nehmen an, wir hätten jetzt eine Lösung gefunden. Der gemeinsame Faktor bedeutet aber, wir können das Rechteck nun in der Sehpyramide parallel verschieben, d. h. die Tiefe ändern ohne dass sich die Bildpunkte ändern. Daher müssen wir noch die Fläche des Rechteckes als A-priori-Information nutzen. Wir können folglich die vier Punkte ansetzen zu:

$$\mathbf{x}_k = \mathbf{x}_c + \mu \lambda_k \mathbf{t}_k. \quad (16.137)$$

Der gemeinsame Faktor ist μ . Dieser ist nun aus der Fläche des Rechteckes zu bestimmen.

Bei der Rekonstruktion haben wir diesmal keine Referenzbeziehungen benötigt. Die Eigenschaft, dass sich die Diagonalen halbieren gilt auch für Parallelogramme. Wieso haben wir dann kein Parallelogramm rekonstruiert, bzw. gibt es zusätzliche Lösungen, die Parallelogramme sein könnten? Wir hatten angenommen, dass unsere Bildpunkte auch tatsächlich von einem beobachteten Rechteck stammen. Mit der Eigenschaft der Halbierung der Diagonalen haben wir eindeutig eine Lösung errechnen können, der Faktor bedeutet nur die Tiefenverschiebung des Objektes. Daher kann das Objekt nur ein Rechteck sein und kein Parallelogramm. Wenn zusätzlich zum rekonstruierten Rechteck noch ein Parallelogramm existieren würde, bedeutet dies, wir haben durch eine 2D projektive Transformation ein Rechteck in ein Parallelogramm überführt. Nun, vier Punkte bestimmen tatsächlich eine 2D projektive Transformation, aber speziell ein Rechteck kann nur durch eine affine Transformation in ein Parallelogramm überführt werden. Das ist kein Widerspruch, da eine affine Transformation eine Untergruppe der projektiven ist. Da wir aber eine Kamera mit endlichem Projektionszentrum angenommen haben, ist diese Entartung nicht möglich, eine affine Transformation würde dann bedeuten, dass das Projektionszentrum im Unendlichen liegt.

Natürlich können wir das gleiche Rekonstruktionsverfahren auch auf die Rekonstruktion von Parallelogrammen anwenden, dann wissen wir aber, dass die Bildpunkte von der Beobachtung eines Parallelogrammes stammen.

16.8.4 Kugeln

Eine Kugel mit bekanntem Radius r soll rekonstruiert werden, d. h. deren Lage im Weltkoordinatensystem berechnet werden. Für die Lage brauchen wir „nur“ die x , y , z -Koordinate des Mittelpunktes der Kugel zu berechnen. Eine Kugel wird durch das zentralprojektive Modell im Bild als eine Ellipse abgebildet. Nun können wir nicht einfach den Mittelpunkt der Ellipse nehmen, den Sehstrahl dieses Punktes berechnen und schlussfolgern, dass der Mittelpunkt der Kugel auf diesem Sehstrahl liegen muss. Dies ist falsch. Der Mittelpunkt der Kugel wird i. Allg. nicht in den Mittelpunkt der Ellipse abgebildet, siehe Abb. 16.13. Wir müssen uns anders helfen. Dazu stellen wir uns vor, wir hätten die Ellipse detektiert und die große Achse der Ellipse bestimmt. Damit haben wir die beiden Scheitelpunkte der Ellipse auf der großen Achse A und B bestimmt. Wir bestimmen nun die Sehstrahlen dieser beiden Scheitelpunkte, die Richtungsvektoren seien \mathbf{t}_A und \mathbf{t}_B . Auf Grund der „totalen“ Symmetrie einer Kugel sind die beiden Sehstrahlen nicht nur Tangenten an die Kugel, sondern der „winkelhalbierende“ Sehstrahl

$$\mathbf{t}_M = \mathbf{t}_A + \mathbf{t}_B \quad (16.138)$$

muss durch den Mittelpunkt der Kugel gehen. Also müssen wir „nur noch“ die Lage des Mittelpunktes auf diesem Sehstrahl bestimmen. Der Winkel zwischen \mathbf{t}_M und \mathbf{t}_A bzw. \mathbf{t}_B werde mit α bezeichnet. Den Winkel zwischen \mathbf{t}_A und \mathbf{t}_B berechnen wir zu:

$$\cos 2\alpha = \frac{\mathbf{t}_A^T \mathbf{t}_B}{|\mathbf{t}_A| \cdot |\mathbf{t}_B|}. \quad (16.139)$$

Wenn wir den zu berechnenden Mittelpunkt \mathbf{x}_M berechnen wollen

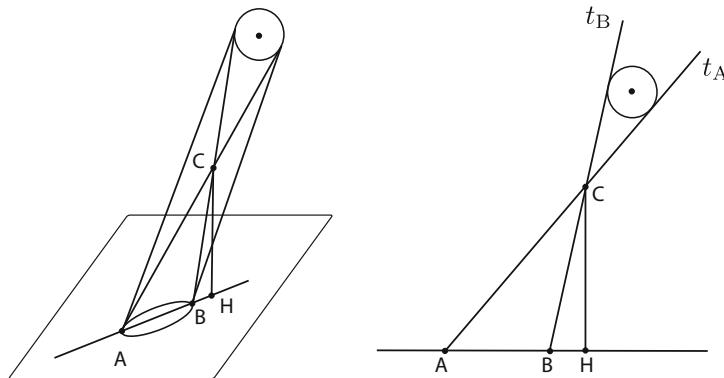
$$\mathbf{x}_M = \mathbf{x}_c + \lambda \mathbf{t}_M, \quad |\mathbf{t}_M| = 1, \quad (16.140)$$

müssen wir nur noch λ bestimmen. Der Sehstrahlen \mathbf{t}_A und \mathbf{t}_B sind tangent an die Kugel. Betrachten wir die tangentialen Punkte und verbinden diese mit dem Mittelpunkt, entstehen zwei rechtwinklige Dreiecke, die in einer gemeinsamen Ebene liegen. Wir beziehen uns nun auf eines dieser beiden rechtwinkligen Dreiecke und erhalten simpel:

$$\sin \alpha = \frac{r}{\lambda \cdot |\mathbf{t}_M|} = \frac{r}{\lambda}. \quad (16.141)$$

Diese Beziehung benutzen wir nun in:

$$\mathbf{x}_M = \mathbf{x}_c + \lambda \mathbf{t}_M = \mathbf{x}_M = \mathbf{x}_c + \frac{r}{\sin \alpha} \mathbf{t}_M = \mathbf{x}_c + \frac{r}{\sqrt{1 - \cos^2 \alpha}} \mathbf{t}_M \quad (16.142)$$

**Abb. 16.13** Abbildung einer Kugel

Mit $\cos^2 \alpha = \frac{\cos 2\alpha + 1}{2}$ erhalten wir schließlich:

$$\mathbf{x}_M = \mathbf{x}_c + \frac{\mathbf{t}_A + \mathbf{t}_B}{|\mathbf{t}_A + \mathbf{t}_B|} \frac{\sqrt{2}r}{\sqrt{1 - \frac{\mathbf{t}_A^T \mathbf{t}_B}{|\mathbf{t}_A| \cdot |\mathbf{t}_B|}}}. \quad (16.143)$$

Was ist also abschließend zu tun:

- Bestimme die Kontur des Objektes im Bild, das das Abbild der Kugel darstellt.
- Fitte an die Konturpunkte eine Ellipse und bestimme deren Scheitelpunkte auf der großen Achse.
- Berechne die Sehstrahlen dieser beiden Punkte mit den Kalibrierdaten der Kamera.
- Berechne nach obiger Formel die Lage des Mittelpunktes der Kugel.

Die Kugelrekonstruktion können wir uns auch bildlich vorstellen: Man bilde von der Kontur der Ellipse im Bild ausgehend den Sichtkegel, der einen Kreiskegel (in Normallage) darstellt. Man nehme nun die Kugel und „schmeiß“ sie von oben in den Kreiskegel. Dort wo die Kugel zur Ruhe kommt, ist die zu rekonstruierende Position der Kugel.

16.8.5 Kreisscheiben

Wesentlich schwieriger als Kugeln sind Kreisscheiben zu rekonstruieren. A-priori muss auch nur deren Radius r bekannt sein. Zur Rekonstruktion müssen wir aber außer dem Kreismittelpunkt \mathbf{x}_M noch deren Orientierung im Raum, z. B. den Normalenvektor \mathbf{n} berechnen. Eine Kreisscheibe im Raum ergibt natürlich im Bild eine Ellipse. Wir haben bezüglich des Mittelpunktes den gleichen Effekt wie bei Kugeln, der Mittelpunkt der Kreisscheibe wird i. Allg. nicht in den Mittelpunkt der Ellipse im Bild abgebildet. Wir können

uns aber bildlich wieder vorstellen: Man bilde von der Ellipse im Bild ausgehend den jetzt elliptischen Sichtkegel (muss kein Kreiskegel in Normallage sein) und schneide diesen Kegel so mit einer Ebene, dass eine Kreisscheibe vom Radius r entsteht.

Zunächst bestimmen wir als erstes wieder die Ellipse im Bild, beschrieben durch die Gleichung in homogenen Koordinaten:

$$\tilde{\mathbf{u}}^T \mathbf{E} \tilde{\mathbf{u}} = 0, \quad \mathbf{E} = \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix}. \quad (16.144)$$

Die Abbildungsgleichungen lauten wie üblich:

$$\tilde{\mathbf{u}} = \mathbf{A}\mathbf{x} + \mathbf{a} = \mathbf{A}(\mathbf{x} - \mathbf{x}_c). \quad (16.145)$$

Diese setzen wir in die Ellipsengleichung ein und erhalten damit die Beschreibung des elliptischen Sichtkegels in \mathbf{x} . Zur Einfachheit legen wir nun den Koordinatenursprung in das Projektionszentrum, damit reduzieren sich die Abbildungsgleichungen zu:

$$\tilde{\mathbf{u}} = \mathbf{A}\mathbf{x}. \quad (16.146)$$

Damit ergibt sich

$$\mathbf{x}^T \mathbf{A}^T \mathbf{E} \mathbf{A} \mathbf{x} = \mathbf{x}^T \mathbf{S} \mathbf{x} = 0 \quad (16.147)$$

als Beschreibung des elliptischen Sichtkegels. Wir schreiben nun die Normalform (in Hauptachsenlage) eines elliptischen Kegels auf:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0. \quad (16.148)$$

Um unseren Elliptischen Sichtkegel in diese Normalform zu transformieren, brauchen wir nur rotieren (nicht translatieren, da Koordinatenursprung im Projektionszentrum), also setzen wir an

$$\mathbf{x} = \mathbf{R}\mathbf{x}' \quad (16.149)$$

mit einer Rotationsmatrix \mathbf{R} . Diese Transformation setzen wir in die Gleichung des elliptischen Sichtkegels ein und erhalten:

$$\mathbf{x}'^T \mathbf{R}^T \mathbf{S} \mathbf{R} \mathbf{x}' = \mathbf{x}'^T \mathbf{S}' \mathbf{x}' = 0. \quad (16.150)$$

Die Matrix $\mathbf{S}' = \mathbf{R}^T \mathbf{S} \mathbf{R}$ muss, wenn es die obige Normalform werden soll, eine Diagonalmatrix werden:

$$\mathbf{S}' = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}, \quad \lambda_1, \lambda_2 > 0, \lambda_3 < 0. \quad (16.151)$$

Die Berechnung von \mathbf{R} und \mathbf{S}' geschieht wie üblich als Eigenwertproblem. Wenn wir nun diesen Kegel mit der Ebene $z' = \text{const}$ schneiden, ergibt sich immer noch kein Kreis, sondern eine Ellipse. Wir setzen im folgenden einmal $\lambda_1 \geq \lambda_2$ voraus, dies bedeutet, dass die große Achse der Schnittellipse die y' -Achse ist. Da der Schnitt mit der z' -Achse eine Kreisscheibe ergeben soll, müssen wir den Kegel jetzt einer speziellen Rotation unterziehen und würden die Normalform wieder verlassen. Wir brauchen doch nur den Kegel um einen Winkel φ bezüglich der y' -Achse drehen und zwar so, dass Schnitte mit der z -Achse Kreise ergeben und keine Ellipsen. Also rotieren wir $\mathbf{x}' = \mathbf{R}'\mathbf{x}''$ mit der speziellen Rotationsmatrix:

$$\mathbf{R}' = \begin{pmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{pmatrix}. \quad (16.152)$$

Wir setzen wieder ein und erhalten:

$$\mathbf{x}''^T \mathbf{R}'^T \mathbf{S}' \mathbf{R}' \mathbf{x}'' = \mathbf{x}''^T \mathbf{S}'' \mathbf{x}'' = 0. \quad (16.153)$$

Dabei errechnet sich \mathbf{S}'' zu:

$$\begin{pmatrix} \lambda_1 \cos^2 \varphi + \lambda_3 \sin^2 \varphi & 0 & (\lambda_1 - \lambda_3)/2 \sin 2\varphi \\ 0 & \lambda_2 & 0 \\ (\lambda_1 - \lambda_3)/2 \sin 2\varphi & 0 & \lambda_1 \sin^2 \varphi + \lambda_3 \cos^2 \varphi \end{pmatrix}. \quad (16.154)$$

Wir schreiben die Gleichung nochmals elementar ohne Matrixnotation auf:

$$\begin{aligned} &(\lambda_1 \cos^2 \varphi + \lambda_3 \sin^2 \varphi)x''^2 + \lambda_2 y''^2 \\ &+ (\lambda_1 \sin^2 \varphi + \lambda_3 \cos^2 \varphi)z''^2 + (\lambda_1 - \lambda_3)(\sin 2\varphi)x''z'' = 0. \end{aligned} \quad (16.155)$$

Nun schneiden wir diesen Kegel mit $z'' = z''_m = \text{const}$ so, dass sich ein Kreis ergeben soll. Die Bedingung dafür ist recht leicht abzulesen:

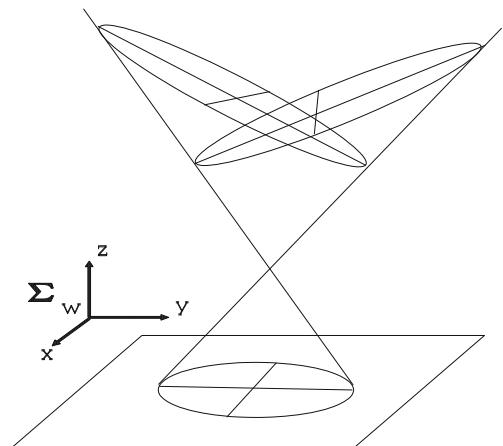
$$\lambda_2 = \lambda_1 \cos^2 \varphi + \lambda_3 \sin^2 \varphi. \quad (16.156)$$

Diese trigonometrische Gleichung in φ besitzt vier Lösungen:

$$\varphi_{1,2} = \pm \frac{1}{2} \arccos \frac{2\lambda_2 - \lambda_1 - \lambda_3}{\lambda_1 - \lambda_3}, \quad \varphi_3 = \varphi_1 + \pi, \quad \varphi_4 = \varphi_2 + \pi. \quad (16.157)$$

Die Unbekannte $z''_m = \text{const}$ können wir berechnen, da der Kreis ja einen vorgegebenen Radius r haben muss. Da die Bestimmungsgleichung in z''_m quadratisch ist, erhalten wir 2 Lösungen, also insgesamt $4 \cdot 2 = 8$ Lösungen. Für diese 8 Lösungen bestimmen wir die Lage des Mittelpunktes und der Normalenvektor in z'' -Richtung simpel angebar. Die 8 Lösungen fallen allerdings zu vier zusammen, 2 Lösungen vor und 2 Lösungen

Abb. 16.14 Rekonstruktion von Kreisscheiben



nach dem Projektionszentrum. Die 2 Lösungen vor dem Projektionszentrum schließen wir aus, so dass endgültig nur 2 Lösungen übrigbleiben, siehe Abb. 16.14. Die Lösungen im x'', y'', z'' -Koordinatensystem transformieren wir nun zurück durch Invertierung unserer Transformationen.

Der Begriff des Tensors wird in der Bildverarbeitung häufig verwendet. Das folgende Kapitel hat zum Ziel die mathematischen Grundlagen der Tensoralgebra vorzustellen, damit diese mathematisch exakt in der Bildverarbeitung eingesetzt werden können. Als Beispiele werden wir den trifokalen Tensor und den Strukturtensor betrachten.

17.1 Grundbegriffe der Tensoralgebra

Tensoren sind indizierte Größen, die eine physikalische oder geometrische Bedeutung besitzen. Diese Größen sind eigentlich unabhängig vom konkreten Koordinatensystem, ändern aber ihre Koordinatenbeschreibung bei Wechsel des Koordinatensystems. In der Bildverarbeitung kommt man im Prinzip mit dem Matrix- und Vektor-Kalkül aus. Nur bei Größen mit mehr als zwei Indizes bereitet die Lineare Algebra gewisse Schwierigkeiten, hier zeigt die Tensoralgebra erst ihre Vorteile. Auch die Schreibweise der Indizes der Tensoren unterscheidet sich etwas von der Schreibweise von Matrizen. Indizes von Tensoren können unten oder oben an Symbole geschrieben werden, dies hat eine tiefere Bedeutung und kann nicht wahllos erfolgen. Häufig wird die verbale Aussage gemacht: Tensoren sind im Prinzip mehrdimensionale Matrizen und umgekehrt. Dies ist natürlich nicht richtig, aber ganz falsch ist es auch nicht.

17.1.1 Vektorraum V

Den Begriff des Vektorraumes und die damit verbundenen Rechenregeln setzen wir als bekannt voraus. Zunächst betrachten wir endlichdimensionale Vektorräume V , d. h. diese Vektorräume besitzen eine endliche Basis. In unserer Vorstellung haben wir dabei den gewöhnlichen Vektorraum von n -Tupeln reeller Zahlen, wobei wir uns die Elemente aus V als Spaltenvektoren vorstellen. Vektoren, also Elemente eines Vektorraumes V , werden wir **fett**

darstellen, die Koordinaten eines Vektors dagegen nicht. Es sei also $\mathbf{x} \in V$, dann entwickeln wir \mathbf{x} nach einer Basis:

$$\mathbf{x} = \sum_{i=1}^n x^i \mathbf{e}_i. \quad (17.1)$$

Die Schreibweise ist nicht zufällig gewählt. Bei den Koordinaten x^i des Vektors \mathbf{x} schreiben wir stets die Indizes oben, bei den Basisvektoren \mathbf{e}_i dagegen unten.

17.1.2 Einsteinsche Summenschreibweise

Wir werden im Folgenden die Schreibweise aus der theoretischen Physik benutzen. Kommt also in einem Ausdruck ein Index in mehreren Faktoren gleichzeitig vor, bedeutet dies, dass über ihn summiert wird. So erhalten wir die Einsteinsche Schreibweise bezüglich der Basiszerlegung:

$$\mathbf{x} = x^i \mathbf{e}_i \stackrel{\text{def}}{=} \sum_{i=1}^n x^i \mathbf{e}_i. \quad (17.2)$$

Dies kann aber zu Verwechslungen führen, insbesondere wenn in einem Term mehrere Indizes gleich sind. Was bedeutet also g_{kk} , entweder $\sum_k g_{kk}$ oder einfach nur die Terme, bei denen der erste Index gleich dem zweiten ist? Dies muss aus dem Zusammenhang eindeutig hervorgehen.

17.1.3 Dualer Vektorraum V^*

Dieser Begriff wird nicht ganz so bekannt sein, daher erfolgt eine kurze Erläuterung. Wir betrachten die Menge aller linearen Vektorfunktionen $\varphi(\mathbf{x})$ über V , d. h.:

$$\varphi \in V^* : V \rightarrow \mathbb{R} \quad (17.3)$$

mit der Linearität

$$\varphi(\lambda \mathbf{x} + \mu \mathbf{y}) = \lambda \varphi(\mathbf{x}) + \mu \varphi(\mathbf{y}); \quad \lambda, \mu \in \mathbb{R}. \quad (17.4)$$

Die Menge aller dieser linearen Vektorfunktionen bildet selbst einen Vektorraum, den so genannten dualen Vektorraum V^* zu V . Dazu müssen die Regeln eines Vektorraumes selbst für die Funktionen gelten. Was ist also eine Addition zweier linearer Vektorfunktionen? Dies definiert man einfach zu:

$$(\alpha \varphi + \beta \psi)(\mathbf{x}) = \alpha \varphi(\mathbf{x}) + \beta \psi(\mathbf{x}). \quad (17.5)$$

Was steckt wieder anschaulich hinter den linearen Vektorfunktionen? Dies stellen wir uns vor als Skalarprodukt eines festen Zeilen-Vektors \mathbf{z} mit dem Spalten-Vektor \mathbf{x} . Jede lineare Vektorfunktion ist also eindeutig durch solch einen Zeilen-Vektor \mathbf{z} bestimmt.

Da nun V^* auch ein Vektorraum ist, hat er eine Basis, die von gleicher Dimension wie die von V ist. Wir wählen wie üblich eine kanonische Basis. Die Basiselemente sind also Vektorfunktionen. Diese Basiselemente (die abstrakt ja auch Vektoren sind) indizieren wir im Gegensatz zu V mit oberen Indizes, und die Koordinaten dagegen mit unteren Indizes (alles gegenläufig zu V). Wir wählen n Basis-Vektorfunktionen mit der folgenden Eigenschaft (notiert in der Einsteinschen Schreibweise):

$$\varphi = \varphi_i \mathbf{e}^i. \quad (17.6)$$

Dabei sind φ_i die Koordinaten (folglich Zahlen) und die Basiselemente \mathbf{e}^i sind lineare Vektorfunktionen aus V^* und nicht aus V für die gelten soll: $\mathbf{e}^i(\mathbf{e}_j) = \delta_j^i$. Unsere Schreibweise mit **fett** war nicht ganz konsequent, als Vektoren hätten wir auch φ fett schreiben müssen.

Wir wenden nun φ auf einen Vektor $\mathbf{x} \in V$ an:

$$\varphi(\mathbf{x}) = \varphi_i \mathbf{e}^i(\mathbf{x}) = \varphi_i \mathbf{e}^i(x^j \mathbf{e}_j) = \varphi_i x^i. \quad (17.7)$$

Gleichzeitig gilt

$$\varphi(\mathbf{x}) = \varphi(x^i \mathbf{e}_i) = x^i \varphi(\mathbf{e}_i) = x^i \varphi_i. \quad (17.8)$$

Damit ist $\varphi_i = \varphi(\mathbf{e}_i)$, d. h. die Koordinaten φ_i sind gerade die Abbildungen, also Zahlen, der Basisvektoren \mathbf{e}_i . Da der duale Vektorraum V^* in unserer Vorstellung identisch mit dem V ist, heißt dies für uns nur, daß wir in V eigentlich zwei Basissysteme gewählt haben, zur Unterscheidung schreiben wir die Basisvektoren einmal mit \mathbf{e}_i und einmal eben mit \mathbf{e}^i , genauso müssen wir dann die Koordinaten eines Vektors \mathbf{x} unterscheiden mit x^i und x_i . Da die Basissysteme für sich allein keine Orthonormalsysteme sein müssen, fordern wir (damit wir überhaupt eine günstige Eigenschaft zum Rechnen haben), dass die Systeme zueinander orthonormal sein sollen, dazu sagt man auch, sie seien *biorthogonale* Basissysteme. Diese Eigenschaft hatten wir oben als: „Wir wählen in V^* eine kanonische Basis mit“

$$\mathbf{e}^j(\mathbf{e}_i) = \delta_i^j \quad (17.9)$$

bezeichnet.

17.1.4 Multilineare Vektorfunktionen

Eine bilineare Vektorfunktion $\varphi(\mathbf{x}, \mathbf{y})$ ist linear in \mathbf{x} , wenn man \mathbf{y} festhält und in \mathbf{y} , wenn man \mathbf{x} festhält. Linearität ist wieder im Sinne eines linearen Operators zu verstehen. Daher

können wir schreiben:

$$\varphi(\mathbf{x}, \mathbf{y}) = \varphi(x^i \mathbf{e}_i, y^j \mathbf{e}_j) = \varphi(\mathbf{e}_i, \mathbf{e}_j) x^i y^j = \varphi_{ij} x^i y^j. \quad (17.10)$$

Ein typischer Vertreter einer bilinearen Vektorfunktion ist das Skalarprodukt zweier Vektoren $\varphi(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$. Multilineare Vektorfunktionen sind dann Vektorfunktionen in n Argumenten, die linear in jedem Argument sind, wenn man die $n - 1$ anderen Argumente festhält, z. B. mit drei Argumenten gilt:

$$\varphi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \varphi(x^i \mathbf{e}_i, y^j \mathbf{e}_j, z^k \mathbf{e}_k) = \varphi(\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k) x^i y^j z^k = \varphi_{ijk} x^i y^j z^k. \quad (17.11)$$

17.1.5 Multilineare Vektorabbildungen

Im Gegensatz zu den Vektorfunktionen betrachten wir jetzt lineare Vektorabbildungen, d. h. die Abbilder sind keine Zahlen mehr, sondern selbst Vektoren aus V :

$$\varphi : V \rightarrow V. \quad (17.12)$$

Da $\varphi(\mathbf{x})$ selbst ein Vektor ist, bezeichnen wir seine j -te Koordinate mit $y^j = \varphi^j(\mathbf{x})$ (also mit Index oben). Damit können wir gleich für die j -te Koordinate die obige Formel benutzen:

$$y^j = \varphi^j(\mathbf{x}) = \varphi^j(x^i \mathbf{e}_i) = x^i \varphi^j(\mathbf{e}_i) = \varphi_i^j x^i. \quad (17.13)$$

Für eine lineare Abbildung rechnen sich also die Koordinaten nach der Formel

$$y^j = \varphi_i^j x^i \quad (17.14)$$

um. Dies kennen wir schon von affinen Abbildungen ohne Verschiebung, wobei dann in Matrixschreibweise gilt: $\mathbf{y} = A \cdot \mathbf{x}$. Die Matrixelemente a_{ij} entsprechen dann gerade den Werten φ_j^i , also gerade der transponierten Matrix, wenn wir den unteren Index als Zeilenindex auffassen. Nun können wir beliebige multilineare Vektorabbildungen benutzen, z. B. bilineare Vektorabbildungen und können für die Koordinaten schreiben:

$$z^k = \varphi^k(\mathbf{x}, \mathbf{y}) = \varphi^k(x^i \mathbf{e}_i, y^j \mathbf{e}_j) = \varphi_{ij}^k x^i y^j. \quad (17.15)$$

Ein Beispiel für eine bilineare Vektorabbildung ist das gewöhnliche Vektorprodukt im dreidimensionalen Raum. Wir sehen, es besteht ein klares Konzept in der Schreibweise der Indizes.

17.1.6 Wechsel des Koordinatensystems

Viele physikalische als auch geometrische Größen sind eigentlich unabhängig vom konkret gewählten Koordinatensystem. Bei Änderung des Koordinatensystems ändern sich aber die Koordinaten der uns interessierenden Größen. Deshalb ist es interessant zu untersuchen, wie sich die Koordinaten bei Wechsel des Koordinatensystems ändern. Dazu schreiben wir auf, wie sich die n Basisvektoren ändern, d. h. wie aus ihnen ein anderes n -Bein gemacht wird. Damit ändern wir das Koordinatensystem durch eine affine Abbildung und nicht nur durch eine Rotation. Schreiben wir dies einmal in üblicher mathematischer Schreibweise auf:

$$\mathbf{e}'_i = \sum_{j=1}^n A_{ij} \mathbf{e}_j. \quad (17.16)$$

Jetzt ändern wir die Schreibweise, nutzen die Einsteinsche Abkürzung, und hängen den „Strich“ (dies ist für uns ungewöhnlich) an die Idizes:

$$\mathbf{e}'_i = A_{i'}^j \mathbf{e}_j. \quad (17.17)$$

Wir haben bei den Matrixelementen das j nach oben gezogen, dadurch ist die Einsteinsche Schreibweise wieder möglich. Wenn wir die Matrix A invertieren, so bezeichnen wir die Elemente der inversen Matrix mit A'^j_i , d. h. wir erkennen dies daran, wenn beim Index oben ein Strich steht. Mit dieser Konvention können wir die alten Basisvektoren durch die neuen ausdrücken durch:

$$\mathbf{e}_i = A'^j_i \mathbf{e}_{j'}. \quad (17.18)$$

Für uns ist nun interessant, wie sich die Koordinaten eines Vektors \mathbf{x} bei Änderung der Basis ändern, daher schreiben wir:

$$\mathbf{x} = x^i \mathbf{e}_i = x^{i'} \mathbf{e}'_{i'} = x^{i'} A_{i'}^j \mathbf{e}_j = x^i A'^j_i \mathbf{e}_{j'}. \quad (17.19)$$

Daraus können wir für die Koordinaten ablesen:

$$x^{i'} = A'^j_i x^j, x^i = A_{i'}^j x^{i'}. \quad (17.20)$$

Wenn wir nun die Basisvektoren transformieren, wie müssen sich dann die kontravarianten Basisvektoren transformieren, so dass nach der Transformation beide Systeme wieder biorthogonal sind? Wir nehmen einmal an, die biorthogonalen Basisvektoren würden sich nach der Vorschrift transformieren

$$\mathbf{e}^{i'} = A_j^{i'} \mathbf{e}^j, \quad (17.21)$$

also genauso wie die Koordinaten $x^{i'}$. Da das biorthogonale System eindeutig bestimmt ist, brauchen wir die Richtigkeit nur überprüfen, indem wir die Biorthogonalität zeigen. Also fragen wir jetzt, ob die Biorthogonalität im transformierten System noch richtig ist:

$$\langle \mathbf{e}^{j'}, \mathbf{e}_{i'} \rangle = \langle A_j^{j'} \mathbf{e}^j, A_{i'}^i \mathbf{e}_i \rangle = A_j^{j'} A_{i'}^i \langle \mathbf{e}^j, \mathbf{e}_i \rangle = A_j^{j'} A_{i'}^i \delta_i^j = \delta_{i'}^{j'}. \quad (17.22)$$

Fassen wir zusammen: beim Übergang zu einer neuen Basis transformieren sich die Basisvektoren und die Koordinaten folgendermaßen:

$$\mathbf{e}_{i'} = A_{i'}^i \mathbf{e}_i, \quad \mathbf{e}_i = A_i^{i'} \mathbf{e}_{i'}, \quad x^{i'} = A_i^{i'} x^i, \quad x^i = A_{i'}^i x^{i'}. \quad (17.23)$$

$$\mathbf{e}^{i'} = A_i^{i'} \mathbf{e}^i, \quad \mathbf{e}^i = A_{i'}^i \mathbf{e}^{i'}, \quad x_{i'} = A_{i'}^i x_i, \quad x_i = A_i^{i'} x_{i'}. \quad (17.24)$$

Formeln (17.23) und (17.24) drücken die wesentlichen Beziehungen aus, die für Tensoren wichtig sind und auf denen vieles beruht. Alles scheint sich gegenläufig zu transformieren. Im kovarianten Basissystem transformieren sich die Basisvektoren wie die kovarianten Koordinaten eines Vektors bezüglich des kontravarianten Basissystems und umgedreht. Daher auch der Sprachgebrauch „kovariant“, es transformiert sich wie Basisvektoren des Ausgangssystems, und „kontravariant“ wie die Basisvektoren des Biorthogonalsystems oder des kontravarianten Systems. Man sieht z. B. dass sich die transformierten Koordinaten aus der transponierten Matrix der inversen Matrix von A berechnen. Damit ergibt sich z. B. ein schöner Spezialfall für die Drehung des Koordinatensystems mittels einer Rotationsmatrix \mathbf{R} . Die Transponierte der Inversen Rotationsmatrix ist wieder \mathbf{R} selbst, daher transformieren sich die Basisvektoren und die Koordinaten in gleicher Art und Weise. Wenn wir nun noch den Spezialfall betrachten, dass unser Ausgangsbasisystem \mathbf{e}_i ein Orthonormalsystem ist und wir nur reine Rotationen betrachten, dann fällt alles in sich zusammen zu einer einzigen Transformation, d. h. das kontravariante Basissystem ist identisch mit dem kovarianten System und die Koordinaten transformieren sich genauso wie die Basisvektoren.

17.1.7 Tensoren

Indizierte Größen, also in unserer Bezeichnung kleine Buchstaben mit Indizes oben und/oder unten, nennen wir Tensoren, wenn sie sich in bestimmter Weise transformieren, sobald sich das Koordinatensystem ändert. Dies wird in der Literatur oft als *Invarianzforderung* bezeichnet. Es hat aber eigentlich mit Invarianten in dem Sinne, dass bestimmte Zahlen invariant sind, überhaupt nichts zu tun. Wie bei Vektoren und Matrizen müssten wir eigentlich streng unterscheiden zwischen der Bezeichnung eines Tensors a und seinen Koordinaten a_i . Da man aber oft noch unterscheiden muss, ist der Tensor kovariant, kontravariant, oder gemischt, ist es einfacher seine Koordinaten aufzuführen.

Beispiel 1

- Wir betrachten Vektoren und deren Koordinaten x^i , also einen Tensor, der einen Index oben trägt. Wie transformieren sich nun die Koordinaten x^i wenn sich das Koordinatensystem ändert? Dies hatten wir schon aufgeschrieben zu:

$$x^{i'} = A_i^{i'} x^i. \quad (17.25)$$

- Wenn sich also ein einstufiger Tensor mit einem Index oben so transformiert, dann nennen wir ihn einen *kontravarianten* Tensor. Man beachte, die $A_i^{i'}$ sind die Elemente der inversen Matrix und der untere Index „läuft“.

Beispiel 2

- Wir betrachten nun nicht die Koordinaten eines Vektors aus V , sondern eines Vektors aus V^* , dessen Koordinaten hatten wir mit φ_i bezeichnet. Wir wollen untersuchen, wie sich diese Koordinaten transformieren:

$$\varphi_{i'} = \varphi(\mathbf{e}_{i'}) = \varphi(A_{i'}^i \mathbf{e}_i) = A_{i'}^i \varphi_i. \quad (17.26)$$

- Damit erhalten wir die Koordinatentransformation:

$$\varphi_{i'} = A_{i'}^i \varphi_i \quad (17.27)$$

und bezeichnen den Tensor erster Stufe φ_i als einen *kovarianten* Tensor.

Es gibt also für einstufige Tensoren nur diese beiden Typen von Tensoren. Für mehrstufige Tensoren benötigen wir mehr Indizes und fordern für jeden Index eine weitere Transformation entsprechend der bisher aufgestellten Regeln. Nehmen wir einen zweistufigen, kovarianten Tensor. Dieser hat zwei untere Indizes und muss sich zweimal so transformieren wie der einstufige Tensor.

Beispiel 3

- Wir wählen einmal eine bilineare Vektorfunktion, die durch die Größen $\varphi_{ij} = \varphi(\mathbf{e}_i, \mathbf{e}_j)$ gekennzeichnet war. Wie transformieren sich diese nun?
- Wir können obige Regel für lineare Vektorfunktionen fast identisch übernehmen und erhalten

$$\varphi_{i'j'} = \varphi(\mathbf{e}_{i'}, \mathbf{e}_{j'}) = A_{i'}^i A_{j'}^j \varphi_{ij}, \quad (17.28)$$

folglich einen zweistufigen, kovarianten Tensor φ_{ij} .

Beispiel 4

- Wir wählen nun lineare Vektorabbildungen, für diese hatten wir eine Umrechnung der Koordinaten in der Form $y^j = \varphi_i^j x^i$ hergeleitet. Ist nun φ_i^j ein Tensor? Dazu müssen wir untersuchen, wie sich diese Größe bei Wechsel des Koordinatensystems transformiert.
- Wir betrachten die Bilder der Einheitsvektoren im transformierten System

$$\varphi(\mathbf{e}_{i'}) = \varphi_i^{j'} \mathbf{e}_{j'}. \quad (17.29)$$

- Nun versuchen wir diese durch die „alten“ Basisvektoren auszudrücken:

$$\begin{aligned} \varphi(\mathbf{e}_{i'}) &= \varphi(A_{i'}^i \mathbf{e}_i) = A_{i'}^i \varphi(\mathbf{e}_i) = A_{i'}^i \varphi_i^j \mathbf{e}_j = A_{i'}^i \varphi_i^j A_j^{j'} \mathbf{e}_{j'} \\ &= A_{i'}^i A_j^{j'} \varphi_i^j \mathbf{e}_{j'} \end{aligned} \quad (17.30)$$

- Vergleichen wir beide Ausdrücke, so können wir ablesen:

$$\varphi_i^{j'} = A_{i'}^i A_j^{j'} \varphi_i^j. \quad (17.31)$$

- Wir sehen: Der Tensor transformiert sich genauso wie wir es fordern, mit dem oberen Index kontravariant, und mit dem unteren Index kovariant. Demnach ist φ_i^j ein einstufig kontravarianter und einstufig kovarianter Tensor, damit aber trotzdem ein zweistufiger Tensor oder Tensor zweiter Stufe.
- Wenn wir a_i^j durch das Kroneckersymbol ersetzen, aber die Indizes genauso setzen, erhalten wir den Tensor δ_i^j . Wenn wir diesen für a_i^j in die Transformation einsetzen, sehen wir, dass δ_i^j sich direkt transformiert zu $\delta_{i'}^{j'}$. Damit ist dieser Tensor invariant gegenüber dem Wechsel des Koordinatensystems, folglich wird δ_i^j auch als Einheitstensor bezeichnet.

Beispiel 5

- Uns fehlt eigentlich nur noch ein Beispiel für einen zweistufigen, kontravarianten Tensor.
- Dazu bilden wir einfach das Tensorprodukt der Koordinaten zweier Vektoren, die ja für sich je einstufige kontravariante Tensoren sind. Das allgemeine Tensorprodukt werden wir noch einführen. Das Tensorprodukt ergibt sich einfach zu:

$$a^{ij} = x^i \cdot y^j. \quad (17.32)$$

- Wir sehen, es ist das Produkt aus allen möglichen Kombinationen. Dieses Tensorprodukt wird oft auch *dydisches* Produkt genannt, wir kennen es als das Produkt eines Spaltenvektors mit einem Zeilenvektor. Dieses ist nicht kommutativ, d. h. bei Vertauschung entsteht ein neuer Tensor.

17.2 Allgemeine Tensoren

Wenn sich eine indizierte Größe $a_{i_1 \dots i_k}^{j_1 \dots j_l}$ nach der Regel

$$a_{i'_1 \dots i'_k}^{j'_1 \dots j'_l} = A_{j_1}^{j'_1} \dots A_{j_l}^{j'_l} A_{i'_1}^{i_1} \dots A_{i'_k}^{i_k} a_{i_1 \dots i_k}^{j_1 \dots j_l} \quad (17.33)$$

transformiert, nennt man sie einen Tensor, der k -fach kovariant und l -fach kontravariant ist.

17.2.1 Tensoralgebra

Da wir bisher wissen, was Tensoren sind, wird es Zeit eine Algebra einzuführen, d. h. Operationen auf Tensoren, die wieder neue Tensoren ergeben. Es gibt nur vier Operationen:

- Addition (Subtraktion)
- Multiplikation
- Verjüngung
- Permutation
- eventuell: Überschiebung (ist aber nur die Hintereinanderausführung von Multiplikation und anschließender Verjüngung).

Addition Die Addition ist nur für gleichartige Tensoren erlaubt, man kann also nicht einen einstufigen und einen zweistufigen Tensor addieren. Man muss nur beweisen, dass nach der Addition wieder ein gleichartiger Tensor entsteht, d. h. dass dieser sich genau so transformiert wie wir das erwarten. Dazu ein

Beispiel

$$c_{ijk}^{pq} = a_{ijk}^{pq} + b_{ijk}^{pq}. \quad (17.34)$$

Der Beweis, dass sich c_{ijk}^{pq} genau so wie a_{ijk}^{pq} und b_{ijk}^{pq} bei Wechsel des Koordinatensystems transformiert ist nahezu trivial und klar. Die Subtraktion erklären wir als Addition, wobei der zweite Summand mit -1 multipliziert wird.

Multiplikation Die Multiplikation ist schon interessanter. Wir multiplizieren alle Kombinationen miteinander, so dass sich die Stufen des Tensors erhöhen. Die Reihenfolge der Multiplikation ist wichtig, es können verschiedene Tensoren entstehen. Also ein einstufiger, kontravarianter und zweistufiger, kovarianter Tensor multipliziert mit einem zweistufigen,

kontravarianten und dreistufigen, kovarianten Tensor ergibt einen dreistufigen, kontravarianten und einen fünfstufigen kovarianten Tensor. Dazu ein anderes Beispiel und die zugehörige Schreibweise:

$$c_{pqr}^{ij} = a_{pq}^i \cdot b_r^j. \quad (17.35)$$

Für die linke Seite von (17.35) müssten wir eigentlich zeigen, dass dies auch ein fünfstufiger Tensor ist. Da dies sehr einfach ist, wollen wir es der Exaktheit halber einmal tun. Die beiden gegebenen Tensoren transformieren sich durch:

$$a_{p'q'}^{i'} = A_i^{i'} A_{p'}^p A_{q'}^q a_{pq}^i, b_{r'}^{j'} = A_{r'}^r A_j^{j'} b_r^j. \quad (17.36)$$

Damit ist

$$c_{p'q'r'}^{i'j'} = a_{p'q'}^{i'} b_{r'}^{j'} = A_i^{i'} A_j^{j'} A_{p'}^p A_{q'}^q A_{r'}^r a_{pq}^i b_r^j = A_i^{i'} A_j^{j'} A_{p'}^p A_{q'}^q A_{r'}^r c_{pqr}^{ij}. \quad (17.37)$$

Wir sehen, der Beweis war wirklich simpel.

Damit können wir beliebig neue Tensoren erzeugen. Ein Beispiel war das sogenannte dyadische Produkt zweier einstufiger, kontravarianter Tensoren. Dazu ein weiteres

Beispiel

- Wenn sich ein Tensor als Produkt darstellt, dann können wir auch umgekehrt formulieren, dass er zerfällt. Es zerfalle z. B. ein einfacher kontravarianter und einfacher kovarianter Tensor in

$$d_j^i = b^i \cdot c_j. \quad (17.38)$$

- Wir erinnern uns, dass für die Koordinaten einer linearen Vektorabbildung gilt:

$$y^i = d_j^i x^j = b^i c_j x^j. \quad (17.39)$$

Die c_j sind ein kovarianter, einstufiger Tensor, folglich deutbar als die Koordinaten eines Vektors aus dem dualen Vektorraum V^* . Wir können somit formal $c_j = \varphi_j = \varphi(\mathbf{e}_j)$ setzen und damit ist $\varphi(\mathbf{x}) = \varphi_j x^j = c_j x^j$.

- Folglich erhalten wir:

$$y^i = d_j^i x^j = b^i c_j x^j = b^i \varphi(\mathbf{x}) \quad (17.40)$$

und damit

$$\mathbf{y} = \mathbf{b} \cdot \varphi(\mathbf{x}). \quad (17.41)$$

Diese Darstellung eines Vektors nennt man *Dyade*.

Verjüngung Die Verjüngung erzeugt das Gegenteil einer Multiplikation, der Tensor verliert an Stufen. Man kann oft mehrfach hintereinander verjüngen, deshalb interessiert uns jetzt nur eine Stufe der Verjüngung. Verjüngung heißt: man nehme genau einen Index oben und einen Index unten, setze sie gleich und summiere darüber. Der Tensor verliert nun zwei Stufen, eine kontravariante und eine kovariante Stufe. Man kann mehrmals verjüngen, aber immer nur nach diesem Schema. Also einen zweistufigen kovarianten Tensor kann man überhaupt nicht verjüngen, weil er oben gar keinen Index hat. Im Gegensatz zur einfachen Addition und Multiplikation von Tensoren muss man nun aber echt beweisen, dass auch nach der Verjüngung wieder ein Tensor entsteht, ansonsten wäre dies alles Unsinn. Wir führen den Beweis für ein Beispiel, die Verallgemeinerung ist selbsterklärend.

Beispiel Wir betrachten den dreistufig kontravarianten und zweistufig kovarianten Tensor a_{pq}^{ijk} und wollen ihn für bestimmte Indizes verjüngen, z. B.

$$a_q^{ik} = a_{rq}^{irk} \stackrel{\text{def}}{=} \sum_r a_{rq}^{irk}. \quad (17.42)$$

Wir verjüngen mit dem ersten Index unten und dem zweiten oben, jede andere Wahl ergibt wieder andere Tensoren. Wir zeigen nun, dass der verjüngte Tensor a_q^{ik} auch wirklich ein Tensor ist. Dazu betrachten wir die Transformation des nicht verjüngten Tensors:

$$a_{p'q'}^{i'j'k'} = A_i^{i'} A_j^{j'} A_k^{k'} A_p^p A_q^q A_{pq}^{ijk}. \quad (17.43)$$

Diesen verjüngen wir nun zu

$$a_{r'q'}^{i'r'k'} = A_i^{i'} A_j^{r'} A_k^{k'} A_{r'}^p A_q^q a_{pq}^{ijk}. \quad (17.44)$$

Nun ist aber gerade

$$A_j^{r'} \cdot A_{r'}^p = \delta_j^p. \quad (17.45)$$

Daraus folgt nun für $j = p = r$

$$a_{r'q'}^{i'r'k'} = A_i^{i'} A_k^{k'} A_q^q a_{rq}^{irk}. \quad (17.46)$$

Daher ist

$$a_{q'}^{i'k'} = A_i^{i'} A_k^{k'} A_q^q a_q^{ik}, \quad (17.47)$$

was zu zeigen war.

Es folgen nun weitere Beispiele.

Beispiel 1 Den Tensor φ_i^j können wir mit einer linearen Vektorabbildung $y = \varphi(\mathbf{x})$ identifizieren und damit als Elemente einer Matrix, die die Koordinaten eines Vektors in die eines anderen Vektors umrechnet. Um Verwechslungen vorzubeugen, bezeichnen wir den Tensor φ_i^j jetzt mit a_i^j . Damit erhalten wir als Verjüngung

$$a = a_i^i \quad (17.48)$$

ein Skalar, also einen Null-stufigen Tensor. Ein Skalar ist immer invariant gegenüber dem Wechsel des Koordinatensystems, folglich haben wir eine affine Invariante gefunden, nämlich die Spur der affinen Matrix. Um die Begriffe deutlich auseinanderzuhalten: der Begriff affine Invariante bezieht sich darauf, dass wir das Koordinatensystem affin transformieren (aus einem n -Bein wird ein anderes n -Bein). Die Spur der affinen Matrix dagegen bezieht sich auf die lineare Vektorabbildung, also auf eine affine Abbildung ohne Verschiebung, die den Tensor repräsentiert.

Beispiel 2 Statt einen Tensor zu verjüngen, kann man auch bestimmte, gegebene Ausdrücke auffassen als Verjüngung von Tensoren. Wir erinnern uns, die skalare Vektorfunktion $\varphi(\mathbf{x})$ lässt sich darstellen zu:

$$\varphi(\mathbf{x}) = \varphi_i x^i. \quad (17.49)$$

Wir sehen, dies kann man als Verjüngung des Tensors $\varphi_i^p = \varphi_i x^p$ auffassen. Damit ist $\varphi(\mathbf{x})$ eine skalare Invariante. Da die linearen Vektorfunktionen als Skalarprodukte gedeutet werden können, könnte man nun glauben, das Skalarprodukt sei affin invariant. Die linearen Vektorfunktionen aus dem dualen Raum sind zwar formal dem Skalarprodukt ähnlich, verhalten sich aber nicht so bei einer Transformationen. Wir transformieren ja gerade die Basisvektoren des Dualraumes gegenläufig zum Vektorraum, beim eigentlichen Skalarprodukt zweier Vektoren ist dies natürlich nicht so. Wir müssen also immer genau überlegen, was eine Verjüngung inhaltlich zu bedeuten hat. Können wir aber nun die Invariante $\varphi(\mathbf{x}) = \varphi_i x^i$ deuten? Dies ist recht einfach, dazu betrachten wir geometrisch eine Hyperebene, die nicht durch den Koordinatenursprung geht:

$$a_i x^i = 1. \quad (17.50)$$

Wenn wir die Koordinaten x^i affin transformieren, also $x^i = A_{i'}^i x^{i'}$ einsetzen, so erhalten wir $a_{i'} = A_{i'}^i a_i$, d. h. die Koeffizienten a_i verhalten sich tatsächlich wie ein kovarianter Tensor. Damit beschreibt unsere lineare Vektorfunktion die linke Seite einer Ebenengleichung und die skalare Invariante $\varphi(\mathbf{x}) = a_i x^i$ ist die rechte Seite der Ebenengleichung, die sich tatsächlich nicht ändert. Es ist

$$a_{i'} x^{i'} = 1 \quad (17.51)$$

die Ebenengleichung nach Wechsel des Koordinatensystems.

Beispiel 3 All dies gilt auch analog für multilinear Vektorfunktionen, z. B. ist $\varphi(\mathbf{x}, \mathbf{y}) = \varphi_{ij}x^i y^j$ die zweifache Verjüngung von $\varphi_{ij}^{pq} = \varphi_{ij}x^p y^q$. Man verwendet dafür eine abkürzende Sprechweise: Der Tensor φ_{ij} wird mit den Tensoren x^i und y^j überschoben, dies ist also dasselbe wie: „Der Tensor φ_{ij} wird mit den Tensoren x^p und y^q multipliziert und anschließend doppelt verjüngt“. Was bedeutet nun wieder geometrisch die Invariante $\varphi(\mathbf{x}, \mathbf{y}) = \varphi_{ij}x^i y^j$? Wir ahnen es schon in Verallgemeinerung, es sind quadratische Formen, die Kurven zweiter Ordnung beziehungsweise Hyperflächen zweiter Ordnung beschreiben:

$$\varphi_{ij}x^i y^j = 1. \quad (17.52)$$

Wenn wir wieder $x^i = A_{i'}^i x^{i'}$, $y^j = A_{j'}^j y^{j'}$ einsetzen, so sehen wir sofort, dass sich die Koeffizienten $a_{i'j'}$ zweifach kovariant transformieren, die Invariante ist wieder weiter nichts als die rechte Seite. Wir sehen, dies gilt nun sofort für alle Hyperflächen n -ter Ordnung bezüglich multilinearer Vektorfunktionen.

Permutationen Jede Permutation der Indizes eines Tensors ergibt einen neuen Tensor. Dies gilt aber nur, wenn die Indizes unten und/oder oben für sich permutiert werden, es dürfen also nicht Indizes von unten mit denen von oben gemeinsam in einer Permutation verändert werden, weil dies anschließend kein Tensor mehr ist. Das liegt daran, dass sich untere und obere Indizes anders transformieren. Dazu ein einfaches **Beispiel**:

- Wenn a^{ij} ein Tensor ist, dann auch a^{ji} .
- Wenn a_{ij} ein Tensor ist, dann auch a_{ji} .
- Wenn a_i^j ein Tensor ist, dann ist a_j^i kein Tensor mehr.

Wenn wir folglich zweistufige Tensoren als Matrizen auffassen, dann existiert nicht in jedem Falle die transponierte Matrix in dem Sinne, dass sie auch einen Tensor repräsentiert. In der Anwendung haben aber nur zwei Typen von Permutationen Bedeutung, nämlich die, die etwas mit symmetrischen und schiefsymmetrischen Tensoren zu tun haben.

• Die Symmetrisierung von Tensoren

Man wählt dazu entweder von den oberen oder den unteren Indizes N Stück aus, bildet alle Permutationen dieser N Indizes, und berechnet das arithmetische Mittel aller dieser Tensoren mit den permutierten Indizes. Beim Ergebnistensor setzen wir die permutierten Indizes in runde Klammern.

Beispiel $N = 2$:

$$a_{(ij)} = \frac{1}{2}(a_{ij} + a_{ji}). \quad (17.53)$$

Beispiel $N = 3$:

$$a_{(ijk)} = \frac{1}{6}(a_{ijk} + a_{jki} + a_{kij} + a_{jik} + a_{ikj} + a_{kji}). \quad (17.54)$$

Einen Tensor nennt man *symmetrisch* (bezüglich der N Indizes), wenn er für alle Permutationen der N Indizes gleich ist.

Im Falle $N = 2$ muss dann $a_{ij} = a_{ji}$ gelten, falls der Tensor symmetrisch ist.

- **Die Alternierung von Tensoren (Antisymmetrisierung)**

Es gilt die gleiche Regel wie bei der Symmetrisierung, nur dass gerade Permutationen positiv und ungerade Permutationen negativ in die Summierung eingehen, die Indizes schreiben wir jetzt in eckige Klammern (bei einer geraden Permutation ist die Anzahl der Inversionen gerade)

Beispiel $N = 2$:

$$a_{[ij]} = \frac{1}{2}(a_{ij} - a_{ji}). \quad (17.55)$$

Beispiel $N = 3$:

$$a_{[ijk]} = \frac{1}{6}(a_{ijk} + a_{jki} + a_{kij} - a_{jik} - a_{kji} - a_{ikj}). \quad (17.56)$$

Einen Tensor nennt man *schiefsymmetrisch*, wenn man eine ungerade Permutation der Indizes bildet, dann wechselt er sein Vorzeichen, bei einer geraden Permutation bleibt sein Vorzeichen erhalten. Oder was dasselbe ist, bildet man eine Transposition zweier Indizes, wechselt er sein Vorzeichen. Eine Transposition ist das Vertauschen zweier Indizes, dabei ändert sich die Anzahl der Inversionen immer um eine ungerade Anzahl, wie man sich leicht überlegen kann. Damit wird bei einer Transposition aus einer geraden Permutation eine ungerade und umgekehrt.

Beispiel $N = 2$:

$$a_{ij} = -a_{ji}. \quad (17.57)$$

Beispiel $N = 3$:

$$a_{ijk} = a_{jki} = a_{kij} = -a_{jik} = -a_{kji} = -a_{ikj}. \quad (17.58)$$

Selbstverständlich sind die alternierten Tensoren $a_{[...]}$ schiefsymmetrisch. Wir nehmen einmal an, ein Tensor sei schiefsymmetrisch. Wenn wir eine Transposition durchführen, also zwei Indizes vertauschen, wechselt der Tensor sein Vorzeichen. Da z. B.

$$a_{iik} = -a_{iik} \rightarrow a_{iik} = 0. \quad (17.59)$$

ist, folgt

$$a_{iik} = a_{ijj} = a_{kkk} = 0. \quad (17.60)$$

Dies können wir verallgemeinern zu n -Indizes eines schiefsymmetrischen Tensors:

Alle Koordinaten eines Tensors, bei denen mindestens zwei Indizes gleich sind, müssen verschwinden. Im Folgenden betrachten wir ein für die Praxis wichtiges Beispiel eines schiefsymmetrischen Tensors.

Beispiel Ein spezieller schiefsymmetrischer Tensor dritter Stufe ist der *Levi-Civita-Tensor*. Er wird häufig benutzt, um das **Vektorprodukt (Kreuzprodukt)** zweier Vektoren im V_3 elegant und kompakt aufzuschreiben, er wird mit ε_{ijk} bezeichnet. Da im 3D-Vektorraum und in der 3D-Bildverarbeitung das Vektorprodukt eine große Bedeutung hat (z. B. bei der sogenannten E-Matrix), sollen noch einige Ausführungen folgen. Dazu betrachten wir eine beliebige Basis \mathbf{e}_i und bilden das Vektorprodukt:

$$\mathbf{a} \times \mathbf{b} = (a^i \mathbf{e}_i) \times (b^j \mathbf{e}_j) = a^i b^j (\mathbf{e}_i \times \mathbf{e}_j). \quad (17.61)$$

Damit bestimmen die Produkte $\mathbf{e}_i \times \mathbf{e}_j$ das Vektorprodukt. Nehmen wir als Beispiel $\mathbf{e}_1 \times \mathbf{e}_2$, dieser Vektor steht auf beiden senkrecht, und dies macht ja gerade der Vektor \mathbf{e}^3 des biorthogonalen Systems. Jetzt normieren wir nur noch die Länge so, dass sich $\langle \mathbf{e}^3, \mathbf{e}_3 \rangle = 1$ ergibt. Damit können wir nun allgemein aufschreiben (dabei ist Spat eine Abkürzung für das bekannte Spatprodukt):

$$\mathbf{e}_i \times \mathbf{e}_j = \begin{cases} \text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \mathbf{e}^k, & i, j, k \text{ gerade Permutation} \\ -\text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \mathbf{e}^k, & i, j, k \text{ ungerade Permutation} \\ 0, & \text{sonst.} \end{cases} \quad (17.62)$$

Damit können wir das Vektorprodukt aufschreiben:

$$\mathbf{a} \times \mathbf{b} = \text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) [a^i b^j - a^j b^i] \mathbf{e}^k, \quad i, j, k \text{ gerade Permutation.} \quad (17.63)$$

Weiterhin ist

$$\mathbf{a} \times \mathbf{b} = C_1 \mathbf{e}^1 + C_2 \mathbf{e}^2 + C_3 \mathbf{e}^3 \quad (17.64)$$

mit

$$C_k = \text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) [a^i b^j - a^j b^i], \quad i, j, k \text{ gerade Permutation.} \quad (17.65)$$

Damit sind die C_k die Koordinaten des Ergebnisvektors, aber bezüglich der anderen biorthogonalen Basis, deshalb stehen die Indizes auch unten. Nun können wir die Koordinaten vertauschen und erhalten analog:

$$\mathbf{a} \times \mathbf{b} = \text{Spat}(\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3) [a_i b_j - a_j b_i] \mathbf{e}_k, \quad i, j, k \text{ gerade Permutation.} \quad (17.66)$$

Oder weiterhin

$$\mathbf{a} \times \mathbf{b} = C^1 \mathbf{e}_1 + C^2 \mathbf{e}_2 + C^3 \mathbf{e}_3 \quad (17.67)$$

mit

$$C^k = \text{Spat}(\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3)[a_i b_j - a_j b_i], \quad i, j, k \text{ gerade Permutation.} \quad (17.68)$$

Wir sehen, die Regel sieht fast so aus wie in kartesischen Koordinaten, es kommt nur noch das Spatprodukt als Faktor dazu und zu beachten sind die verschiedenen Basen. Daher können wir diese Regeln auch leicht wie üblich in Determinantenschreibweise formulieren:

$$\mathbf{a} \times \mathbf{b} = \text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \begin{vmatrix} \mathbf{e}^1 & \mathbf{e}^2 & \mathbf{e}^3 \\ a^1 & a^2 & a^3 \\ b^1 & b^2 & b^3 \end{vmatrix}. \quad (17.69)$$

Daraus lässt sich sogar gleich die Regel für das Spatprodukt selbst ableiten:

$$\text{Spat}(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \begin{vmatrix} a^1 & a^2 & a^3 \\ b^1 & b^2 & b^3 \\ c^1 & c^2 & c^3 \end{vmatrix}. \quad (17.70)$$

Durch Vertauschen der Indizes folgen die anderen Regeln. Nun wollen wir das Vektorprodukt in Tensorschreibweise darstellen. Dazu betrachten wir den schiefsymmetrischen, dreistufigen Tensor ε_{klm} . Man beachte, da wir im 3-dimensionalen Raum sind, kann jeder Index nur die Werte 1, 2, 3 annehmen und nicht mehr. Daher kann der Tensor nur $3^3 = 27$ Werte annehmen. Nach unseren allgemeinen Regeln für schiefsymmetrische Tensoren gilt nun:

$$\varepsilon_{123} = \varepsilon_{231} = \varepsilon_{312} = -\varepsilon_{132} = -\varepsilon_{321} = -\varepsilon_{213}. \quad (17.71)$$

Alle anderen Werte sind Null, weil dann mindestens zwei Indizes gleich sein müssen. Demzufolge hat dieser Tensor nur einen freien Parameter. Wenn wir z. B. ε_{123} irgendwie wählen, dann sind alle anderen fest bestimmt. Wenn wir den Tensor ε_{klm} nach unseren üblichen Tensorregeln transformieren, also das Koordinatensystem wechseln, wie transformiert sich eigentlich dann dieser eine freie Parameter? Dies kann man leicht aufschreiben und erhält:

$$\varepsilon_{l'2'3'} = \det(A) \varepsilon_{123}, \quad (17.72)$$

beziehungsweise

$$\varepsilon^{l'2'3'} = \det(A') \varepsilon^{123} \quad (17.73)$$

(A' war die inverse Transformationsmatrix). Wir wollen nun das Vektorprodukt mit diesem Tensor ausdrücken. Wir nehmen einamal an, es sei $\varepsilon_{123} = 1$ und wir hätten nur Orthogonalsysteme und nur Rotationen, so dass die Determinante immer +1 ist. Dann gilt offensichtlich:

$$[\mathbf{a} \times \mathbf{b}]_k = \varepsilon_{ijk} a^i b^j. \quad (17.74)$$

Umgekehrt natürlich auch:

$$[\mathbf{a} \times \mathbf{b}]^k = \varepsilon^{ijk} a_i b_j. \quad (17.75)$$

Dabei könnten wir jetzt die Indizes auch nur oben oder nur unten schreiben, da es in diesem Falle keinen Unterschied mehr zwischen kovarianten und kontravarianten Tensoren gibt.

Können wir diese Formeln auch für den affinen Fall benutzen? So einfach natürlich nicht, wir sehen, dass sich das Vektorprodukt mit dem Vorfaktor „Spatprodukt“ berechnet und damit ist obige Formel natürlich nicht richtig. Wir müssen also ε_{123} so wählen, dass das Spatprodukt berücksichtigt wird. Wie können wir das geschickt tun? Wie transformiert sich eigentlich das Spatprodukt der drei Basisvektoren? Dies kann man leicht ausrechnen und erhält:

$$\text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) = \det(A) \text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3), \quad (17.76)$$

bzw.

$$\text{Spat}(\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3) = \det(A') \text{Spat}(\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3). \quad (17.77)$$

Wir sehen, das Spatprodukt transformiert sich genauso wie ε_{123} , daher können wir für ε_{123} auch das Spatprodukt nutzen. Das Spatprodukt wollen wir nun nur noch der besseren Ausdruckweise wegen durch eine andere Größe ersetzen. Wir erinnern uns an den metrischen Tensor g_{ij} , der weiter nichts als die Skalarprodukte der Basisvektoren darstellt, analog g^{ij} , welche auch die Elemente der inversen Matrix zu g_{ij} sind. Der metrische Tensor g_{ij} ist nichts anderes als die Gramsche Matrix, siehe Abschnitt „Euklidischer Vektorraum“. Man kann nun zeigen:

$$g \stackrel{\text{def}}{=} [\text{Spat}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)]^2 = \det(g_{ij}) \quad (17.78)$$

und

$$\frac{1}{g} = [\text{Spat}(\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3)]^2 = \det(g^{ij}). \quad (17.79)$$

Folglich können wir das Spatprodukt durch die Wurzel aus der Determinante der Gramschen Matrix (Gramsche Determinante g) ersetzen, damit ergibt sich:

$$\varepsilon_{123} = \sqrt{g}, \varepsilon^{123} = \frac{1}{\sqrt{g}}. \quad (17.80)$$

Mit diesen Werten können wir also auch im affinen Fall die Tensornotationen (17.74), (17.75) für das Vektorprodukt benutzen.

Wenn wir uns g noch einmal anschauen und aufschreiben wie sich g transformiert, so sehen wir:

$$g' = [\det(A)]^2 \cdot g. \quad (17.81)$$

Damit ist g keine skalare Invariante und damit auch kein Tensor Nullter Stufe. Alle indizierten Größen, die sich mit einer Potenz der Determinante von A bzw. A' transformieren, heißen relative Tensoren vom Gewicht der entsprechenden Potenz. Unser g ist also ein relativer, nullstufiger Tensor vom Gewicht 2.

17.2.2 m -Vektoren

Definition 17.1 (Bivektor) *Einen zweifach kontravarianten, schiefsymmetrischen Tensor*

$$a^{ij} = -a^{ji} \quad (17.82)$$

nennen wir Bivektor. Ein Bivektor heißt zerlegbar oder zerfallend, wenn er aus zwei beliebigen Vektoren $\mathbf{a}_1, \mathbf{a}_2$ mit den Koordinaten a_1^i, a_2^j nach der Formel

$$a^{ij} = \frac{1}{2}(a_1^i a_2^j - a_1^j a_2^i) = \frac{1}{2} \begin{vmatrix} a_1^i & a_1^j \\ a_2^i & a_2^j \end{vmatrix} \quad (17.83)$$

entsteht. Der zerlegbare Bivektor a^{ij} entsteht also aus dem Tensorprodukt $a_1^i \cdot a_2^j$, wobei die Reihenfolge der Multiplikation wesentlich ist und anschließender Alternierung.

Definition 17.2 (Äußeres Produkt) *Ein zerlegbarer Bivektor, der gemäß der Reihenfolge $\mathbf{a}_1, \mathbf{a}_2$ gebildet wurde, heisst äußeres Produkt der Vektoren $\mathbf{a}_1, \mathbf{a}_2$. Es wird oft mit $[\mathbf{a}_1, \mathbf{a}_2]$ oder mit $\mathbf{a}_1 \wedge \mathbf{a}_2$ bezeichnet.*

Wir betrachten nun Eigenschaften des äußeren Produktes. Wir sehen sofort

$$[\mathbf{a}_1, \mathbf{a}_2] = -[\mathbf{a}_2, \mathbf{a}_1] \rightarrow [\mathbf{a}, \mathbf{a}] = 0. \quad (17.84)$$

Weiterhin sieht man leicht, dass das äußere Produkt linear in beiden Vektoren für sich ist, d. h. es ist bilinear, daher gilt z. B.

$$[\alpha \mathbf{a}_1, \mathbf{a}_2] = \alpha [\mathbf{a}_1, \mathbf{a}_2], \quad \alpha \in R \quad (17.85)$$

und

$$[\mathbf{a}_1 + \mathbf{a}_3, \mathbf{a}_2] = [\mathbf{a}_1, \mathbf{a}_2] + [\mathbf{a}_3, \mathbf{a}_2], \quad \alpha \in R. \quad (17.86)$$

Mit obiger Determinante als Minor sieht man leicht:

Zwei Vektoren $\mathbf{a}_1, \mathbf{a}_2$ sind genau dann linear abhängig, wenn das äußere Produkt verschwindet, d. h. $[\mathbf{a}_1, \mathbf{a}_2] = 0$. Das äußere Produkt können wir auch als Verallgemeinerung des gewöhnlichen Vektorproduktes $\mathbf{a}_1 \times \mathbf{a}_2$ im R^3 auffassen. Wählen wir also als Spezialfall einmal $n = 3$, so besteht der Bivektor a^{ij} zwar aus 9 Werten, auf Grund der Schiefsymmetrie aber nur aus 3 Werten. Wir können daher ablesen: a^{23} ist die x -Koordinate, a^{31} ist die y -Koordinate und a^{12} ist die z -Koordinate (wir sehen vom Faktor $\frac{1}{2}$ einmal ab). Wir wollen nun einmal zwei Vektoren $\mathbf{a}_1, \mathbf{a}_2$ linear transformieren, d. h.

$$\begin{aligned} \mathbf{a}_{1'} &= A_{1'}^1 \mathbf{a}_1 + A_{1'}^2 \mathbf{a}_2 \\ \mathbf{a}_{2'} &= A_{2'}^1 \mathbf{a}_1 + A_{2'}^2 \mathbf{a}_2. \end{aligned} \quad (17.87)$$

Wie transformiert sich nun das äußere Produkt:

$$[\mathbf{a}_{1'}, \mathbf{a}_{2'}] = A_{1'}^1 A_{2'}^2 [\mathbf{a}_1, \mathbf{a}_2] + A_{1'}^2 A_{2'}^1 [\mathbf{a}_2, \mathbf{a}_1] = \begin{vmatrix} A_{1'}^1 & A_{1'}^2 \\ A_{2'}^1 & A_{2'}^2 \end{vmatrix} [\mathbf{a}_1, \mathbf{a}_2]. \quad (17.88)$$

Wir sehen, es transformiert sich mit der Determinante der Transformation. Damit kann man allgemein feststellen:

Eine zweidimensionale Ebene im n -dimensionalen Raum ist durch das äußere Produkt zweier Vektoren in der Ebene bis auf einen Faktor eindeutig bestimmt. Alle parallelen Ebenen besitzen bis auf einen Faktor das gleiche äußere Produkt bzw. bis auf einen Faktor den gleichen, zerlegbaren Bivektor.

Alles über Bivektoren notierte kann sofort auf mehrere Vektoren übertragen werden. Es entstehen dann die m -Vektoren, z. B. Trivektoren usw.. Ein Trivektor ist demnach ein dreistufiger, schiefsymmetrischer, kontravarianter Tensor.

17.2.3 Rang von Tensoren

Eng mit der Zerlegbarkeit von Tensoren ist auch der Begriff des Ranges verwandt, der bei Matrizen aus der linearen Algebra allgemein bekannt ist. Dazu sei ein n -stufiger Tensor gegeben, ob er kovariant oder kontravariant ist, ist im Folgenden egal. Daher nehmen wir

als Beispiel einen N -stufigen kovarianten Tensor $a_{i_1 i_2 \dots i_N}$. Ohne Indizes bezeichnen wir den Tensor mit **a**. Der Tensor **a** hat den Rang 1, wenn gilt:

$$\mathbf{a} = \mathbf{u}_1 \otimes \mathbf{u}_2 \otimes \dots \otimes \mathbf{u}_N \leftrightarrow a_{i_1 i_2 \dots i_N} = u_{1 i_1} \cdot \dots \cdot u_{N i_N}, \quad (17.89)$$

d. h. wenn er sich als Tensorprodukt (dyadisches Produkt) von N einstufigen Tensoren (Vektoren) darstellen lässt. Den Rang R des Tensors **a** definieren wir dann als minimale Summe von Tensoren mit Rang 1:

$$\mathbf{a} = \sum_{r=1}^R \sigma_r u_1^{(r)} \otimes u_2^{(r)} \otimes \dots \otimes u_N^{(r)}. \quad (17.90)$$

Wir suchen folglich unter allen Zerlegungen das kleinste R . Wir sehen sofort, dass bei Matrizen rein äußerlich dies die SVD darstellt. Dort müssen allerdings die Vektoren $\mathbf{u}_i^{(r)}$ ein Orthonormalsystem bilden (i fest, r läuft). Dies muss bei der Rangzerlegung nicht gelten. Folglich ist eine SVD eine Rangzerlegung, aber eine beliebige Rangzerlegung ist noch keine SVD. Bei einer Matrix kennen wir noch die Begriffe Zeilenrang und Spaltenrang, das Minimum aus beiden ist der Rang der Matrix, wobei beide bei quadratischen Matrizen gleich sind. Den Begriff des Zeilen- beziehungsweise Spaltenranges kann man nun auch verallgemeinern. Dazu vereinbaren wir, dass der Wertebereich des Indexes i_n von 1 bis I_n läuft, d. h. es gibt für i_n genau I_n verschiedene Werte. Nun führen wir den Begriff **mode-n**-Vektor. Der **mode-n**-Vektor ergibt sich aus dem Tensor N -ter Stufe, indem wir den Index i_n variieren, aber gleichzeitig alle anderen Indizes festhalten. Je nachdem, welche Werte die anderen Indizes gerade haben, ergibt sich immer ein anderer Vektor oder Tensor 1. Stufe, nämlich der **mode-n**-Vektor. Wir können aber soviel **mode-n**-Vektoren bilden, wie wir eben Kombinationen unter den anderen Indizes bilden können. Nun können wir doch gedanklich alle diese **mode-n**-Vektoren anordnen als Spalten einer entstehenden Matrix $A_{(n)}$, dies nennt man **flattening** des Tensors. Wir müssen nur noch festlegen, in welcher Reihenfolge wir diese **mode-n**-Vektoren anordnen wollen. Wir legen die lexikografische Reihenfolge fest und setzen zyklisch fort. Nehmen wir als Beispiel einen 3-stufigen Tensor. Die Matrix $A_{(1)}$ entsteht, indem wir den 1. Index variieren und den 2. und 3. festhalten, aber systematisch verändern, so dass immer ein anderer Vektor entsteht. Also, der 2. und 3. Index variieren dann folgendermaßen:

$$(., 1, 1), (., 2, 1), \dots, (., I_2, 1), (., 1, 2), (., 2, 2), \dots, (., I_2, 2) \dots \quad (17.91)$$

Bei $A_{(2)}$ variieren die Indizes entsprechend:

$$(1, ., 1), (1, ., 2), \dots, (1, ., I_3), (2, ., 1), (2, ., 2), \dots, (2, ., I_3) \dots \quad (17.92)$$

Damit können wir den n -Rang von **a** definieren:

$$R_n = \text{Rang}_n(\mathbf{a}) = \text{Rang}(A_{(n)}). \quad (17.93)$$

Nun führen wir das Produkt zwischen einem Tensor und einer Matrix ein, das laut Literatur einen Tensor ergeben soll. Wenn wir dies im Folgenden anschauen, dann ist das weiter nichts als das überschieben des Tensors mit einem Tensor 2. Stufe, also Tensormultiplikation und eine anschließende Verjüngung. Da alle Indizes unten stehen, ist aber eine Verjüngung nur möglich, wenn wir nur orthogonale Basisysteme betrachten und nur Rotationen als Basiswechsel zulassen, sonst ist die Verjüngung in diesem Fall nicht richtig, beziehungsweise ergibt keinen Tensor. Unter diesen Annahmen notieren wir in der Einsteinischen Schreibweise:

$$(\mathbf{a} \otimes_n M)_{i_1, \dots, i_{n-1}, j_n, i_{n+1}, \dots, i_N} = a_{i_1, \dots, i_{n-1}, i_n, i_{n+1}, \dots, i_N} \cdot m_{j_n i_n}. \quad (17.94)$$

In allemeiner nicht-indizierter Schreibweise

$$\mathbf{b} = \mathbf{a} \otimes_n M, \quad (17.95)$$

oder direkt mit der Matrixalgebra

$$B_{(n)} = M \cdot A_{(n)}. \quad (17.96)$$

Natürlich setzen wir voraus, dass die Indizes, über die summiert wird, den gleichen Wertevorrat haben, dass alles „passt“. Nun kann man zwei sehr einfache Rechenregeln aufschreiben, dazu seien zwei Matrizen U und V gegeben (die passend sind bezüglich ihrer Dimensionen), so gilt:

- $\mathbf{a} \otimes_m U \otimes_n V = \mathbf{a} \otimes_n V \otimes_m U$
- $(\mathbf{a} \otimes_n U) \otimes_n V = \mathbf{a} \otimes_n (VU)$.

17.2.4 Zerlegung von Tensoren

Wir erinnern zunächst an die bekannte SVD einer geg. Matrix \mathbf{D} :

$$\mathbf{D} = \mathbf{U}_1 \Sigma \mathbf{U}_2^T, \quad (17.97)$$

wobei \mathbf{U}_1 spaltenorthogonal, Σ eine Diagonalmatrix und \mathbf{U}_2^T zeilenorthogonal ist. (Bei quadratischen Matrizen gilt: Ist die Matrix zeilenorthogonal, dann ist sie auch spaltenorthogonal und umgekehrt). Mit dem **mode-n**-Produkt kann man dies nun schreiben als:

$$\mathbf{D} = \Sigma \otimes_1 \mathbf{U}_1 \otimes_2 \mathbf{U}_2. \quad (17.98)$$

Dies lässt sich sehr einfach verifizieren. Nun kann man versuchen, diese Zerlegung auf N -stufige Tensoren zu erweitern und zwar in der Form:

$$\mathbf{d} = \mathbf{z} \otimes_1 U_1 \otimes_2 U_2 \cdots \otimes_N U_N. \quad (17.99)$$

Die Spalten aller U_j bilden je ein Orthonormalsystem, diese Verallgemeinerung nennt man **N-mode-SVD**. Das Analogon zur Diagonalmatrix Σ , der sogenannte **core tensor z**, muss nicht diagonal sein, er ist ein beliebiger Tensor. Die Matrizen U_n bestehen aus orthonormalen Spaltenvektoren, diese Spaltenvektoren spannen den gleichen Raum auf, wie die Spaltenvektoren von $D_{(n)}$, wobei diese aus dem *flattening* von \mathbf{d} entstehen. Der Tensor \mathbf{z} stellt dabei den Zusammenhang zwischen diesen orthonormalen Räumen her.

Algorithmus zur N-mode-SVD

- Berechne die Matrizen U_i , $i = 1, \dots, N$, indem wir für die Matrizen $D_{(i)}$, $i = 1, \dots, N$ die gewöhnliche SVD durchführen. Wir nehmen für U_i , $i = 1, \dots, N$ die „linken“ Matrizen (oft mit U bezeichnet, die „rechten“ oft mit V bezeichnet).
- Anschließend bestimmen wir den *core tensor z* durch

$$\mathbf{z} = \mathbf{d} \otimes_1 U_1^T \otimes_2 U_2^T \dots U_n^T \otimes_n \dots \otimes_N U_N^T. \quad (17.100)$$

17.3 Der Euklidische Raum

Bisher haben wir alle Überlegungen im affinen Raum durchgeführt, d. h. die Basis bildet ein n -Bein, das wieder in ein n -Bein übergeht und nichts mehr. Führen wir metrische Eigenschaften ein, z. B. Streckenmessungen, so sprechen wir vom Euklidischen Raum (hat zunächst nichts mit einer orthogonale Basis zu tun). Zur Streckenmessung führen wir das Skalarprodukt ein. Wir bezeichnen das Skalarprodukt wie üblich mit $\langle \mathbf{x}, \mathbf{y} \rangle$. Wir erinnern uns, eine bilineare Vektorfunktion war durch $\varphi_{ij} = \varphi(\mathbf{e}_i, \mathbf{e}_j)$ eindeutig bestimmt. Speziell für das Skalarprodukt bezeichnen wir diese jetzt mit g_{ij} , wobei durch die Symmetrieverforderung $g_{ij} = g_{ji}$ gelten muss. Die Größen g_{ij} werden auch *metrischer Tensor* genannt, in der Algebra und Analysis dagegen oft *Gramsche Matrix*. Im eigentlich Euklidischen Raum ist die Matrix, gebildet aus den Koordinaten g_{ij} , positiv definit, damit invertierbar. Wir bezeichnen die Elemente der inversen Matrix mit g^{ij} , sie transformieren sich (wie man leicht überprüfen kann) nach der Regel

$$g^{i'j'} = A_i^{i'} A_j^{j'} g^{ij} \quad (17.101)$$

und bilden demnach einen zweistufigen kontravarianten Tensor. Wir haben also jetzt im Euklidischen Raum im Gegensatz zum affinen Raum die beiden Tensoren g_{ij} und g^{ij} gegeben. Dies gestattet uns nun Indizes an Tensoren zu „senken“ oder zu „heben“. Wir zeigen dies an den Koordinaten von Vektoren:

$$x_i = g_{ij} x^j = \langle \mathbf{e}_i, \mathbf{e}_j \rangle x^j = \langle \mathbf{e}_i, x^j \mathbf{e}_j \rangle \rightarrow x_i = \langle \mathbf{x}, \mathbf{e}_i \rangle. \quad (17.102)$$

Das Skalarprodukt von \mathbf{x} mit den Basisvektoren führt zu den kovarianten Koordinaten eines Vektors. Umgekehrt, können wir diese wieder in die kontravarianten Koordinaten überführen:

$$x^i = g^{ij} x_j. \quad (17.103)$$

Wir wollen das Senken noch einmal plausibel mit den dualen Räumen V^* erklären. Unser Skalarprodukt in V , das wir eben eingeführt haben, können wir auch nochmals aufschreiben:

$$\varphi(\mathbf{x}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{x} \rangle = g_{ij} x^j x^i = |\mathbf{x}|^2. \quad (17.104)$$

Nun erinnern wir uns an den dualen Raum V^* und an die biorthogonalen Basissysteme. Eine lineare Vektorfunktion $\varphi(\mathbf{x})$ ist weiter nichts als: wir wählen aus V einen festen Vektor \mathbf{z} und bilden $\varphi(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle$. Für festes \mathbf{z} ist die lineare Vektorfunktion eindeutig bestimmt, für andere lineare Vektorfunktionen müssen wir eben ein anderes \mathbf{z} nehmen. Für die lineare Vektorfunktion hatten wir auf Grund der biorthogonalen System die Beziehung $\varphi(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle = \varphi_i x^i = z_i x^i$ ausgerechnet. Da wir für \mathbf{x} auch selbst \mathbf{z} einsetzen können, entsteht die Beziehung:

$$\varphi(\mathbf{z}) = \langle \mathbf{z}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{x} \rangle = x_i x^i = |\mathbf{x}|^2. \quad (17.105)$$

Wir vergleichen den Ausdruck mit der bilinearen Form des Skalarproduktes oben und können sofort ablesen:

$$x_i = g_{ij} x^j. \quad (17.106)$$

Daraus können wir nun formal die Regel für das „Senken“ oder „Heben“ eines Index bei einem beliebigen Tensor ableiten. Dazu müssen wir gedanklich alle Indizes durchnummerieren, d. h. wenn z. B. der dritte Index oben steht, dann muss der dritte unten „leer“ bleiben, dazu ein Beispiel:

$$a_{ij,l}^{..k..}. \quad (17.107)$$

Wir wollen den ersten Index einmal „heben“, so gilt

$$a_{..j,l}^{i..k..} = g^{ip} a_{pj,l}^{..k..}. \quad (17.108)$$

oder wir „senken“ den dritten Index

$$a_{ijk,l}^{..} = g_{kp} a_{ij,l}^{..p..}. \quad (17.109)$$

Ein wichtiges Beispiel ist der metrische Tensor selbst g_{ij} . Heben wir z. B. den zweiten Index, so ist

$$g_{i..}^{..j} = g^{jp} g_{ip} = \delta_i^j. \quad (17.110)$$

Heben wir jetzt auch noch den ersten Index, so ist

$$g_{..}^{ij} = g^{ip} \delta_p^j = g^{ij}. \quad (17.111)$$

Betrachten wir noch als Beispiel eine lineare Vektorabbildung, die durch den Tensor a_j^i beschrieben wurde. Mit der Nummerierung $a_{\cdot j}^i$ lautete die lineare Vektorabbildung in Koordinatenschreibweise

$$y^i = a_{\cdot j}^i x^j. \quad (17.112)$$

„Senken“ wir den ersten Index i, so ist

$$y_i = a_{ij} x^j \quad \text{mit } a_{ij} = g_{ip} a_{\cdot j}^p. \quad (17.113)$$

Gehen wir nun davon aus, dass als Basis ein orthonormiertes n -Bein vorliegt, so soll dies natürlich auch vorliegen, wenn wir das Koordinatensystem wechseln. Daher vereinfachen sich nun alle Formeln wesentlich

$$x_i = g_{ij} x^j = \delta_{ij} x^j = x^i \quad (17.114)$$

und damit gilt auch

$$A_{i'}^i = A_i^{i'}. \quad (17.115)$$

Damit ist die affine Matrix eine Orthogonalmatrix, sowohl Vektoren als auch Koordinaten transformieren sich also mit genau der gleichen Matrix. Für diesen Fall verschwindet nun auch der Unterschied zwischen kovarianten und kontravarianten Indizes vollständig.

17.4 Tensorfelder

Ein Tensorfeld liegt vor, wenn der Tensor ortsabhängig ist, d. h. er ist eine Funktion der Punkte P eines Gebietes D eines vorgegebenen Raumes. Allgemein könnten wir also notieren:

$$a_{i_1 \dots i_k}^{j_1 \dots j_l} = a_{i_1 \dots i_k}^{j_1 \dots j_l}(P), \quad (17.116)$$

oder wenn ein Koordinatensystem gegeben ist, dann:

$$a_{i_1 \dots i_k}^{j_1 \dots j_l} = a_{i_1 \dots i_k}^{j_1 \dots j_l}(x^1, x^2, \dots, x^n) \quad (17.117)$$

mit Koordinaten x^1, \dots, x^n bezüglich der kovarianten Basis e_i . Elementare Operationen in Tensorfeldern sind natürlich Ableitungsoperationen. Als erstes bilden wir das totale Differential, wobei für den Index i die Einsteinsche Schreibweise verwendet wird:

$$da_{i_1 \dots i_k}^{j_1 \dots j_l} = \frac{\partial a_{i_1 \dots i_k}^{j_1 \dots j_l}(x^1, x^2, \dots, x^n)}{\partial x^i} dx^i. \quad (17.118)$$

Wir behaupten, das totale Differential ist ein Tensor des gleichen Typs wie $a_{i_1 \dots i_k}^{j_1 \dots j_l}$. Dies lässt sich leicht mit Hilfe der Transformationsregel zeigen:

$$a_{i'_1 \dots i'_k}^{j'_1 \dots j'_l}(P) = A_{j_1}^{j'_1} \dots A_{j_l}^{j'_l} \dots a_{i_1 \dots i_k}^{j_1 \dots j_l}(P). \quad (17.119)$$

Wir bilden davon einfach das Differential und beachten, dass die Transformationskoeffizienten A_i^j Konstanten sind und erhalten demzufolge

$$da_{i'_1 \dots i'_k}^{j'_1 \dots j'_l}(P) = A_{j_1}^{j'_1} \dots A_{j_l}^{j'_l} \dots da_{i_1 \dots i_k}^{j_1 \dots j_l}(P), \quad (17.120)$$

was zu zeigen war.

Wir betrachten nun in jedem Punkt P alle partiellen Ableitungen des Tensors und bezeichnen diese mit:

$$\nabla_i a_{i_1 \dots i_k}^{j_1 \dots j_l} = \frac{\partial a_{i_1 \dots i_k}^{j_1 \dots j_l}(x^1, x^2, \dots, x^n)}{\partial x^i}. \quad (17.121)$$

Nun gilt die interessante Schlussfolgerung: Die $\nabla_i a_{i_1 \dots i_k}^{j_1 \dots j_l}$ bilden selbst einen Tensor mit der gleichen Anzahl oberer Indizes, während sich die Anzahl der unteren Indizes um Eins vergrößert. Die Vergrößerung der unteren Indizes wird durch den Differentiationsindex i hervorgerufen.

Auch dieses Ergebnis ist einfach zu zeigen. Es ist

$$\frac{\partial a_{i'_1 \dots i'_k}^{j'_1 \dots j'_l}(P)}{\partial x^{i'}} = A_{j_1}^{j'_1} \dots A_{j_l}^{j'_l} \dots \frac{\partial a_{i_1 \dots i_k}^{j_1 \dots j_l}(P)}{\partial x^{i'}} = A_{j_1}^{j'_1} \dots A_{j_l}^{j'_l} \dots \frac{\partial a_{i_1 \dots i_k}^{j_1 \dots j_l}(P)}{\partial x^i} \cdot \frac{\partial x^i}{\partial x^{i'}}. \quad (17.122)$$

(Beachtung: Kettenregel, über i ist zu summieren)

Da aber $x^i = A_{i'}^i x^{i'}$ und somit $\frac{\partial x^i}{\partial x^{i'}} = A_{i'}^i$ ist, setzen wir dies ein und erhalten:

$$\nabla_{i'} a_{i'_1 \dots i'_k}^{j'_1 \dots j'_l}(P) = A_{j_1}^{j'_1} \dots A_{j_l}^{j'_l} \dots A_{i'}^i \nabla_i a_{i_1 \dots i_k}^{j_1 \dots j_l}(P), \quad (17.123)$$

sodass sich der Differentiationsindex i tatsächlich kovariant transformiert.

Bilden wir als Spezialfall einmal den Gradienten eines Skalarfeldes $a(P)$, so erhalten wir $\nabla_i a$ als einen kovarianten Tensor 1. Stufe. Die Koordinaten $\nabla_i a$ dieses Vektors beziehen sich folglich auf das kontravariante Basissystem, wir müssten also schreiben

$$\nabla a = \nabla_i a e^i. \quad (17.124)$$

Da wir Vektoren aus V immer mit Spaltenvektoren identifiziert haben und Vektoren aus V^* mit Zeilenvektoren, müssten wir im Matrixkalkül den Gradienten eigentlich immer als Zeilenvektor betrachten. Wenn wir folglich weiter eine Vektorfunktion betrachten $\mathbf{a} = a^i \mathbf{e}_i$, dann können wir auch den Gradienten einer Vektorfunktion aufschreiben und erhalten den zweistufigen Tensor $\nabla_i a^j$, im Matrixkalkül Jakobimatrix genannt. Wir sehen, dies ist ein einstufiger kovarianter und einstufig kontravarianter Tensor zweiter Stufe.

17.5 Anwendungen in der Bildverarbeitung

17.5.1 Der trifokale Tensor

Der trifokale Tensor beschreibt das Problem der Fundamentalmatrix für drei Kameras, folglich die Epipolargeometrie bezüglich drei Kameras. Dabei bietet sich die Tensornotation an.

Zunächst soll der trifokale Tensor in üblicher Notation hergeleitet werden. Als elementare Feststellung gilt folgendes:

Schneidet eine 3D-Ebene die Bildebene und wir haben die Bildgerade \tilde{l} bestimmt, so lässt sich auch die 3D-Ebene eindeutig beschreiben. Mit

$$\tilde{l}^T \tilde{u} = 0, \quad \tilde{u} = Px + p = \tilde{P}\tilde{x} \quad (17.125)$$

ist

$$\tilde{l}^T \tilde{P}\tilde{x} = 0, \quad \tilde{x}^T \tilde{P}^T \tilde{l} = 0, \quad (17.126)$$

so dass die 3D-Ebene durch

$$\tilde{m} = \tilde{P}^T \tilde{l} = \begin{pmatrix} P^T \tilde{l} \\ p^T \tilde{l} \end{pmatrix} \quad (17.127)$$

beschrieben ist.

Nun seien 3 Kameras gegeben. Wenn wir Eigenschaften untersuchen, die 3D-projektiv invariant sind, dann können wir die 3 Projektionsmatrizen folgendermaßen aufschreiben:

$$\tilde{P} = [I, 0], \quad \tilde{P}' = [A, a_4], \quad \tilde{P}'' = [B, b_4]. \quad (17.128)$$

Nun betrachten wir eine Gerade im Raum und deren Abbilder $\tilde{l}, \tilde{l}', \tilde{l}''$ in den 3 Kameras. Dann können wir zu jeder Geraden die Ebenen aufschreiben

$$\tilde{m}_1 = \begin{pmatrix} \tilde{l} \\ 0 \end{pmatrix}, \quad \tilde{m}_2 = \begin{pmatrix} A^T \tilde{l}' \\ a_4^T \tilde{l}' \end{pmatrix}, \quad \tilde{m}_3 = \begin{pmatrix} B^T \tilde{l}'' \\ b_4^T \tilde{l}'' \end{pmatrix}. \quad (17.129)$$

Diese 3 Ebenen müssen sich in der einzigen 3D-Geraden schneiden, deshalb muss die (4×3) -Matrix (m_1, m_2, m_3) den Rang 2 haben, da sich 3 Ebenen normalerweise nicht in einer Geraden schneiden. Daher können wir für diese Rangbedingung sofort die 4. Zeile dieser Matrix nutzen und schreiben:

$$\alpha a_4^T \tilde{l}' + \beta b_4^T \tilde{l}'' = 0. \quad (17.130)$$

Daher können wir setzen

$$\alpha = C(b_4^T \tilde{l}''), \quad \beta = -C(a_4^T \tilde{l}') \quad (17.131)$$

mit einem Faktor C . Wenn wir dies nun in $m_1 = \alpha m_2 + \beta m_3$ einsetzen, erhalten wir die Bedingung für die ersten 3 Zeilen:

$$\tilde{l} = C b_4^T \tilde{l}'' A^T \tilde{l}' - C a_4^T \tilde{l}' B^T \tilde{l}'' = (\tilde{l}''^T b_4) A^T \tilde{l}' - (\tilde{l}'^T a_4) B^T \tilde{l}'' . \quad (17.132)$$

Da die linke Seite bis auf einen Faktor bestimmt ist, lassen wir C einfach weg. Nun schreiben wir einfach die Komponenten von \tilde{l} auf:

$$\tilde{l}_i = \tilde{l}''^T (b_4 a_i^T) \tilde{l}' - \tilde{l}'^T (a_4 b_i^T) \tilde{l}'' = \tilde{l}''^T (a_i b_4^T) \tilde{l}'' - \tilde{l}'^T (a_4 b_i^T) \tilde{l}'' . \quad (17.133)$$

Mit der Abkürzung

$$T_i = a_i b_4^T - a_4 b_i^T, \quad i = 1, 2, 3 \quad (17.134)$$

können wir schreiben:

$$\tilde{l}_i = \tilde{l}''^T T_i \tilde{l}'', \quad i = 1, 2, 3 . \quad (17.135)$$

Wir haben also eine Beziehung zwischen allen drei Bildgeraden gefunden. Diese Menge von 3 Matrizen $\{T_1, T_2, T_3\}$ nennt man den **trifokalen Tensor** in Matrixnotation. Da wir 3 Matrizen haben, haben wir Schwierigkeiten den trifokalen Tensor als ganzes zu bezeichnen, so wird oft versucht:

$$\tilde{l}^T = \tilde{l}''^T [T_1, T_2, T_3] \tilde{l}'' . \quad (17.136)$$

Besser ist es die Tensorschreibweise zu benutzen, was wir im Folgenden machen wollen.

Wenn wir im 2D-projektiven Raum arbeiten, dann gehen wir einfach zu homogenen Koordinaten über und befinden uns im 3D-Vektorraum. Analog zum 3D-projektiven Raum nehmen wir homogene Koordinaten und sind im 4D-Vektorraum. Wechsel der Basis bedeutet dann einfach eine Homographie, Verschiebungen gibt es ja nicht extra zu modellieren. Wir können also vollständig unsere Notation mit den A_i^j bei Wechsel der Basis benutzen. Weiterhin wissen wir, dass eine lineare Abbildung ein einfacher kovarianter und einfacher kontravarianter Tensor ist, dies gilt nun für eine Homographie. Unsere Matrixnotation

$$\tilde{x}' = H \tilde{x} \quad (17.137)$$

für eine Homographie geht dann über in die Tensorschreibweise

$$\tilde{x}^{i'} = h_j^i \tilde{x}^j , \quad (17.138)$$

wobei in dieser Notation i eine Zeile und j eine Spalte von H kennzeichnet. Vektoren \tilde{x} sind wie immer kontravariant, die Koeffizienten von Geraden und Ebenen \tilde{l}_k sind wie immer dagegen kovariante Tensoren. Damit werden Geraden durch eine Homographie mit

$$\tilde{l}_i = h_i^j \tilde{l}_j \quad (17.139)$$

transformiert. Nun wollen wir den trifokalen Tensor T_i

$$T_i = a_i b_4^T - a_4 b_i^T, \quad i = 1, 2, 3 \quad (17.140)$$

umschreiben zu

$$t_i^{jk} = a_i^j b_4^k - a_4^j b_i^k \quad (17.141)$$

und

$$\tilde{l}_i = \tilde{l}'^T T_i \tilde{l}'', \quad i = 1, 2, 3 \quad (17.142)$$

wird zu

$$\tilde{l}_i = \tilde{l}'_j \tilde{l}''_k t_i^{jk}. \quad (17.143)$$

Daher können wir schreiben

$$\tilde{l}_i = \tilde{l}'_j \tilde{l}''_k t_i^{jk} = \tilde{l}''_k (\tilde{l}'_j t_i^{jk}) = \tilde{l}''_k h_i^k \quad (h_i^k = \tilde{l}'_j t_i^{jk}). \quad (17.144)$$

Daher ist dies eine Homographie zwischen dem 1. und dem 3. Bild:

$$\tilde{x}^{k''} = h_i^k \tilde{x}^i. \quad (17.145)$$

Wir können aber auch schreiben:

$$\tilde{l}_i = \tilde{l}'_j (\tilde{l}''_k t_i^{jk}) = \tilde{l}'_j h_i^j \quad (h_i^j = \tilde{l}''_k t_i^{jk}). \quad (17.146)$$

Damit beschreiben wir eine Homographie vom 1. zum 2. Bild:

$$\tilde{x}^{j'} = h_i^j x^i. \quad (17.147)$$

Im Folgenden sind einige Beziehungen aufgeführt in Tensornotation:

- Gerade-Gerade-Gerade-Korrespondenzen

$$(\tilde{l}_r \epsilon^{ris}) \tilde{l}'_j \tilde{l}''_k t_i^{jk} = o^s, \quad (17.148)$$

- Punkt-Gerade-Gerade-Korrespondenzen

$$\tilde{x}^i \tilde{l}'_j \tilde{l}''_k t_i^{jk} = o, \quad (17.149)$$

- Punkt-Gerade-Punkt-Korrespondenzen

$$\tilde{x}^i \tilde{l}'_j (\tilde{x}^k \varepsilon_{kqs}) t_i^{jq} = o_s, \quad (17.150)$$

- Punkt-Punkt-Gerade-Korrespondenzen

$$\tilde{x}^i (\tilde{x}^j \varepsilon_{jpr}) \tilde{l}''_k t_i^{pk} = o_r, \quad (17.151)$$

- Punkt-Punkt-Punkt-Korrespondenzen

$$\tilde{x}^i (\tilde{x}^j \varepsilon_{jpr}) (\tilde{x}^k \varepsilon_{kqs}) t_i^{pq} = o_{rs}. \quad (17.152)$$

17.5.2 Der Strukturtensor

Häufig wird in Anwendungen das Matrix-Vektor-Kalkül benutzt und nicht die Tensornotation, aber trotzdem wird die Anwendung in Verbindung mit dem Begriff Tensor gebracht. Dies ist sicher nicht falsch, aber ohne Tensornotation auch nicht ganz einsichtig. Eine solche typische Anwendung ist z. B. der sogenannte **Strukturtensor**. Die Mathematik dazu ist recht einfach und soll kurz dargestellt werden. Wir betrachten im Bild ein Fenster der Größe $n \times n$ und wollen für dieses Fenster die „typische Gradientenrichtung“ und den „typischen Gradientenbetrag“ feststellen. Dazu betrachten wir den Gradienten pro Pixel (x, y) der Grauwertfunktion $f(x, y)$ und bezeichnen ihn mit $\nabla f(x, y)$. Um zur Tensorrechnung konsistent zu sein, ist dies nun ein Zeilenvektor der Länge zwei (kovarianter Tensor). Es seien \mathbf{u} und \mathbf{v} zwei Vektoren (Spaltenvektoren) der Länge eins, wobei \mathbf{v} auf \mathbf{u} senkrecht stehen möge, also $\mathbf{u}^T \mathbf{v} = 0$. Wir betrachten nun im Fenster folgendes Optimierungsproblem:

$$z(\mathbf{u}) = \sum_{x,y} (\nabla f(x, y) \cdot \mathbf{u})^2 \rightarrow \text{Maximum bei } |\mathbf{u}| = 1. \quad (17.153)$$

Wir suchen also den Einheitsvektor \mathbf{u} , für den die Summe der Quadrate aller Skalarprodukte mit den Gradienten maximal ist, der also am wenigsten in diesem Sinne von allen Gradienten abweicht. Dieses Optimierungsproblem zu lösen ist eine Standardaufgabe, d. h. wir bilden die Lagrange-Funktion, bilden den Gradienten der Lagrange-Funktion und setzen ihn zu Null, erhalten ein Gleichungssystem, welches ein Eigenwertproblem darstellt. Dieses Eigenwertproblem schreiben wir nun einmal auf. Dazu bilden wir die $n^2 \times 2$ Matrix, die aus allen Grauwertgradienten des Fensters besteht und bezeichnen sie mit

$$\mathbf{F} = \begin{pmatrix} \nabla f(1,1) \\ \nabla f(1,2) \\ \vdots \\ \nabla f(n,n) \end{pmatrix}. \quad (17.154)$$

Mit dieser Matrix bilden wir nun die 2×2 Matrix $\mathbf{S} = \mathbf{F}^T \cdot \mathbf{F}$ und nennen diese Größe **Strukturtensor** in Matrixnotation:

$$\mathbf{S} = \mathbf{F}^T \cdot \mathbf{F} = \begin{pmatrix} \sum_{x,y \in U} f_x^2 & \sum_{x,y \in U} f_x \cdot f_y \\ \sum_{x,y \in U} f_x \cdot f_y & \sum_{x,y \in U} f_y^2 \end{pmatrix}. \quad (17.155)$$

Wenn wir den Gradienten als Zufallsvektor auffassen und die Realisierungen des Zufallsvektors pro Pixel betrachten, dann ist der Strukturtensor weiter nichts (bis auf einen Faktor) als eine Schätzung der Kovarianzmatrix dieses Zufallsvektors, damit symmetrisch und positiv semidefinit. Wir können also von \mathbf{S} die beiden Eigenwerte λ_1 und λ_2 sowie die beiden orthogonalen Eigenvektoren \mathbf{u} und \mathbf{v} ausrechnen, die die Lösung unseres Optimierungsproblems bilden. Eigentlich haben wir mit dem Eigenwertproblem zwei Optimierungsprobleme gelöst

$$z(\mathbf{u}) \rightarrow \text{Maximum und } z(\mathbf{v}) \rightarrow \text{Minimum.} \quad (17.156)$$

Der Eigenvektor mit dem „großen“ Eigenwert löst das Maximumproblem, der andere das Minimumproblem. Anhand des Verhältnisses der beiden Eigenwerte kann man nun Aussagen treffen über die ein- oder zweidimensionale Struktur im Bildfenster. Um dies zu verdeutlichen, nutzen wir einmal die Taylorentwicklung:

$$f(x + \Delta x, y + \Delta y) \approx f(x, y) + \nabla f(x, y) \cdot \Delta \mathbf{x}. \quad (17.157)$$

Wenn wir $\|\Delta \mathbf{x}\| = 1$ wählen, dann beschreibt

$$D(\Delta \mathbf{x}) = \sum_{x,y} (f(x + \Delta x, y + \Delta y) - f(x, y))^2 = \Delta \mathbf{x}^T \mathbf{S} \Delta \mathbf{x} \quad (17.158)$$

das mittlere Differenzenquadrat aller Grauwerte in einem Fenster bezüglich einer gewählten Richtung $\Delta \mathbf{x}$. Folglich gilt:

$$\begin{aligned} D(\mathbf{u}) &= \mathbf{u}^T \mathbf{S} \mathbf{u} = \lambda_1 \\ D(\mathbf{v}) &= \mathbf{v}^T \mathbf{S} \mathbf{v} = \lambda_2 \end{aligned} \quad (17.159)$$

und damit sind die beiden Eigenwerte gerade die Summe der Differenzenquadrate der Grauwerte bez. der berechneten Hauptrichtungen. Somit kann man schlussfolgern:

- Wenn $\lambda_1 \approx 0, \lambda_2 \approx 0$ gilt, dann sind die Differenzen in beiden Hauptrichtungen klein, folglich liegt keinerlei Struktur vor.
- Wenn $\lambda_1 \gg 0, \lambda_2 \approx 0$ gilt, dann haben wir eine stabile Gradientenrichtung, was auf eine eindimensionale Struktur schließen lässt, z. B. eine Kante.
- Wenn $\lambda_2 \gg 0$, d. h. λ_2 groß genug ist, dann sind beide Eigenwerte groß und damit sind die Differenzen in beiden Hauptrichtungen groß. Folglich liegt eine zweidimensionale Struktur vor, d. h. eine Ecke oder ein Peak.

Den Strukturtensor kann man folglich für zwei wesentliche Aufgaben benutzen:

- Pro Pixel kann man eine stabile Gradientenrichtung oder Kantenrichtung berechnen und die „Stabilität“ auch numerisch bewerten, siehe dazu auch „Mittelung von Orientierungen“ in Abschn. 8.2.
- Mittels der berechneten Eigenwerte kann man die lokale Struktur bewerten, z. B. kann man dies zur Eckendetektion nutzen, siehe Kanade-Tomasi-Detektor in Abschn. 8.2.

Wir sehen, wir haben das Problem formuliert ohne je die Tensorrechnung zu benutzen. Wir wollen aber noch zum Schluss klären, was für ein Tensor der Strukturtensor \mathbf{S} nun eigentlich ist, oder ob er überhaupt ein Tensor ist. Wir wissen, nicht jede Matrix ist ein Tensor, nur dann, wenn sich die Matrixkoeffizienten entsprechend transformieren. Da \mathbf{S} eine symmetrische Matrix ist, muss \mathbf{S} ein symmetrischer Tensor zweiter Stufe sein, also zwei Indizes besitzen. Wo schreiben wir diese Indizes aber hin? Da \mathbf{S} aus dem dyadischen Produkt des Gradienten mit sich selbst entsteht, der Gradient aber ein kovarianter Tensor ist, muss \mathbf{S} also ein zweistufig kovarianter Tensor sein. Er ist also mit s_{ij} zu bezeichnen, wobei i, j nur die Werte eins und zwei annehmen können (wenn f nur von zwei Variablen abhängt). Da der Strukturtensor sich also tatsächlich aus dem Tensorprodukt des kovarianten Gradienten-Tensors mit sich selbst berechnet, ist er auch tatsächlich ein Tensor.

Dies können wir nun weiter verallgemeinern. Statt einem Grauwertbild $f(x, y)$ betrachten wir ein vektorwertiges Bild $\mathbf{f}(x, y)$, z. B. ein Farbbild. Wir können den Strukturtensor nun tatsächlich auch auf Vektorfunktionen verallgemeinern, bei Farbbildern wird er **Farbtensor** oder **color tensor** genannt. Dazu bilden wir statt dem Gradienten der Skalarfunktion f den Gradienten der Vektorfunktion \mathbf{f} , was einen einstufig kontravarianten und einstufig kovarianten Tensor zweiter Stufe ergibt oder in Matrixnotation die Jakobimatrix $\nabla \mathbf{f} = J$. Analog zu f bilden wir die Matrix \mathbf{F}

$$\mathbf{F} = \begin{pmatrix} \nabla \mathbf{f}(1,1) \\ \nabla \mathbf{f}(1,2) \\ \vdots \\ \nabla \mathbf{f}(N,N) \end{pmatrix} \quad (17.160)$$

und nennen das Produkt $\mathbf{S} = \mathbf{F}^T \mathbf{F}$ bei Farbbildern **Farbtensor**. Für ein Pixel ist $\mathbf{S} = \nabla \mathbf{f}^T \cdot \nabla \mathbf{f}$. Dies schreiben wir nun tensoriell:

$$s_{ij} = \nabla_i f^k \cdot \nabla_j f^k. \quad (17.161)$$

Wir sehen, der Verjüngungsindex k steht nur „oben“, damit ist dies keine zulässige Verjüngung. Damit ist der Farbtensor im allgemeinen Fall gar kein Tensor. Nur wenn wir speziell orthogonale Basen und nur Rotationen zulassen, dann ist er ein zweistufig (kovarianter) Tensor.

Bemerkung Wenn wir eine gewöhnliche Matrix \mathbf{A} betrachten, so können wir uneingeschränkt Produkte mit anderen geeigneten Matrizen bilden, es ergeben sich immer wieder Matrizen. So können wir auch z. B. die Produkte $\mathbf{A}^T \mathbf{A}$ und $\mathbf{A} \mathbf{A}^T$ immer bilden und es entsteht eine quadratische Matrix. Wir nehmen nun einmal an, \mathbf{A} repräsentiere einen Tensor 2. Stufe, dann bedeutet das Tensorprodukt immer, dass der Ergebnistensor die Stufe 4 hat. Wenn wir dann verjüngen, erhalten wir wieder einen Tensor 2. Stufe, also womöglich die gewünschte Ergebnismatrix. Wenn aber \mathbf{A} z. B. einen Tensor a_{ij} repräsentiert, dann gibt es keine Verjüngung des Produktes $a_{ij} \cdot a_{kl}$, dann ist obiges Matrixprodukt kein Tensor. Dies gilt ebenso für Tensoren a^{ij} ! Nur bei Tensoren a_j^i könnte man das Produkt verjüngen und das Matrixprodukt ergibt tatsächlich auch wieder einen Tensor. Dem Matrixprodukt $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$ entspricht dann tatsächlich ein Tensor, wenn \mathbf{A} und \mathbf{B} solche Tensoren a_i^j und b_k^l darstellen. Multiplizieren wir auch in diesem Falle etwa $\mathbf{C} = \mathbf{A}^T \cdot \mathbf{A}$, dann ist dies auch in diesem Falle kein Tensor. Man könnte dies auch tatsächlich auf die Fälle ausdehnen, wenn die Indizes nur unten stehen oder nur oben stehen, dann müssen wir aber orthogonale Basen voraussetzen und der Wechsel der Basis darf nur über Rotationen erfolgen. In diesem Falle ist es egal wo die Indizes stehen, dann ergeben obige Matrixprodukte auch tatsächlich immer wieder Tensoren.

Teil III

Objekterkennung

Im folgenden Kapitel werden wir einen Überblick über Verfahren des maschinellen Lernens bieten. Dabei werden wir uns auf etablierte Methoden konzentrieren, welche vor allem im Bereich der Bildverarbeitung Anwendung finden. Die Fähigkeit eine Art der automatischen Bildanalyse und -erkennung von Objekten durchzuführen, ist sowohl in der Robotik als auch bei zahlreichen Anwendungen zwingend notwendig. In den letzten Jahren lässt sich ein drastischer Anstieg an komplexen industriellen Problemstellungen verzeichnen, welche ohne Verfahren des maschinellen Lernens nicht realisierbar sind. Als prägnantes Beispiel sei hier die Fußgängerdetektion [20] und zahlreiche andere Fahrerassistenzsysteme aufgeführt. Weiterhin stößt die manuelle Optimierung von Parametern eines Algorithmus für eine gegebene Aufgabenstellung schnell an ihre Grenzen und die meisten Algorithmen der Bildverarbeitung besitzen Kontrollparameter welchen einen entscheidenden Einfluss auf die Ergebnisse haben. Methoden des maschinellen Lernens ermöglichen es einen Teil dieser Parameter automatisch aus gegebenen Daten zu lernen.

Das folgende Kapitel führt zunächst die unterschiedlichen Problemfelder des maschinellen Lernens ein, um dann aktuelle Klassifikationstechniken zu diskutieren. Insgesamt werden wir in diesem Buch nur die Grundzüge des maschinellen Lernens bei einfachen Klassifikationsaufgaben kennenlernen. Dem interessierten Leser sei zur weiteren Lektüre das englischsprachige Buch von Chris Bishop [3] empfohlen.

18.1 Grundprinzipien des maschinellen Lernens

Ziel des maschinellen Lernens ist es, den funktionalen Zusammenhang zwischen Eingabedaten $\mathbf{x} \in \mathcal{X}$ und Ausgabedaten $y \in \mathcal{Y}$ zu schätzen. Dabei werden wir uns in diesem Buch hauptsächlich mit überwachten Verfahren beschäftigen, welche versuchen, aus Lerndaten diesen Zusammenhang zu schließen. An dieser Stelle wollen wir gleich einmal die Erwartung an Verfahren des maschinellen Lernens ein bisschen zügeln: Für praktische Anwendungen gibt es kein perfektes Verfahren, viel mehr noch es ist nicht möglich anhand

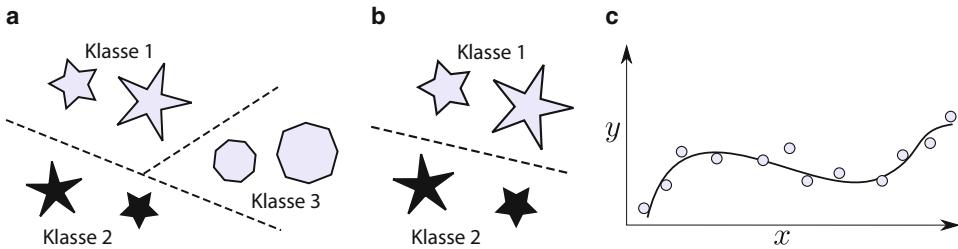


Abb. 18.1 Beispiele für Anwendungsszenarien des maschinellen Lernens mit unterschiedlichen Ausgabedaten $y \in \mathcal{Y}$. **a** Mehrklassen-Klassifikation, **b** binäre Klassifikation und **c** Regression

von theoretischen Betrachtungen ein für eine gegebene Aufgabenstellung am Besten geeignetes Verfahren auszuwählen. Diese Tatsache, welche auch oft als *no free lunch theorem* bekannt ist und aus den mathematisch wesentlich präziseren Resultaten von David Wolpert [90] entspringt, führt dazu, dass die Wahl des Verfahrens immer anhand empirischer Untersuchungen und den technischen Anforderungen (verfügbarer Speicher, Rechenzeit, etc.) der Aufgabenstellung getroffen werden muss.

Bei visuellen Erkennungsaufgaben, welche in Kap. 19 näher betrachtet werden, dienen Bilder als Eingabedaten. Die Art der Ausgaben führt zu ganz unterschiedlichen Aufgabenstellungen und Bereichen des maschinellen Lernens. In Abb. 18.1 wird dies anhand von einfachen Beispielen deutlich. Bei reellen Ausgabedaten $y \in \mathbb{R}$ handelt es sich um klassische *Regressionsprobleme*, d. h. der Zusammenhang zwischen Ein- und Ausgaben lässt sich durch eine reelle Funktion beschreiben. Im Folgenden werden wir uns hauptsächlich mit *Klassifikationsproblemen* beschäftigen, d. h. Aufgaben welche durch einen diskreten Ausgaberaum charakterisiert sind. Ziel der Klassifikation ist es, Eingabedaten automatisch in vordefinierte Klassen einzuteilen. Dabei sprechen wir von binärer Klassifikation, wenn eine Unterscheidung in zwei Klassen $y = -1$ und $y = 1$ erfolgt. Die Wahl der Werte -1 und 1 ist eine reine Festlegung und wir werden im weiteren Verlauf sehen, dass dadurch die mathematische Beschreibung oft in einfacherer Form gelingt. Wenn eine Unterscheidung in mehr als zwei Klassen erforderlich ist, wird von einem *Mehrklassen-Klassifikationsproblem* gesprochen.

Die Unterscheidung in die einzelnen Problemklassen ist wichtig für die Auswahl der Verfahren und kann an einem einfachen Beispiel verdeutlicht werden: Eine sehr häufige Anwendung des maschinellen Lernens ist die Biometrie. So kann man zum Beispiel anhand eines digitalen Bildes eine Identifikation von Personen vornehmen und es handelt sich bei mehr als zwei Personen um ein Mehrklassen-Klassifikationsproblem mit einer Klasse pro Person. Soll hingegen das Geschlecht einer Person anhand eines Bildes erkannt werden, so ist die Unterscheidung in weiblich und männlich eine klassische binäre Aufgabenstellung. Im Gegensatz zu dieser Aufgabenstellung mit diskreten Ausgaben kann aber auch eine Schätzung des Alters der Person durchgeführt werden, welches auf ein Regressionsproblem führt.

Im Folgenden werden wir die Menge der Lerndaten mit $\mathcal{D} \subseteq (\mathcal{X} \times \mathcal{Y})^n$ als geordnete Menge von n Eingabe-Ausgabepaaren (\mathbf{x}_i, y_i) bezeichnen. Der zu schätzende Zusammenhang zwischen Ein- und Ausgabedaten kann auf zwei Arten beschrieben werden: als funktionaler Zusammenhang oder auf stochastische Weise durch Angabe der A-Posteriori-Verteilung.

18.1.1 Funktionale Modellierung

Bei der ersten Variante wird angenommen, dass sich jeder Eingabe \mathbf{x} genau eine entsprechende Ausgabe $\tilde{h}(\mathbf{x})$ zuweisen lässt. Ziel ist es daher die Funktion $\tilde{h} : \mathcal{X} \rightarrow \mathcal{Y}$ durch Informationen aus dem Lerndatensatz \mathcal{D} möglichst gut zu approximieren. Da Regressionsprobleme mathematisch einfacher zu handhaben sind als Klassifikationsprobleme, führt man bei Klassifikationsproblemen die Approximation von \tilde{h} mit diskreter Ausgabemenge \mathcal{Y} oft auf eine Schätzung dieser zurück. Bei der binären Klassifikation ($y \in \{-1, 1\}$) wird zum Beispiel bei vielen Verfahren zunächst eine Schätzung einer reellwertigen Funktion $f : \mathcal{X} \rightarrow \mathbb{R}$ durchgeführt, um dann die Entscheidungsfunktion durch $h(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$ zu approximieren. Die Funktion $\text{sign}(z)$ sei dabei 1 für alle positiven z und -1 in allen anderen Fällen. Die Ausgabe $f(\mathbf{x})$ kann aber auch direkt als kontinuierliche Bewertung genutzt werden. Dadurch ist die Erstellung eines Rankings von Beispielen möglich, wie es zum Beispiel bei der Anwendung der inhaltsbasierten Bildsuche (engl.: Image Retrieval) notwendig ist. Bei dieser Aufgabe ist gefordert anhand gegebener Bilder, ähnliche Bilder in einer Datenbank zu finden und zurückzuliefern. Dabei soll natürlich eine Ordnung der Ergebnisse aufgrund einer Bewertungsfunktion f erfolgen, welche die Ähnlichkeit zu den gegebenen Bildern angibt.

Eine ähnliche Idee der Zurückführung auf Regressionsprobleme kann bei der Mehrklassen-Klassifikation mit M Klassen angewendet werden, Ziel ist es dabei, für jede Klasse κ eine Bewertungsfunktion $f^{(\kappa)}(\mathbf{x})$ zu schätzen, welche besonders große Werte für die korrekte Klasse liefert. Eine Klassifikationsentscheidung kann dann durch Auswahl der Klasse erfolgen, welche die maximale Bewertung („score“) erhält:

$$h(\mathbf{x}) = \underset{1 \leq \kappa \leq M}{\operatorname{argmax}} f^{(\kappa)}(\mathbf{x}). \quad (18.1)$$

Ein Vorteil dieser Modellierung ist die Möglichkeit auch zweit- oder drittplatzierte Klassen für die Weiterverarbeitung miteinzubeziehen.

18.1.2 Stochastische Modellierung

Im letzten Abschnitt haben wir bereits die Vorteile von „weichen“ Entscheidungen aufgezählt, welche aber bei der funktionalen Modellierung nur durch Umwege über die Re-

gression zu erhalten sind. Im Gegensatz dazu sind bei der stochastischen Modellierung weiche Entscheidungen direkt durch Wahrscheinlichkeiten für die Ausgabewerte gegeben. Grundidee ist es, Ein- und Ausgaben als Zufallsvariablen zu betrachten. Eine Klassifikationsaufgabe ist dann durch die gemeinsame Verteilung $p(y, \mathbf{x})$ definiert und es wird angenommen, dass die Elemente (y_i, \mathbf{x}_i) der Lerndaten \mathcal{D} von dieser Verteilung gezogen wurden. Es ist leicht zu sehen, dass diese Art der Modellierung die funktionale Modellierung verallgemeinert. Sei zum Beispiel ein funktionaler Zusammenhang durch \hat{h} gegeben, so kann die gemeinsame Verteilung mit $p(y, \mathbf{x}) = \delta[\hat{h}(\mathbf{x}) - y] p(\mathbf{x})$ angegeben werden. Die Funktion $\delta[\cdot]$ repräsentiert dabei die Diracfunktion oder den Einheitsimpuls, der bereits in Abschn. 4.18 eingeführt wurde.

Das Ziel des maschinellen Lernverfahrens ist es nun die A-Posteriori-Verteilung $p(y_* | \mathbf{x}_*, \mathcal{D})$ ¹ für die Ausgabe y_* eines neuen Beispiels $\mathbf{x}_* \in \mathcal{X}$ zu ermitteln. Eine Entscheidung kann dann durch die Maximierung der A-Posteriori-Verteilung realisiert werden. Bei der Klassifikation ist dies äquivalent zur Minimierung der Wahrscheinlichkeit einer Fehlklassifikation:

$$h(\mathbf{x}_*) = \operatorname{argmin}_{\kappa \in \mathcal{Y}} p(y \neq \kappa | \mathbf{x} = \mathbf{x}_*, \mathcal{D}) \quad (18.2)$$

$$= \operatorname{argmin}_{\kappa \in \mathcal{Y}} (1 - p(y = \kappa | \mathbf{x} = \mathbf{x}_*, \mathcal{D})) \quad (18.3)$$

$$= \operatorname{argmax}_{\kappa \in \mathcal{Y}} p(y = \kappa | \mathbf{x} = \mathbf{x}_*, \mathcal{D}). \quad (18.4)$$

Im weiteren Verlauf des Buches werden wir Zufallsvariablen nur bei Uneindeutigkeiten explizit angeben. Das bedeutet, dass zum Beispiel anstatt der langen Form $p(\mathbf{x} = \mathbf{x}_*)$ die Kurzform $p(\mathbf{x}_*)$ verwendet wird.

18.1.3 Diskriminative und generative Klassifikatoren

Optimale Entscheidungen lassen sich im stochastischen Sinne nur treffen, wenn wir die genaue A-Posteriori-Verteilung kennen. Leider ist dies nicht der Fall und die einzige Information die wir zur Verfügung haben sind die gegebenen Lerndaten, aus denen wir ein entsprechendes Modell ableiten müssen.

Diskriminative Klassifikatoren modellieren die A-Posteriori-Verteilung direkt. Einige bekannte Vertreter sind hier die Support-Vektor-Klassifikation (Abschn. 18.4.6) und das maschinelle Lernen mit Gaußprozessen (Abschn. 18.5.5). *Generative Klassifikatoren* gehen einen anderen Weg und schätzen die A-Posteriori-Verteilung indirekt über die bedingte Verteilung $p(\mathbf{x}|y)$ der Eingaben \mathbf{x} . Grundidee ist hierbei die Anwendung des Gesetzes

¹ Wir verwenden an dieser Stelle bewusst den uneindeutigen Begriff „Verteilung“. Für diskrete Ausgaben y handelt es sich bei $p(y_* | \mathbf{x}_*, \mathcal{D})$ um die bedingte Wahrscheinlichkeitsverteilung. Im Gegensatz dazu ist bei kontinuierlichen Werten y eine bedingte Wahrscheinlichkeitsdichte gegeben.



Abb. 18.2 Ein Beispiel für die Modellverifikation mit generativen Modellen: die abgebildeten Gesichter wurden synthetisch aufgrund eines gelernten Modells der Datenverteilung von Gesichtsbildern erzeugt. Zum Schätzen des Modells wurden Bilder aus der freien Datenbank von [70] verwendet

von Bayes um im zweiten Schritt die A-Posteriori-Verteilung der Ausgaben zu berechnen:

$$p(y_* | \mathbf{x}_*, \mathcal{D}) = \frac{p(\mathbf{x}_* | y_*, \mathcal{D}) p(y_* | \mathcal{D})}{p(\mathbf{x}_* | \mathcal{D})} \quad (18.5)$$

$$= \frac{p(\mathbf{x}_* | y_*, \mathcal{D}) p(y_* | \mathcal{D})}{\sum_{y \in \mathcal{Y}} p(\mathbf{x}_* | y, \mathcal{D}) p(y | \mathcal{D})}. \quad (18.6)$$

Der Nenner in (18.5) kann für die Maximierung bezüglich der Ausgabe y_* ignoriert werden und es ergibt sich folgende Klassifikationsentscheidung:

$$h(\mathbf{x}_*) = \operatorname{argmax}_{\kappa \in \mathcal{Y}} p(\mathbf{x}_* | y_* = \kappa, \mathcal{D}) \cdot p(y_* = \kappa | \mathcal{D}). \quad (18.7)$$

Die direkte Verwendung der Dichtewerte hat oft numerische Nachteile und um diese zu verhindern, kann die logarithmierte A-Posteriori-Verteilung zur Entscheidung herangezogen werden:

$$h(\mathbf{x}_*) = \operatorname{argmax}_{\kappa \in \mathcal{Y}} \log p(\mathbf{x}_* | y_* = \kappa, \mathcal{D}) + \log p(y_* = \kappa | \mathcal{D}). \quad (18.8)$$

Im Gegensatz zu diskriminativen Ansätzen haben generative Ansätze den großen Vorteil, dass ein Modell der Verteilung der Eingabedaten zur Verfügung steht. Dadurch ist es zum Beispiel möglich, eine Ausreißerdetektion durchzuführen oder neue Datenpunkte von der Verteilung zu ziehen. Die zweite Eigenschaft ist besonders sinnvoll für die Verifikation des Modells. Nehmen wir einfach einmal an, dass wir aufgrund von gegebenen Lernbildern die Verteilung von Gesichtern modelliert und geschätzt haben. Durch das Ziehen einer Eingabe von der erlernten Verteilung und deren Inspektion lässt sich leicht erkennen, ob das Modell den Anforderungen der Aufgabenstellung genügt. Ein entsprechendes Beispiel dieses Szenarios lässt sich in Abb. 18.2 finden.

Der klare Vorteil von diskriminativen Ansätzen ist die direkte Modellierung der gewünschten A-Posteriori-Verteilung ohne den Umweg über die Likelihood der Eingabedaten. Dieser Vorteil ist in folgender Aussage von Vladimir Vapnik, dem Begründer der Lerntheorie und der Support-Vektor-Klassifikation, schön zusammengefasst:

When solving a given problem, try to avoid solving a more general problem. [79], Abschn. 1.9, S. 28.

18.2 Herausforderungen des Maschinellen Lernens

Beim maschinellen Lernen mit gegebenen beschrifteten Lernbeispielen gibt es viele Aspekte zu beachten, welche die Auswahl des Klassifikators, der Merkmale oder des Lerndatensatzes beeinflussen. Bei der Entwicklung eines Erkennungssystems ist es daher zwingend notwendig, sich mit bestimmten Grundkonzepten vertraut zu machen, da dadurch viele Phänomene erklärt werden können.

18.2.1 Modellkomplexität, Überanpassung und Generalisierung

Im Folgenden wollen wir uns die Begriffe *Modellkomplexität*, *Überanpassung* und *Generalisierung* näher ansehen. Wir betrachten dazu ein einfaches binäres Klassifikationsproblem so wie es in Abb. 18.3 dargestellt ist. Es lässt sich sehr schnell erkennen, dass eine lineare Trennebene nicht ausreicht, um die Lerndaten vollständig voneinander zu trennen. Dies liegt an ihrer *Modellkomplexität*. Das Modell ist durch die Annahme der Linearität nicht flexibel genug. Eine nichtlineare Trennung ermöglicht hingegen die Separierung der zwei Klassen.

Aufgrund dieses Beispiels könnte man natürlich schlussfolgern, dass immer ein möglichst komplexes Modell verwendet werden sollte. Dies ist aber nicht der Fall, da komplexe Modelle schnell zu einer *Überanpassung* führen können. Dieser Überanpassungseffekt lässt sich gut am sogenannten *bias-variance tradeoff* erläutern, welcher den Fehler bei einer Schätzung in einen Fehlerterm der Lerndaten (*bias*) und die Varianz der Schätzung (*variance*) zerlegt.

Für die Herleitung untersuchen wir den zu erwartenden quadratischen Fehler eines gerünteten Modells in einem einzelnen Punkt \mathbf{x} , d. h. die quadratische Differenz der Ausgaben $\tilde{h}(\mathbf{x})$ und $h(\mathbf{x}; \mathcal{D})$:

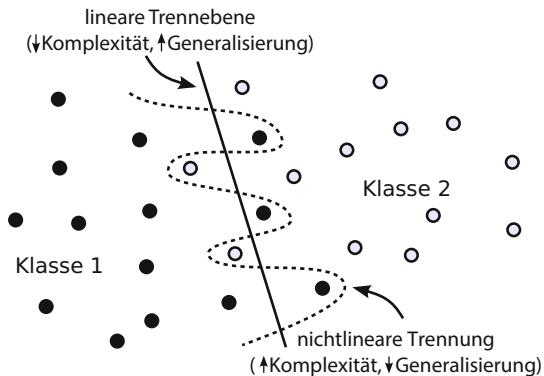
$$\text{MSE}(\mathbf{x}) = E_{\mathcal{D}} \left((\tilde{h}(\mathbf{x}) - h(\mathbf{x}; \mathcal{D}))^2 \right). \quad (18.9)$$

Der obige Erwartungswert wird bezüglich des Lerndatensatzes \mathcal{D} berechnet und wird daher mit $E_{\mathcal{D}}(\cdot)$ notiert. Eine Betrachtung des mittleren Fehlers ist natürlich auch möglich, aber wir wollen an dieser Stelle aus Gründen der Übersichtlichkeit darauf verzichten. Sei nun $h^{\text{avg}}(\mathbf{x}) = E_{\mathcal{D}}(h(\mathbf{x}; \mathcal{D}))$ die durchschnittliche Schätzung des Modells für eine Eingabe \mathbf{x} bezogen auf zufällige Lerndaten \mathcal{D} . Dieser Term wird nun genutzt um eine neutrale Null zu erzeugen und den Fehler zu zerlegen:

$$\text{MSE}(\mathbf{x}) = E_{\mathcal{D}} \left((\tilde{h}(\mathbf{x}) - h^{\text{avg}}(\mathbf{x}) + h^{\text{avg}}(\mathbf{x}) - h(\mathbf{x}; \mathcal{D}))^2 \right) \quad (18.10)$$

$$\begin{aligned} &= (\tilde{h}(\mathbf{x}) - h^{\text{avg}}(\mathbf{x}))^2 \\ &\quad + 2 \cdot (E_{\mathcal{D}}(h^{\text{avg}}(\mathbf{x}) - h(\mathbf{x}; \mathcal{D}))) (\tilde{h}(\mathbf{x}) - h^{\text{avg}}(\mathbf{x})) \\ &\quad + E_{\mathcal{D}}(h^{\text{avg}}(\mathbf{x}) - h(\mathbf{x}; \mathcal{D}))^2. \end{aligned} \quad (18.11)$$

Abb. 18.3 Komplexität und Generalisierungseigenschaften bei der Klassifikation



Aufgrund der Definition von h^{avg} entfällt nun der mittlere Summand und wir erhalten:

$$\text{MSE}(\mathbf{x}) = (\tilde{h}(\mathbf{x}) - h^{\text{avg}}(\mathbf{x}))^2 + E_{\mathcal{D}}(h^{\text{avg}}(\mathbf{x}) - h(\mathbf{x}; \mathcal{D}))^2 \quad (18.12)$$

$$= \underbrace{(\tilde{h}(\mathbf{x}) - h^{\text{avg}}(\mathbf{x}))^2}_{\text{bias}} + \underbrace{\sigma_{\mathcal{D}}^2(h(\mathbf{x}; \mathcal{D}))}_{\text{variance}}. \quad (18.13)$$

Der vordere Term, welcher als *bias* bezeichnet wird, misst die Abweichung der durchschnittlichen Schätzung zur echten Funktion \tilde{h} . Der zweite Term hingegen ist die Varianz der Schätzungen bei unterschiedlichen Lerndatensätzen und lässt sich als Maß der Komplexität des Modells begreifen.

Die Minimierung des Schätzfehlers führt also immer auf einen Kompromiss zwischen unterschiedlichen Modellkomplexitäten. Ein komplexes Modell wird immer einen niedrigen *bias*-Term haben, da es sich flexibel an gegebene Lerndaten anpassen kann, dieser Tatsache steht aber immer eine Erhöhung des *variance*-Term entgegen.

18.2.2 Lernen mit wenigen Beispielen

Das Problem bei wenigen Lerndaten manifestiert sich als schlecht gestelltes Optimierungsproblem, welches im Folgenden näher erläutert werden soll: Das Lernen von visuellen Aufgaben kann mathematisch als das Schätzen einer Abbildung $h : \mathcal{X} \rightarrow \mathcal{Y}$ von der Menge \mathcal{X} aller Bilder in die Menge \mathcal{Y} aller möglichen Beschriftungen angesehen werden. Die Schätzung basiert dabei auf einem gegebenen Lerndatensatz \mathcal{D} , welcher n Bilder $\mathbf{x}_i \in \mathcal{X}$ und deren Beschriftungen $y_i \in \mathcal{Y}$ enthält. Wird die Aufgabe als reines Schätzproblem betrachtet, führt dies unmittelbar zum entscheidenden Dilemma der Objekterkennung: auf der einen Seite ist die Menge aller möglichen Funktionen h und der Eingaberaum \mathcal{X} selbst hochdimensional, auf der anderen Seite existieren nur wenige gegebene Datenpunkte. Ohne weitere Zusatzinformationen ist diese Situation vergleichbar mit der Regression einer

komplexen Funktion (z. B. Polynom hohen Grades) mit einer geringen Anzahl von Abtastwerten. Die Schätzung eines komplexen Modells erfordert daher auch immer eine große Anzahl von Lernbeispielen.

18.3 Statistische Schätzer

Wie wir bereits in Abschn. 18.1.2 gesehen haben, lassen sich die Problemstellungen des maschinellen Lernens auch stochastisch modellieren. Daher werden wir uns im Folgenden den Grundprinzipien von statistischen Schätzern zu wenden, da diese zum Lernen von stochastischen Modellen eingesetzt werden können. Weiterhin sind diese Schätzer nicht nur im Bereich des maschinellen Lernens wichtig. In vielen Anwendungen der Bildverarbeitung werden Schätzer benötigt, so ist zum Beispiel das Wiener-Filter ein MMSE-Schätzer (Abschn. 9.5.3), Geometrische Objekte werden mit LSE oder LSA-Schätzern aus detektierten Pixeln im Bild geschätzt (Abschn. 21.2) und MAP-Schätzer werden oft in der Bildrestauration benutzt (Abschn. 9.4).

Für das weitere Vorgehen wählen wir einen sehr allgemeinen Ansatz und betrachten eine Menge von Modellen, welche durch eine Variable $\boldsymbol{\theta} \in \Theta$ parametrisiert sind. Mögliche Parameter sind zum Beispiel der Mittelwert und die Standardabweichung $\boldsymbol{\theta} = (\mu, \sigma^2)$ eines Normalverteilungsmodells oder der Normalenvektor einer Hyperebene. Das Modell einer Likelihood oder A-Posteriori-Verteilung ist also immer von einem Parametervektor $\boldsymbol{\theta}$ abhängig und wir verwenden im Folgenden die Schreibweise $p(y|\mathbf{x}, \boldsymbol{\theta})$ für diskriminative Modelle und $p(\mathbf{x}|y, \boldsymbol{\theta})$ für generative Modelle.

18.3.1 Maximum-Likelihood und Maximum-A-Posteriori-Schätzung

Viele Verfahren des maschinellen Lernens sind zweistufig aufgebaut. Zunächst werden die Modellparameter durch die Optimierung eines passenden Zielkriteriums bestimmt, welches von den Lerndaten abhängt. Anschließend werden die Modellparameter und das daraus resultierende Modell verwendet, um die Ausgaben zukünftiger neuer Eingabedaten zu schätzen.

Ein Standardkriterium für die Optimierung während des Lernschrittes ist dabei die Likelihood der Lerndaten bei gegebenen Modellparametern:

$$\hat{\boldsymbol{\theta}}^{\text{ML}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}} p(\mathcal{D}|\boldsymbol{\theta}). \quad (18.14)$$

Diese Schätzung wird als *Maximum-Likelihood-Schätzung* (*ML-Schätzung*) bezeichnet. Die Optimierung führt dabei auf eine sogenannte *Punktschätzung*, indem sie den Modalwert der Likelihood-Verteilung bestimmt. Eine entscheidende Eigenschaft der ML-Schätzung ist es, dass keine A-Priori-Wahrscheinlichkeiten des Modells eingehen. Egal wie unwahr-

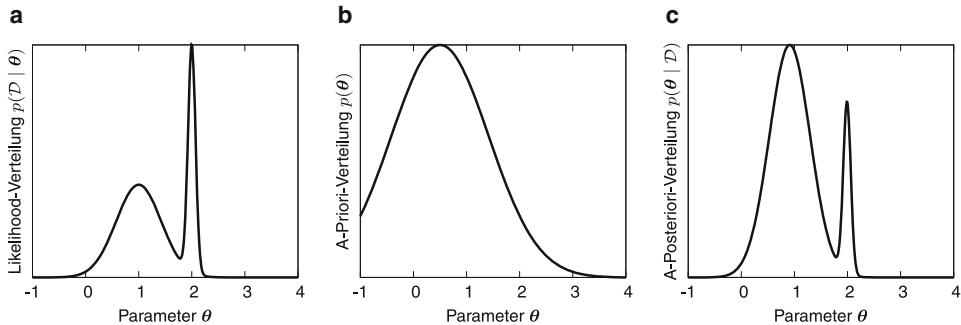


Abb. 18.4 Ein einfaches Beispiel für eine ML- und MAP-Schätzung. **a** Likelihood mit einem Modalwert und einer geringen Varianz, **b** Normalverteilung als A-Priori-Verteilung, **c** resultierende A-Posteriori-Verteilung mit einem anderen Modalwert

scheinlich die Modellparameter unabhängig von den Lerndaten sein mögen, das Kriterium orientiert sich allein an den gegebenen Beispielen in \mathcal{D} . Dadurch wird die Modellkomplexität nicht eingeschränkt und es kann im Allgemeinen schnell zu einer Überanpassung kommen (Abschn. 18.2.1).

Eine Möglichkeit diese Effekte zu reduzieren ist es, die A-Priori-Verteilung $p(\theta)$ direkt miteinzubeziehen und ein Maximum der A-Posteriori-Verteilung des Modellparameters zu finden:

$$\hat{\theta}^{\text{MAP}} = \underset{\theta \in \Theta}{\operatorname{argmax}} p(\theta | \mathcal{D}) = \underset{\theta \in \Theta}{\operatorname{argmax}} p(\mathcal{D} | \theta) p(\theta). \quad (18.15)$$

Diese Schätzung wird auch als *Maximum-A-Posteriori-Schätzung (MAP-Schätzung)* bezeichnet. In Abb. 18.4 werden beide Arten der Schätzung an einem kleinen Beispiel verdeutlicht. Die Likelihood in Abb. 18.4 besitzt ein Maximum bei $\theta = 2$ mit einer kleinen Breite, welches höchstwahrscheinlich aus einer Überanpassung entstanden ist. Wenn wir nun die A-Priori-Verteilung des Parameters θ verwenden, erhalten wir eine A-Posteriori-Verteilung mit einem Modalwert in der Nähe von $\theta = 1$. Der MAP-Schätzwert unterscheidet sich also vom ML-Schätzwert.

Beide Schätzverfahren sind äquivalent, wenn eine Gleichverteilung für $p(\theta)$ angenommen wird. Dies ist nur möglich, wenn der Parameterraum ein beschränktes Volumen besitzt, da ansonsten keine Gleichverteilung auf Θ existieren kann. Um eine MAP-Schätzung durchzuführen, ist es natürlich zunächst notwendig eine entsprechende A-Priori-Verteilung zu modellieren und zu bestimmen. Die Wahl ist dabei abhängig von der Aufgabenstellung.

Das direkte Verwenden einer Punktschätzung des Lernschrittes bei der Schätzung der Ausgabe eines neuen Beispiels wird auch als *Plugin-Prinzip* bezeichnet. Ein großer Nachteil dieses Ansatzes ist es, dass die Varianz der Schätzwertes dabei ignoriert wird.

18.3.2 Bayes-Ansätze und Marginalisierung

Im Gegensatz zu Punktschätzungen mit ML oder MAP verwenden *Bayes-Ansätze* (englisch: *Bayesian inference methods*) keine direkten Parameterschätzungen sondern marginalisieren den Parameter θ , d. h. wir integrieren über diesen Parameter:

$$p(y_* | \mathbf{x}_*, \mathcal{D}) = \int_{\Theta} p(y_*, \theta | \mathbf{x}_*, \mathcal{D}) d\theta \quad (18.16)$$

$$= \int_{\Theta} p(y_* | \mathbf{x}_*, \theta) p(\theta | \mathcal{D}) d\theta \quad (18.17)$$

$$= \int_{\Theta} p(y_* | \mathbf{x}_*, \theta) \left(\frac{p(\mathcal{D} | \theta) p(\theta)}{p(\mathcal{D})} \right) d\theta. \quad (18.18)$$

Die zweite Umformung verwendet die Annahme, dass die Likelihood eines neuen Beispiels von den Lerndaten unabhängig ist, wenn die Modellparameter θ gegeben sind. Diese Annahme wurde auch bereits indirekt bei der ML und MAP-Schätzung verwendet und ist intuitiv, schließlich sollte ein Modell möglichst so entworfen werden, dass alle wichtigen Informationen der Lerndaten für die Schätzung enthalten sind. Eine analoge Formulierung zu (18.17) lässt sich auch für die Likelihood $p(\mathbf{x}_* | y_*, \mathcal{D})$ durchführen.

Verglichen mit dem Plugin-Prinzip ist der Bayes-Ansatz wesentlich allgemeiner, so lassen sich ML und MAP-Schätzung als Spezialfall darstellen. Wir betrachten dazu die A-Posteriori-Verteilung $p(\theta | \mathcal{D})$ des Modells mit einem einzelnen Peak (Maximum) in $\hat{\theta}^{\text{MAP}}$. Das Integral in (18.17) lässt sich nun folgendermaßen berechnen (siehe Abschn. 2.1.2):

$$p(y_* | \mathbf{x}_*, \mathcal{D}) = \int_{\Theta} p(y_* | \mathbf{x}_*, \theta) \left(\delta \left[\theta - \hat{\theta}^{\text{MAP}} \right] \right) d\theta \quad (18.19)$$

$$= p(y_* | \mathbf{x}_*, \hat{\theta}^{\text{MAP}}). \quad (18.20)$$

Der Diracimpuls $\delta [\cdot]$ als Dichte interpretiert hat eine Varianz von Null. Daher wird bei der Punktschätzung die Varianz der Schätzung nicht beachtet. In Abb. 18.4a ist dies anhand der „schmalen Spitze“ verdeutlicht. Die Wahrscheinlichkeit (Fläche unterhalb der Likelihood-Funktion), dass der ML-Schätzwert die Lerndaten richtig repräsentiert ist trotz des hohen Wertes der Dichte sehr klein. Zwar wird durch die MAP-Schätzung eine Reduzierung dieses Phänomens erzielt, dennoch besteht das Problem der Punktschätzung weiterhin.

Im Gegensatz dazu kann der Bayes-Ansatz in (18.17) als unendliche Summe von verschiedenen modellspezifischen Likelihoods interpretiert werden, welche alle mit dem Wert der A-Posteriori-Verteilung des Modells gewichtet werden. Daher ist der Bayes-Ansatz robust bezüglich der in Abb. 18.4 beobachteten Problematik.

Genauso wie bei der MAP-Schätzung ist die Wahl der A-Priori-Verteilung entscheidend und dies nicht nur für die Genauigkeit der Schätzung, sondern auch für die Laufzeit und effiziente Realisierbarkeit. Der Parametervektor des Modells ist in vielen Fällen hochdimensional. Eine Auswertung des Integrals in (18.17) ist daher mit Standardmitteln der

numerischen Mathematik oft nicht möglich und es müssen Techniken wie *Markov chain Monte Carlo (MCMC)* [3], *Laplace-Approximation* (Abschn. 18.5.8) oder *expectation propagation* [57] genutzt werden.

Eine analytische Lösung ist nur in Spezialfällen möglich. So besitzen zum Beispiel bestimmte parametrische Familien von Likelihood-Verteilungen eine korrespondierende parametrische Familie von A-Priori-Verteilungen, welche auf eine geschlossene Lösung der Marginalisierung in (18.17) führen. Diese A-Priori-Verteilungen werden auch als *conjugate prior* bezeichnet.

18.3.3 MMSE-Schätzer

Wir haben gesehen, dass bei Bayes-Ansätzen keine einzelne Schätzung des Modellparameters durchgeführt wird. Dennoch ist es manchmal vorteilhaft eine Einzelschätzung vorzunehmen, um zum Beispiel die Plausibilität des Modells zu überprüfen. Der *Bayes-Schätzwert* des Parameters ist definiert als der Wert, welcher den Erwartungswert eines Fehlerterms $L : \Theta^2 \rightarrow \mathbb{R}$ bezüglich der A-Posteriori-Verteilung des Modells minimiert:

$$\hat{\theta}^{\text{Bayes}} = \underset{\tilde{\theta} \in \Theta}{\operatorname{argmin}} E_{\theta}(L(\theta, \tilde{\theta}) | \mathcal{D}) \quad (18.21)$$

$$= \underset{\tilde{\theta} \in \Theta}{\operatorname{argmin}} \int_{\Theta} L(\theta, \tilde{\theta}) \cdot p(\theta | \mathcal{D}) d\theta. \quad (18.22)$$

Wird ein quadratischer Fehlerterm gewählt, so ist der Bayes-Schätzwert äquivalent zum sogenannten MMSE-Schätzwert (englisch: *Minimum Mean Square Error*) und der Minimierung der Varianz [19]:

$$\hat{\theta}^{\text{MMSE}} = \underset{\tilde{\theta} \in \Theta}{\operatorname{argmin}} E_{\theta} \left(\|\theta - \tilde{\theta}\|^2 | \mathcal{D} \right). \quad (18.23)$$

Um dieses Kriterium zu minimieren, leiten wir es nun bezüglich $\tilde{\theta}$ ab:

$$\frac{\partial}{\partial \tilde{\theta}} E_{\theta} \left(\|\theta - \tilde{\theta}\|^2 | \mathcal{D} \right) = 2(E_{\theta}(\theta) - \tilde{\theta}) \quad (18.24)$$

und erhalten durch Nullsetzen folgenden Schätzwert:

$$\hat{\theta}^{\text{MMSE}} = E(\theta | \mathcal{D}) = \int_{\Theta} \theta \cdot p(\theta | \mathcal{D}) d\theta. \quad (18.25)$$

Dabei handelt es sich um den Mittelwert der A-Posteriori-Verteilung des Modells. Bei einem normalverteilten Modell lässt sich schnell die Äquivalenz zur MAP-Schätzung erkennen, da in diesem Fall der Mittelwert und der Modalwert identisch sind. Zu MMSE-Schätzern siehe auch die Abschn. 9.5.3 und 21.2.

18.3.4 Kombination von Modellen mit Bagging

Bei der Schätzung von Ausgaben oder Parametern können auch Techniken der Modellkombination angewendet werden, wie etwa das von Leo Breiman [8] vorgestellte *Bagging*, welches eine Abkürzung für *bootstrap aggregation* darstellt. Dieser Ansatz, welcher für die Regression und die Klassifikation eingesetzt werden kann, gehört zu der Klasse der *Ensemble-Klassifikatoren*. Wir werden in der Boosting-Methode noch einen weiteren Vertreter dieser Klasse in Abschn. 18.4.5 kennenlernen.

Ziel von Ensemble-Klassifikatoren ist es, eine Menge von mehreren Basisklassifikatoren, welche als Ensemble bezeichnet werden, zu kombinieren. In der englischen Literatur wird auch meist von *weak classifier* gesprochen, um zu verdeutlichen, dass selbst sehr einfache Klassifikatoren zu einem komplexen Modell kombiniert werden können. Die Idee beim Bagging-Ansatz ist es, mehrere Modelle $(\boldsymbol{\theta}_t)_{t=1}^T$ mit einer zufälligen Auswahl von Lerndaten zu bestimmen, d. h. es werden nur $r_B \cdot n$ Lernbeispiele verwendet mit $0 < r_B < 1$.

Nach dem unabhängigen Lernen der einzelnen Modelle kann die Schätzung einfach durch eine Mittelung erfolgen:

$$h^{\text{Bagging}}(\mathbf{x}_*) = \frac{1}{T} \sum_{t=1}^T h(\mathbf{x}_*; \boldsymbol{\theta}_t). \quad (18.26)$$

Die Motivation bei dieser Technik ist es, eine Überanpassung zu vermeiden, indem viele sehr unterschiedliche Modelle verwendet werden. Diese Eigenschaft von Bagging-Methoden wollen wir uns im Folgenden einmal näher ansehen. Wir untersuchen dazu den Fehler bei der Schätzung ähnlich zu [3, Abschn. 14.2] und bezeichnen den wahren Zusammenhang zwischen Ein- und Ausgabedaten mit \tilde{h} . Die Ausgabe des Modells lässt sich dann mit entsprechenden Fehlertermen ε_t schreiben als:

$$h(\mathbf{x}; \boldsymbol{\theta}_t) = \tilde{h}(\mathbf{x}) + \varepsilon_t(\mathbf{x}). \quad (18.27)$$

Der Fehler eines einzelnen Modells wird durch die quadratische Differenz zur wahren Funktion \tilde{h} bestimmt analog zur Betrachtung in Abschn. 18.2.1:

$$E_t = E_{\mathbf{x}} \left((h(\mathbf{x}; \boldsymbol{\theta}_t) - \tilde{h}(\mathbf{x}))^2 \right) = E_{\mathbf{x}} (\varepsilon_t(\mathbf{x})^2). \quad (18.28)$$

Dies führt direkt auf den durchschnittlichen Fehler des Ensembles, wenn jedes Modell unabhängig voneinander betrachtet wird:

$$E_{\text{average}} = \frac{1}{T} \sum_{t=1}^T E_t = \frac{1}{T} \sum_{t=1}^T E_{\mathbf{x}} (\varepsilon_t(\mathbf{x})^2). \quad (18.29)$$

Der Fehler des Bagging-Ansatzes ergibt sich nun unter der Annahme *unkorrelierter* Fehlerterme mit Mittelwert Null:

$$\begin{aligned}
 E_{\text{bagging}} &= E_{\mathbf{x}} \left(\left(h_{\mathcal{D}}^{\text{Bagging}}(\mathbf{x}_*) - \tilde{h}(\mathbf{x}) \right)^2 \right) \\
 &= E_{\mathbf{x}} \left(\left(\frac{1}{T} \sum_{t=1}^T \varepsilon_t(\mathbf{x}) \right)^2 \right) \\
 &= \frac{1}{T^2} \left(E_{\mathbf{x}} (\varepsilon_1(\mathbf{x})^2 + \dots + \varepsilon_T(\mathbf{x})^2) + E_{\mathbf{x}} \left(\sum_{t \neq t'} \varepsilon_t(\mathbf{x}) \varepsilon_{t'}(\mathbf{x}) \right) \right) \\
 &= \frac{1}{T^2} (E_{\mathbf{x}} (\varepsilon_1(\mathbf{x})^2 + \dots + \varepsilon_T(\mathbf{x})^2)) \\
 &= \frac{1}{T} \cdot E_{\text{average}}.
 \end{aligned} \tag{18.30}$$

Die theoretische Untersuchung zeigt, dass Bagging bei der Reduzierung des zu erwartenden Fehlers hilft besonders wenn ein großes Ensemble gewählt wird. Die wichtigste Annahme, die wir getroffen haben, war die fehlende Korrelation zwischen den Fehlertermen. Im allgemeinen Fall ist diese Annahme nicht gültig. Das Bagging-Verfahren versucht aber durch die randomisierte Auswahl von Lerndaten möglichst unterschiedliche und dadurch auch unkorrelierte Modelle zu gewinnen.

Bei stochastischen Modellen, wie wir sie vorwiegend in den letzten Abschnitten betrachtet hatten, kann die Bagging-Idee ebenfalls angewendet werden. Die geschätzte A-Posteriori-Verteilung einer Ausgabe y_* eines neuen Beispiels \mathbf{x}_* ist dann gegeben durch:

$$p(y_* | \mathbf{x}_*, \mathcal{D}) = \frac{1}{T} \sum_{t=1}^T p(y_* | \mathbf{x}_*, \theta_t). \tag{18.31}$$

Diese Schätzgleichung lässt sich aber nicht als stochastische Version von (18.26) interpretieren.

18.4 Klassifikatoren

Im folgenden Abschnitt wollen wir uns näher mit Klassifikationsverfahren beschäftigen. Die Verfahren unterscheiden sich auf unterschiedlichen Ebenen:

1. Art des Modells (diskriminativ oder generativ),
2. Grundlegende Modellannahmen,
3. Nebenbedingungen und Regularisierung,
4. Verfahren zur Schätzung der Parameter.

Die in diesem Buch besprochenen Verfahren sind nur eine kleine Teilmenge der verfügbaren und im Laufe der letzten Jahrzehnte entwickelten Klassifikatoren, welche besonders

für Aufgabenstellungen in der Bildverarbeitung wesentlich sind. Für weitere Beispiele, wie etwa neuronale Netze, sei an dieser Stelle auf das Buch von Chris Bishop [3] verwiesen, welches einen exzellenten und detaillierten Einblick in die Verfahren des maschinellen Lernens und deren Zusammenhänge bietet.

18.4.1 Nächster-Nachbar-Klassifikator

Der Nächste-Nachbar-Klassifikator (NN-Klassifikator) ist der einfachste aber auch bekannteste Klassifikator. Gegeben sei ein Lerndatensatz \mathcal{D} eines Mehrklassen-Klassifikationsproblems mit $y \in \{1, \dots, M\}$. Weiterhin nehmen wir an, dass wir D -dimensionale reellwertige Eingabedaten $\mathbf{x} \in \mathbb{R}^D$ klassifizieren möchten.

Wir gehen im Folgenden von der Existenz eines passenden Abstandsmaßes $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ zwischen Eingabedaten aus, welches kleine Distanzwerte zurück liefert, wenn zwei Beispiele im Sinne der Aufgabenstellung ähnlich sind. Der Nächste-Nachbar-Klassifikator verwendet direkt dieses Abstandsmaß um das Beispiel $\pi(\mathbf{x}_*)$ der Lernmenge zu bestimmen, welches maximal zum neuen Beispiel \mathbf{x}_* ähnlich ist:

$$\pi(\mathbf{x}_*) = \operatorname{argmin}_{1 \leq i \leq n} d(\mathbf{x}_*, \mathbf{x}_i). \quad (18.32)$$

Das neue Beispiel kann dann direkt mit der Klasse $y_{\pi(\mathbf{x}_*)}$ des nächsten Nachbarn $\pi(\mathbf{x}_*)$ klassifiziert werden und es ergibt sich folgende Klassifikationsregel:

$$h(\mathbf{x}_*) = y_{\pi(\mathbf{x}_*)}. \quad (18.33)$$

Die Entscheidungsfunktion des Nächster-Nachbar-Klassifikators bei der Anwendung auf eine zweidimensionale Lernstichprobe ergibt eine Voronoi-Zerlegung. Eine weitere Eigenschaft des Klassifikators ist die hohe Komplexität (Abschn. 18.2.1) der möglichen Trennung der Klassengebiete. Dadurch ist dieser Klassifikator auch besonders anfällig gegenüber Fehlern in der Lernstichprobe.

k -NN oder demokratische Lokalpolitik Eine robustere Variante des NN-Klassifikators ermöglicht die Betrachtung mehrerer Nachbarn. Seien $\pi^{(1)}, \dots, \pi^{(k)}$ die k nächsten Nachbarn von \mathbf{x}_* , welche analog zu (18.32) bestimmt wurden. Wir nehmen im Folgenden an, dass diese Menge eindeutig ist oder allgemeiner dass die Abstände zu den Lerndaten paarweise verschieden sind. Aus der Statistik der zu den Nachbarn zugehörigen Ausgaben $\mathbf{s}(\mathbf{x}_*) = (y_{\pi^{(1)}}, \dots, y_{\pi^{(k)}})$ lassen sich nun mehrere unterschiedliche Entscheidungsregeln ableiten. Die einfachste Regel ist der Mehrheitsentscheid, d. h. wir nehmen die Klasse, die unter den k nächsten Nachbarn am häufigsten auftritt:

$$h(\mathbf{x}_*) = \operatorname{argmax}_{\kappa \in \mathcal{Y}} |\{i \mid s_i(\mathbf{x}_*) = \kappa\}|. \quad (18.34)$$

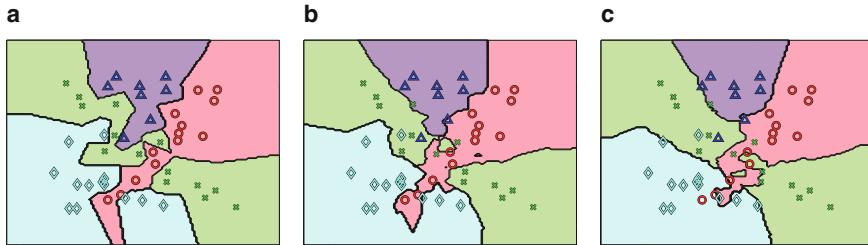


Abb. 18.5 Klassifikation mit einem k -Nächster-Nachbar Klassifikator und unterschiedlichen Werten von k (**a** $k = 1$, **b** $k = 3$, **c** $k = 5$). Farblich dargestellt sind die durch die Klassifikationsentscheidung entstehenden Klassengebiete

Abbildung 18.5 zeigt anhand eines zweidimensionalen Beispiels die Ergebnisse beim k -Nächster-Nachbar-Klassifikator. Eine optimale Bestimmung des Parameters k kann durch Kreuzvalidierung (Abschn. 18.6.1) durchgeführt werden. Weiterhin ermöglicht die Bestimmung der k nächsten Nachbarn auch eine stochastische Modellierung durch die empirische Bestimmung der Wahrscheinlichkeiten aufgrund des Histogramms von \mathbf{s} :

$$p(y_* = \kappa | \mathbf{x}_*, \mathcal{D}) = \frac{|\{i | s_i(\mathbf{x}_*) = \kappa\}|}{k}. \quad (18.35)$$

Ein Nachteil dieses Modelles ist, dass die genauen Abstände der nächsten Nachbarn nicht miteinbezogen werden. Jeder Nachbar wird gleichwertig behandelt und geht mit dem gleichen Gewicht in die Berechnung der Wahrscheinlichkeit ein. Ein Ausweg aus dieser Situation bietet die Verwendung der sogenannten *softmax* Funktion:

$$p(y_* = \kappa | \mathbf{x}_*, \mathcal{D}) = \frac{\sum_{s_i=\kappa} \exp(-\alpha \cdot d(\mathbf{x}_*, \mathbf{x}_{\pi^{(i)}}))}{\sum_{j=1}^k \exp(-\alpha \cdot d(\mathbf{x}_*, \mathbf{x}_{\pi^{(j)}}))}. \quad (18.36)$$

Der Parameter $\alpha \geq 0$ bewirkt eine Art Glättung der sich ergebenen Wahrscheinlichkeitsverteilung. Für $\alpha = 0$ ergibt sich das Modell aus (18.35) und für $\alpha \rightarrow \infty$ erhalten wir eine 0/1-Verteilung.

Abstandsmaße Die Wahl eines Abstandsmaßes ist oft abhängig von der Anwendung. Eine Übersicht dazu ist in Abschn. 19.8.1 zu finden. Eine Standardwahl ist der quadrierte Euklidische Abstand zwischen den Merkmalsvektoren:

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2. \quad (18.37)$$

Dies erfordert zwingend, dass alle Merkmale gleich skaliert sind, zum Beispiel durch $0 \leq x_j \leq 1$, da ansonsten die Änderungen in den einzelnen Komponenten der Eingaben eine unterschiedliche Auswirkung auf die Distanz haben. Ein allgemeineres Abstandsmaß ist die Mahalanobisdistanz:

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T \mathbf{A} (\mathbf{x} - \mathbf{x}'), \quad (18.38)$$

dabei kann die Matrix \mathbf{A} eine unterschiedliche Gewichtung der einzelnen Merkmale bewirken. Ein typisches Beispiel ist die Verwendung einer Diagonalmatrix $\mathbf{A} = \text{diag}(\sigma_1^{-2}, \dots, \sigma_D^{-2})$ mit den inversen Varianzen der einzelnen Merkmale. Allgemeiner kann auch die inverse Kovarianzmatrix der Lerndaten als Matrix \mathbf{A} verwendet werden. Die Verwendung der inversen Kovarianzmatrix und des Mahalanobisabstandes erinnert sehr an die Dichte einer mehrdimensionalen Normalverteilung und ist daher verwandt zum Normalverteilungsklassifikator, welcher im nächsten Abschnitt genauer vorgestellt wird.

18.4.2 Normalverteilungsklassifikator

Der Normalverteilungsklassifikator ist ein typischer Vertreter der generativen Klassifikatoren (Abschn. 18.1.3). Oft wird in diesem Zusammenhang auch vom Bayes-Klassifikator gesprochen, obwohl dieser Begriff mehrdeutig ist und daher vermieden werden sollte.

Wir gehen im Folgenden von der Annahme aus, dass die Beispiele einer Klasse normalverteilt sind. Daraus ergibt sich direkt die folgende Angabe der Verteilungsdichte der Eingabedaten \mathbf{x} :

$$\begin{aligned} p(\mathbf{x} | y = \kappa) &= \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa) \\ &= \frac{1}{\sqrt{(2\pi)^D \det(\mathbf{S}_\kappa)}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_\kappa)^T \mathbf{S}_\kappa^{-1} (\mathbf{x} - \boldsymbol{\mu}_\kappa)\right). \end{aligned} \quad (18.39)$$

Dabei ist $\boldsymbol{\mu}_\kappa \in \mathbb{R}^D$ der Mittelwertvektor der Verteilung und $\mathbf{S}_\kappa \in \mathbb{R}^{D \times D}$ die Kovarianzmatrix. Aufgabe des Lernschrittes ist es, diese Parameter aus den gegebenen Lerndaten zu schätzen. Um die Maximum-Likelihood-Schätzwerte $(\hat{\boldsymbol{\mu}}_\kappa, \hat{\mathbf{S}}_\kappa)$ der Parameter einer multivariaten Normalverteilung zu erhalten, bestimmen wir zunächst die negative logarithmierte Likelihood-Funktion:

$$\begin{aligned} L(\boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa) &= - \sum_{\mathbf{x} \in \mathbf{X}_\kappa} \log \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa) \\ &\propto \sum_{\mathbf{x} \in \mathbf{X}_\kappa} \left(\log(\det(\mathbf{S}_\kappa)) + (\mathbf{x} - \boldsymbol{\mu}_\kappa)^T \mathbf{S}_\kappa^{-1} (\mathbf{x} - \boldsymbol{\mu}_\kappa) \right). \end{aligned} \quad (18.40)$$

Wir werden auf die genaue Herleitung verzichten, aber es lässt sich zeigen, dass die folgenden Schätzwerte die obige Funktion minimieren:

$$\hat{\boldsymbol{\mu}}_\kappa = \frac{1}{n_\kappa} \sum_{\mathbf{x} \in \mathbf{X}_\kappa} \mathbf{x}, \quad (18.41)$$

$$\hat{\mathbf{S}}_\kappa = \frac{1}{n_\kappa} \sum_{\mathbf{x} \in \mathbf{X}_\kappa} (\mathbf{x} - \hat{\boldsymbol{\mu}}_\kappa) (\mathbf{x} - \hat{\boldsymbol{\mu}}_\kappa)^T. \quad (18.42)$$

Das Lernen beschränkt sich demnach auf einfache algebraische Operationen.

Wie wir bereits in Abschn. 18.1.3 kennengelernt haben, ist die Klassifikation eines neuen Beispiels \mathbf{x}_* mit generativen Methoden äußerst intuitiv und es wird die Klasse y_* ausgewählt, welche die A-Posteriori-Verteilung maximiert. Wir folgen also unseren früheren Überlegungen und verwenden die Entscheidungsfunktion aus (18.8) zusammen mit unserer Normalverteilungsannahme in (18.39):

$$\begin{aligned} h(\mathbf{x}_*) &= \operatorname{argmax}_{\kappa \in \mathcal{Y}} (\log p(\mathbf{x}_* | y_* = \kappa, \mathcal{D}) + \log p(y_* = \kappa | \mathcal{D})) \\ &= \operatorname{argmin}_{\kappa \in \mathcal{Y}} \left(\log (\det(\mathbf{S}_\kappa)) + (\mathbf{x} - \boldsymbol{\mu}_\kappa)^T \mathbf{S}_\kappa^{-1} (\mathbf{x} - \boldsymbol{\mu}_\kappa) - \log p(y_* = \kappa | \mathcal{D}) \right). \end{aligned} \quad (18.43)$$

Zur Bestimmung der einzelnen Terme in der obigen Optimierung ist zum einen die Ermittlung der logarithmierten Determinante der Kovarianzmatrix nötig sowie deren Invertierung und Multiplikation mit $(\mathbf{x} - \boldsymbol{\mu}_\kappa)$.

Implementierung Eine Kovarianzmatrix ist immer positiv semidefinit, daher bietet sich das Cholesky-Verfahren für beide Probleme an. Eine Beschreibung des Verfahrens ist in vielen Standardwerken der numerischen Mathematik zu finden [56] und eine der Basismethoden in Programmierbibliotheken.

Nicht zu vernachlässigen sind bei diesem Verfahren dennoch die numerischen Schwierigkeiten. Stellen wir uns einmal vor, wir hätten viele Merkmale berechnet und erhalten jeweils Eingabevektoren der Dimension 1000. Dann sind auch mindestens 1000 Lernbeispiele pro Klasse erforderlich, um die Regularität der Kovarianzmatrizen $\hat{\mathbf{S}}_\kappa$ sicherzustellen, ansonsten lässt sich die Invertierung oder das Lösen des zugehörigen Gleichungssystems (Kap. 22) nicht bewerkstelligen. Die Anforderung mindestens D Lernbeispiele pro Klasse zu verwenden, ist bei vielen Anwendungen nicht realisierbar.

Eine einfache Lösung ist die Regularisierung der Matrizen, d. h. anstatt $\hat{\mathbf{S}}_\kappa$ verwenden wir die Matrix $\hat{\mathbf{S}}_\kappa + \lambda \mathbf{I}$ mit geeignet gewähltem $\lambda > 0$. Diese Art der Regularisierung wird in Abschn. 22.2 näher erläutert und motiviert. Eine weitere Möglichkeit ist die Berechnung der Pseudoinversen (Abschn. 22.1) oder die Verwendung einer Mischung aus Klassenkovarianzmatrix und der Kovarianzmatrix aller Daten.

Erweiterungen Ein großer Nachteil des Normalverteilungsklassifikators ist die Annahme der Unimodalität der Klassenverteilung. Es gibt immer ein eindeutiges Maximum jeder Klassenverteilung repräsentiert durch den Mittelwertvektor. In vielen Anwendungen ist diese Annahme nicht korrekt, so gibt es zum Beispiel bei der Bildklassifikation (Abschn. 20.2) oft unterschiedliche Arten von Objektdarstellungen, wie etwa Fotos oder Zeichnungen. Die Flexibilität des Klassifikators kann durch die Schätzung einer Mischverteilung erhöht werden (Abschn. 18.7.2). Eine weitere Möglichkeit ist die Anwendung des Kernel-Tricks, welcher in Abschn. 18.5 näher diskutiert wird.

Ein Normalverteilungsklassifikator kann auch bezüglich seiner Laufzeit verbessert werden. So kann zum Beispiel die Annahme getroffen werden, dass alle Merkmale unabhängig voneinander sind. Daraus ergibt sich eine diagonale Kovarianzmatrix und die Laufzeit während der Klassifikation verringert sich von $\mathcal{O}(D^2)$ auf $\mathcal{O}(D)$.

18.4.3 Entscheidungsbäume

Entscheidungsbäume gehören zu den ersten Methoden des maschinellen Lernens und wir werden im weiteren Verlauf des Abschnittes noch sehen, dass die angewandten Grundprinzipien auf viele andere Konzepte übertragbar sind. So ermöglicht das randomisierte Lernen von Entscheidungsbäumen, welches in Abschn. 18.4.4 näher erläutert wird, eine effiziente Klassifikation mit einfachsten algorithmischen Mitteln.

Zunächst werden wir uns aber kurz den ursprünglichen Ideen von Entscheidungsbäumen zuwenden. Wie der Name schon suggeriert, versuchen Entscheidungsbaumklassifikatoren den Entscheidungsprozess der Klassifikation als Baum abzubilden. Es werden dazu (binäre) Fragen gestellt die nacheinander angewendet zu einer Entscheidung führen. Aus Sicht des maschinellen Lernens zerlegen Entscheidungsbäume den Merkmalsraum rekursiv mit binären Klassifikatoren, welche wir als Basisklassifikatoren bezeichnen, bis eine (eindeutige) Klassifikationsentscheidung getroffen werden kann. Diese binären Klassifikatoren sind mit den inneren Knoten des Baumes (Trennknoten) assoziiert und bestimmen so den Pfad eines Beispiels \mathbf{x}_* im Baum. Es bestimmen daher immer mehrere Basisklassifikatoren die Gesamtklassifikation. Dieses Konzept wird Modellkombination genannt und erlaubt es, auch einfache Klassifikatoren zu verwenden. Die einfachsten Vertreter sind Schwellwertentscheidungen bei einem einzelnen Merkmal $r \in \{1, \dots, D\}$:

$$h(\mathbf{x}; r, \zeta) = \text{sign}(x_r - \zeta) = \begin{cases} 1 & \text{wenn } x_r > \zeta \\ -1 & \text{wenn } x_r \leq \zeta \end{cases} \quad (18.44)$$

Die Trennknoten zerlegen den Eingaberaum \mathcal{X} in einzelne Bereiche, welche gedanklich den m_ℓ Blattknoten zugeordnet werden können. In den Blattknoten eines Entscheidungsbäumes sind nun Informationen über die A-Posteriori-Verteilung der einzelnen Klassen für diese Region enthalten. Wir werden diese Verteilung im Folgenden mit $p(y = \kappa | \vartheta(\mathbf{x}))$ bezeichnen, wobei $\vartheta(\mathbf{x})$ das resultierende Blatt des Beispiels \mathbf{x} bezeichnet. Die eingeführten Bezeichnungen sind in Abb. 18.6 noch einmal dargestellt.

Durch die rekursive Zerlegung des Eingaberaumes ist theoretisch die Approximation beliebiger Klassenregionen möglich. Ein Entscheidungsbau kann daher eine hohe Modellkomplexität aufweisen. Ein weiterer Vorteil ist die schnelle Laufzeit bei der Klassifikation eines Beispiels. Für diese müssen nur höchstens $\mathcal{O}(\log m_\ell)$ Basisklassifikatoren ausgewertet werden.

Lernen von Entscheidungsbäumen Das Lernen eines Entscheidungbaumes umfasst zum einen die Struktur sowie die A-Posteriori-Wahrscheinlichkeiten in den Blattknoten. Die Suche nach einem optimalen Baum für einen gegebenen Lerndatensatz ist ein schwieriges kombinatorisches Problem, welches im allgemeinen Fall schwierig handhabbar ist.

Daher kommt folgende „gierige“ Strategie zum Einsatz: Ausgehend vom Wurzelknoten werden Basisklassifikatoren ausgewählt, welche die Lerndaten optimal nach einem gewis-

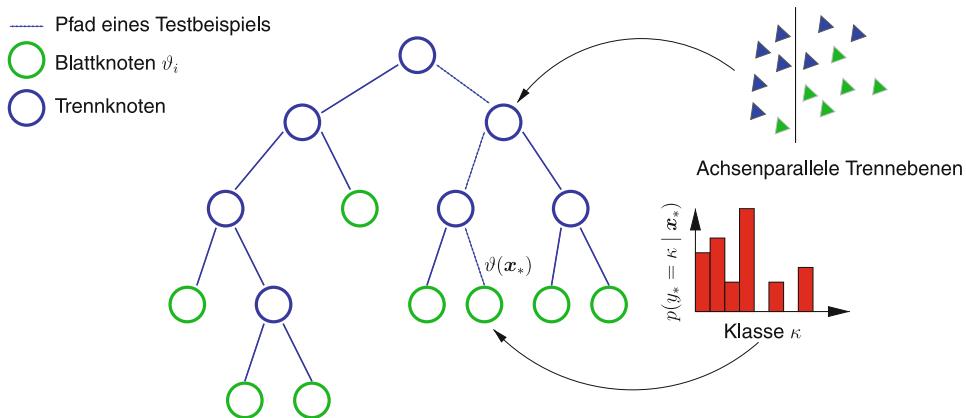


Abb. 18.6 Allgemeines Prinzip und verwendete Notationen bei Entscheidungsbäumen. Das Diagramm auf der rechten Seite veranschaulicht die A-Posteriori-Verteilung des Blattknotens. Der Pfad eines Testbeispiels im Baum (gestrichelte Linie) wird durch die Basisklassifikatoren in den Trennknoten bestimmt

sen Trennkriterium Γ auftrennen. In jedem Kindknoten wird dann dieser Schritt mit dem reduzierten Lerndaten rekursiv wiederholt. Dies erfolgt solange bis ein Abbruchkriterium erfüllt ist.

Trennkriterien basieren oft auf einem Maß für die Heterogenität einer Lernprobe. Dieses liefert hohe Werte für den Fall, dass Beispiele vieler verschiedener Klassen einen Knoten erreicht haben. Das bekannteste Beispiel für solch ein Maß ist die Entropie der Klassenbeschriftungen:

$$\mathcal{J}^{\text{Entropie}}(\nu) = - \sum_{\kappa=1}^M p_\kappa \log p_\kappa. \quad (18.45)$$

Dabei verwenden wir p_κ als Notation für den Anteil von Beispielen der Klasse κ im aktuellen Knoten ν . Ein mögliches Trennkriterium ergibt sich nun aus dem Vergleich der Heterogenität des aktuellen Knotens und der resultierenden Kindknoten:

$$\Gamma(\nu; r, \zeta) = \mathcal{J}(\nu) - \sum_{\nu' \text{ ist Kind von } \nu} p(\nu' | \nu) \mathcal{J}(\nu'). \quad (18.46)$$

Die Summe erfolgt dabei über jeden Kindknoten, welcher durch einen Basisklassifikator $h(\cdot; r, \zeta)$ erzeugt wird. Die Übergangswahrscheinlichkeit $p(\nu' | \nu)$ ist die Wahrscheinlichkeit, dass ein Beispiel des Knotens ν dem Kindknoten ν' zugeordnet wird. Diese Wahrscheinlichkeit kann durch einfaches Auszählen ermittelt werden.

Die Verwendung der Entropie liefert auch noch eine zusätzliche Interpretation des resultierenden Trennkriteriums. In diesem Fall ist Γ gleich bedeutend mit der wechselseitigen

Information der Klassenbeschriftung und der Ausgabe des Klassifikators. Die Optimierung der Zielfunktion in (18.46) bezüglich der Parameter der einfachen Schwellwertklassifikatoren wird meist durch eine vollständige Suche realisiert:

$$\left(\hat{r}_v, \hat{\zeta}_v \right) = \underset{1 \leq r \leq D, \zeta \in \mathbb{R}}{\operatorname{argmax}} \Gamma(v; r, \zeta). \quad (18.47)$$

Zu beachten ist, dass auch der Wertebereich des Schwellwertes ζ beschränkt ist, da nur eine Trennung zwischen Merkmalswerten unterschiedlicher Klassen sinnvoll ist.

Eine wesentliche Bedeutung kommt beim Lernen von Entscheidungsbäumen den Abbruchkriterien zu. Der einfachste Fall tritt auf, wenn nach der rekursiven Unterteilung der Lerndaten nur Beispiele einer Klasse in einem Knoten vorhanden sind. Zusätzlich ist eine weitere Auf trennung einer Lernmenge mit einer großen Anzahl von Beispielen einer Klasse und einer sehr geringen Anzahl von Beispielen einer anderen Klasse ebenfalls nicht sinnvoll. Als Abbruchkriterium kann auch das Unterschreiten des Heterogenitätsmaßes verwendet werden, d. h. es gilt $\mathcal{J}(v) < \xi_{\mathcal{J}}$ für den aktuellen Knoten v .

Ein weiteres Abbruchkriterium ist eine Mindestanzahl an Lernbeispielen. Dies ist erforderlich, da eine Ermittlung eines geeigneten Basisklassifikators mit nur wenigen Lernbeispielen nicht mehr robust ist. Wir hatten dieses Grundproblem der Mustererkennung in Abschn. 18.2.2 bereits kennengelernt. Bei Anwendungen mit einer hohen Anforderung an die Laufzeit während der Klassifikation ist eine zusätzliche Beschränkung der Baumtiefe sinnvoll.

Einer der Hauptkritikpunkte an Entscheidungsbäumen ist die Neigung zur starken Überanpassung. Diese entsteht ganz automatisch durch die hohe Modellkomplexität und die geringe Regularisierung durch einfache Abbruchkriterien. Der bekannteste Ansatz zur Umgehung dieser Schwäche sind sogenannte „Pruning“-Verfahren [10]. Diese verwenden einen zusätzlichen Validierungsdatensatz um den Baum „zurückzuschneiden“. Es wird also nachträglich die Komplexität des Klassifikators reduziert ohne seine Erkennungsleistung massiv zu beeinträchtigen.

18.4.4 Randomisierte Entscheidungsbäume

Randomisiertes Lernen ermöglicht es, Überanpassungseffekte zu verringern und die Laufzeit des Lernschrittes oft drastisch zu reduzieren. Bei randomisierten Entscheidungsbäumen kann eine Randomisierung auf mehreren Ebenen erfolgen.

Eine Möglichkeit haben wir bereits in Abschn. 18.3.4 beim sogenannten Bagging-Ansatz kennengelernt. Die Anwendung dieses Konzeptes bei Entscheidungsbäumen geht auf Breiman [9] zurück. Anstatt nur einen Entscheidungsbau zu lernen, wird eine Menge von Bäumen gelernt, welche auch als Ensemble oder zufälliger Wald (engl. *random forest*) bezeichnet werden. Um die Diversität der Bäume sicherzustellen, wird zum Lernen eines Baumes immer eine zufällige Auswahl der Lerndaten verwendet.

Die Entscheidungen der einzelnen Entscheidungsbäume müssen bei der Klassifikation zusammengeführt werden. Beim Bagging Ansatz erfolgt dies durch eine Mittelung der jeweiligen A-Posteriori-Verteilungen. Wir werden im Folgenden das Blatt des Baumes t , welches von Beispiel \mathbf{x} erreicht wird, mit $\vartheta^{(t)}(\mathbf{x})$ bezeichnen. Die Klassifikationsentscheidung kann dann wie folgt dargestellt werden:

$$p(y_* | \mathbf{x}_*, \mathcal{D}) = \frac{1}{T} \sum_{t=1}^T p(y_* | \mathbf{x}_*, \vartheta^{(t)}(\mathbf{x}_*)). \quad (18.48)$$

Dieser Ansatz ist vor allem bei unsicheren Beschriftungen sinnvoll.

Der Parameter r_B des Bagging-Ansatzes, welcher den Anteil der zufällig gezogenen Lernbeispiele angibt, muss vorsichtig gewählt werden. Wählen wir einen kleinen Wert für r_B stehen relativ wenige Lernbeispiele zur Verfügung. Zusätzlich erlaubt es aber eine geringe Laufzeit des Lernschrittes.

Out-of-Bag-Schätzungen Ein weiterer Vorteil des Bagging Konzeptes ist die Verfügbarkeit von Out-of-Bag-Schätzungen. Da nur ein gewisser Anteil von Lerndaten verwendet wird, kann der restliche Teil dazu verwendet werden, eine Vorhersage der voraussichtlichen Erkennungsrate zu treffen. Jedes Beispiel \mathbf{x}_i der Lerndaten wird einfach mit der Menge an Entscheidungsbäumen klassifiziert, die \mathbf{x}_i nicht zum Lernen verwendet haben. Durch die geschätzten Erkennungsraten kann eine Modellauswahl erfolgen und die Parameter des Klassifikators können ohne die Verwendung einer expliziter Validierungsmenge optimiert werden.

Zufällige Unterräume Eine ähnliche randomisierte Methode ist der *random subspace* Ansatz von [31]. Jeder Baum wird zwar mit allen Beispielen gelernt, aber nur mit einer beschränkten Menge von Merkmalen. Eine Verallgemeinerung ist die Verwendung von zufällig linear transformierten Merkmalen [38]. Diese Methode ist auch als *rotation forest* bekannt und suggeriert bereits die Motivation, die diesem Verfahren zugrunde liegt. Wenn wir zum Beispiel eine Klassenverteilung haben, die nicht in achsenparalleler Lage ist, dann benötigt ein Entscheidungsbaum eine Vielzahl von Schwellwertklassifikatoren (18.44) um die Klassifikationsgrenze zu approximieren. Werden hingegen die Eingabedaten vorher rotiert, so ist eine Abtrennung des Klassengebietes mit nur wenigen Basisklassifikatoren möglich. Genau diese Grundidee versucht ein *rotation forest* auszunutzen um kompaktere Entscheidungsbäume zu erhalten, die eine bessere Generalisierung ermöglichen.

Randomisiertes Lernen der Basisklassifikatoren Anstatt eine zufällige Auswahl von Merkmalen für einen kompletten Baum vorzunehmen, kann auch eine randomisierte Optimierung der Basisklassifikatoren erfolgen [28]. Wir haben in Abschn. 18.4.3 gesehen, dass bei den traditionellen Lernverfahren für Entscheidungsbäume eine vollständige Parametersuche erfolgt. Der Ansatz von Geurts et al. [28] wählt daher zufällig eine Untermenge

\mathcal{R}_v aller Merkmale sowie eine Auswahl Q_v von möglichen Schwellwerten in jedem Knoten v aus. Nach der Auswahl wird die Optimierung auf diese beiden Mengen eingeschränkt:

$$\left(\hat{r}_v, \hat{\zeta}_v \right) = \underset{r \in \mathcal{R}_v, \zeta \in Q_v}{\operatorname{argmax}} \Gamma(v; r, \zeta). \quad (18.49)$$

Diese Strategie ist besonders bei großen Lernmengen sinnvoll, wie sie zum Beispiel bei der semantischen Segmentierung auftreten (Abschn. 20.4.2) und wird in der Literatur auch unter dem Namen *extra trees* geführt. Die Randomisierung dient wie beim Bagging-Ansatz wieder dazu die Diversität der Klassifikatoren in einem Ensemble zu erhöhen und dadurch die Korrelation zu verringern (siehe Analyse in Abschn. 18.3.4).

18.4.5 Boosting

Ähnlich wie bei Entscheidungsbäumen werden auch beim *Boosting*-Verfahren mehrere Basisklassifikatoren h_t miteinander kombiniert. Die einzelnen Ausgaben der Basisklassifikatoren werden gewichtet addiert um zunächst eine kontinuierliche Bewertung zu erhalten:

$$f(\mathbf{x}_*; \boldsymbol{\alpha}, (\boldsymbol{\theta}_t)_{t=1}^T) = \sum_{t=1}^T \alpha_t \cdot h_t(\mathbf{x}_*; \boldsymbol{\theta}_t). \quad (18.50)$$

Eine übliche Wahl für die Basisklassifikatoren ist wieder eine einfache Schwellwertentscheidung (18.44) mit den Parametern $\boldsymbol{\theta} = (r, \zeta)$. Die Grundidee von Boosting-Verfahren ist es nun die Gewichte α_t und die Basisklassifikatoren h_t (bestimmt durch Parameter $\boldsymbol{\theta}_t$) sequentiell auszuwählen, so dass ein gewichtetes Fehlermaß für die Lerndaten minimiert wird. Jedem Lernbeispiel \mathbf{x}_i wird ein Gewicht w_i zugeordnet. Nehmen wir einfach einmal an, dass wir bei gegebenen Gewichten diese Minimierung durchführen können und immer genau einen Basisklassifikator h_t und einen dazu passenden Koeffizienten α_t erhalten. Wie wir bereits beim Bagging-Verfahren in Abschn. 18.3.4 gesehen haben, ist es zwingend notwendig, möglichst unterschiedliche Basisklassifikatoren miteinander zu kombinieren. Die Erzwingung dieser Variabilität wird beim Boosting-Verfahren durch eine Adaption der Gewichte erreicht. Nach jedem Schritt werden die Gewichte der Lernbeispiele so geändert, dass bisher falsch klassifizierte Beispiele ein hohes Gewicht bekommen und bereits korrekt klassifizierte Beispiele ein niedriges Gewicht. Diese allgemeine Verfahrensweise ist bei allen Boosting-Verfahren gleich. Im Folgenden wollen wir die genaue Funktionsweise von AdaBoost herleiten, in dem wir den Zusammenhang zur Optimierung herstellen.

Wenn wir den Fehler des aktuellen Klassifikators auf den Lerndaten bestimmen wollen, dann bietet es sich an einzelne Fehlerterme für jedes einzelne Lernbeispiel zu summieren:

$$E\left(\boldsymbol{\alpha}, (\boldsymbol{\theta}_t)_{t=1}^T\right) = \sum_{i=1}^n L(y_i \cdot f(\mathbf{x}_i)), \quad (18.51)$$

wobei die Funktionen f analog zu (18.50) definiert sind, wir aber auf die explizite Angabe der Parameter aus Gründen der Übersichtlichkeit verzichtet haben. Die Funktion L ist eine Fehlerfunktion für ein einzelnes Beispiel. Wir erinnern uns an dieser Stelle noch an die

Definitionen bei der aktuell betrachteten binären Aufgabenstellung. Die Beschriftungen y_i sind -1 für negative und $+1$ für positive Beispiele. Während des Lernens versuchen wir eine Funktion zu finden, welche ähnliche Werte für neue Testbeispiele ohne Beschriftung ermittelt. Das Vorzeichen der Ausgabe $f(\mathbf{x}_i)$ des Klassifikators sollte demnach den Beschriftungen y_i entsprechen um einen möglichst geringen Fehler auf der Lernstichprobe zu ergeben. Aus diesem Grund wird auch nur das Produkt dieser beiden Werte als Argument der Fehlerfunktion L in (18.51) übergeben. Eine Funktion L sollte demnach so konstruiert werden, so dass $L(z)$ für positive Werte z klein ist und für negative Werte z einen großen Wert und Fehler zurück gibt. Wir werden in den weiteren Abschnitten noch sehen, dass es mehrere Möglichkeiten gibt, L festzulegen. Bei der direkten Optimierung des gesamten Fehlers in (18.51) ist es hilfreich eine stetige und differenzierbare Funktion L zu wählen um eine gradientenbasierte Optimierung durchführen zu können.

Wenden wir uns nun wieder dem konkreten Fall des Boosting-Verfahrens zu. Wir verwenden in diesem Fall eine Exponentialfunktion um den Fehler L auszudrücken:

$$E\left(\alpha, (\boldsymbol{\theta}_t)_{t=1}^T\right) = \sum_{i=1}^n \exp(-y_i \cdot f(\mathbf{x}_i)). \quad (18.52)$$

Ein üblicher Ansatz in der Optimierung ist es, sequentiell nur eine bestimmte Teilmenge der Parameter zu optimieren. Um die Basisklassifikatoren der Reihe nach anzulernen, nehmen wir an, dass die ersten $T - 1$ Basisklassifikatoren und Koeffizienten konstant sind und wir nur bezüglich $\boldsymbol{\theta}_T$ und α_T optimieren. Der gesamte Fehlerterm (18.52) lässt sich dann schreiben als:

$$E\left(\alpha, (\boldsymbol{\theta}_t)_{t=1}^T\right) = \sum_{i=1}^n \exp\left(-y_i \alpha_T h(\mathbf{x}_i; \boldsymbol{\theta}_T) - y_i \sum_{t=1}^{T-1} \alpha_t \cdot h(\mathbf{x}_i; \boldsymbol{\theta}_t)\right) \quad (18.53)$$

$$= \sum_{i=1}^n w_i \cdot \exp(-y_i \alpha_T h(\mathbf{x}_i; \boldsymbol{\theta}_T)). \quad (18.54)$$

Dabei haben wir den konstanten Teil der Ausgabe des Klassifikators als Gewichte w_i definiert. Wir gehen im Folgenden davon aus, dass die Basisklassifikatoren diskrete Ausgaben zurück liefern, d. h. $h(\mathbf{x}; \boldsymbol{\theta}_T) \in \{-1, 1\}$. Dies ermöglicht die Aufteilung der Summe in (18.54) in einen Teil \mathcal{P} der Lerndaten welcher durch den Basisklassifikator korrekt klassifiziert wurde ($h(\mathbf{x}; \boldsymbol{\theta}_T) \cdot y_i = 1$) und einen anderen Teil \mathcal{N} der Lerndaten welcher falsch klassifiziert wurde:

$$E\left(\alpha, (\boldsymbol{\theta}_t)_{t=1}^T\right) = \exp(\alpha_T) \cdot \left(\sum_{i \in \mathcal{N}} w_i \right) + \exp(-\alpha_T) \cdot \left(\sum_{i \in \mathcal{P}} w_i \right) \quad (18.55)$$

$$\begin{aligned} &= (\exp(\alpha_T) - \exp(-\alpha_T)) \sum_{i=1}^n w_i \cdot \delta(h(\mathbf{x}_i; \boldsymbol{\theta}_T) \neq y_i) \\ &\quad + \exp(-\alpha_T) \sum_{i=1}^n w_i. \end{aligned} \quad (18.56)$$

Betrachten wir zunächst die Minimierung dieser Fehlerfunktion bezüglich des Basisklassifikators $h(\cdot; \theta_T)$. Für die Optimierung ist der multiplikative Term am Anfang sowie der additive Term am Ende von (18.56) irrelevant. Daher genügt es die Parameter des Klassifikators durch folgendes Optimierungsproblem zu finden:

$$\hat{\theta}_T = \operatorname{argmin}_{\theta_T} \sum_{i=1}^n w_i \cdot \delta(h(\mathbf{x}_i; \theta_T) \neq y_i). \quad (18.57)$$

Im Falle von einfachen Schwellwertentscheidungen als Basisklassifikatoren lässt sich diese Optimierung ähnlich wie bei Entscheidungsbäumen mit einer einfachen vollständigen Suche in der sortierten Liste von Merkmalswerte durchführen. Werden als Basisklassifikatoren SVM Klassifikatoren verwendet (Abschn. 18.4.6), so erfolgt ein Lernen dieser Klassifikatoren mit Gewichten für jedes Beispiel, aber dazu später mehr.

Um den aktuellen Schritt im Boostingverfahren zu vervollständigen fehlt noch die Bestimmung des Koeffizienten α_T . Dazu leiten wir die Fehlerfunktion nach α_T ab und setzen die Ableitung auf Null:

$$0 = \exp(\alpha_T) \cdot \left(\sum_{i \in \mathcal{N}} w_i \right) - \exp(-\alpha_T) \cdot \left(\sum_{i \in \mathcal{P}} w_i \right) \quad (18.58)$$

Eine kurze Rechnung ergibt:

$$\exp(-\alpha_T) \cdot \left(\sum_{i \in \mathcal{P}} w_i \right) = \exp(\alpha_T) \cdot \left(\sum_{i \in \mathcal{N}} w_i \right) \quad (18.59)$$

$$-\alpha_T + \log \left(\sum_{i \in \mathcal{P}} w_i \right) = \alpha_T + \log \left(\sum_{i \in \mathcal{N}} w_i \right). \quad (18.60)$$

Diese Gleichung führt dann zu einer geschlossenen Formel für den Koeffizienten α_T :

$$\alpha_T = \frac{1}{2} \left(\log \left(\sum_{i \in \mathcal{P}} w_i \right) - \log \left(\sum_{i \in \mathcal{N}} w_i \right) \right). \quad (18.61)$$

Anhand dieser Formel lässt sich die Bedeutung der Koeffizienten gut ablesen: Ist das Gesamtgewicht der korrekt klassifizierten Beispiele durch den aktuellen Basisklassifikator hoch, so erhält dieser ein hohes Gewicht. Ist das Gesamtgewicht der falsch klassifizierten Beispiele höher als das der korrekt klassifizierten Beispiele, so erhält dieser sogar ein negatives Vorzeichen im Gesamtklassifikator f um seine Entscheidungen umzukehren. Es lässt sich leicht nachprüfen, dass obige Formel für α_T äquivalent zu folgender Berechnungsvorschrift ist, welche man üblicherweise in der Literatur findet:

$$\varepsilon_T = \left(\sum_{i \in \mathcal{N}} w_i \right) / \left(\sum_{i=1}^n w_i \right), \quad (18.62)$$

$$\alpha_T = \log \left(\frac{1 - \varepsilon_T}{\varepsilon_T} \right). \quad (18.63)$$

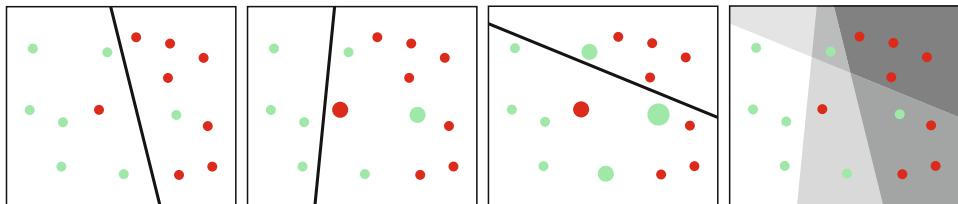


Abb. 18.7 Einzelne Schritte des AdaBoost-Verfahrens mit binären SVMs als Basisklassifikatoren: Einzelne Punkte entsprechen Lernbeispielen und die Größe dieser ist proportional zum aktuellen Gewicht des Beispiels gewählt. Dargestellt sind in den ersten drei Abbildungen die gelernten Basisklassifikatoren sowie die dafür verwendeten gewichteten Beispiele. Die vierte Abbildung zeigt die kontinuierliche Ausgabe des kombinierten Klassifikators anhand von Graustufen

Boostingverfahren im Überblick Wir haben nun alle notwendigen Teile um das Boostingverfahren zu realisieren und wollen dies nun als Algorithmus kurz schematisch zusammenfassen:

1. Setzen der Gewichte auf eine Gleichverteilung: $\forall 1 \leq i \leq n : w_i = \frac{1}{n}$
2. Sequentielle Bestimmung aller Basisklassifikatoren durch:
 - (a) Bestimmung des Basisklassifikators durch Lösen des Optimierungsproblems (18.57)
 - (b) Ermitteln des Koeffizienten durch (18.62) und (18.63)
 - (c) Neuberechnung der Gewichte durch:

$$w_i := w_i \cdot \exp(-\alpha_T \delta(h(\mathbf{x}_i; \boldsymbol{\theta}_T) \neq y_i)) \quad (18.64)$$

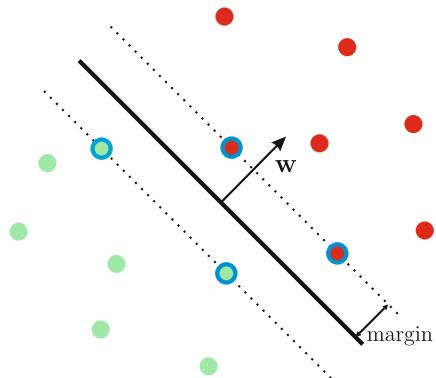
Die Gleichung für die Aktualisierung der Gewichte ergibt sich durch die indirekte Definition von w_i in (18.54). Abbildung 18.7 zeigt die sequentielle Bestimmung der Basisklassifikatoren anhand eines zweidimensionalen Beispiels.

Die Klassifikation beschränkt sich auf die Auswertung der Basisklassifikatoren und der linearen Kombination der Ergebnisse. Da wir Basisklassifikatoren verwenden, bei denen das Vorzeichen die Klassifikationsentscheidung bestimmt, gilt das Gleiche für den resultierenden Boosting-Klassifikator. An dieser Stelle ist zu beachten, dass manche Präsentation des Boosting-Verfahrens abweichen und Basisklassifikatoren mit Ausgaben im Bereich von 0 bis 1 annehmen.

18.4.6 Support-Vektor-Klassifikator

Der Support-Vektor-Klassifikator (engl.: *support vector machines*, SVM oder SVM-Klassifikator) hat sich im Laufe der Jahre zu einem der meist verwendeten Klassifikatoren entwickelt, welcher sich besonders bei visuellen Erkennungsaufgaben durch ihre Generalisierungsfähigkeit und der geringen Rechenzeit des Lern- und Testschrittes auszeichnen. In

Abb. 18.8 Darstellung der Maximierung des Gebietes zwischen den Klassen (engl. *margin*) durch einen linearen SVM-Klassifikator: Punkte repräsentieren die zweidimensionalen Lerndaten und sind zusätzlich markiert, falls sie Support Vektoren darstellen



seiner ursprünglichen Version ist ein SVM-Klassifikator ein binärer linearer Klassifikator, d. h. zwei Klassen werden durch eine Hyperebene voneinander getrennt. Im Laufe dieses Kapitels werden wir noch Möglichkeiten kennenlernen, wie zum einen die Erweiterung auf nicht-lineare Trennfunktionen und wie zum anderen eine Mehrklassen-Klassifikation ermöglicht wird.

Wenden wir uns zunächst der ursprünglichen Formulierung des SVM-Ansatzes zu. Die Beschriftungen y sind zunächst binär und wir wählen $y = -1$ für die Kennzeichnung der negativen Beispiele und $y = 1$ für die der positiven Klasse. Wie wir bereits schon im vorherigen Abschnitt erwähnt haben, versucht ein SVM-Klassifikator eine lineare Trennebene für die Trennung zu verwenden:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b. \quad (18.65)$$

Die Trennebene wird dabei durch die Normale $\mathbf{w} \in \mathbb{R}^D$ und den Offset $b \in \mathbb{R}$ repräsentiert und wir verwenden die Schreibweise $\langle \cdot, \cdot \rangle$ als Notation eines Skalarproduktes. Die Funktion $f(\mathbf{x})$ beschreibt unsere kontinuierliche Bewertungsfunktion, von der das Vorzeichen die Klassenentscheidung treffen soll. Für den Lernschritt ist $\mathcal{D} = ((\mathbf{x}_i, y_i))_{i=1}^n \subset \mathcal{X} \times \{-1, 1\}$ gegeben. Wir werden zunächst annehmen, dass diese Lerndaten theoretisch linear trennbar sind. Dies bedeutet, dass es eine Hyperebene (\mathbf{w}, b) gibt, so dass für jedes Lernbeispiel \mathbf{x}_i das Vorzeichen von $f(\mathbf{x}_i)$ mit dem von y_i übereinstimmt.

Eine Festlegung einer trennenden Hyperebene ist auf viele Arten möglich. Daher benötigen wir noch eine zusätzliche Anforderung an die Hyperebene. Die charakteristische zusätzliche Bedingung bei SVM-Klassifikatoren ist die Maximierung des kleinsten Abstandes der Hyperebene zu einem Lernbeispiel. Dieser Abstand wird auch als *margin* bezeichnet und ist für SVM-Klassifikatoren von entscheidender Bedeutung, daher werden diese auch oft als *maximum margin* Klassifikatoren bezeichnet. Der Name Support-Vektor-Klassifikator leitet sich hingegen von sogenannten Support Vektoren ab, welche ebenfalls eine besondere Rolle besitzen. Ein Lernbeispiel wird als Support Vektor bezeichnet, wenn es den kleinsten Abstand zur Hyperebene hat. Diese Beispiele definieren die Hyperebene. Dieses Grundprinzip ist in Abb. 18.8 noch einmal dargestellt.

Die offene Frage ist nun, wie sich diese beschriebenen Aspekte mathematisch beschreiben lassen, um einen Algorithmus zur Bestimmung der Parameter (\mathbf{w}, b) der Hyperebene aus den Lerndaten abzuleiten. Die zwei Forderungen nach linearer Separierung und der Maximierung des *margin* führen direkt zu folgendem Optimierungsproblem:

$$\max_{\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}} \text{mg}_{\mathcal{D}}(\mathbf{w}, b) \quad (18.66)$$

Nb. $\forall i = 1 \dots n : y_i \cdot f(\mathbf{x}_i) > 0$.

Die Ungleichungen in den Nebenbedingungen stellen die lineare Separierung sicher, d. h. jedes Lernbeispiel sollte sich auf der richtigen Seite der Hyperebene befinden. Der *margin* lässt sich wie folgt definieren:

$$\text{mg}_{\mathcal{D}}(\mathbf{w}, b) = \min \{ \|\mathbf{x} - \mathbf{x}_i\| : 1 \leq i \leq n; \langle \mathbf{w}, \mathbf{x} \rangle + b = 0; \mathbf{x} \in \mathcal{X} \} \quad (18.67)$$

$$= \min_{i=1 \dots n} \frac{|\langle \mathbf{w}, \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|}. \quad (18.68)$$

Zu beachten ist nun, dass eine Skalierung der Normale \mathbf{w} und des Offsets b mit einem skalaren Wert $\lambda > 1$ zu keiner Verletzung der Nebenbedingungen oder zu einer Änderung des *margin* führt. Aus diesem Grund kann die Skalierung theoretisch beliebig festgelegt werden. Wenn wir aber die Skalierung genau so festlegen, dass $|\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$ für jedes Lernbeispiel \mathbf{x}_i mit minimaler Distanz zur Hyperebene gilt, dann erhalten wir so eine deutliche Vereinfachung des *margin* zu $\text{mg}_{\mathcal{D}}(\mathbf{w}, b) = 1 / \|\mathbf{w}\|$.

Das SVM Optimierungsproblem lässt sich nun folgendermaßen formulieren:

$$\min_{\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}} \frac{1}{2} \|\mathbf{w}\|^2 \quad (18.69)$$

Nb. $\forall i = 1 \dots n : y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$.

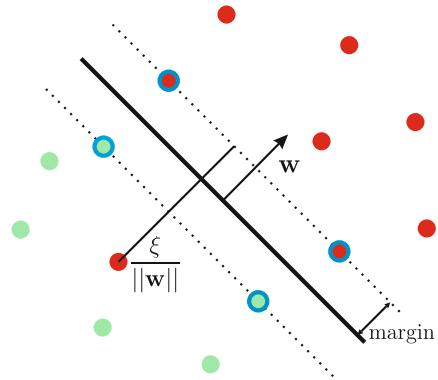
Der zusätzliche Vorfaktor $\frac{1}{2}$ wird sich im weiteren Verlauf als vorteilhaft für eine kompaktere Herleitung herausstellen. Eine wichtige Änderung ist der Wechsel der rechten Seite der Nebenbedingung von > 0 zu ≥ 1 . Wenn wir die alte Nebenbedingung $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$ betrachten und zusätzlich die Definition der Skalierung von \mathbf{w} und b einfließen lassen, lässt sich dieser Schritt schnell begründen:

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = |y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)| = |y_i| \cdot |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \quad (18.70)$$

$$= |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \geq 1. \quad (18.71)$$

Das Optimierungsproblem des SVM-Klassifikators ist ein quadratisches Optimierungsproblem, oft auch als quadratisches Programm bezeichnet. Für diese Art von Optimierungsproblemen gibt es zahlreiche effiziente Lösungsverfahren [63]. Es sei aber an dieser Stelle darauf hingewiesen, dass für eine wirklich effiziente Implementierung allgemeine Verfahren für quadratische Optimierungsprobleme oft nicht geeignet sind. Besonders für große Datenmengen wird oft auf schnelle einfache Abstiegsverfahren zurückgegriffen.

Abb. 18.9 Darstellung der SVM-Klassifikation bei nicht-linear-separierbaren Lerndaten und Einführung von Schlupfvariablen



Nicht-linear-separierbare Lerndaten Wir hatten am Anfang des vorherigen Abschnittes die Annahme getroffen, dass unsere Lernmenge linear separierbar ist. Dies trifft natürlich nicht immer zu, gerade bei einer hohen Intraklassenvarianz und geringen Interklassendistanz oder bei niedrig dimensionalnen Daten. Als Standardbeispiel für ein nicht-linear-separierbares Problem wird immer das Lernen der XOR-Funktion angeführt.

Um die Forderung der linearen Separierbarkeit abzuschwächen, ist eine Änderung der Nebenbedingung in (18.69) notwendig. Wir führen Schlupfvariablen $\xi_i \geq 0$ in das Optimierungsproblem (18.69) ein, welche es ermöglichen, dass ein Lernbeispiel \mathbf{x}_i die Ungleichung in den Nebenbedingungen nicht zwangsweise erfüllen muss:

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i. \quad (18.72)$$

Natürlich soll eine Verletzung der Nebenbedingung bei $\xi_i > 0$ bestraft werden. Dies gelingt durch die Inklusion eines zusätzlichen Straftermes in die Zielfunktion und wir erhalten folgendes Optimierungsproblem:

$$\min_{\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad (18.73)$$

$$\text{Nb. } \forall i = 1 \dots n : y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \text{ und } \xi_i \geq 0 .$$

Der resultierende Klassifikator wird oft als *soft margin classifier* [63, S. 16] bezeichnet und ist in Abb. 18.9 für ein zweidimensionales Beispiel dargestellt. Der Parameter $C > 0$ kann verwendet werden, um einen Ausgleich zwischen einem großen *margin* und der Anzahl der Fehlklassifikationen des Lerndatensatzes zu schaffen. Dabei ist der optimale Wert sehr spezifisch für eine Klassifikationsaufgabe und den gegebenen Lerndaten. Wir werden die Auswirkungen dieses Parameters in späteren Abschnitten noch untersuchen (Abschn. 18.5.9).

Das obige Optimierungsproblem kann auch ohne Nebenbedingungen formuliert werden. Dies ist für den späteren Vergleich mit anderen Verfahren vorteilhaft (Abschn. 18.5.9).

Zunächst definieren wir die sogenannte *hinge loss* H :

$$H(z) = \max(1 - z, 0). \quad (18.74)$$

Diese Funktion erlaubt die Darstellung von (18.73) als einzelne Zielfunktion:

$$\min_{\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}} \sum_{i=1}^n H(y_i \cdot f(\mathbf{x}_i)) + \lambda \|\mathbf{w}\|^2. \quad (18.75)$$

Dabei haben wir aus Gründen der Übersichtlichkeit die Abkürzung $\lambda = (2C)^{-1}$ verwendet. Wie wir in Abschn. 18.5.9 noch genauer sehen werden, approximiert die *hinge loss* die binäre Fehlerfunktion $\delta(z < 0)$. Daher minimiert der SVM Ansatz einen Fehlerterm mit einer zusätzlichen Regularisierung, welche die Komplexität der Entscheidungsfunktion beeinflusst. Dieses Prinzip werden wir noch häufiger sehen und es ist auch verwandt zu der Maximum-A-Posteriori Schätzung in Abschn. 18.3.1. Der Fehlerterm korrespondiert zu einer negativen logarithmierten Likelihood und $\|\mathbf{w}\|^2$ entspricht der negativen logarithmierten Normalverteilung als A-Priori-Annahme für \mathbf{w} . Zu beachten ist, dass diese Analogie keine mathematische Äquivalenz darstellt, da es zur *hinge loss* als Fehlerterm keine korrespondierende Wahrscheinlichkeitsverteilung gibt.

Mehrklassen-Klassifikation Bisher haben wir nur die binäre Klassifikation mit Beschriftungen $y_i \in \{-1, 1\}$ betrachtet. Eine Erweiterung für den Mehrklassenfall mit $y_i \in \{1, \dots, M\}$ ist auf unterschiedliche Art und Weise möglich. So gibt es zum Beispiel Ansätze, welche das SVM Optimierungsproblem für den Mehrklassenfall adaptieren. Eine Beschreibung dieser Ansätze ist in [63, Abschn. 7.6.4] zu finden. Im Folgenden werden wir Methoden betrachten die im Gegensatz dazu mehrere binäre Klassifikatoren kombinieren. Die Methoden lassen sich dadurch für die Erweiterung aller möglichen binären Klassifikatoren auf den Mehrklassenfall einsetzen.

Sei ein Lerndatensatz $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ mit $y_i \in \{1, \dots, M\}$ und $M > 2$ gegeben. Der sogenannte *one-vs-all* or *one-vs-rest* Ansatz lernt insgesamt M binäre SVM Klassifikatoren f^κ , welche jeweils versuchen die Klasse κ von allen anderen Klassen zu trennen. Die Klassifikatoren werden mit den binären Datensätzen $\mathcal{D}^\kappa = \{(\mathbf{x}_1, \tilde{y}_1), \dots, (\mathbf{x}_n, \tilde{y}_n)\}$ angelernt, wobei die binäre Beschriftung \tilde{y}_i genau dann 1 ist, wenn \mathbf{x}_i ein Beispiel für Klasse κ ist, d. h. $y_i = \kappa$. In allen anderen Fällen wird \tilde{y}_i auf -1 gesetzt. Für die Klassifikation eines neuen Beispiels \mathbf{x}_* wird nach der Lernphase jeder binäre Klassifikator $f^{(\kappa)}$ ausgewertet und es wird die Klasse κ gewählt, welche die höchste Bewertung $f^{(\kappa)}(\mathbf{x}_*)$ erhält:

$$h(\mathbf{x}_*) = \operatorname{argmax}_{\kappa=1 \dots M} f^{(\kappa)}(\mathbf{x}_*). \quad (18.76)$$

Entscheidend ist an dieser Stelle also die Verwendung kontinuierlicher Bewertungsfunktionen $f^{(\kappa)}$. Die Verwendung von diskreten Entscheidungen der M binären Klassifikatoren könnte zu Mehrdeutigkeiten führen, wenn zum Beispiel mehrere dieser Klassifikatoren die Entscheidung treffen, dass \mathbf{x}_* ein positives Beispiel ist. Eine zusätzliche Möglichkeit ist die

Anwendung des Algorithmus von Platt [55], um die Bewertungen $f^{(\kappa)}$ in approximierte Wahrscheinlichkeiten umzuwandeln.

Eine weitere Methode ist das sogenannte *one-vs-one* Prinzip. Dabei werden insgesamt $\binom{M}{2}$ Klassifikatoren $f^{(\kappa, \kappa')}$ angelernt, welche jeweils versuchen Klasse κ und κ' zu trennen. Für die Klassifikation eines neuen Beispiels \mathbf{x}_* werden wieder alle Klassifikatoren ausgewertet und die Anzahl der Klassifikationsentscheidungen für Klasse κ gezählt. Eine Klasse, welche die meisten Stimmen bekommt, kann als Ergebnis des Mehrklassenklassifikators verwendet werden. Bei dieser Technik ist die Berechnung von Wahrscheinlichkeiten für jede Klasse direkt durch eine Normierung der Auszählung möglich.

18.5 Kernelbasierte Klassifikation

Im Folgenden werden wir uns mit nicht-linearen Klassifikatoren beschäftigen, welche auf sogenannten Kernelfunktionen basieren. Wie wir bereits in Abschn. 18.2 sehen werden, erfordert die visuelle Objekterkennung eine hohe Komplexität der Klassifikationsmodelle. Mittels randomisierter Entscheidungsbäume (Abschn. 18.4.4) ist es zwar möglich nicht-lineare Entscheidungsfunktionen zu realisieren, dies erfolgt aber durch eine stückweise lineare Approximation und nicht durch eine direkte nicht-lineare Beschreibung. Kernelklassifikatoren, wie etwa Gauß-Prozess-Klassifikatoren und SVM mit Kernelfunktionen, ermöglichen hingegen die Verwendung nicht-lineare Entscheidungsfunktionen ohne eine explizite Parametrisierung.

18.5.1 Kernelfunktionen und nichtlineare Klassifikation

Viele Klassifikatoren nehmen an, dass die Lerndaten linear trennbar sind und eine lineare Entscheidungsfunktion daher ausreichend ist. In Abschn. 18.4.6 haben wir dies am Beispiel eines SVM-Klassifikators kennengelernt. Wir wollen im Folgenden diese Klassifikatoren als *lineare Klassifikatoren* bezeichnen, obwohl sie oft zusätzlich noch einen affinen Anteil beinhalten.

Abbildung 18.10 zeigt ein binäres Klassifikationsbeispiel, bei welchem die Eigenschaft der linearen Separierbarkeit nicht gegeben ist. Ein linearer Klassifikator kann in diesem Beispiel daher keine vernünftige Trennung der Daten durchführen. Im Folgenden werden wir bei solchen Problemstellungen von *nicht-linearen Klassifikationsproblemen* sprechen. Ein explizites nicht-lineares Klassifikationsmodell kann die Lernbeispiele natürlich sehr wohl voneinander trennen. Zum Beispiel ist es möglich, einen Kreis zu bestimmen, welcher nur die Beispiele der einen Klasse beinhaltet:

$$h(\mathbf{x}) = \text{sign} (w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2 + b) \quad (18.77)$$

$$= \begin{cases} 1 & \text{wenn } w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2 > -b \\ -1 & \text{sonst.} \end{cases} \quad (18.78)$$

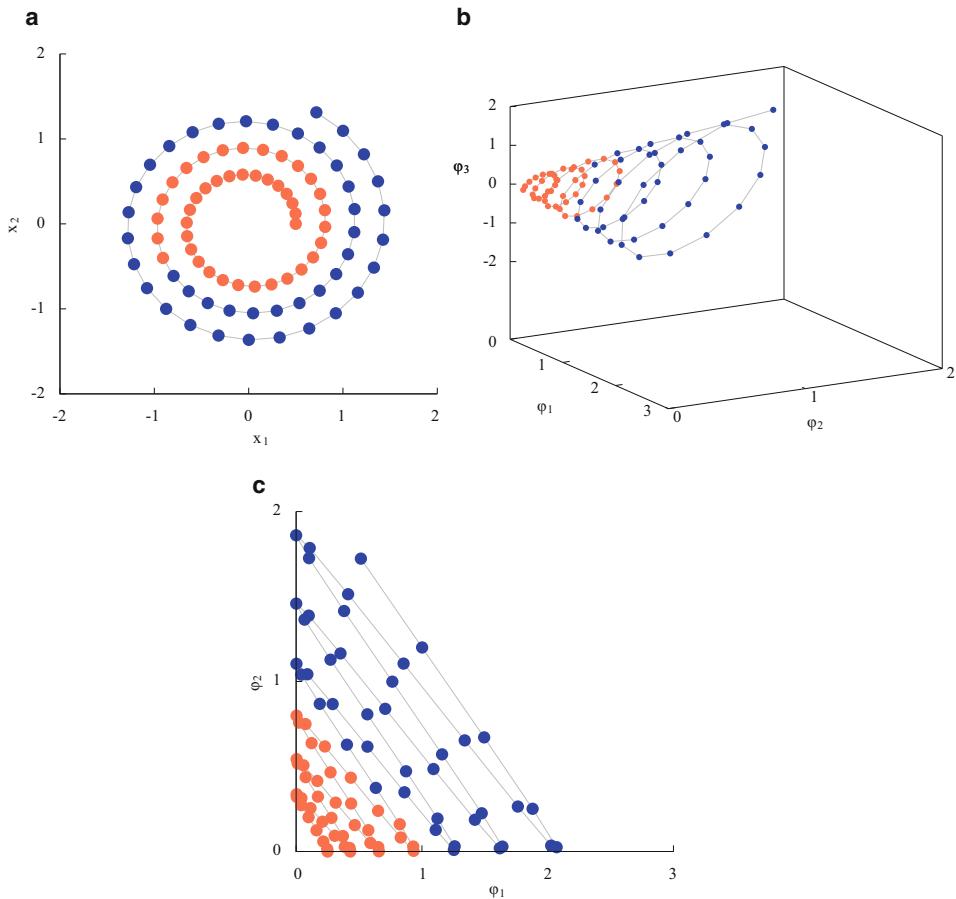


Abb. 18.10 Veranschaulichung des Prinzips von Kernfunktionen für nicht-lineare Entscheidungsfunktionen. **a** Eingaberaum \mathcal{X} , **b** zugehöriger Merkmalsraum \mathcal{H} der polynomielles Kernfunktion und **c** die erste und dritte Komponente des Merkmalsraumes. Die Verbindung der Punkte durch eine Linie dient zur besseren Visualisierung der Struktur der Daten

Einerseits ist diese Entscheidungsfunktion nicht-linear, andererseits ist auch eine Interpretation als lineare Funktion mit transformierten Eingabevektoren $\varphi(\mathbf{x})$ möglich:

$$\varphi(\mathbf{x}) = (x_1^2, x_1 \cdot x_2, x_2^2)^T, \quad (18.79)$$

$$h(\mathbf{x}) = \text{sign} (\langle \mathbf{w}, \varphi(\mathbf{x}) \rangle + b). \quad (18.80)$$

Dabei haben wir $\langle \cdot, \cdot \rangle$ als Notation für das Standardskalarprodukt verwendet. Die Transformation φ ist in Abb. 18.10b dargestellt und Abb. 18.10c zeigt eine zweidimensionale Projektion der transformierten Daten. Anhand der Projektion ist die lineare Trennbarkeit der Daten in diesem neuen Eingaberaum gut zu erkennen.

Wenn wir nun von D -dimensionalen Eingabevektoren $\mathbf{x} \in \mathbb{R}^D$ ausgehen, so würde die Transformation die Berechnung von $\mathcal{O}(D^2)$ Termen erforderlich machen. Dies ist einfach unpraktisch in vielen Anwendungsfällen mit hochdimensionalen Merkmalsvektoren und beinhaltet auch nur Terme mit jeweils zwei Variablen. Eine Möglichkeit, die explizite Berechnung zu umgehen, ist die Verwendung von Kernelfunktionen. Dadurch ist auch eine explizite Angabe von φ nicht notwendig.

Wir werden an dieser Stelle ein bisschen vorausgreifen und das verallgemeinerte *representer* Theorem [63] verwenden. Die Grundaussage dieses Theorems ist es, dass viele Lernmethoden, wie zum Beispiel SVM (Abschn. 18.4.6) und die lineare Regression und Ausgleichsrechnung (Abschn. 21.2.1), eine Hyperebene \mathbf{w} liefern, welche sich wie folgt darstellen lässt:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i \cdot \varphi(\mathbf{x}_i), \quad (18.81)$$

mit Koeffizienten $\alpha_i \in \mathbb{R}$ und den Lernbeispielen \mathbf{x}_i . Wie wir bereits in (18.80) gesehen haben, spielt die Auswertung des Skalarproduktes $\langle \mathbf{w}, \varphi(\mathbf{x}) \rangle$ eine entscheidende Rolle und kann wie folgt ausgewertet werden:

$$\langle \mathbf{w}, \varphi(\mathbf{x}) \rangle = \sum_{i=1}^n \alpha_i \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}) \rangle. \quad (18.82)$$

In diesem Fall haben wir die Bilinearität des Skalarproduktes ausgenutzt, um das Skalarprodukt in die Summe zu bringen. Die Auswertung der Entscheidungsfunktion basiert nur auf der Berechnung von Skalarprodukten im Raum \mathcal{H} , welcher durch die Transformation $\varphi : \mathcal{X} \rightarrow \mathcal{H}$ induziert wird. Diesen Raum werden wir im Folgenden als Merkmalsraum \mathcal{H} bezeichnen. Ist ein gegebener Vektorraum mit einem Skalarprodukt ausgestattet, so wird von einem Hilbertraum gesprochen. Wir verwenden im Folgenden die Notation $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ für ein Skalarprodukt in \mathcal{H} .

Wenn wir eine Abbildung (Kernelfunktion) $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ finden, welche das Skalarprodukt in \mathcal{H} direkt berechnet:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}: \quad K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}, \quad (18.83)$$

vermeiden wir die explizite und aufwendige Transformation der Lernbeispiele mit φ . Wie wir bereits in (18.82) gesehen haben, ist für die Auswertung der Entscheidungsfunktion oft nur die Berechnung der Skalarprodukte notwendig. Die Idee ist es nun bei einem geeigneten Lernverfahren, die Auswertung von Skalarprodukten durch die Kernelfunktion zu ersetzen. Diese Methodik wird als *Kernel-Trick* bezeichnet und hat viele Vorteile:

1. Eine konkrete Angabe der Transformation φ ist nicht notwendig.
2. Eine aufwendige explizite Transformation der Lernbeispiele ist nicht notwendig.
3. Es lässt sich auch mit komplexen strukturierten Daten lernen, da nur die Ähnlichkeit zwischen zwei gegebenen Beispielen berechnet werden muss.

Die letzte Eigenschaft werden wir bei der Bildklassifikation in Abschn. 20.2 ausnutzen.

Eine offene Fragestellung ist es, wie Kernelfunktionen K charakterisiert werden können, d. h. wie kann man bei einem gegebenen K die Eigenschaft in (18.83) nachweisen. Eine Möglichkeit ist es natürlich eine entsprechende Transformation φ direkt anzugeben. Dies ist aber oft schwierig mathematisch durchführbar. Einen Ausweg bietet die sogenannte Mercer-Bedingung, welche wir im nächsten Abschnitt näher betrachten wollen.

18.5.2 Mercer-Bedingung und Konstruktion von Kernelfunktionen

Die Mercer-Bedingung ist eine der wichtigsten Resultate für das Lernen mit Kernelfunktionen. Sie ermöglicht es für eine gegebene Abbildung nachzuweisen, dass es sich um eine Kernelfunktion im Sinne von (18.83) handelt, ohne direkt auf die Transformation φ Bezug zu nehmen. Oft wird eine Kernelfunktion verwendet, von der man überhaupt nicht die Transformation φ kennt. So ist es möglich die Modellierung allein anhand der Kernelfunktion vorzunehmen. Zunächst wollen wir aber eine sehr wichtige Klasse von Kernelfunktionen definieren:

Definition 18.1 (Positiv definite Kernelfunktion) *Eine Abbildung $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ heißt positiv definite Kernelfunktion, wenn für alle $n \in \mathbb{N}$ und alle endlichen Teilmengen $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ von \mathcal{X} die quadratische Matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ mit $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ positiv semi-definit ist. Eine solche Matrix \mathbf{K} werden wir als Kernmatrix bezeichnen.*

Weiterhin, verwenden wir die Notation $K(\mathbf{X}, \mathbf{X}')$ mit $\mathbf{X}' = (\mathbf{x}'_j)_{j=1}^m$ für die Matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, welche die paarweisen Werte der Kernelfunktion $A_{ij} = K(\mathbf{x}_i, \mathbf{x}'_j)$ enthält. Wenn \mathbf{X}' nur ein Element \mathbf{x} beinhaltet, schreiben wir auch $K(\mathbf{X}, \mathbf{x})$.

Ausgehend von dieser Definition lässt sich sofort folgender Satz beweisen:

Satz 18.2 (Lineare Kombination von Kernelfunktionen) *Seien K_1, \dots, K_R positiv definite Kernelfunktionen und $\beta_1, \dots, \beta_R \in \mathbb{R}$ nicht-negative Koeffizienten, dann ist die lineare Kombination $K = \beta_1 K_1 + \dots + \beta_R K_R$ ebenfalls eine positiv definite Kernelfunktion.*

Der Satz zeigt, dass wir eine positiv definite Kernelfunktion K mit einem Faktor $v_0 > 0$ skalieren können, ohne die Eigenschaft der positiven Definitheit von K zu verletzen:

$$\tilde{K}(\cdot, \cdot) = v_0 \cdot K(\cdot, \cdot) + v_1. \quad (18.84)$$

Wenn wir eine lineare Kernelfunktion $K(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ einsetzen, so können wir leicht sehen, dass jede Kernmatrix, welche durch K induziert wird, positiv definit ist. Diese Beobachtung lässt einen starken Zusammenhang zwischen der Bedingung in (18.83) und der Definition 18.1 vermuten. Dieser Zusammenhang manifestiert sich in der Mercer-Bedingung.

Satz 18.3 (Mercer-Bedingung [46, 79]) Eine Abbildung $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ ist eine positiv definite Kernelfunktion, genau dann wenn ein Hilbertraum \mathcal{H} und eine Transformation $\varphi : \mathcal{X} \rightarrow \mathcal{H}$ existieren, so dass folgende Eigenschaft gilt:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} : K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}. \quad (18.85)$$

Beweis (a) Richtung 1: „ \rightarrow “:

Wir werden im Folgenden annehmen, dass der Eingaberaum \mathcal{X} endlich ist und insgesamt $|\mathcal{X}| = n$ Elemente besitzt. Diese Annahme ermöglicht es, den Beweis auf der Grundlage von einfachen Sachverhalten der linearen Algebra zu führen. Für einen exakten Beweis für den allgemeinen Fall sei auf [63] verwiesen. Für jede positiv definite Kernelfunktion ist die Kernmatrix \mathbf{K} aller Elemente des Eingaberaums positiv semi-definit. Wir können daher \mathbf{K} mittels der Cholesky-Faktorisierung wie folgt zerlegen: $\mathbf{K} = \mathbf{G}\mathbf{G}^T$. Wir definieren weiterhin die Transformation $\varphi(\mathbf{x}_i)$ als Spaltenvektor, welcher die Elemente der i -ten Zeile von \mathbf{G} enthält:

$$(\varphi(\mathbf{x}_i))_k = G_{ik}. \quad (18.86)$$

Diese Definition führt bereits zu einer passenden Transformation für K :

$$K(\mathbf{x}_i, \mathbf{x}_j) = K_{ij} = \sum_{k=1}^n G_{ik} G_{jk} = \sum_{k=1}^n (\varphi(\mathbf{x}_i))_k (\varphi(\mathbf{x}_j))_k = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle$$

Die Kernelfunktion K und die Transformation φ erfüllen also die Eigenschaft in (18.85).

(b) Richtung 2: „ \leftarrow “:

Sei \mathbf{K} eine beliebige Kernmatrix, welche durch die Kernelfunktion K berechnet wurde. Die Kernelfunktion erfüllt die Bedingung in (18.85). Für jeden Vektor $\mathbf{a} \in \mathbb{R}^n$ gilt nun Folgendes:

$$\begin{aligned} \mathbf{a}^T \mathbf{K} \mathbf{a} &= \sum_{i,j=1}^n \alpha_i \alpha_j \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i), \sum_{j=1}^n \alpha_j \varphi(\mathbf{x}_j) \right\rangle_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i) \right\|_{\mathcal{H}}^2 \geq 0. \end{aligned} \quad (18.87)$$

Dabei ist $\|\cdot\|_{\mathcal{H}}$ die induzierte Norm in \mathcal{H} . Die Kernmatrix ist demnach positiv semi-definit und dadurch, dass wir keine weiteren Anforderungen an die Matrix gestellt haben, folgt unmittelbar, dass die Kernelfunktion K positiv definit ist. \square

In den folgenden Abschnitten werden wir den Begriff *Kernelfunktion* anstatt positiv definite Kernelfunktion verwenden. Wir gehen also immer davon aus, dass die Mercer-Bedingung gültig ist. Eine Kernelfunktion kann als eine Art Ähnlichkeitsmaß zwischen zwei Eingaben \mathbf{x} und \mathbf{x}' interpretiert werden. Bei der Anwendung des Kernel Tricks ist also

nur die Ähnlichkeit zwischen den Lernbeispielen von Bedeutung und nicht deren explizite Repräsentation. Wir werden nun ein paar Klassen von Kernelfunktionen kennenlernen, welche im Bereich des maschinellen Lernens eine besondere Bedeutung haben.

Definition 18.4 (Eigenschaften von Kernelfunktionen) Eine Kernelfunktion $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ ist

1. stationär oder verschiebungsinvariant wenn ihre Werte nur von der Differenz der beiden Eingabevektoren abhängt:

$$K(\mathbf{x}, \mathbf{x}') = K^{\text{stat}}(\mathbf{x} - \mathbf{x}'), \quad (18.88)$$

2. homogen oder ist eine verallgemeinerte RBF-Kernelfunktion, wenn ihre Werte nur von der Entfernung der beiden Eingabevektoren abhängt:

$$K(\mathbf{x}, \mathbf{x}') = K^{\text{hom}}(d(\mathbf{x}, \mathbf{x}')). \quad (18.89)$$

Die Entfernung wird dabei mit einer allgemeinen Metrik $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ berechnet. In der englischen Fachliteratur wird an dieser Stelle von „generalized radial basis function kernels“ gesprochen, daher leitet sich die Abkürzung RBF ab.

3. eine RBF-Kernelfunktion, wenn sie homogen ist und ihre Werte nur von der Euklidischen Distanz der beiden Eingabevektoren abhängen.²

Die Kernelfunktion die am häufigsten eingesetzt wird, ist der sogenannte Gaußkern, welcher zur Familie der RBF-Kernelfunktionen gehört:

$$K^{\text{gauss}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right). \quad (18.90)$$

Die Kernelfunktion ist abhängig vom Parameter $\gamma > 0$, welcher oft einen entscheidenden Einfluss auf einen Klassifikator haben kann. Dies kann man anhand des Verhaltens an den Grenzen gut nachvollziehen: Für $\gamma = 0$ ist der Wert der Kernelfunktion immer 1 unabhängig von den Eingabedaten. Betrachtet man hingegen das Verhalten für $\gamma \rightarrow \infty$, so sieht man, dass die Kernelfunktion zur Diracfunktion konvergiert (Abschn. 4.18). Parameter der Kernelfunktionen werden allgemein als *Hyperparameter* bezeichnet und deren Bestimmung ist ein wichtiger Bestandteil des Lernschrittes.

Zwei weitere Kernelfunktionen, welche vor allem bei der Bildklassifikation in Abschn. 20.2 mit Histogrammen eine wichtige Rolle spielen werden, sind der *Chi-Quadrat-Kern*:

$$K^{\chi^2}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \sum_{i=1}^D \frac{(x_i - x'_i)^2}{x_i + x'_i}\right) \quad (18.91)$$

² Das Buch von Bishop [3] verwendet den Begriff der RBF-Kernelfunktion als Synonym für homogene Kernelfunktionen.

und der *Minimumkern* oder *Histogrammschnittkern*:

$$K^{\min}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^D \min(x_i, x'_i). \quad (18.92)$$

Letztere Kernelfunktion ist auch direkt mit dem Histogramm-Schnitt als Abstandsmaß verknüpft (Abschn. 19.8.2).

18.5.3 Kernaldichteschätzung und Parzen-Klassifikator

Im Folgenden sei ein sehr einfacher Klassifikator betrachtet, welcher zunächst linear formuliert wird, um dann den Kerneltrick anzuwenden. Die Klassifikationsentscheidung für ein neues Beispiel \mathbf{x}_* wird aufgrund von folgenden Ähnlichkeiten $f^{(\kappa)}(\mathbf{x}_*)$ zu den Klassen κ getroffen:

$$f^{(\kappa)}(\mathbf{x}_*) = \mathbf{x}_*^T \left(\frac{1}{n_\kappa} \sum_{y_i=\kappa} \mathbf{x}_i \right). \quad (18.93)$$

Die Ähnlichkeit zu einer gegebenen Klasse wird also mittels der Korrelation zwischen dem neuen Beispiel und dem Schwerpunkt der Klasse bestimmt. Wir wenden nun eine Merkmalstransformation φ an, um die Merkmale in einen hochdimensionalen Raum zu transformieren und gleichzeitig den Kerneltrick anzuwenden:

$$\begin{aligned} f^{(\kappa)}(\mathbf{x}_*) &= \varphi(\mathbf{x}_*)^T \left(\frac{1}{n_\kappa} \sum_{y_i=\kappa} \varphi(\mathbf{x}_i) \right) \\ &= \frac{1}{n_\kappa} \sum_{y_i=\kappa} \varphi(\mathbf{x}_*)^T \varphi(\mathbf{x}_i) = \frac{1}{n_\kappa} \sum_{y_i=\kappa} K(\mathbf{x}_*, \mathbf{x}_i). \end{aligned} \quad (18.94)$$

Die letzte Schätzgleichung (18.94) wird als Kernaldichteschätzung bezeichnet und lässt sich mit beliebigen Kernelfunktionen durchführen. Üblicherweise wird der Gaußkern (18.90) verwendet und die Kernaldichteschätzung ist in diesem Fall gegeben durch:

$$f^{(\kappa)}(\mathbf{x}_*) = \frac{1}{n_\kappa} \sum_{y_i=\kappa} \exp\left(-\gamma \|\mathbf{x}_* - \mathbf{x}_i\|^2\right). \quad (18.95)$$

Der Parameter γ hat eine entscheidende Bedeutung für den Einflussbereich jedes Lernbeispiels. Bei $\gamma \rightarrow \infty$ erhalten wir eine Summe von Diracfunktionen, welche auch oft als empirische Dichte bezeichnet wird, und bei $\gamma \rightarrow 0$ ergibt sich eine konstante Bewertungsfunktion $f^{(\kappa)}(\mathbf{x}_*) = 1$. Der Hyperparameter γ des Gaußkerns steuert daher die „Glättigkeit“ der Bewertungsfunktion. Ein Beispiel für zweidimensionale Eingabevektoren ist in Abb. 18.11 dargestellt. In der Abbildung ist sehr gut der Einfluss der Hyperparameter der

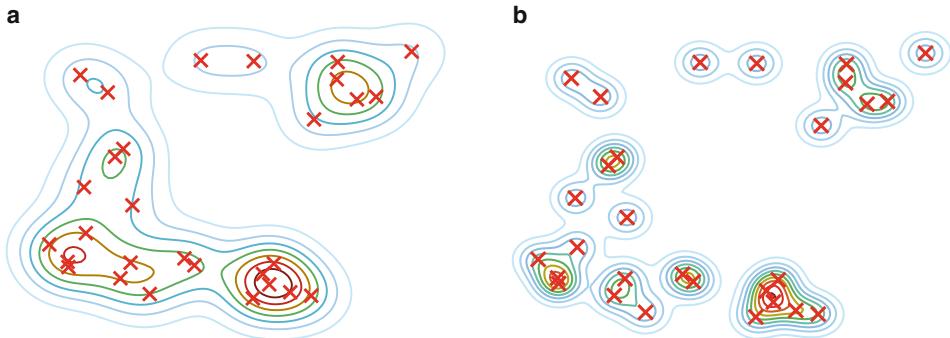


Abb. 18.11 Darstellung der Höhenlinien einer Kerneldichteschätzung für ein Beispiel mit zweidimensionalen Eingabevektoren. Die beiden Grafiken zeigen die Schätzungen für eine unterschiedliche Wahl der Hyperparameter der Kernelfunktion (**a** niedriger Wert von γ , **b** hoher Wert von γ ; oft als Bandbreite in diesem Fall bekannt)

Kernelfunktion zu erkennen. Ein Hyperparameter der Kernelfunktion wird in diesem Fall oft als Bandbreite bezeichnet, wenn der den Einfluss der Nachbarn reguliert. Eine Klassifikation kann mittels einer Bestimmung der Klasse mit dem höchsten Wert der Kerneldichteschätzung erfolgen.

Wenn als Kernelfunktion der Gaußkern verwendet wird, so kann mit einer geeigneten Normierung der Wert $f^{(\kappa)}(\mathbf{x}_*)$ als bedingte Verteilung der Eingabedaten interpretiert werden:

$$p(\mathbf{x}_* \mid y_* = \kappa, \mathcal{D}) = \frac{1}{C \cdot n_{\kappa}} \sum_{y_i=\kappa} K(\mathbf{x}_*, \mathbf{x}_i). \quad (18.96)$$

Diese Modellierung kann direkt für einen generativen Klassifikator eingesetzt werden. Die Normierungskonstante C führt zu einer korrekt normierten Verteilung und ist folgendermaßen bestimmt:

$$C = \int_{\mathbf{x} \in \mathcal{X}} \exp(-\gamma \|\mathbf{x}\|^2) = \sqrt{\frac{\pi^D}{\gamma}}. \quad (18.97)$$

Gleichung (18.94) kann auch als Mischverteilung interpretiert werden und ermöglicht im Gegensatz zum Normalverteilungsklassifikator (Abschn. 18.4.2) eine Modellierung von Verteilungen mit mehreren Maxima. Eine Normalverteilung hat immer eine sehr einfache Form mit genau einem klar definierten Maximum (Mode), daher ist es nicht möglich, Klassen zu modellieren, welche mehrere unterschiedliche Ausprägungen besitzen. Dies ist zum Beispiel der Fall wenn die Klasse *Auto* mit Front- und Seitenaufnahmen angelernt wird. Wir haben also gesehen, dass der Kerneltrick, angewendet auf einen sehr einfachen Klassifikator, zu einem flexiblen Ansatz führt.

18.5.4 Support-Vektor-Klassifikator mit Kernfunktionen

In Abschn. 18.4.6 hatten wir gesehen, dass Support-Vektor-Klassifikatoren (engl.: *support vector machines*, SVM) eine lineare Trennebene bestimmen, welche den minimalen Abstand zu den Lernbeispielen maximiert. Bei nicht-linear-separierbaren Lerndaten hatten wir auch die Verwendung von Schlupfvariablen kennengelernt, um Ausreißer in den Lerndaten zu behandeln. Wir werden uns jetzt der Anwendung des Kerneltricks bei Support-Vektor-Klassifikatoren zuwenden, welche es ermöglichen, direkt eine nicht-lineare Trennebene zu schätzen.

Wir betrachten im Folgenden die Anwendung der Transformation $\varphi : \mathcal{X} \rightarrow \mathcal{H}$ und die Schätzung linearer Trennebenen im resultierenden hochdimensionalen Raum \mathcal{H} :

$$f(\mathbf{x}) = \langle \mathbf{w}, \varphi(\mathbf{x}) \rangle_{\mathcal{H}} + b. \quad (18.98)$$

Im Abschn. 18.5.1 wurde schon das *representer Theorem* von [63] angesprochen. Dieses Theorem zeigt, dass es möglich ist, für viele unterschiedliche Lernverfahren die Hyperebene \mathbf{w} als Linearkombination der transformierten Lernbeispiele auszudrücken (18.81). Wir wollen jedoch an dieser Stelle diesen wichtigen Sachverhalt direkt für den SVM-Ansatz herleiten. Dafür bestimmen wir zunächst das duale Gegenstück des Optimierungsproblems (18.69). Für eine gute Einführung in die mathematische Optimierung sei an dieser Stelle auf [7] verwiesen.

Das duale Problem lässt sich durch das Aufstellen der Lagrange-Funktion herleiten:

$$L(\mathbf{w}, b, \boldsymbol{\lambda}) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^n \lambda_i (1 - y_i \cdot (\langle \mathbf{w}, \varphi(\mathbf{x}_i) \rangle_{\mathcal{H}} + b)). \quad (18.99)$$

Um nun die duale Lagrange-Funktion zu bestimmen, berechnen wir erst die Gradienten um L bezüglich \mathbf{w} und b zu minimieren [7, Kapitel 5]:

$$\{\nabla_{\mathbf{w}} L\}(\mathbf{w}, b, \boldsymbol{\lambda}) = \mathbf{w} - \sum_{i=1}^n \lambda_i y_i (\varphi(\mathbf{x}_i)), \quad (18.100)$$

$$\{\nabla_b L\}(\mathbf{w}, b, \boldsymbol{\lambda}) = - \sum_{i=1}^n \lambda_i y_i = -\boldsymbol{\lambda}^T \mathbf{y}. \quad (18.101)$$

Werden nun beide Gradienten auf Null gesetzt, so erhält man:

$$\tilde{\mathbf{w}} = \sum_{i=1}^n \lambda_i y_i \varphi(\mathbf{x}_i) = \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i), \quad (18.102)$$

mit den Koeffizienten $\alpha_i = \lambda_i y_i$ und der Eigenschaft $\sum_{i=1}^n \alpha_i = 0$. Das ermittelte Ergebnis in (18.102) ist eine Instanz des *representer Theorems* für SVM-basiertes Lernen. Die Koeffizienten α_i spielen eine ganz zentrale Rolle, da sie genau dann Null sind, wenn das

korrespondierende Lernbeispiel kein Supportvektor ist und demnach die Nebenbedingung in (18.69) nicht aktiv ist. Die Norm der Hyperebene $\tilde{\mathbf{w}}$ lässt sich nun mit Auswertungen der Kernelfunktion bestimmen:

$$\|\tilde{\mathbf{w}}\|^2 = \sum_{i,j=1}^n \alpha_i \alpha_j \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}} = \mathbf{a}^T \mathbf{K} \mathbf{a}. \quad (18.103)$$

Diesen Sachverhalt werden wir später noch benötigen. Zunächst kombinieren wir erst einmal alle Resultate und erhalten die modifizierte duale Lagrange-Funktion $g(\mathbf{a}) = L(\tilde{\mathbf{w}}, \tilde{b}, [\alpha_i / y_i]_{i=1}^n)$ in Abhängigkeit von den Koeffizienten α_i :

$$\begin{aligned} g(\mathbf{a}) &= \frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{a} + \sum_{i=1}^n \left(\frac{\alpha_i}{y_i} \right) \left(1 - y_i \cdot \left(\sum_{j=1}^n \alpha_j \langle \varphi(\mathbf{x}_j), \varphi(\mathbf{x}_i) \rangle_{\mathcal{H}} + \tilde{b} \right) \right) \\ &= \frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{a} + \left(\sum_{i=1}^n \frac{\alpha_i}{y_i} \right) - \left(\sum_{i,j=1}^n \alpha_i \alpha_j \langle \varphi(\mathbf{x}_j), \varphi(\mathbf{x}_i) \rangle_{\mathcal{H}} \right) - \tilde{b} \left(\sum_{i=1}^n \alpha_i \right) \end{aligned} \quad (18.104)$$

$$= -\frac{1}{2} \underbrace{\mathbf{a}^T \mathbf{K} \mathbf{a}}_{=\alpha_i y_i} + \underbrace{\sum_{i=1}^n \left(\frac{\alpha_i}{y_i} \right)}_{=0} - \tilde{b} \left(\sum_{i=1}^n \alpha_i \right) \quad (18.105)$$

$$= -\frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{a} + \mathbf{a}^T \mathbf{y}. \quad (18.106)$$

Dabei haben wir die Kernelmatrix \mathbf{K} der Lerndaten verwendet, welche die paarweisen Skalarprodukte der transformierten Lernbeispiele beinhaltet, d. h. $K_{ij} = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}}$. Das (modifizierte) duale Problem maximiert $g(\mathbf{a})$ unter der Nebenbedingung, dass die Lagrange-Faktoren λ_i nicht negativ sind und dass sich die Koeffizienten α_i zu Null summieren:

$$\begin{aligned} \max_{\mathbf{a} \in \mathbb{R}^n} \quad g(\mathbf{a}) &= -\frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{a} + \mathbf{a}^T \mathbf{y} \\ \text{Nb.} \quad \forall i = 1 \dots n : \alpha_i \cdot y_i &\geq 0 \text{ und } \sum_{i=1}^n \alpha_i = 0 . \end{aligned} \quad (18.107)$$

Die zweite Nebenbedingung folgte aus dem Nullsetzen des Gradienten für b (18.101). Eine Herleitung der dualen Kernelvariante des Optimierungsproblems (18.73) mit den Schlupfvariablen ξ_i erfolgt vollkommen analog und führt nur zu zusätzlichen oberen Schranken von α_i : $0 \leq \alpha_i \cdot y_i \leq C$ [3, S. 333].

Einfluss des Parameters C Im Folgenden wollen wir kurz den Einfluss des Regularisierungspараметers C auf die Entscheidungsfunktion untersuchen. Wie wir bereits in Abschn. 18.4.6 gesehen haben, ermöglicht es dieser Parameter, den Einfluss von Ausreißern zu kontrollieren. Für die Analyse verwenden wir das folgende Theorem:

Satz 18.5 (Obere Schranke der Standardabweichung) Sei $\varphi : \mathcal{X} \rightarrow \mathcal{H}$ eine Abbildung in den hochdimensionalen Raum \mathcal{H} mit der Kernelfunktion K . Weiterhin, sei eine Menge $\mathbf{X} \subseteq \mathcal{X}$ von Beispielen gegeben. Die Funktion $f : \mathcal{X} \rightarrow \mathbb{R}$ sei nun folgendermaßen definiert:

$$f(\mathbf{x}) = \langle \mathbf{w}, \varphi(\mathbf{x}) \rangle_{\mathcal{H}} + b \quad (18.108)$$

$$\mathbf{w} = \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i), \quad (18.109)$$

Diese Darstellung ermöglicht eine Beschränkung der Standardabweichung dieser Funktion:

$$\sigma_{\mathbf{x}}(f) \leq \|\boldsymbol{\alpha}\| \cdot \sqrt{\lambda_{\max}(\mathbf{K})} \cdot \zeta_K, \quad (18.110)$$

dabei ist ζ_K ein Term, welcher nur von der Kernelfunktion und der Verteilung der Eingabedaten $\mathbf{x} \in \mathcal{X}$ abhängt.

Beweis Um die obere Schranke für die Standardabweichung zu beweisen, versuchen wir zunächst diese anhand der Norm der Hyperebene \mathbf{w} im Raum \mathcal{H} auszudrücken:

$$\sigma_{\mathbf{x}}^2(f(\mathbf{x})) = \int_{\mathcal{X}} (f(\mathbf{x}') - E_{\mathbf{x}}(f(\mathbf{x})))^2 p(\mathbf{x}') d\mathbf{x}' \quad (18.111)$$

$$= \int_{\mathcal{X}} (\langle \mathbf{w}, \varphi(\mathbf{x}') - E_{\mathbf{x}}(\varphi(\mathbf{x})) \rangle_{\mathcal{H}})^2 p(\mathbf{x}') d\mathbf{x}' \quad (18.112)$$

$$\leq \int_{\mathcal{X}} \|\mathbf{w}\|_{\mathcal{H}}^2 \cdot \|\varphi(\mathbf{x}') - E_{\mathbf{x}}(\varphi(\mathbf{x}))\|^2 p(\mathbf{x}') d\mathbf{x}' \quad (18.113)$$

$$\leq \|\mathbf{w}\|_{\mathcal{H}}^2 \cdot \left(\int_{\mathcal{X}} K(\mathbf{x}', \mathbf{x}') p(\mathbf{x}') d\mathbf{x}' - \int_{\mathcal{X} \times \mathcal{X}} K(\mathbf{x}', \mathbf{x}) p(\mathbf{x}) p(\mathbf{x}') d\mathbf{x}' d\mathbf{x} \right). \quad (18.114)$$

Zur Herleitung haben wir die Ungleichung von Cauchy verwendet um das Skalarprodukt mit Hilfe der Norm abzuschätzen. Der letzte Faktor der oberen Schranke hängt nur von der Verteilung $p(\mathbf{x})$ und der Kernelfunktion ab. Wir werden diesen Term daher mit ζ_K^2 bezeichnen. Wie wir bereits bei dem Resultat des *representer theorems* gesehen haben, lässt sich die Hyperebene \mathbf{w} mittels der Lerndaten \mathbf{X} ausdrücken. Die Norm von \mathbf{w} lässt sich dann auch durch den größten Eigenwert der Kernmatrix \mathbf{K} und der Norm der Koeffizienten $\boldsymbol{\alpha}$ nach oben beschränken:

$$\|\mathbf{w}\|_{\mathcal{H}}^2 = \left\| \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i) \right\|_{\mathcal{H}}^2 = \sum_{i,j=1}^n \alpha_i \alpha_j \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}} \quad (18.115)$$

$$= \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \leq \|\boldsymbol{\alpha}\|^2 \cdot \lambda_{\max}(\mathbf{K}). \quad (18.116)$$

□

Das Theorem gibt eine obere Schranke für die Standardabweichung der Entscheidungsfunktion f in Abhängigkeit von den Koeffizienten α an. Wenn wir jetzt zusätzlich die Bedingung $0 \leq \alpha_i \cdot y_i \leq C$ einfließen lassen, so können wir diese obere Schranke in der Variable C ausdrücken:

$$\sigma_x(f(\mathbf{x})) \leq \|\alpha\| \cdot \sqrt{\lambda_{\max}(\mathbf{K})} \cdot \zeta_K \leq \sqrt{n} \cdot C \cdot \sqrt{\lambda_{\max}(\mathbf{K})} \cdot \zeta_K, \quad (18.117)$$

dabei ist ζ_K ein Term, welcher nur von der verwendeten Kernelfunktion abhängt und $\lambda_{\max}(\mathbf{K})$ steht für den größten Eigenwert der Kernelmatrix. Die obere Schranke verdeutlicht den Einfluss von C auf die Entscheidungsfunktion. Kleine Werte des Parameters bewirken eine geringere obere Schranke für die Standardabweichung von f und daher auch ein Klassifikationsmodell mit einer geringeren Modellkomplexität, wie etwa Hyperebenen im Eingaberaum. Ein großer Wert von C bewirkt hingegen eine große Variabilität der Entscheidungsfunktion.

Komplexität der Berechnung Vergleicht man das primale Problem (18.69) mit der dualen Version (18.107), so ist vor allem die Änderung der Größe des zu optimierenden Vektors auffällig. Bei der primalen Version erfolgte die Optimierung bezüglich \mathbf{w} und b , also von insgesamt $D + 1$ veränderlichen Größen. Demgegenüber müssen bei der dualen Version n Parameterwerte optimiert werden. Dabei ist zu beachten, dass die effektive Größe des dualen Problems die Anzahl der Supportvektoren ist [63]. Daher wird der SVM-Ansatz mit Kernelfunktionen auch oft als semi-parametrisch bezeichnet. Das Lösen des dualen Problems ist dem primalen Problem überlegen, wenn eine kleine Anzahl von Lernbeispielen aber eine große Anzahl von Merkmalen zur Verfügung steht. Bei aktuellen Erkennungssystemen mit Lerndatensätzen von mehreren Tausend Beispielen ist die direkte Anwendung des Kernelprinzips natürlich unpraktisch. Aus diesem Grund wird aktuell oft der Ansatz verfolgt, dass man ausgehend von einer gegebenen Kernelfunktion eine approximierte Merkmalstransformation berechnet [80]. Diese kann dann auf die Lernbeispiele angewendet werden, um ausgehend davon einen linearen SVM-Klassifikator zu schätzen.

Klassifikation von neuen Beispielen Nach der Bestimmung der Koeffizienten α_i durch das Lösen des dualen Problems (18.107) können neue Beispiele \mathbf{x}_* klassifiziert werden, indem zunächst die Entscheidungsfunktion $f(\mathbf{x}_*)$ ausgewertet wird:

$$f(\mathbf{x}_*) = \langle \mathbf{w}, \varphi(\mathbf{x}_*) \rangle_{\mathcal{H}} + b = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}_*) + b, \quad (18.118)$$

um dann anschließend eine diskrete Klassifikationsentscheidung durch Überprüfung des Vorzeichens zu erhalten:

$$h(\mathbf{x}_*) = \text{sign}(f(\mathbf{x}_*)) = \text{sign}\left(\sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}_*) + b\right). \quad (18.119)$$

Der Wert der Verschiebung b der Hyperebene kann ebenfalls in Abhängigkeit der Kernelfunktion angegeben werden. Eine entsprechende Herleitung ist in [63] und [3] zu finden. In manchen Anwendungen ist eine weiche Entscheidungsfunktion vorteilhaft. Bei SVM-Ansätzen kann dafür direkt der Wert von $f(\mathbf{x}_*)$ verwendet werden.

Optimierung von Hyperparametern In Abschn. 18.5.2 hatten wir bereits schon das Problem angesprochen, optimale Kernel-Hyperparameter zu finden. Bei den SVM-basierten Ansätzen aus den vorherigen Abschnitten haben wir zusätzlich noch den freien Parameter $C > 0$, welcher den Bestrafungsterm $\sum_i \xi_i$ der Schlupfvariablen ξ gewichtet.

Eine übliche Methode den Parameter C und die Hyperparameter η einer Kernelfunktion, wie etwa der Parameter γ des Gaußkerns (18.90), zu optimieren, ist die k -fache Kreuzvalidierung. Um diese Methode anzuwenden, wird der Lerndatensatz \mathcal{D} in k Untermengen \mathcal{D}^j eingeteilt und Lernen und Testen des Klassifikators genau k -mal durchgeführt. In jeder Iteration j wird der Datensatz \mathcal{D}^j zum Testen des Klassifikators verwendet, welcher mit verbliebenen Beispielen aus $\mathcal{D} \setminus \mathcal{D}^j$ angelernt wurde. Anschließend stehen k Werte eines Gütemaßes (Abschn. 18.6) zur Verfügung, welche gemittelt werden können, um ein Maß für die Güte der Klassifikation für einen gegebenen Satz von Hyperparametern zu erhalten. Zur Bestimmung der optimalen Hyperparameter wird meist eine vollständige Suche mit vorher festgelegten einzelnen Parameterwerten durchgeführt. Dies ist natürlich äußerst aufwendig, und bei mehr als zwei Parametern oft praktisch nicht anwendbar. Alternativ können Ansätze aus dem *multiple kernel learning* [69] und der Gauß-Prozess-Klassifikation (Abschn. 18.5.11) verwendet werden.

18.5.5 Modellierung mit Gauß-Prozessen

Im folgenden Abschnitt werden wir uns mit der Anwendung von Gauß-Prozessen (GP) beim maschinellen Lernen beschäftigen. Das Buch von Rasmussen und Williams [57] ist hierbei die Standardliteratur für den Bereich Modellierung mit GP. Grundsätzlich wird oft bei den Herleitungen zwischen zwei unterschiedlichen Herangehensweisen unterschieden: der Zugang über die Modellierung der Wahrscheinlichkeitsverteilung der Hyperebene (*weight space view*) und die Modellierung der latenten Funktion als Zufallsvariable (*process view* oder *function space view*). In der folgenden Darstellung wählen wir die letztere Variante und stellen den Zusammenhang zu der Schätzung von Hyperbenen später her.

Gauß-Prozesse gehören zur Klasse der stochastischen Prozesse, welche als eine Familie von Zufallsvariablen definiert sind. Einen Gauß-Prozess kann man als Verallgemeinerung einer multivariaten Normalverteilung auf unendlich viele Eingabedimensionen ansehen. Diese Vorstellung spiegelt sich auch in der folgenden formalen Definition wieder, welche ähnlich wie die Definition von positiv definiten Kernelfunktionen auf endliche Untermen gen zurückgreift:

Definition 18.6 (Gauß-Prozess) Eine Familie von Zufallsvariablen $f = (f(\mathbf{x}))_{\mathbf{x} \in \mathcal{X}}$ ist ein Gauß-Prozess mit Mittelwertfunktion $\mu : \mathcal{X} \rightarrow \mathbb{R}$ und Kovarianzfunktion $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, genau dann wenn für jede endliche Untermenge $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{X}$ die korrespondierenden Zufallsvariablen gemeinsam normal verteilt sind, d. h. es gilt

$$\mathbf{f} = (f(\mathbf{x}_i))_{i=1}^n \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{S})$$

mit $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n))^T$ und Kovarianzen $S_{ij} = E(f(\mathbf{x}_i)f(\mathbf{x}_j)) = K(\mathbf{x}_i, \mathbf{x}_j)$. Wir verwenden in diesem Fall die Notation $f \sim \mathcal{GP}(\mu, K)$.

Im Folgenden beschränken wir uns auf mittelwertfreie Gauß-Prozesse, d. h. $\mu \equiv 0$. Oft können die Ausgabedaten ohne Weiteres vorher normiert werden, um mittelwertfreie Daten zu erhalten.

Die Art der Definition von Gauß-Prozessen und positiv definiten Kernelfunktionen suggeriert sofort einen starken Zusammenhang. So muss eine valide Kovarianzfunktion positiv definite Kovarianzmatrizen induzieren und sollte daher selbst positiv definit sein. Aus diesem Grund ist jede positiv definite Kernelfunktion nach Definition 18.1 eine valide Kovarianzfunktion und jede valide Kovarianzfunktion auch eine positiv definitive Kernelfunktion. Im restlichen Teil des Buches verwenden wir daher den Begriff Kernelfunktion und Kovarianzfunktion als Synonyme.

Eine wichtige Eigenschaft von Gauß-Prozessen im Gegensatz zu vielen anderen bekannten stochastischen Prozessen, wie etwa Poisson- oder Gamma-Prozesse, ist die Definition anhand von mehrdimensionalen Indexmengen \mathcal{X} . Oft werden stochastischen Prozesse nur auf eindimensionalen Zeitachsen definiert. Dadurch eignen sich diese nicht für die Modellierung der Klassifikation mehrdimensionaler Daten. Ein Gauß-Prozess kann hingegen so als Wahrscheinlichkeitsverteilung von Funktionen angesehen werden. Die Kernelfunktion K eines Gauß-Prozesses bestimmt die Kovarianz zweier Funktionswerte $f(\mathbf{x})$ und $f(\mathbf{x}')$ und hat somit einen entscheidenden Einfluss auf die Modellierung.

Einfluss von Hyperparametern Abbildung 18.12 zeigt ein paar Beispiele von Funktionen, welche von einem mittelwertfreien Gauß-Prozess zufällig gezogen wurden. Dabei wurden mehrere Kernelfunktionen und Hyperparameter verwendet, um die Flexibilität der entstehenden Funktionen aufzuzeigen. Fokussieren wir uns zunächst auf die erste Zeile in der Abbildung, welche mit einem Gaußkern (18.90) erzeugt wurde. Durch die Stationarität der Kernelfunktion ist das Verhalten der gezogenen Funktionen unabhängig von der absoluten Position. Weiterhin können wir beobachten, dass die Variabilität der Funktion mit zunehmenden Werten des Hyperparameters γ steigt. Dieses Verhalten wollen wir jetzt genauer analytisch untersuchen, indem wir eine Approximation des zu erwartenden quadratischen Gradienten von f betrachten. Seien zunächst Punkte $x_1 < \dots < x_n \in \mathbb{R}$ und ihr korrespondierender normalverteilter Zufallsvektor $\mathbf{f} = (f(x_1), \dots, f(x_n))^T \in \mathbb{R}^n$ gege-

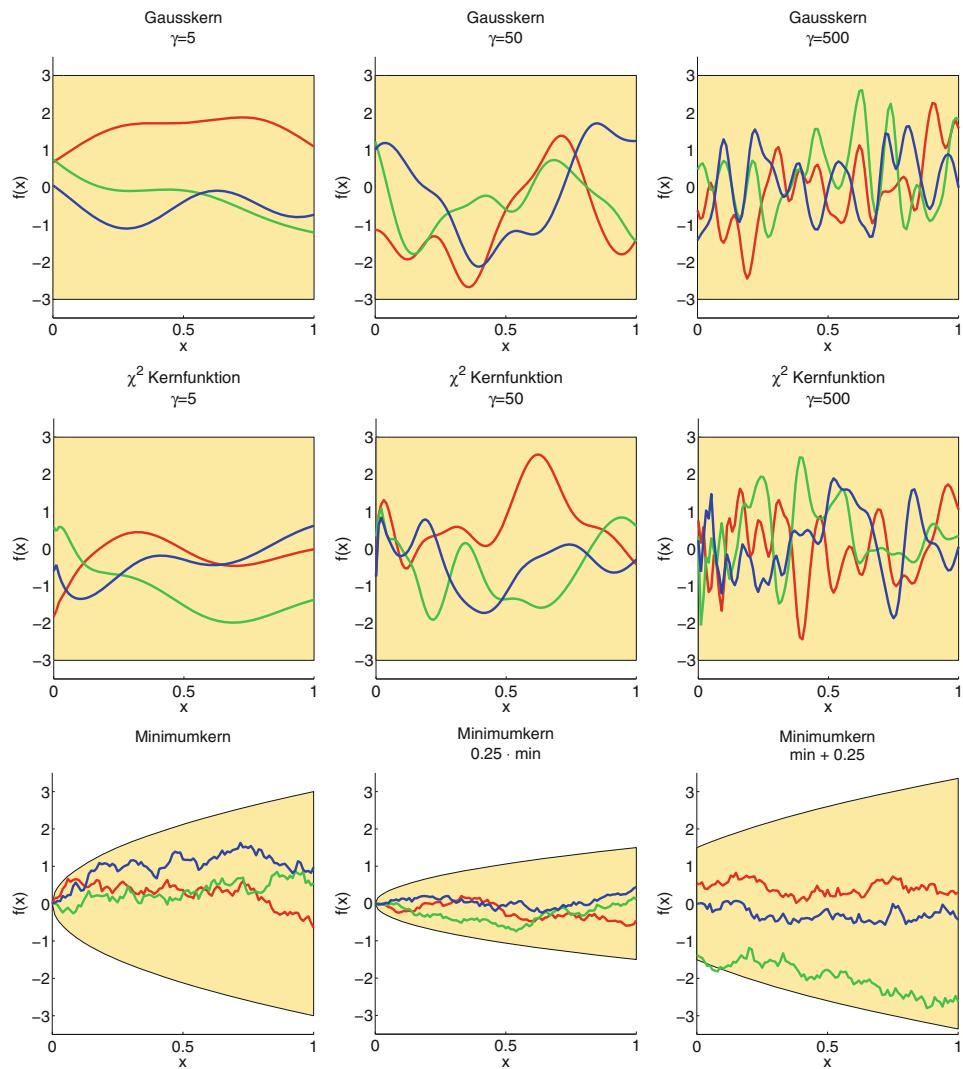


Abb. 18.12 Drei Funktionen werden von einem mittelwertfreien Gauß-Prozess mit unterschiedlichen Kernelfunktionen gezogen (Zeilen: Gaußkern, χ^2 -Kernelfunktion, Minimum-Kernelfunktion) und variierenden Hyperparametern (Spalten). Die markierte Fläche zeigt das Konfidenzintervall an, welches mittels der 3σ -Regel abgeleitet wurde

ben. Durch die Betrachtung einer endlichen Anzahl von Eingabedaten ist es uns möglich, den Erwartungswert bezüglich aller möglichen Funktionen f auf einen wohldefinierten Erwartungswert einfacher Zufallsvariablen zu reduzieren. Der Erwartungswert des quadratischen Differenzenquotienten lässt sich schreiben als:

$$g_K(x_1, \dots, x_n) = \int_{\mathbb{R}^n} \left(\frac{1}{n-1} \sum_{i=1}^{n-1} \left(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \right)^2 \right) \cdot p(\mathbf{f}) d\mathbf{f} \quad (18.120)$$

$$= \frac{1}{n-1} \sum_{i=1}^{n-1} \left(\frac{1}{(x_{i+1} - x_i)^2} \int_{\mathbb{R}^n} (f(x_{i+1}) - f(x_i))^2 p(\mathbf{f}) d\mathbf{f} \right) \quad (18.121)$$

$$= \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{K(x_{i+1}, x_{i+1}) - 2K(x_{i+1}, x_i) + K(x_i, x_i)}{(x_{i+1} - x_i)^2}. \quad (18.122)$$

Wir verwenden nun den Gaußkern und nehmen äquidistante Werte x_i mit Abstand h an:

$$g_{\text{gauss}}(h) = g_K(x_1, x_1 + h, x_1 + 2h, \dots) = \frac{2}{h^2} (1 - \exp(-\gamma h^2)). \quad (18.123)$$

Durch die Anwendung der Regel von l'Hospital erhalten wir als Grenzwert:

$$g_{\text{gauss}} = \lim_{h \rightarrow 0} g_{\text{gauss}}(h) = \lim_{h \rightarrow 0} \frac{4\gamma h \exp(-\gamma h^2)}{2h} = 2\gamma. \quad (18.124)$$

Dies entspricht genau unserer Beobachtung in der ersten Zeile von Abb. 18.12. Lokale Variationen der Funktionen sind um so stärker, je größer der Hyperparameter γ gewählt wird.

Die zweite Zeile in Abb. 18.12 zeigt zufällig gezogene Funktionen unter Verwendung der χ^2 -Kernelfunktion (18.91). Dieses Verhalten ist sehr ähnlich wie bei der Gaußkernfunktion, außer dass hier die Nichtstationarität beobachtet werden kann. Lokale Variationen der Funktionen sind wesentlich häufiger in den Bereichen um den Punkt $x = 0$.

Charakteristische Funktionen eines Gauß-Prozesses mit der Minimum-Kernelfunktion (18.92) sind in der letzten Zeile in Abb. 18.12 dargestellt. Für eindimensionale Eingaberäume $\mathcal{X} = \mathbb{R}$, ist ein GP mit Minimum-Kernelfunktion äquivalent zu einem Wiener-Prozess. Die interessante Eigenschaft dieses Prozesses ist, dass alle gezogenen Funktionen stetig aber nicht differenzierbar sind [65, Abschn. 2.1.1.3]. Dies lässt sich auch an unserem Maß $g_K(x_1, \dots, x_n)$ erkennen:

$$g_{\text{min}}(x_1, \dots, x_n) = \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{x_{i+1} - 2x_i + x_i}{(x_{i+1} - x_i)^2} \quad (18.125)$$

$$= \frac{1}{n-1} \sum_{i=1}^{n-1} \frac{1}{x_{i+1} - x_i}. \quad (18.126)$$

Der Wert g_{min} konvergiert demnach nicht, wenn die Abtastung der Positionen x_i feiner gewählt wird. Dies wird deutlich, wenn man wieder äquidistante Positionen mit dem Abstand

h betrachtet: $g_{\min}(x_1, \dots, x_n) = \frac{1}{h}$. Die Standardabweichung der Funktionswerte $f(x)$ ist \sqrt{x} und führt unmittelbar dazu, dass die Funktion fast sicher durch den Ursprung geht, d. h. $f(0) = 0$. Im Gegensatz zur Gauß- oder χ^2 -Kernelfunktion, ist der Minimumkern nicht parametrisiert und nicht abhängig von einem Parameter γ . Daher ist in der Abb. 18.12 der Einfluss eines Skalierungs- und eines Offsetparameters (18.84) dargestellt.

Grundprinzip der Schätzung In Abschn. 18.1 haben wir gesehen, dass das Grundproblem des maschinellen Lernens die Bestimmung der Abhängigkeit von Eingabedaten \mathbf{x} und Ausgabedaten y ist. Dieser Zusammenhang wird oft mit einer kontinuierlichen Funktion f modelliert:

$$y = f(\mathbf{x}) + \varepsilon. \quad (18.127)$$

Der Term ε ist dabei eine Zufallsvariable, welche die Störung der Ausgabedaten modelliert und daher oft als Rauschmodell bezeichnet wird. Viele Verfahren der Mustererkennung, wie etwa Support-Vektor-Klassifikatoren (Abschn. 18.4.6) aber auch Entscheidungsbäume (Abschn. 18.4.3), parametrisieren die latente Funktion f durch $f(\cdot; \theta)$ und schätzen dann die Parameter θ mit Maximum-Likelihood-Verfahren.

Im Gegensatz zu diesem Vorgehen verwenden GP-Ansätze in der Mustererkennung das Konzept der Marginalisierung, welches wir in Abschn. 18.3.2 kennengelernt haben. Ein wichtiger Bestandteil ist dabei ein Modell für die A-Priori-Verteilung. Weiterhin werden wir die latente Funktion f nicht parametrisieren, sondern direkt als Zufallsvariable interpretieren und diese marginalisieren. Es ist also kein parametrisches Modell notwendig. Wie wir bereits in Abschn. 18.5.5 gesehen haben, eignen sich Gauß-Prozesse hervorragend um die Verteilung von ganzen Funktionen zu beschreiben. Vor allem die Eigenschaft der Stetigkeit ist in diesem Fall entscheidend. So ist es möglich, eine der Hauptannahmen des maschinellen Lernens direkt zu modellieren: Ähnliche Eingabedaten sollten zu ähnlichen Ausgabedaten führen.

Modellannahmen bei der Schätzung Zunächst werfen wir einen Blick auf die zwei wichtigsten Annahmen, welche beim Lernen mit Gauß-Prozessen sowohl bei Regressions- als auch bei Klassifikationsaufgaben getroffen werden:

- Bedingte Unabhängigkeit:** Es existiert eine latente Funktion $f : \mathcal{X} \rightarrow \mathbb{R}$, so dass Ausgaben y unabhängig von ihren Eingaben \mathbf{x} sind, wenn der Wert $f(\mathbf{x})$ der latenten Funktion gegeben ist. Diese bedingte Verteilung der Ausgaben ist durch das *Rauschmodell* $p(y | f(\mathbf{x}))$ festgelegt.
- A-Priori-Modell:** Die Verteilung der Funktion f wird durch einen mittelwertfreien Gauß-Prozess mit der Kernelfunktion $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ beschrieben, d. h. $f \sim \mathcal{GP}(\mathbf{0}, K)$.

Seien nun n Lernbeispiele $\mathbf{x}_i \in \mathbf{X} \subset \mathcal{X}$ mit entsprechenden Ausgaben $y_i \in \mathcal{Y}$ gegeben. Die Ausgaben werden wir im Folgenden in einem Vektor $\mathbf{y} \in \{-1, 1\}^n$ zusammenfassen. Ziel ist

es, die A-Posteriori-Verteilung der Ausgabe y_* eines neuen Beispiels $\mathbf{x}_* \in \mathcal{X}$ zu bestimmen. Dazu marginalisieren wir zunächst den latenten Funktionswert $f_* = f(\mathbf{x}_*)$:

$$p(y_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int_{\mathbb{R}} p(y_* | f_*) p(f_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}) df_*. \quad (18.128)$$

Zu beachten ist, dass wir dabei schon explizit die Annahme der bedingten Unabhängigkeit von y_* und \mathbf{x}_* verwendet haben. Zur Bestimmung der bedingten Verteilung von f_* müssen wir nun eine zweite Marginalisierung durchführen, da die latenten Funktionswerte $\mathbf{f} = (f(\mathbf{x}_i))_{i=1}^n$ des Lerndatensatzes ebenfalls nicht bekannt sind:

$$p(f_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int_{\mathbb{R}^n} p(f_* | \mathbf{x}_*, \mathbf{f}) p(\mathbf{f} | \mathbf{y}, \mathbf{X}) d\mathbf{f}. \quad (18.129)$$

An dieser Stelle können wir jetzt das Rauschmodell verwenden und die bedingte Unabhängigkeit der Lernbeispiele verwenden:

$$p(\mathbf{f} | \mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{f} | \mathbf{X})}{p(\mathbf{y} | \mathbf{X})} \left(\prod_{i=1}^n p(y_i | f_i) \right). \quad (18.130)$$

Die Verteilung $p(\mathbf{f} | \mathbf{X})$ ist eine n -dimensionale Normalverteilung mit Mittelwert Null und der Kovarianzmatrix $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$. Das Rauschmodell $p(y_i | f_i)$ kann unterschiedliche Formen annehmen und hängt von der Art der Ausgaben y_i ab. Bei kontinuierlichen Werten $y \in \mathbb{R}$ handelt es sich um ein Regressionsproblem.

18.5.6 Gauß-Prozess-Regression

Im folgenden Abschnitt zeigen wir die Anwendung der vorgestellten Grundideen auf das Problem der Regression mit kontinuierlichen Ausgaben $y \in \mathbb{R}$. Ein übliches Rauschmodell ist dabei die Annahme eines additiven normalverteilten Rauschens mit Mittelwert Null, d. h. $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Weiterhin nehmen wir an, dass die Fehlerterme ε für jede Eingabe \mathbf{x} unabhängig und identisch verteilt sind:

$$p(y | f(\mathbf{x})) = \mathcal{N}(y | f(\mathbf{x}), \sigma_\varepsilon^2). \quad (18.131)$$

Die Annahme von normalverteiletem Rauschen ermöglicht es, die Marginalisierungen in geschlossener Form durchzuführen. Diese wichtige Eigenschaft resultiert aus der Beobachtung, dass eine Ausgabe y eine Summe der normalverteilten Zufallsvariablen $f(\mathbf{x})$ und ε ist, und daher auch selbst einer Normalverteilung unterliegt. Die gemeinsame Verteilung der Ausgabe y_* und der Ausgaben \mathbf{y} der Lerndaten kann also wie folgt angegeben werden:

$$\begin{aligned} p(y_*, \mathbf{y} | \mathbf{X}) &= \mathcal{N}\left(\begin{bmatrix} y_* \\ \mathbf{y} \end{bmatrix} \mid \mathbf{0}, K((\mathbf{x}_*, \mathbf{X}), (\mathbf{x}_*, \mathbf{X})) + \sigma_\varepsilon^2 \cdot \mathbf{I}\right) \\ &= \mathcal{N}\left(\begin{bmatrix} y_* \\ \mathbf{y} \end{bmatrix} \mid \mathbf{0}, \begin{bmatrix} K(\mathbf{x}_*, \mathbf{x}_*) + \sigma_\varepsilon^2 & \mathbf{k}_*^T \\ \mathbf{k}_* & \mathbf{K} + \sigma_\varepsilon^2 \cdot \mathbf{I} \end{bmatrix}\right). \end{aligned}$$

Um die bedingte Verteilung zu ermitteln, verwenden wir die Resultate aus Abschn. 22.5, welche zeigen, dass die A-Posteriori-Verteilung von y_* gegeben \mathbf{y} ebenfalls normalverteilt ist, d. h. $y_* \sim \mathcal{N}(\mu_*, \sigma_*^2)$. Der Mittelwert lässt sich anhand folgender Vorschrift berechnen:

$$\mu_* = \mathbf{k}_*^T (\mathbf{K} + \sigma_\epsilon^2 \cdot \mathbf{I})^{-1} \mathbf{y} = \mathbf{k}_*^T \boldsymbol{\alpha}. \quad (18.132)$$

Die zentrale Bedeutung der Koeffizienten $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_\epsilon^2 \cdot \mathbf{I})^{-1} \mathbf{y}$ haben wir schon in Abschn. 18.5.1 diskutiert. Der Mittelwert μ_* ist zugleich das Maximum der A-Posteriori-Verteilung und daher der Schätzwert für eine Eingabe \mathbf{x}_* . Wir werden diesen Schätzwert im Folgenden als prädiktiver Mittelwert oder als Regressionsfunktion bezeichnen.

Ein wichtiger Vorteil des GP-Ansatzes ist die Möglichkeit, auch die Varianz des Schätzwertes zu ermitteln. Dies ist ein großer Vorteil der durchgeführten Marginalisierung.

$$\sigma_*^2 = K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \sigma_\epsilon^2 \cdot \mathbf{I})^{-1} \mathbf{k}_* + \sigma_\epsilon^2. \quad (18.133)$$

Oft spricht man in diesem Zusammenhang auch von der Unsicherheit der Schätzung. Der Begriff der Unsicherheit wird meistens mit dem Begriff der Entropie assoziiert. Normalverteilungen mit Varianz σ_*^2 haben eine differentielle Entropie von $\frac{1}{2} \log(2\pi \cdot e \cdot \sigma_*^2)$, welches den Zusammenhang der Begriffe verdeutlicht.

Die Berechnung des prädiktiven Mittelwertes erfordert die vorherige Bestimmung der Koeffizienten $\boldsymbol{\alpha}$. Dies kann mit Hilfe der Cholesky Faktorisierung der regularisierten Kernmatrix erfolgen. Insgesamt ergibt sich so eine kubische Laufzeitkomplexität von $\mathcal{O}(n^3)$. Für jedes Testbeispiel sind danach $\mathcal{O}(n)$ Operationen notwendig, um das Skalarprodukt $\mathbf{k}_*^T \boldsymbol{\alpha}$ auszurechnen. Zur Bestimmung der prädiktiven Varianz sind $\mathcal{O}(n^2)$ Operationen für jedes Testbeispiel notwendig, wenn die Cholesky-Faktorisierung verfügbar ist.

Abbildung 18.13 zeigt ein paar Beispiele für die GP-Regression bei einem kleinen eindimensionalen Beispiel mit dem Gaußkern und unterschiedlichen Werten des Hyperparameters γ . Fokussieren wir uns zunächst auf die Regressionsfunktion (roter Graph in der Abbildung), welche durch den prädiktiven Mittelwert gegeben ist. In der oberen Grafik ist gut die Auswirkung des Rauschmodells erkennbar. Die Regressionsfunktion muss nicht zwangsläufig durch die Lernbeispiele gehen, da die Annahme getroffen wurde, dass diese einem additiven Rauschen unterliegen. Weiterhin ist erkennbar, dass die Funktion sehr stark vom gewählten Hyperparameter γ abhängt und das gleiche Verhalten zeigt, welches wir schon in Abschn. 18.5.5 gesehen haben. Mit einem steigenden Wert von γ ermöglicht die GP-Verteilung eine höhere Flexibilität der entstehenden Regressionsfunktion.

Der markierte Bereich in den Grafiken entspricht dem Konfidenzintervall, welches durch die prädiktive Varianz ermittelt wurde. Eine hohe Varianz der A-Posteriori-Verteilung ist vor allem an Punkten \mathbf{x}_* zu beobachten, welche sehr weit von den Lerndaten entfernt liegen, d. h. am Rand der jeweiligen Grafiken in unserem Fall. Durch die Tatsache, dass die Matrix $(\mathbf{K} + \sigma_\epsilon^2 \cdot \mathbf{I})$ positiv definit ist und für den Gaußkern $K(\mathbf{x}, \mathbf{x}) = 1$ gilt, lässt sich $1 + \sigma_\epsilon^2$ als obere Schranke für die prädiktive Varianz angeben. Die Schranke ist außerdem der Grenzwert für einen Punkt \mathbf{x}_* , wenn sich dieser von den Lerndaten entfernt.

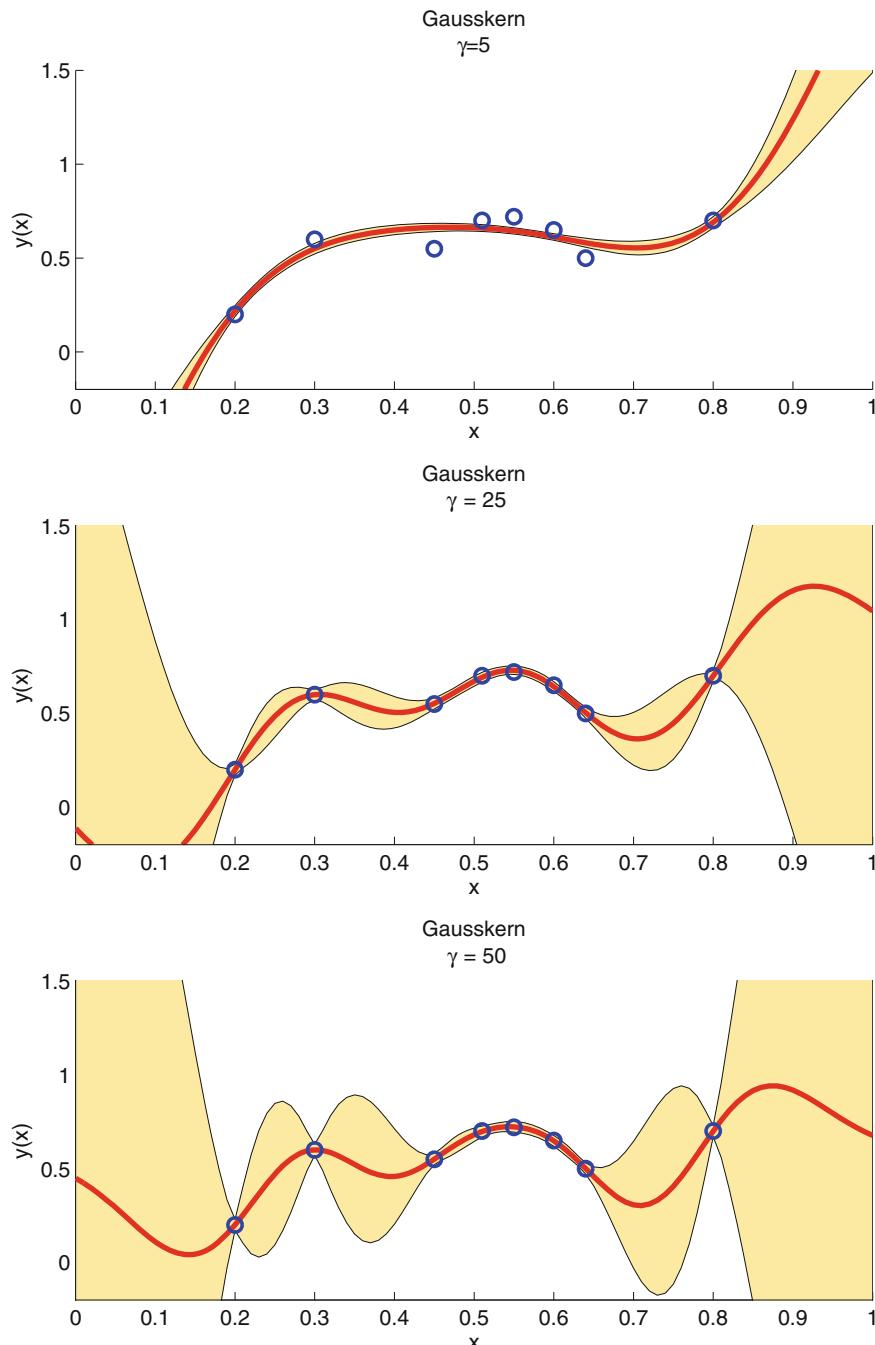


Abb. 18.13 Gauß-Prozess-Regression mit einem Gaußkern (18.90) und unterschiedlichen Werten des Hyperparameters γ . Die rote Kurve zeigt den prädiktiven Mittelwert (18.132) für jedes Testbeispiel \mathbf{x}_* und die markierte Fläche charakterisiert das Konfidenzintervall $[\mu_* - 3\sigma_*, \mu_* + 3\sigma_*]$. Die Standardabweichung des Rauschens wurde auf 0,01 festgelegt

18.5.7 Klassifikation mit Gauß-Prozessen

Wie wir gesehen haben, führt die Annahme des normalverteilten Rauschens zu einer geschlossenen Lösung der Integrale in (18.128) und (18.129) und damit zu direkten Schätzgleichungen. Bei binären Klassifikationsproblemen mit $y \in \{-1, 1\}$, kann natürlich auch die Gauß-Prozess-Regression direkt angewendet werden. Die Ausgaben werden dann einfach als kontinuierliche Werte angenommen. Als Wert der weichen Klassifikationsentscheidung ist es zum Beispiel möglich, den prädiktiven Mittelwert μ_* oder den Wert $p(y_* = 1 | \mathbf{x}_*, \mathbf{y}, \mathbf{X})$ der Dichtefunktion zu verwenden. Dieser Ansatz wird als *label regression* [50] bezeichnet und erfordert nur ein paar einfache algebraische Operationen.

Aus theoretischer Sichtweise erscheint diese Methode jedoch nicht angebracht. Schließlich ist das normalverteilte Rauschen nicht mit dem diskreten Wertebereich der Ausgaben y vereinbar. Ein Rauschmodell für die binäre Klassifikation sollte eher folgende Bedingung erfüllen:

$$\forall f \in \mathbb{R}: p(y = 1 | f) + p(y = -1 | f) = 1. \quad (18.134)$$

Nur in diesem Fall ist eine korrekte diskrete Wahrscheinlichkeitsverteilung gegeben. Das *probit*-Modell ist solch ein mögliches Modell und kann wie folgt motiviert und hergeleitet werden. Um aus einem kontinuierlichen Wert f eine Klassifikationsentscheidung abzuleiten, wird oft einfach eine Schwellwertoperation durchgeführt, z. B. durch $y = \text{sign}(f + \varepsilon)$. Dabei ist $\varepsilon \sim \mathcal{N}(0, \sigma_c^2)$ ein entsprechender Rauschterm. Auf der Grundlage dieser Überlegung können wir die diskrete Wahrscheinlichkeitsverteilung von y bestimmen und mittels der kumulativen Gaußfunktion Φ und der Fehlerfunktion $\text{erf}(z)$ ausdrücken:

$$p(y | f) = \begin{cases} p(f + \varepsilon \geq 0) & \text{wenn } y = 1 \\ p(f + \varepsilon < 0) & \text{wenn } y = -1 \end{cases} \quad (18.135)$$

$$= \int_{-\infty}^{y \cdot f} \mathcal{N}(z | 0, \sigma_c^2) dz \quad (18.136)$$

$$= \frac{1}{2} \left(\text{erf} \left(\frac{y \cdot f}{\sqrt{2} \cdot \sigma_c} \right) + 1 \right) = \Phi \left(\frac{y \cdot f}{\sigma_c} \right). \quad (18.137)$$

In den meisten Fällen wird der Skalierungsfaktor σ_c des Rauschmodells einfach auf eins gesetzt, da er auch durch einen multiplikativen Faktor bei der Kernelfunktion ausgedrückt werden kann. Alternativ lässt sich eine Sigmoidfunktion verwenden und führt auf das sogenannte logistische Rauschmodell [57]:

$$p(y | f) = \frac{1}{2} (\text{sig}(y \cdot f) + 1). \quad (18.138)$$

Das logistische Modell findet zum Beispiel auch bei neuronalen Netzwerken Verwendung [3, Abschn. 5].

Der große Nachteil beider Klassifikationsmodelle ist, dass die bedingte Verteilung $p(\mathbf{f} | \mathbf{y}, \mathbf{X})$ (18.130) der Werte \mathbf{f} der latenten Funktion nicht mehr einer Normalverteilung entspricht. Dadurch wird die Auflösung der Marginalisierung in (18.129) nicht mehr möglich. Dieses Problem ist äußerst typisch für Bayes-Ansätze (Abschn. 18.3.2). In den folgenden Abschnitten werden wir als Lösung dieses Problems die Laplace-Approximation als Verfahren für die approximative Inferenz kennenlernen. Eine weitere wichtige Methode in diesem Bereich ist *expectation propagation* [47], welches in [57, Abschn. 3.6] gut beschrieben ist.

18.5.8 Laplace-Approximation

Im Folgenden konzentrieren wir uns auf die Marginalisierung der Zufallsvariable \mathbf{f} . Zur Lösung der ein-dimensionalen Integration in (18.128) (Marginalisierung von f_*) können einfache numerische Verfahren, wie die Monte-Carlo Simulation verwendet werden. Für das probit-Modell existiert sogar eine Lösung in geschlossener Form [57, Abschn. 3.4.2].

Wie wir gesehen haben, ist die Marginalisierung in (18.129) effizient möglich, wenn die bedingte Verteilung $p(\mathbf{f} | \mathbf{y}, \mathbf{X})$ eine Normalverteilung ist. Die Grundidee der Laplace-Approximation ist es daher allgemeine Verteilungen mit einer Normalverteilung $q(\mathbf{f} | \mathbf{y}, \mathbf{X})$ zu approximieren. Aus diesem Grund ist es die Hauptaufgabe der Laplace-Approximation, eine geeignete Verteilung $q(\mathbf{f} | \cdot)$ zu finden, welche zur Ausgangsverteilung $p(\mathbf{f} | \cdot)$ am Besten passt.

Wenn q eine Normalverteilung ist, dann ist $\log q$ eine quadratische Funktion. Diese Tatsache verwenden wir um q zu finden, indem wir die Taylor-Approximation von $L(\mathbf{f}) = \log p(\mathbf{f} | \mathbf{y}, \mathbf{X})$ durchführen. Sei $\hat{\mathbf{f}}$ das Maximum der exakten bedingten Verteilung, d.h. das Ergebnis des MAP-Schätzers, welches sich in wenigen Iterationen mit einem einfachen Newton-Verfahren finden lässt [57]. Die Taylor-Approximation von L um die Position $\hat{\mathbf{f}}$ lässt sich wie folgt angeben:

$$\begin{aligned} L(\mathbf{f}) &= L(\hat{\mathbf{f}}) + [\{\nabla L\}(\hat{\mathbf{f}})]^T (\mathbf{f} - \hat{\mathbf{f}}) \\ &\quad + \frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^T [\{\nabla^2 L\}(\hat{\mathbf{f}})] (\mathbf{f} - \hat{\mathbf{f}}) + \dots \end{aligned} \quad (18.139)$$

$$= L(\hat{\mathbf{f}}) + \frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^T [\{\nabla^2 L\}(\hat{\mathbf{f}})] (\mathbf{f} - \hat{\mathbf{f}}) + \dots \quad (18.140)$$

In obiger Gleichung haben wir den Gradienten der Funktion L mit ∇L und die Hessematrix mit $\nabla^2 L$ bezeichnet. Außerdem haben wir explizit ausgenutzt, dass der Gradient an der Stelle $\hat{\mathbf{f}}$ des Maximums Null sein muss. Durch die Taylor-Approximation kann folgende Approximation der bedingten Verteilung angegeben werden:

$$q(\mathbf{f} | \mathbf{y}, \mathbf{X}) = \mathcal{N}\left(\mathbf{f} | \hat{\mathbf{f}}, -[\{\nabla^2 L\}(\hat{\mathbf{f}})]^{-1}\right). \quad (18.141)$$

Die Kovarianzmatrix ist positiv definit, daher wissen wir auch, dass $\hat{\mathbf{f}}$ ein lokales Maximum ist. Die exakte Formel für die Hessematrix können wir durch die Definition von L und der Anwendung des Gesetzes von Bayes finden:

$$L(\mathbf{f}) = \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}|\mathbf{X}) - \log p(\mathbf{y}|\mathbf{X}) \quad (18.142)$$

$$\{\nabla^2 L\}(\hat{\mathbf{f}}) = \{\nabla_{\mathbf{f}}^2 \log p(\mathbf{y}|\mathbf{f})\}(\hat{\mathbf{f}}) - \mathbf{K}^{-1} \quad (18.143)$$

$$= -\mathbf{W} - \mathbf{K}^{-1}. \quad (18.144)$$

Die Matrix \mathbf{W} ist eine $n \times n$ große Diagonalmatrix, welche die Werte der zweiten Ableitungen des negativen Logarithmus des Rauschmodells enthält:

$$\forall 1 \leq i \leq n : W_{ii} = - \left\{ \frac{d^2}{d^2 f} \log p(y_i | f) \right\}(\hat{f}_i). \quad (18.145)$$

Die Diagonalität der Matrix entsteht durch die Annahme der bedingte Unabhängigkeit der Ausgaben, welche wir ganz am Anfang des Abschnittes getroffen hatten.

Für die Klassifikation müssen wir die bedingte Verteilung des latenten Funktionswertes f_* bestimmen. Zunächst fassen wir aber noch einmal alle Informationen über den Integranden in (18.129) zusammen:

$$p(f_* | \mathbf{x}_*, \mathbf{f}) = \mathcal{N}(f_* | \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{f}, K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k}_*), \quad (18.146)$$

$$p(\mathbf{f} | \mathbf{y}, \mathbf{X}) \approx q(\mathbf{f} | \mathbf{y}, \mathbf{X}) = \mathcal{N}(\mathbf{f} | \hat{\mathbf{f}}, (\mathbf{W} + \mathbf{K}^{-1})^{-1}). \quad (18.147)$$

Wenn wir jetzt das Lemma 22.6 anwenden, gelangen wir direkt zu der Bestimmung des prädiktiven Mittelwertes von f_* :

$$E(f_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \mathbf{k}_*^T \mathbf{K}^{-1} \hat{\mathbf{f}}, \quad (18.148)$$

und der korrespondierenden Varianzschätzung:

$$\begin{aligned} \sigma^2(f_* | \cdot) &= K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k}_* + \mathbf{k}_*^T \mathbf{K}^{-1} (\mathbf{W} + \mathbf{K}^{-1})^{-1} \mathbf{K}^{-1} \mathbf{k}_* \\ &= K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K}^{-1} - \mathbf{K}^{-1} (\mathbf{W} + \mathbf{K}^{-1})^{-1} \mathbf{K}^{-1}) \mathbf{k}_* \\ &\stackrel{\text{(Lemma 22.3)}}{=} K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_*. \end{aligned} \quad (18.149)$$

Bei der Implementierung sind viele numerische Details zu beachten, welche in [57, Abschn. 3.4.3] erläutert werden.

18.5.9 Zusammenhang zum SVM Ansatz

Wie bei vielen Ansätzen in der Mustererkennung besteht trotz der unterschiedlichen Formulierung von GP- und SVM-Klassifikatoren ein starker Zusammenhang zwischen beiden

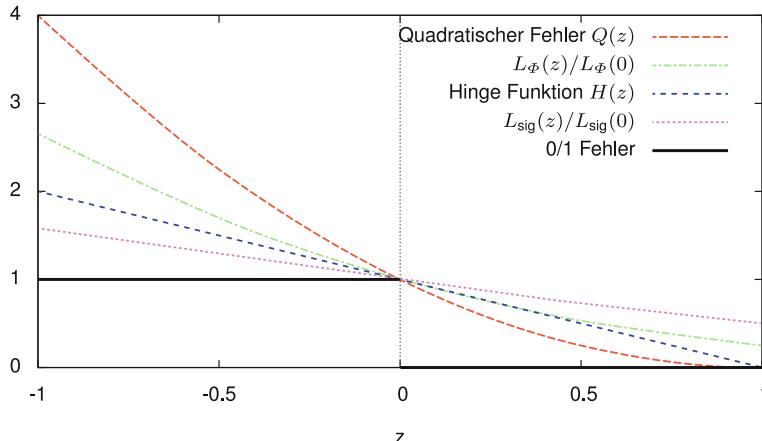


Abb. 18.14 Darstellung verschiedener Fehlerterme: *hinge loss* des SVM-Ansatzes, quadratischer Fehler bei der GP-Regression, sowie L_Φ und L_{sig} bei der GP-Klassifikation. In der Abbildung werden diese Funktionen mit der Funktion zur Bestimmung von Fehlklassifikationen verglichen

Methoden. Wir betrachten im Folgenden die Kernelversion des SVM-Klassifikator, wie er in Abschn. 18.4.6 vorgestellt wurde. Durch eine geschickte Umformulierung, welche in [57, S. 144] detailliert dargestellt ist, kann das Lernen des SVM-Klassifikators auf folgendes Optimierungsproblem zurückgeführt werden:

$$\min_{\mathbf{f} \in \mathbb{R}^n} C \sum_{i=1}^n H(y_i \cdot f_i)) + \frac{1}{2} \mathbf{f}^T \mathbf{K}^{-1} \mathbf{f} . \quad (18.150)$$

Zusätzlich haben wir die Konstante $\lambda = (2C)^{-1}$ definiert. Ein ähnliches Problem erhalten wir bei der Maximierung der Verteilung $p(\mathbf{f} | \mathbf{y}, \mathbf{X})$ (oder der äquivalenten Minimierung von $-\log p(\mathbf{f} | \mathbf{y}, \mathbf{X})$) bezüglich des Vektors \mathbf{f} :

$$\min_{\mathbf{f} \in \mathbb{R}^n} - \sum_{i=1}^n \log p(y_i | f_i) + \frac{1}{2} \mathbf{f}^T \mathbf{K}^{-1} \mathbf{f} . \quad (18.151)$$

Vergleicht man beide Optimierungsprobleme so fällt sofort auf, dass diese sich nur in der Wahl des Fehlerterms unterscheiden, welcher von den Produkten $z = y_i f_i$ abhängig ist. Der SVM-Klassifikator verwendet zum Lernen die *hinge loss* $H(z)$ und der GP-Ansatz unter Annahme eines normalverteilten Rauschens die quadratische Funktion $Q(z) = (1-z)^2$. Bei der GP-Klassifikation werden hingegen die Funktionen $L_{\text{sig}}(z) = -\log \text{sig}(z)$ und $L_\Phi(z) = -\log \Phi(z)$ zur Approximation des Fehlers eingesetzt. In Abb. 18.14 sind die einzelnen Fehlerfunktionen dargestellt und mit der Stufenfunktion $M(z) = \delta(z \geq 0)$ zur Bestimmung von Fehlklassifikationen verglichen. Es lässt sich gut erkennen, dass alle Funktionen die Funktion M von oben beschränken und glatt und stetig sind. Eine direkte Minimierung von M ist oft nicht möglich, da diese nicht stetig und schwierig zu optimieren ist. Daher ist es ein

üblicher Trick in der Mustererkennung stattdessen eine stetige und differenzierbare obere Schranke zu minimieren. Die Wahl dieser Schranke führt automatisch zu unterschiedlichen Ansätzen der Klassifikation, mit eigener Herleitung und Motivation. Welche dieser Funktionen sich besser eignet, ist vom Klassifikationsproblem abhängig und es lässt sich keine allgemeine Aussage treffen.

Kehren wir aber noch einmal zum Vergleich der Optimierungsprobleme zurück und schauen uns die GP-Regression mit $p(y_i | f_i) = \mathcal{N}(y_i | f_i, \sigma_\epsilon^2)$ näher an. Das Optimierungsproblem (18.151) lässt sich dann wie folgt spezifizieren:

$$\min_{\mathbf{f} \in \mathbb{R}^n} \quad \frac{1}{2\sigma_\epsilon^2} \|\mathbf{y} - \mathbf{f}\|^2 + \frac{1}{2} \mathbf{f}^T \mathbf{K}^{-1} \mathbf{f}. \quad (18.152)$$

Bei einem Vergleich von (18.152) und (18.150) fällt sofort auf, dass C^{-1} und σ_ϵ^2 anscheinend den gleichen Einfluss auf das Zielkriterium ausüben. Eine Erhöhung der Standardabweichung σ_ϵ^2 im Rauschmodell bewirkt ein stärkeres Gewicht auf den Regularisierungsterm und damit eine Bevorzugung von latenten Funktionen mit geringer Modellkomplexität.

18.5.10 Mehrklassen-Klassifikation

Ähnlich zur Darstellung des SVM-Ansatzes haben wir auch bei der GP-Klassifikation zunächst nur den Fall der binären Klassifikation betrachtet. Für die Mehrklassen-Klassifikation gibt es Ansätze, die auf der GP-Modellierung basieren und mit Ausgaben $y \in \mathcal{Y} = \{1, \dots, M\}$ direkt umgehen können. Diese Methoden sind aber oft sehr aufwendig und wir verweisen an dieser Stelle wieder auf das Buch von Rasmussen und Williams [57].

Im Gegensatz zur direkten Modellierung ist es natürlich auch wieder möglich, den *one-vs-all* Ansatz zu verwenden (Abschn. 18.4.6). Für jede Klasse $\kappa \in \mathcal{Y}$ wird also ein GP-Klassifikator gelernt, welcher alle Beispiele der Klasse κ als positive und alle übrigen Beispiele als negative Lernbeispiele verwendet. Sei $\mathbf{y}^{(\kappa)} \in \{-1, 1\}^n$ der Vektor der binären Beschriftungen für die Klasse $\kappa \in \{1, \dots, M\}$, welcher aus den Beschriftungen \mathbf{y} generiert wird, indem $y_i^{(\kappa)}$ auf 1 gesetzt wird, wenn das Beispiel i zur Klasse κ gehört, und ansonsten mit -1 festgelegt ist. Für die Klassifikation mit GP-Regression (*label regression*) ergibt sich dann die Wahl der Klasse anhand des größten geschätzten Mittelwertes der Beschriftung y_* eines neuen Beispiels \mathbf{x}_* :

$$y_*^{\text{multi}} = \underset{\kappa=1 \dots M}{\operatorname{argmax}} \mu_*^\kappa = \underset{\kappa=1 \dots M}{\operatorname{argmax}} \mathbf{k}_*^T (\mathbf{K} + \sigma_\epsilon^2 \cdot \mathbf{I})^{-1} \mathbf{y}^\kappa. \quad (18.153)$$

Ein wichtiger Vorteil dieses Ansatzes ist es, dass die Kernmatrix unverändert bleibt und so zum Beispiel nur einmal die Cholesky-Faktorisierung durchgeführt werden muss. Daher ist es für die Laufzeit des Klassifikators beim Lernen nicht entscheidend wie viele Klassen zur Klassifikation verwendet werden.

18.5.11 Hyperparameter-Schätzung

Wie wir bereits in Abschn. 18.5.5 und besonders in Abb. 18.12 und 18.13 gesehen haben, spielen die Hyperparameter η der Kernfunktion eine ganz entscheidende Rolle und haben einen starken Einfluss auf das Ergebnis. Für SVM Ansätze hatten wir bereits die Kreuzvalidierung zur Bestimmung von Hyperparametern vorgestellt. Ein großer Nachteil dieser Methode ist die vollständige Suche im Parameterraum. Bei vielen Parametern ist die Suche durch die hohe Laufzeit nicht anwendbar.

Durch die wahrscheinlichkeitstheoretische Modellierung ist es im Gegensatz zu SVM-Ansätzen bei der GP-Modellierung möglich, ein geschlossenes Zielkriterium zur Optimierung von Hyperparametern zu definieren. Die Anwendung der Maximum-Likelihood Schätzung für die Hyperparameter bei der GP-Regression ergibt sofort:

$$\hat{\eta}^{\text{ML}} = \underset{\eta}{\operatorname{argmax}} p(\mathbf{y} | \mathbf{X}, \eta) \quad (18.154)$$

$$= \underset{\eta}{\operatorname{argmin}} -\log p(\mathbf{y} | \mathbf{X}, \eta) \quad (18.155)$$

$$= \underset{\eta}{\operatorname{argmin}} \frac{1}{2} \log \det(\tilde{\mathbf{K}}_{\eta}) + \frac{1}{2} \mathbf{y}^T \tilde{\mathbf{K}}_{\eta}^{-1} \mathbf{y}, \quad (18.156)$$

Bei der Herleitung haben wir additive Konstanten vernachlässigt, welche nicht für die Optimierung notwendig sind und die Verteilung des Beschriftungsvektors $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{K}}_{\eta})$ ausgenutzt. Die Matrix $\tilde{\mathbf{K}}_{\eta}$ ist dabei die regularisierte Kernmatrix, $\tilde{\mathbf{K}}_{\eta} = \mathbf{K}_{\eta} + \sigma_{\varepsilon}^2 \cdot \mathbf{I}$, welche von den Hyperparametern abhängt. Das obige Kriterium kann mit einem nichtlinearen Optimierungsverfahren, wie etwa dem Newton-Verfahren optimiert werden. Die dafür notwendige Ableitung lässt sich sogar analytisch bestimmen.

Bei der Hyperparameter-Optimierung im Mehrklassen-Fall ist es möglich, die Summe der Terme für jeden einzelnen binären Klassifikator zu minimieren:

$$\hat{\eta}^{\text{ML}} = \underset{\eta}{\operatorname{argmax}} p(\mathbf{y}^1, \dots, \mathbf{y}^M | \mathbf{X}, \eta) \quad (18.157)$$

$$= \underset{\eta}{\operatorname{argmin}} -\sum_{\kappa=1}^M \log p(\mathbf{y}^{\kappa} | \mathbf{X}, \eta) \quad (18.158)$$

$$= \underset{\eta}{\operatorname{argmin}} \frac{M}{2} \log \det(\tilde{\mathbf{K}}_{\eta}) + \frac{1}{2} \sum_{\kappa=1}^M (\mathbf{y}^{\kappa})^T \tilde{\mathbf{K}}_{\eta}^{-1} \mathbf{y}^{\kappa} \quad (18.159)$$

$$= \underset{\eta}{\operatorname{argmin}} \frac{M}{2} \log \det(\tilde{\mathbf{K}}_{\eta}) + \frac{1}{2} \operatorname{Spur}\left(\tilde{\mathbf{K}}_{\eta}^{-1} \sum_{\kappa=1}^M \mathbf{y}^{\kappa} (\mathbf{y}^{\kappa})^T\right). \quad (18.160)$$

Grundsätzlich ist zu sagen, dass die Hyperparameter-Optimierung in dieser Form nicht die einzige wahre Lösung ist, sondern eher eine mögliche Alternative zur Kreuzvalidierung darstellt. Eine andere Möglichkeit ist es auch, sogenannte *leave-one-out* Schätzungen zu verwenden [57].

18.6 Bewertung von Klassifikatoren

Im folgenden Abschnitt wollen wir uns näher mit der Bewertung von Klassifikatoren und Klassifikationssystemen beschäftigen. Wie schon am Anfang des Kapitels erwähnt, ist die quantitative empirische Auswertung für die Wahl des Klassifikators sowie etwaiger Hyperparameter wesentlich.

18.6.1 Wahl der Testbeispiele

Wir gehen im Folgenden von einer maschinellen Lernaufgabe mit einem vollständig beschrifteten (Lern-)Datensatz \mathcal{D} aus. Wird ein Klassifikator auf dem gleichen Datensatz angewendet mit dem dieser auch gelernt wurde, so spricht man von Reklassifikation. Der ermittelte Fehler der resultierenden Schätzungen der Ausgaben auf diesem Datensatz wird Reklassifikationsfehler genannt. Eines der fundamentalen Irrtümer bei der Evaluation von maschinellen Lernverfahren ist es, den Reklassifikationsfehler zur Evaluation der Leistungsfähigkeit eines Klassifikators zu verwenden. Warum dies ein Irrtum ist, lässt sich schnell anhand des NN-Klassifikators (Abschn. 18.4.1) erläutern. Für jedes Lernbeispiel liefert dieser Klassifikator eine perfekte Schätzung, da der nächste Nachbar im Lerndatensatz immer das Beispiel selbst ist. Der Reklassifikationsfehler lässt demnach keine Rückschlüsse über die Leistungsfähigkeit in der Praxis, d. h. auf neuen unbekannten Daten, zu. Vielmehr erhält man durch diesen ein Maß für die Modellkomplexität des Klassifikators.

Die Auswertung eines Klassifikationssystems aber auch eines Regressionsverfahrens muss daher immer auf einem Testdatensatz erfolgen, welcher disjunkt zur eigentlichen Lernstichprobe ist. Bei der Aufteilung eines gegebenen Datensatzes in Lern- und Testdatensatz sind mehrere Aspekte zu beachten:

1. Testdatensatz und Lerndatensatz sollten repräsentativ für die in der Praxis zu erwartenden Testbeispiele sein (gleiche Verteilung der Ein- und Ausgaben).
2. Im Lerndatensatz müssen ausreichend Beispiele vorhanden sein, damit die Leistungsfähigkeit des erlernten Klassifikators mit der eines auf der kompletten Stichprobe erlernten Klassifikators annähernd vergleichbar ist.
3. Der Testdatensatz sollte genügend Beispiele enthalten um eine statistisch robuste Schätzung des Fehlers durchführen zu können.

Bei einem relativ zur Schwierigkeit der Aufgabenstellung kleinen Datensatz (Abschn. 18.2.2) hat die Wahl der Lernstichprobe einen großen Einfluss auf die Leistungsfähigkeit des Klassifikators. Aus diesem Grund werden oft mehrere Einteilungen (*data splits*) vorgenommen um dadurch einen mittleren Fehler oder den Mittelwert eines Gütekriteriums zu bestimmen. Ähnlich zum Bagging-Ansatz, welchen wir in Abschn. 18.3.4 kennengelernt haben um die Varianz der Schätzungen eines Klassifikators zu verringern, lässt sich auch bei der Auswertung von Klassifikatoren durch eine zufällige Auswahl von Lern- und Teststichprobe die Robustheit der Fehlerschätzung erhöhen. In diesem Fall wählen wir einfach aus

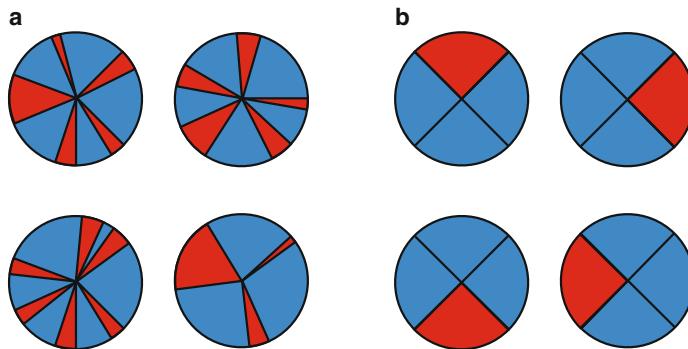


Abb. 18.15 Techniken zur Auswahl von Lern- und Testbeispielen. **a** Zufällige Auswahl, **b** Kreuzvalidierung. Der Kreis repräsentiert den gesamten Datensatz und die Auswahl der Lernbeispiele ist mit blauen Kreissektoren gekennzeichnet

dem gesamten Datensatz gleichverteilt zufällig Lernbeispiele aus und die Bestimmung des Fehlers des auf diesen Daten gelernten Klassifikators erfolgt dann auf den restlichen Beispielen. Ein anderer Ansatz ist die k -fache Kreuzvalidierung, bei welcher der Datensatz in k gleichgroße Teilmengen aufgeteilt wird. Es erfolgt dann das Lernen auf jeweils $k-1$ dieser Mengen und ein Testen auf der verbleibenden Menge. Beide Konzepte für die Auswahl von Lern- und Testbeispielen sind in Abb. 18.15 dargestellt.

Ein Extremfall der k -fachen Kreuzvalidierung ist die Bestimmung von *leave-one-out* Fehlern. Wenn insgesamt n Beispiele im Datensatz \mathcal{D} zur Verfügung stehen, wird ein Klassifikator mit $n-1$ Beispielen gelernt und auf dem verbleibenden Beispiel getestet. Dies kann insgesamt genau n -Mal erfolgen und die Schätzungen für jedes der n Beispiele können dann mit einem Gütekriterium bewertet werden. Ein Nachteil dieses Verfahrens ist der Rechenaufwand beim Lernen von insgesamt n Klassifikatoren. Im Falle des k -NN Klassifikators oder der Gauß-Prozess-Regression ist aber eine effiziente Bestimmung von *leave-one-out* Fehlern möglich [57].

Die Evaluation von Klassifikationssystemen ist nicht nur für die Gesamtbewertung des entwickelten Verfahrens wichtig, sondern auch für den Lernschritt und die Wahl von Parametern selbst. Oft ist es notwendig einen zusätzlichen Validierungsdatensatz oder eine Kreuzvalidierung auf der Lernstichprobe zu verwenden, damit keine Überanpassung des Klassifikationssystems an den eigentlichen Testdatensatz erfolgt. Ein typisches Beispiel für einen Parameter, welcher oft mittels Kreuzvalidierung bestimmt wird, ist die Anzahl k der Nachbarn beim k -NN Verfahren oder der Regularisierungsparameter C beim SVM Klassifikator (Abschn. 18.4.6).

18.6.2 Erkennungsraten bei mehreren Klassen

Während bei Regressionsaufgaben oft einfach der quadratische Fehler zwischen geschätzten Ausgaben und exakten Ausgaben berechnet wird, erfordert die Auswertung bei Klassi-

fikationsaufgaben andere Auswertungsmaße. Der quadratische Fehler ist bei den diskreten Ausgaben eines Klassifikators nicht sinnvoll, da ansonsten die Reihenfolge der Klassen Auswirkung auf das Auswertungsmaß hat.

Gehen wir zunächst von der Klassifikation von mehr als zwei Klassen aus. Ein wichtiges Werkzeug der Auswertung ist die Vertauschungsmatrix (*confusion matrix*) $C \in \mathbb{N}^{M \times M}$, wobei M die Anzahl der Klassen in der Aufgabenstellung ist. Die Vertauschungsmatrix zählt wie oft ein Klassifikator ein Beispiel der Klasse i der Klasse j zugewiesen hat:

$$C_{ij} = |\{\mathbf{x} \in \mathcal{D}^{\text{test}} \mid \mathbf{x} \text{ gehört zur Klasse } j \text{ wurde aber als } i \text{ klassifiziert}\}|. \quad (18.161)$$

Bei einem perfekten Klassifikator ist die Vertauschungsmatrix eine reine Diagonalmatrix. Hohe positive Werte $C_{ij} > 0$ welche nicht auf der Diagonale ($i \neq j$) stehen, signalisieren eine schwierige Unterscheidung der Klassen i und j aufgrund der gegebenen Merkmale und des Klassifikationsmodells. Ausgehend von der Vertauschungsmatrix können wir jetzt auch Gütemaße für die Bewertung des Klassifikators definieren. Ein typischer und intuitiver Wert ist die Erkennungsrate, d. h. der relative Anteil der Falschklassifikationen an den Testbeispielen:

$$\text{err-ov} = \frac{1}{n^{\text{(test)}}} \sum_{j=1}^M C_{jj}. \quad (18.162)$$

Die Angabe dieses Wertes für die Bewertung besitzt aber einen entscheidenden Nachteil. Gehören zum Beispiel im Testdatensatz 99% der Beispiele zur Klasse 1, so wird ein trivialer Klassifikator, welcher immer nur als Klassifikationsentscheidung die Klasse 1 zurück gibt, mit der Erkennungsrate $\text{err-ov} = 0,99$ bewertet. Dieser Wert ist jedoch irreführend bei der Bewertung. Eine Alternative ist die Bestimmung der mittleren Erkennungsrate der einzelnen Klassen:

$$\text{err-avg} = \frac{1}{M} \cdot \sum_{j=1}^M \frac{C_{jj}}{n_j^{\text{(test)}}}. \quad (18.163)$$

Bei diesem Gütemaß wird die Erkennungsrate zunächst für jede einzelne Klasse betrachtet und dann gemittelt. Im beschriebenen Extremfall eines trivialen „konstanten“ Klassifikators erhalten wir $\text{err-avg} = \frac{1}{M}$. Im Falle von gleichverteilten Klassen im Testdatensatz ergeben beide Gütemaße den gleichen Wert.

18.6.3 Bewertung von binären Klassifikatoren

Bei der Unterscheidung von nur zwei Klassen, d. h. bei der binären Klassifikation, können wir natürlich auch die zwei Varianten der Erkennungsrate zur Bewertung verwenden. Die

Vertauschungsmatrix C hat in diesem Fall nur vier Einträge:

$$C = \begin{bmatrix} \text{TN(true negatives)} & \text{FN(false negatives)} \\ \text{FP(false positives)} & \text{TP(true positives)} \end{bmatrix}. \quad (18.164)$$

Zur Bezeichnung der vier Werte haben wir die üblichen Begriffe aus der englischsprachigen Literatur angegeben. Wie wir in den vorherigen Abschnitten kennengelernt haben, liefern viele binäre Klassifikatoren einen kontinuierlichen Ausgabewert $f(\mathbf{x}_*)$ zurück, welcher dann per Schwellwertoperation zur Klassifikationsentscheidung führt. Bei linearen Klassifikatoren ist dies der algebraische vorzeichenbehaftete Abstand $f(\mathbf{x}_*) = \mathbf{w}^T \mathbf{x}_* + b$, welcher zu folgender diskreten Klassifikationsentscheidung führt:

$$h(\mathbf{x}_*) = \begin{cases} -1 & \mathbf{w}^T \mathbf{x}_* + b < t \\ 1 & \mathbf{w}^T \mathbf{x}_* + b \geq t \end{cases}. \quad (18.165)$$

Erhöht man den Schwellwert t so werden automatisch mehr Beispiele in der Testphase als -1 klassifiziert. Eine Veränderung des Schwellwertes zieht demnach meist eine Veränderung der Vertauschungsmatrix nach sich. In vielen Anwendungsfällen ist eine sehr niedrige Falschpositivrate (*false positive rate*) $FPR = \frac{FP}{n_{\text{neg}}}$ erforderlich. Ein typisches Beispiel ist hier die Fußgängerdetektion, bei der Fußgänger in Kamerabildern lokalisiert werden müssen. Diese Aufgabenstellung ist ein binäres Klassifikationsproblem und als positive Beispiele zählen hier Kamerabilder in denen sich ein Fußgänger direkt vor dem Fahrzeug befindet. Ein Algorithmus mit einer hohen Falschpositivrate wird viele Kamerabilder, in denen sich keine Personen befinden, als positiv klassifizieren. Eine Fahrzeugelektronik die aufgrund der Aussagen solch eines Klassifikators zu ständigen Notbremsungen führt, erzeugt vermutlich mehr Unfälle als es welche vermeiden kann.

Ziel ist es daher eine Auswertungsstrategie zu verfolgen, welche zunächst unabhängig von konkreten Angaben des Schwellwertes ist. Sogenannte ROC-Kurven (*receiver operator characteristic*) betrachten alle möglichen Falschpositivraten und Richtigpositivraten (*true positive rate*) welche durch Wahl eines Schwellwertes t entstehen können:

$$FPR(t) = \frac{FP(t)}{n_{\text{neg}}}, \quad TPR(t) = \frac{TP(t)}{n_{\text{pos}}}. \quad (18.166)$$

Ein Beispiel für solch eine Kurve ist in Abb. 18.16a zu sehen. Für eine gegebene Mindestanforderung an die Falschpositivrate lässt sich in der Kurve die zugehörige Richtigpositivrate ablesen. Die Bestimmung solcher Kurven kann sehr effizient erfolgen. Ähnlich wie bei der Bestimmung von einfachen Basisklassifikatoren in Abschn. 18.4.3 gibt es nur eine endliche Anzahl von Schwellwerten die entweder die Falschpositivrate oder die Richtigpositivrate ändern, diese sind durch die Anzahl der Testbeispiele beschränkt. Eine ROC-Kurve kann direkt aus der Sortierung der kontinuierlichen Klassifikatorausgaben und den zugehörigen korrekten Klassenangaben ermittelt werden.

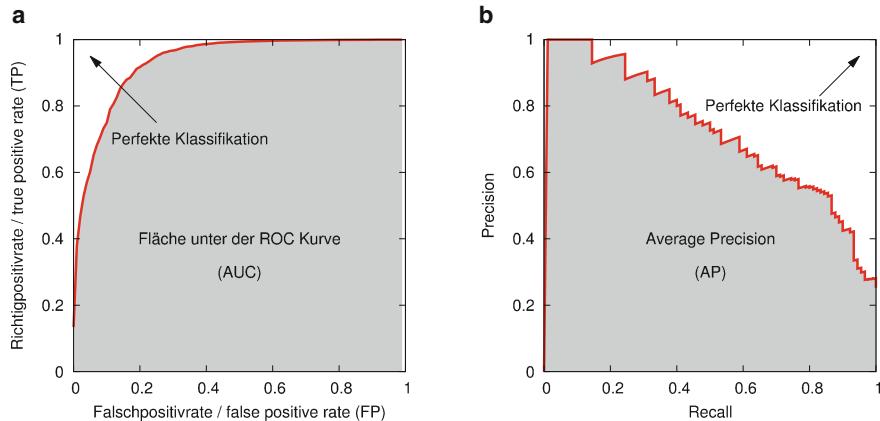


Abb. 18.16 Beispiele für die Auswertung von binären Klassifikationsproblemen mit **a** ROC- und **b** Recall-Precision-Kurven, sowie den Flächeninhalten unter diesen Kurven

Ein ähnliches Werkzeug zur Beurteilung von Klassifikatoren sind Recall-Precision-Kurven:

$$\text{recall}(t) = \text{TPR}(t) = \frac{\text{TP}(t)}{n_{\text{pos}}}, \quad \text{precision}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FP}(t)}. \quad (18.167)$$

Ein Vorteil dieser Kurve ist es, dass die Anzahl der negativen Beispiele in der Stichprobe nicht bekannt sein muss. Dies ist besonders bei der Auswertung von Objektlokalisationsverfahren notwendig (Abschn. 20.3). Ein Beispiel für das charakteristische Muster solch einer Kurve ist in Abb. 18.16b zu sehen. Um ein einzelnes Gütemaß zu berechnen, unabhängig von der Wahl des Schwellwertes, kann die Fläche unter einer ROC Kurve (*area under ROC curve*, AUC) oder unter der Recall-Precision Kurve (*average precision*) verwendet werden.

18.7 Unüberwachte Verfahren und Gruppierung

Im Folgenden wollen wir einen kurzen Abstecher in den Bereich des unüberwachten Lernens machen. Wir werden uns dabei nur auf zwei Grundkonzepte fokussieren, welche im Kap. 19 als Basisalgorithmen notwendig sind.

Im Gegensatz zu den bisher betrachteten überwachten Verfahren, sind beim unüberwachten Lernen meist nur die Eingabedaten $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ gegeben. Die meisten Zielsetzungen beim unüberwachten Lernen reduzieren sich auf eine Gruppierung der Daten, d. h. wir versuchen die gegebenen Daten sinnvoll in M Teilmengen zu zerlegen. Wir werden uns im Folgenden auch genau auf diese Aufgabenstellung beschränken. Die Gruppierung kann als Verallgemeinerung der Bildsegmentierung (Einteilung der Pixel in Regionen) auf beliebige mehrdimensionale Datenpunkte angesehen werden und oft wird auch dafür der Begriff der Quantisierung synonym verwendet.

18.7.1 Gruppierung mit k -Means

Gegeben sei eine Menge von Eingabevektoren $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n \subset \mathcal{U}$ aus der Basismenge \mathcal{U} , welche im Normalfall der D -dimensionale reelle Raum ist. Ziel soll es nun sein, eine Funktion $q : \mathcal{U} \rightarrow \{1, \dots, M\}$ zu finden, welche jedem möglichen Eingabevektor einen von M Gruppen zuweist. Wir werden im Folgenden, für den Begriff Cluster auch synonym für Gruppe verwenden.

Ein Standardverfahren der Gruppierung ist das k -Means-Verfahren. Ausgangspunkt ist, dass jedem der M Cluster ein Vektor $\boldsymbol{\mu}_k \in \mathcal{U}$ als Repräsentant zugewiesen ist. In diesem Sinne, spricht man auch von einem Prototypen. Wir gehen zunächst einmal davon aus, dass diese Prototypen bereits gegeben sind. Die Zuweisung eines Vektors zu einem Cluster kann dann relativ einfach erfolgen, indem wir den Cluster wählen bei dem der Abstand zum Prototypen am kleinsten ist:

$$q(\mathbf{x}) = \operatorname{argmin}_{1 \leq k \leq M} \|\mathbf{x} - \boldsymbol{\mu}_k\|^2. \quad (18.168)$$

Zur Bestimmung der Prototypen gehen wir von einer bereits erfolgten Einteilung der gegebenen Eingabevektoren in Cluster \mathcal{C}_k aus. Eine sinnvolle Wahl eines Prototyps $\boldsymbol{\mu}_k$ wäre dann der Mittelwertvektor eines jeden Clusters \mathcal{C}_k :

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{x}_i. \quad (18.169)$$

Wie man sieht liegt ein typisches „Henne-Ei-Problem“ vor: Die Bestimmung der Clusterzugehörigkeit ist bei gegebenen Prototypen einfach zu realisieren und die Bestimmung der Prototypen ist bei gegebenen Clusterzugehörigkeit nur eine Bestimmung von Mittelwertvektoren. Ein klassisches Konzept bei dieser Art von Problemen ist die iterative Bestimmung der unbekannten Größen. Ausgehend von einer initialen Wahl von Prototypen bestimmen wir die Clusterzugehörigkeit der gegebenen Vektoren, danach erfolgt die Neubestimmung der Prototypen auf Grundlage der vorherigen Einteilung der Daten. Diesen Prozess wiederholen wir solange bis sich die Prototypen nach der Neubestimmung nicht mehr signifikant ändern.

Insgesamt lässt sich dieses Verfahren als stückweise Optimierung der folgenden Fehlerfunktion verstehen:

$$E(\mathbf{D}, \mathbf{V}) = \sum_{i=1}^n \|\mathbf{D}\mathbf{v}_i - \mathbf{x}_i\|^2. \quad (18.170)$$

Die Matrix $\mathbf{D} = [\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M]$ enthält dabei die Prototypen und die Vektoren $\mathbf{v}_i \in \{0, 1\}^M$ sind binäre Vektoren, welche die Zugehörigkeit zu den einzelnen Clustern festlegen. Diese Vektoren sind in der Matrix \mathbf{V} zusammengefasst und wir müssen als Nebenbedingung fordern, dass die Vektoren \mathbf{v}_i genau an einer Stelle eine 1 besitzen um die Clusterzugehörigkeit eindeutig festzulegen. Der Ausdruck $\mathbf{D}\mathbf{v}_i$ entspricht demnach genau dem Prototypen

des aktuellen Clusters von \mathbf{x}_i . Die Fehlerfunktion $E(\mathbf{D}, \mathbf{V})$ misst daher den Quantisierungsfehler der aktuellen Gruppierung gegeben durch (\mathbf{D}, \mathbf{V}) . Da die Optimierung von E gleichzeitig nach \mathbf{D} und \mathbf{V} schwierig ist, erfolgt beim k -Means-Verfahren eine zyklische Optimierung zunächst nach den Clusterzugehörigkeiten \mathbf{V} und dann nach den Prototypen \mathbf{D} .

Hinweise zur Implementierung Die Wahl der initialen Prototypen ist beim k -Means-Verfahren entscheidend. Oft werden zufällig aus der gegebenen Datenmenge M Beispiele als Prototypen ausgewählt. Das k -Means-Verfahren liefert dann einen Quantisierungsfehler der ermittelten Gruppierung und praktisch wählt man oft aus mehreren Durchläufen mit unterschiedlicher Initialisierung die Gruppierung mit dem kleinsten Quantisierungsfehler. Bei der Implementierung muss auch darauf geachtet werden, dass bei der Bestimmung der Clusterzugehörigkeit durchaus leere Cluster entstehen können, welche keine Beispiele aus der gegebenen Datenmenge erhalten. In diesem Fall erfolgt oft ein Neustart des Verfahrens mit einer neuen zufälligen Initialisierung.

18.7.2 Schätzung von Mischverteilungen

Eine weitere Möglichkeit eine Gruppierung vorzunehmen, ist es eine Mischverteilung der Daten zu schätzen. Eine Mischverteilung ist nichts anderes als eine gewichtete Kombination von M Basisverteilungen. Ziel ist es dadurch eine multimodale Verteilung modellieren zu können, bei denen die einzelnen lokalen Modi den Clustern entsprechen. Eine Gaußsche Mischverteilung ist gegeben, wenn wir als Basisverteilung eine multivariate Normalverteilung wählen mit verschiedenen Mittelwertvektoren $\boldsymbol{\mu}_\kappa$ und Kovarianzmatrizen \mathbf{S}_κ :

$$p(\mathbf{x} | (\pi_\kappa, \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa)_{\kappa=1}^M) = \sum_{\kappa=1}^M \pi_\kappa \cdot \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa). \quad (18.171)$$

Damit diese Verteilung valide ist, müssen die Koeffizienten π_κ die Bedingungen einer Multinomialverteilung erfüllen: $\forall \kappa \ 0 \leq \pi_\kappa \leq 1$ und $\sum_\kappa^M \pi_\kappa = 1$. In Abschn. 18.3 hatten wir die Maximum-Likelihood-Schätzung zur Bestimmung von Parametern eines Modells kennengelernt. Dieses Konzept können wir jetzt auch anwenden um ausgehend von einer Menge von Vektoren $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ die Parameter $(\pi_\kappa, \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa)_{\kappa=1}^M$ der Mischverteilung zu schätzen:

$$\underset{(\pi_\kappa, \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa)_{\kappa=1}^M}{\operatorname{argmin}} \sum_{i=1}^n \log p(\mathbf{x}_i | (\pi_\kappa, \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa)_{\kappa=1}^M) \quad (18.172)$$

Im Gegensatz zu den Schätzwerten bei einer einzelnen Normalverteilung (Abschn. 18.4.2) gibt es für Gaußsche Mischverteilungen mit $M > 1$ nur iterative Formeln für die Schätzwer-

te der Parameter. Es ergeben sich Schätzgleichungen, welche von den normierten Dichtewerten $\gamma_{i,\kappa}$ der κ -Komponente für das Beispiel \mathbf{x}_i abhängen:

$$\gamma_\kappa(\mathbf{x}_i) = \frac{\pi_\kappa \cdot \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa)}{\sum_{\kappa'=1}^M \pi_{\kappa'} \cdot \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_{\kappa'}, \mathbf{S}_{\kappa'})}. \quad (18.173)$$

Die Gewichte $\gamma_\kappa(\mathbf{x}_i)$ lassen sich als Wahrscheinlichkeiten für die Zugehörigkeit eines Beispiels \mathbf{x}_i zur Komponente κ der Mischverteilung interpretieren. Mit der Definition der $\gamma_\kappa(\mathbf{x}_i)$ lassen sich die Schätzwerte für die Parameter der Mischverteilung wie folgt berechnen:

$$\pi_\kappa = \frac{1}{n} \sum_{i=1}^n \gamma_\kappa(\mathbf{x}_i), \quad (18.174)$$

$$\boldsymbol{\mu}_\kappa = \frac{1}{n \cdot \pi_\kappa} \sum_{i=1}^n \gamma_\kappa(\mathbf{x}_i) \cdot \mathbf{x}_i, \quad (18.175)$$

$$\mathbf{S}_\kappa = \frac{1}{n \cdot \pi_\kappa} \sum_{i=1}^n \gamma_\kappa(\mathbf{x}_i) \cdot (\mathbf{x}_i - \boldsymbol{\mu}_\kappa)(\mathbf{x}_i - \boldsymbol{\mu}_\kappa)^T. \quad (18.176)$$

In den Schätzgleichungen wird jedes Beispiel mit $\gamma_\kappa(\mathbf{x}_i)$ gewichtet, d. h. ein Beispiel mit einer hohen Wahrscheinlichkeit der Zugehörigkeit zur Komponente κ erhält auch einen starken Einfluss auf die gewichtete Schätzung des Mittelwertvektors. Bei näher Betrachtung erkennen wir wieder ein Henne-Ei-Problem, die Bestimmung der Gewichte $\gamma_\kappa(\mathbf{x}_i)$ in (18.173) erfordert bereits die Parameter der Mischverteilung und die Berechnung der Schätzwerte dieser Parameter in (18.174) bis (18.176) erfordert Gewichte $\gamma_\kappa(\mathbf{x}_i)$. Dieses Problem ist uns schon vom k -Means-Algorithmus bekannt und wir können es daher ähnlich angehen. Zunächst wählen wir initiale Werte für die Parameter der Mischverteilung. Dies lässt sich zum Beispiel realisieren, in dem wir die Kovarianzmatrizen \mathbf{S}_κ auf Einheitsmatrizen setzen, die Mischungskoeffizienten auf eine Gleichverteilung $\pi_\kappa = \frac{1}{M}$ festlegen und die Mittelwertvektoren $\boldsymbol{\mu}_\kappa$ zufällig aus den gegebenen Eingabevektoren wählen. Ausgehend von diesen Festlegungen lassen sich dann die Gewichte $\gamma_\kappa(\mathbf{x}_i)$ in (18.173) bestimmen, ein algorithmischer Schritt welcher oft als *Expectation* bezeichnet wird. Danach erfolgt die Bestimmung der Schätzwerte durch (18.174) bis (18.176) (*Maximization* Schritt) und die anschließende Iteration von *Expectation* und *Maximization* Schritt bis zur Konvergenz des Verfahrens.

Das obige Verfahren ist eine Instanz des *Expectation-Maximization*-Ansatzes (EM) und wir verweisen den interessierten Leser an dieser Stelle auf das Buch von Chris Bishop [3] für eine allgemeine Herleitung des Verfahrens. Abbildung 18.17 zeigt ein paar Beispielergebnisse der Schätzung einer Gaußschen Mischverteilung in einem zweidimensionalen Eingaberaum und mit unterschiedlich vielen Komponenten der Mischverteilung. An den dargestellten Höhenlinien der Verteilung lassen sich auch ganz gut die Mittelwertvektoren der einzelnen Komponenten erkennen. Bei der Anwendung eines GMMs bei der Gruppierung können wir einfach jedem Beispiel \mathbf{x}_* die Komponente (Gruppe, Cluster) κ mit dem

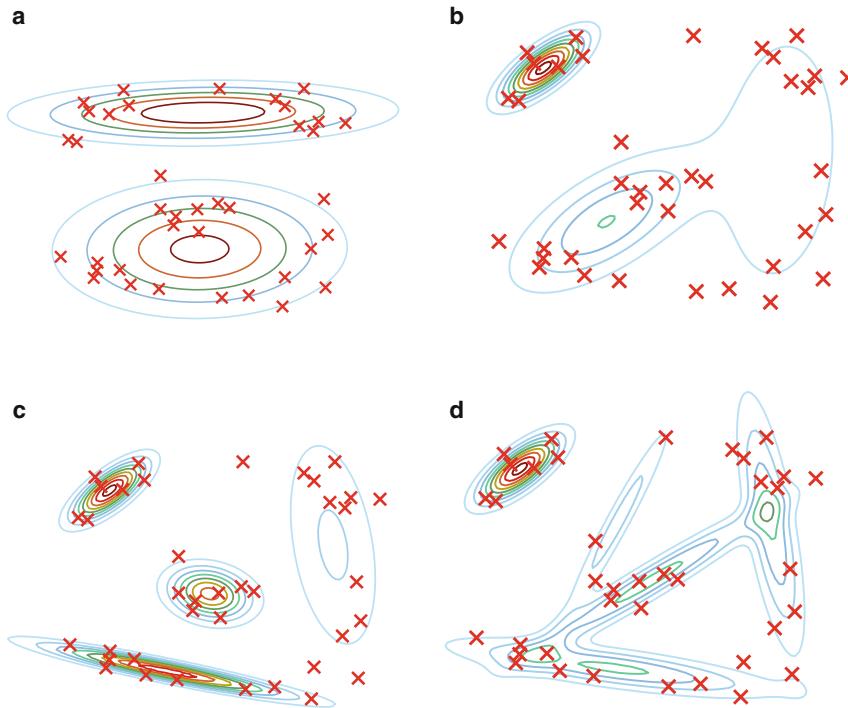


Abb. 18.17 Schätzung einer Gaußschen Mischverteilung mit einer unterschiedlichen Anzahl an Komponenten (**a** $M = 2$, **b** $M = 3$, **c** $M = 4$, **d** $M = 5$) und zweidimensionalen Eingabevektoren

höchsten Wert des Gewichtes $\gamma_\kappa(\mathbf{x}_*)$ zuordnen:

$$\begin{aligned}
 q(\mathbf{x}_*) &= \operatorname{argmax}_{1 \leq \kappa \leq M} \log \gamma_\kappa(\mathbf{x}_*) \\
 &= \operatorname{argmax}_{1 \leq \kappa \leq M} \log(\pi_\kappa \cdot \mathcal{N}(\mathbf{x}_* | \boldsymbol{\mu}_\kappa, \mathbf{S}_\kappa)) \\
 &= \operatorname{argmin}_{1 \leq \kappa \leq M} \left(\log(\det(\mathbf{S}_\kappa)) + (\mathbf{x}_* - \boldsymbol{\mu}_\kappa)^T \mathbf{S}_\kappa^{-1} (\mathbf{x}_* - \boldsymbol{\mu}_\kappa) - \log \pi_\kappa \right).
 \end{aligned} \tag{18.177}$$

Diese Form der Entscheidung mittels der Auswertung der Dichtefunktion einer Normalverteilung haben wir schon in Abschn. 18.4.2 bei Normalverteilungsklassifikatoren kennengelernt. Der zweite Term in obiger Gleichung entspricht dem Mahalanobisabstand zwischen \mathbf{x}_* und $\boldsymbol{\mu}_\kappa$. Wenn wir annehmen, dass alle Kovarianzmatrizen der Einheitsmatrix entsprechen, d. h. $\mathbf{S}_1 = \dots = \mathbf{S}_M = \mathbf{I}$, dann erhalten wir:

$$q(\mathbf{x}_*) = \operatorname{argmin}_{1 \leq \kappa \leq M} \left(\|\mathbf{x}_* - \boldsymbol{\mu}_\kappa\|^2 - \log \pi_\kappa \right). \tag{18.178}$$

Dies entspricht bis auf den zusätzlichen Gewichten π_k , der Wahl des Clusters beim k -Means-Verfahren (siehe (18.168)). Auch das EM-Verfahren zur Parameterbestimmung von Gaußschen Mischverteilungen kann als Verallgemeinerung des k -Means-Verfahrens um folgende Aspekte angesehen werden:

1. Modellierung der Größe, Lage und Ausdehnung eines Clusters in den einzelnen Dimensionen mittels Kovarianzmatrizen,
2. Verwendung „weicher“ Clusterzugehörigkeiten $\gamma_k(\mathbf{x}_i)$ bei der Bestimmung der Parameter anstatt von „harten“ Zuweisungen,
3. Modellierung unterschiedlicher A-Priori-Wahrscheinlichkeiten für die einzelnen Cluster durch die Mischungskoeffizienten π_k .

Das Verfahren eignet sich daher besonders gut bei unterschiedlich großen Clustern und bei Eingabevektoren mit unterschiedlichen Wertebereichen für jede Dimension. In der Praxis werden oft zusätzliche Annahmen verwendet um die Parameterbestimmung zu vereinfachen, wie etwa die Verwendung von Kovarianzmatrizen mit Diagonalgestalt.

Im folgenden Kapitel werden wir uns mit der Berechnung von Merkmalen und Statistiken sowie dem Lösen von Zuordnungsproblemen (Matching) beschäftigen. Ein besonderer Schwerpunkt liegt auf der Untersuchung von Invarianzeigenschaften von Momenten und davon abgeleiteten Größen. Wir werden auch lokale Merkmale kennenlernen, welche besonders für die Anwendung in der Objekterkennung im nächsten Kapitel wichtig sind.

19.1 Momente

Momente sind inzwischen Standardmerkmale in der Bildverarbeitung geworden. Sie werden auch in der Physik und der Stochastik verwendet. In der Bildverarbeitung sind die Hauptanwendungsgebiete:

- Nutzung zu Lage- und Orientierungsbeschreibung von „Freiformobjekten“.
- Sie dienen als Merkmale zur Klassifikation von Objekten. Insbesondere lassen sich daraus Invarianten ableiten.
- Sie dienen zum momentenbasierten Fitting von Freiformobjekten durch elementare geometrische Grundformen, wie z. B. Kreise, Ellipsen, Dreiecke, Parallelogramme, Superellipsen, Kreissegmente, Quadrate und Rechtecke.

Man unterscheidet Momente an Hand der geometrischen Einteilung der Objekte in:

- Flächenmomente von Objekten mit geschlossenen Konturen.
- Linienmomente von Kurvensegmenten.
- Punktmomente von endlichen, diskreten, ungeordneten Punktmengen.

Eine völlig andere Einteilung, die unabhängig von der Geometrie erfolgt, ist folgende:

- orthogonale oder nichtorthogonale Momente.
 - komplexe oder reelle Momente.
 - Momente von Binärobjekten oder Grauwertobjekten oder gar Grauwertbildern.

Betrachten wir schon einmal die Definition der Flächenmomente (19.2), so erkennen wir, dass die Summe $p + q$ die Ordnung der Momente bestimmt. Man sieht sofort in der Übersicht (19.1) wieviele Momente es für jede Ordnung gibt:

In der Praxis benutzt man die Momente höchstens bis zur vierten Ordnung.

Flächenmomente Die Flächenmomente der Ordnung $p + q$ für die Grauwertfunktion $f(x, y)$ eines Bildes B oder eines Objektes B mit einer geschlossenen Kontur sind definiert zu:

$$m_{p,q} = \iint_B x^p y^q f(x,y) dx dy. \quad (19.2)$$

Der Integrationsbereich B ist entweder das gesamte Bild oder eine Untermenge, welches ein Objekt sein kann. Ist B sogar ein Binärobject, dann setzen wir $f(x, y) = 1 \forall x, y \in B$. Bestimmte Momente haben konkrete Bedeutungen:

- $A = m_{0,0}$ ist die Fläche von B , falls $f(x, y) = 1 \forall x, y \in B$.
 - $x_c = \frac{m_{1,0}}{m_{0,0}}$ ist die x -Koordinate des Schwerpunktes von B .
 - $y_c = \frac{m_{0,1}}{m_{0,0}}$ ist die y -Koordinate des Schwerpunktes von B .
 - $m_{2,0}, m_{0,2}$ sind die Trägheitsmomente bezüglich der Koordinatenachsen.
 - $m_{2,0} + m_{0,2}$ ist das polare Trägheitsmoment bezüglich des Koordinatenursprungs.

Wenn man die Grauwertfunktion normiert, d. h. $f'(x, y) = \frac{f(x, y)}{m_{0,0}}$, dann sind die Momente (19.2) mit den Momenten einer Verteilungsfunktion identisch, dann gilt:

- $m_{0,0} = 1$.
 - Der Schwerpunkt ist der Erwartungswert.

- $\sigma_X^2 = m_{2,0}$ und $\sigma_Y^2 = m_{0,2}$ sind die Varianzen.
- $\sigma_{X,Y} = m_{1,1}$ ist die Kovarianz. Die Kovarianzmatrix ist die Matrix der Momente zweiter Ordnung.
- Normiert man diese Kovarianz bezüglich der beiden Varianzen, dann nennt man diese Kovarianz normierter Korrelationskoeffizient.

Speziell für die Flächenmomente bedeutet $m_{0,0}$ die Fläche von Binärobjekten. Die Momente erster Ordnung bestimmen den Schwerpunkt eines Bildes oder Objektes und die Momente zweiter Ordnung beschreiben die sogenannte Trägheitsellipse oder Streuungselipse:

$$m_{0,2}x^2 - 2m_{1,1}x \cdot y + m_{2,0}y^2 = 1. \quad (19.3)$$

Es ist für die Trägheitsellipse unbedingt angeraten, den Koordinatenursprung in den Schwerpunkt zu legen, weil sonst die Orientierung der Trägheitsellipse von Translationen abhängig ist. Nun kann man die Orientierung dieser Ellipse, d. h. den Winkel zwischen der Hauptachse der Ellipse und der x -Achse, als Orientierung des Objektes B verwenden. Dieser Winkel ergibt sich zu:

$$\varphi = \frac{1}{2} \arctan \frac{2m_{1,1}}{m_{0,2} - m_{2,0}}. \quad (19.4)$$

Wenn nun die Trägheitsellipse zum Trägheitskreis zumindestens näherungsweise entartet, dann kann man die Orientierung nicht mehr mit den Momenten zweiter Ordnung beschreiben. Es werden in diesem Falle der Zähler und der Nenner in (19.4) gleichzeitig Null bzw. näherungsweise Null. Als weitere Möglichkeit bietet sich dann an die Momente dritter bis vierter Ordnung zur Orientierung des Objektes heranzuziehen, siehe [71]. Oft herrscht der Irrglaube, dass diese Entartung nur bei symmetrischen Objekten auftreten kann. In Abschn. 19.3 wird gezeigt: jedes Objekt kann durch eine Scherung und eine anisotrope Skalierung derart transformiert werden, dass die Trägheitsellipse zum Trägheitskreis entartet.

Linienmomente Die Liniensegmente sind analog zu den Flächenmomenten definiert, sie beziehen sich aber auf ein Liniensegment L :

$$m_{p,q} = \int_L x^p y^q f(x, y) ds. \quad (19.5)$$

Im Gegensatz zum Flächenintegral (19.2) wird in (19.5) ein Kurvenintegral erster Ordnung bezüglich der Bogenlänge s verwendet.

Punktmomente Die Punktmomente beziehen sich auf eine endliche, diskrete und ungeordnete Punktmenge P :

$$m_{p,q} = \sum_{(x_i, y_i) \in P} x_i^p y_i^q f(x_i, y_i). \quad (19.6)$$

Komplexe, axiale und orthogonale Momente Die Einteilung der Momente in Flächen-, Linien- und Punktmomente erfolgte auf Grundlage der Geometrie der Objekte als Bezugssystem. Man kann auch eine völlig andere Einteilung vornehmen: man bezieht sich auf gedrehte Achsen oder man verwendet komplexe Grauwertfunktionen. Diese Momente müsste man dann wieder unterteilen nach der Art der verwendeten geometrischen Objekte. Wir wollen zunächst als Flächenmomente die sogenannten *komplexen Momente* der Ordnung $p + q$ einführen:

$$C_{p,q} = \iint_B (x + iy)^p (x - iy)^q f(x, y) dx dy. \quad (19.7)$$

Man sieht in (19.7), dass man die komplexen Momente durch „auspotenzieren“ auf die gewöhnlichen Flächenmomente zurückführen kann. Warum benutzt man dann eigentlich komplexe Momente? Weil man dann in sehr übersichtlicher und kompakter Schreibweise formulieren kann, wie sich diese komplexen Momente bei einer Rotation transformieren, siehe Abschn. 19.2.

Als eine weitere Möglichkeit betrachten wir die *axialen Momente*. Die Momente $m_{p,0}$ integrieren die p -te Potenz des Abstandes eines beliebigen Punktes zur y -Achse. Nun kann man analog die p -ten Potenzen des Abstandes einer beliebigen Punktes zu einer beliebigen Geraden durch den Koordinatenursprung integrieren. Die Gerade $x \cos \varphi - y \sin \varphi = 0$ ist die Hessesche Normalform einer Geraden durch den Koordinatenursprung. Setzt man einen Punkt (x, y) ein, so erhält man immer den vorzeichenbehafteten Abstand des Punktes zu dieser Geraden. Deshalb betrachten wir die Abstandsmomente von dieser Geraden:

$$\begin{aligned} A_p(\varphi) &= \iint_B (x \cos \varphi - y \sin \varphi)^p dx dy \\ &= \sum_{k=0}^p (-1)^{p-k} \binom{p}{k} \cos^k \varphi \cdot \sin^{p-k} \varphi \cdot m_{k,p-k}. \end{aligned} \quad (19.8)$$

Substituieren wir in dieser Beziehung $\cos \varphi$ und $\sin \varphi$ mit Hilfe der Eulerschen Beziehung (4.2), so erhalten wir nach Umordnung der Summen

$$A_p(\varphi) = 2^{-p} \sum_{k=0}^p e^{i\varphi(2k-p)} \binom{p}{k} \sum_{l=0}^k \sum_{j=0}^{p-k} (-1)^{p-k-j} \binom{k}{l} \binom{p-k}{j} i^{p-j-l} m_{l+j, p-l-j}. \quad (19.9)$$

Damit können wir die axialen Momente p -ter Ordnung durch die *bandbegrenzte Fourierreihe*

$$A_p(\varphi) = \sum_{k=0}^p \beta_k^{[p]} e^{i\varphi(2k-p)} \quad (19.10)$$

mit den Fourierkoeffizienten

$$\beta_k^{[p]} = \sum_{l=0}^k \sum_{j=0}^{p-k} 2^{-p} (-1)^{p-k-j} \binom{k}{l} \binom{p-k}{j} i^{p-j-l} m_{l+j, p-l-j} \quad (19.11)$$

ausdrücken. Damit haben wir erstmals einen Zusammenhang zwischen Momenten und der Fouriertransformation hergestellt.

Eine völlig andere Einteilung basiert auf der Verwendung der Polynome, die bei der Momentdefinition benutzt werden. Die „gewöhnlichen“ Polynome $P_{p,q} = x^p y^q$ sind nicht orthogonal, daher gibt es auch Momente, die auf orthogonalen Polynomen beruhen. Dies sind hauptsächlich die *Legendre Momente* und die *Zernike Momente*, siehe [48].

Momente und Fouriertransformation Die Momente sind eine bestimmte Repräsentationsform von Information. Das Spektrum ist eine andere Repräsentationsform der gleichen Information. Um dies zu zeigen, benutzen wir die Fourier-Integraltransformation (3.59) für zwei unabhängige Variable und entwickeln den e -Term in eine Potenzreihe:

$$\begin{aligned} \alpha(v_1, v_2) &= \iint_B f(x, y) e^{-2\pi i(v_1 x + v_2 y)} dx dy \\ &= \iint_B f(x, y) \sum_{j=0}^{\infty} \frac{(-2\pi i v_1 x)^j}{j!} \sum_{k=0}^{\infty} \frac{(-2\pi i v_2 y)^k}{k!} dx dy. \end{aligned} \quad (19.12)$$

Wir nehmen einmal an, wir dürfen Integration und Summation vertauschen, dann ist:

$$\alpha(v_1, v_2) = \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} \left(\frac{(-2\pi i)^{j+k}}{j! k!} m_{j,k} \right) \cdot v_1^j v_2^k. \quad (19.13)$$

Folglich haben wir das Spektrum $\alpha(v_1, v_2)$ in eine Potenzreihe entwickelt, wobei die Koeffizienten bis auf einen Faktor genau die Momente der Funktion $f(x, y)$ sind. Damit haben wir das Spektrum vollständig durch die Momente beschrieben. Da das Spektrum aber vollständig das Original $f(x, y)$ beschreibt, haben wir gezeigt, dass dies nur andere Darstellungen der gleichen Information sind. Da das Integralspektrum im Wesentlichen die *charakteristische Funktion* einer Zufallsvariablen darstellt, kann man diesen Sachverhalt auch in der Stochastik finden.

19.2 Transformation der Momente

Wenn wir ein Bild/Objekt B transformieren, dann ändern sich natürlich auch die Momente. Diese könnte man einfach vom transformierten Bild B' berechnen. Wenn man aber die Transformation kennt, dann braucht man nur ausrechnen, wie sich die Momente mit dieser Transformation ändern und braucht sie nicht von B' zu berechnen. Die einfachste Transformation ist die Translation, siehe Abschn. 1.4. Nun kommt es darauf an, welche Momente wir transformieren wollen. Nehmen wir die Flächenmomente, so müssen wir wissen, wie sich Flächenintegrale bei Koordinatentransformtionen transformieren. Wir substituieren einfach und müssen noch die Differentiale umrechnen. Bei Flächenintegralen ändern sich diese mit dem Betrag der Funktionaldeterminante. Diese Determinante ist bei Translatio-
nen Eins. Folglich gilt:

$$m'_{p,q} = \iint_{B'} x'^p y'^q f'(x', y') dx' dy' = \iint_B (x + a_{10})^p (y + a_{20})^q f(x, y) dx dy. \quad (19.14)$$

Für konkrete Momente ergibt sich daher

$$m'_{00} = m_{00}, m'_{10} = m_{10} + a_{10}m_{00}, m'_{01} = m_{01} + a_{01}m_{00}. \quad (19.15)$$

Nun kann man beliebige Transformationen nehmen, z. B. affine Transformationen und die Momente umrechnen. Bei affinen Transformationen ist die Funktionaldeterminante gleich $\det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21}$, deshalb transformiert sich $m_{0,0}$ bei Flächenmomenten nach

$$m'_{0,0} = \det(\mathbf{A})m_{0,0}. \quad (19.16)$$

Linienmomente sind schwieriger umzurechnen als die Flächenmomente, da man die Bogen-differentiale umrechnen muss. Dazu müssen wir die fundamentale Beziehung bezüg-lich der Bogenlänge

$$ds^2 = dx^2 + dy^2 \quad (19.17)$$

benutzen. Für die Translation (1.21) ergibt sich dann:

$$\begin{aligned} m'_{p,q} &= \int_{L'} x'^p y'^q f'(x', y') ds' = \int_L (x + a_{10})^p (y + a_{01})^q f(x, y) \sqrt{dx^2 + dy^2} \\ &= \int_L (x + a_{10})^p (y + a_{01})^q f(x, y) ds. \end{aligned} \quad (19.18)$$

Für Rotationen (1.23) kann man leicht $ds' = ds$ zeigen, sodass bis einschließlich Eukli-dischen Transformationen Flächenmomente und Linienmomente sich in gleicher Weise transformieren. Aber schon bei Ähnlichkeitstransformationen (1.29) treten Unterschie-de auf und zwar bezüglich des Skalierungsfaktors s . Während die Funktionaldeterminante

gleich s^2 ist, transformieren sich die Bogendifferentiale zu $ds' = s \cdot ds$. Folglich transformieren sich die Flächenmomente anders als die Linienmomente im Falle von Ähnlichkeitstransformationen. Benutzen wir sogar affine Transformationen (1.30), dann können wir analytisch die Bogendifferentiale nicht mehr umrechnen, es gibt dafür keine geschlossene Formel mehr.

Am einfachsten ist der Sachverhalt bei den Punktmomenten. Da hier überhaupt kein Differential auftritt, brauchen wir auch kein Differential umrechnen, wir setzen nur die Koordinaten ein und drücken die transformierten Momente durch die originalen Momente aus. Die Punktmomente transformieren sich daher wie die Flächenmomente ohne Funktionaldeterminante. Nehmen wir z. B. Ähnlichkeitstransformationen, so sind folgende Faktoren zu berücksichtigen:

- Bei der Transformation von Flächenmomenten: Faktor s^2 .
- Bei der Transformation von Linienmomenten: Faktor s .
- Bei der Transformation von Punktmomenten: kein Faktor, bzw. s^0 .

Nun betrachten wir noch eine wichtige Transformationsgruppe, die „reinen“ Rotationen, siehe (1.24). Jetzt könnten wir aufschreiben, wie sich die Momente mit reinen Rotationen transformieren. Dies geht aber eleganter, wenn wir die komplexen Flächenmomente (19.7) und die komplexe Form der Rotation (1.25) benutzen. Die Funktionaldeterminante der Rotation ist gleich Eins, so dass sich die komplexen Flächenmomente (19.7) durch

$$C'_{q,p} = C_{p,q} \cdot e^{i(p-q)\varphi} \quad (19.19)$$

transformieren. Nun schreiben wir die Transformation (19.19) einmal konkret bis zu den Momenten fünfter Ordnung auf und führen sie gleichzeitig auf die „gewöhnlichen“ Flächenmomente zurück:

$$C'_{0,0} = m_{0,0}, \quad (19.20)$$

$$\begin{aligned} C'_{1,0} &= (m_{1,0} + im_{0,1}) \cdot e^{i\varphi}, \\ C'_{0,1} &= (m_{1,0} - im_{0,1}) \cdot e^{-i\varphi}, \end{aligned} \quad (19.21)$$

$$\begin{aligned} C'_{2,0} &= (m_{2,0} + 2im_{1,1} - m_{0,2}) \cdot e^{2i\varphi}, \\ C'_{1,1} &= (m_{2,0} + m_{0,2}), \\ C'_{0,2} &= (m_{2,0} - 2im_{1,1} - m_{0,2}) \cdot e^{-2i\varphi}, \end{aligned} \quad (19.22)$$

$$\begin{aligned} C'_{3,0} &= (m_{3,0} + 3im_{2,1} - 3m_{1,2} - im_{0,3}) \cdot e^{3i\varphi}, \\ C'_{2,1} &= (m_{3,0} + im_{2,1} + m_{1,2} + im_{0,3}) \cdot e^{i\varphi}, \\ C'_{1,2} &= (m_{3,0} - im_{2,1} + m_{1,2} - im_{0,3}) \cdot e^{-i\varphi}, \\ C'_{0,3} &= (m_{3,0} - 3im_{2,1} - 3m_{1,2} + im_{0,3}) \cdot e^{-3i\varphi}, \end{aligned} \quad (19.23)$$

$$\begin{aligned} C'_{4,0} &= (m_{4,0} + 4im_{3,1} - 6m_{2,2} - 4im_{1,3} + m_{0,4}) \cdot e^{4i\varphi}, \\ C'_{3,1} &= (m_{4,0} + 2im_{3,1} + 2im_{1,3} - m_{0,4}) \cdot e^{2i\varphi}, \\ C'_{2,2} &= (m_{4,0} + 2m_{2,2} + m_{0,4}), \end{aligned} \quad (19.24)$$

$$\begin{aligned} C'_{1,3} &= (m_{4,0} - 2im_{3,1} - 2im_{1,3} - m_{0,4}) \cdot e^{-2i\varphi}, \\ C'_{0,4} &= (m_{4,0} - 4im_{3,1} - 6m_{2,2} + 4im_{1,3} + m_{0,4}) \cdot e^{-4i\varphi}, \\ C'_{5,0} &= (m_{5,0} + 5im_{4,1} - 10m_{3,2} - 10im_{2,3} + 5m_{1,4} + im_{0,5}) \cdot e^{5i\varphi}, \\ C'_{4,1} &= (m_{5,0} + 3im_{4,1} - 2m_{3,2} + 2im_{2,3} - 3m_{1,4} - im_{0,5}) \cdot e^{3i\varphi}, \\ C'_{3,2} &= (m_{5,0} + im_{4,1} + 2m_{3,2} + 2im_{2,3} + m_{1,4} + im_{0,5}) \cdot e^{i\varphi}, \\ C'_{2,3} &= (m_{5,0} - im_{4,1} + 2m_{3,2} - 2im_{2,3} + m_{1,4} - im_{0,5}) \cdot e^{-i\varphi}, \\ C'_{1,4} &= (m_{5,0} - 3im_{4,1} - 2m_{3,2} - 2im_{2,3} - 3m_{1,4} + im_{0,5}) \cdot e^{-3i\varphi}, \\ C'_{0,5} &= (m_{5,0} - 5im_{4,1} - 10m_{3,2} + 10im_{2,3} + 5m_{1,4} - im_{0,5}) \cdot e^{-5i\varphi}. \end{aligned} \quad (19.25)$$

Für „reine“ Rotationen gelten diese Transformationsformeln auch für die komplexen Linien- und Punktmomente.

19.3 Normalisierung der Momente

Beispiel Schwerpunkt Wenn man Merkmale bezüglich einer Transformation normiert, dann sind die transformierten Merkmale Invarianten bezüglich der Transformation. Ein einfaches Beispiel soll das Grundprinzip demonstrieren. Wir wählen die Flächenmomente (19.2) und die Translation (1.21). Die Momente erster Ordnung transformieren sich dann zu:

$$m'_{1,0} = m_{1,0} + a_{10}m_{0,0}, \quad m'_{0,1} = m_{0,1} + a_{01}m_{0,0}. \quad (19.26)$$

Nun legen wir als Normierung $m'_{1,0} = m'_{0,1} = 0$ fest. Dadurch erhalten wir ein Gleichungssystem mit zwei Gleichungen und zwei Unbekannten

$$0 = m_{1,0} + a_{10}m_{0,0}, \quad 0 = m_{0,1} + a_{01}m_{0,0}. \quad (19.27)$$

Aufgelöst ergibt sich:

$$a_{10} = \frac{m_{1,0}}{m_{0,0}}, \quad a_{01} = \frac{m_{0,1}}{m_{0,0}}. \quad (19.28)$$

Wenn wir mit (19.28) translatieren, verschieben wir den Koordinatenursprung in den Schwerpunkt. Beziehen wir im Folgenden alle Momente auf diesen Koordinatenursprung, dann sind alle Momente *translationsinvariant*. Die bezüglich des Schwerpunktes berechneten Momente heißen *zentrale Momente* und werden oft mit $\mu_{p,q}$ bezeichnet.

Beispiel Trägheitsellipse Ein weiteres einfaches Normierungsbeispiel bieten die Momente zweiter Ordnung. Wir transformieren ein Bild/Objekt B derart, dass die Trägheitsellipse (19.3) zum Trägheitskreis entartet. In der analytischen Geometrie ist das folgende Normierungsprinzip als Hauptachsentransformation von Kurven zweiter Ordnung wohlbekannt. Wir müssen zuerst durch eine Transformation die Normierung

$$m'_{1,1} = 0 \quad (19.29)$$

erreichen. Dazu eignet sich eine Rotation, aber auch eine x -Scherung, siehe (1.26). Daher müssen wir nun wissen, wie sich die Momente bei x -Scherungen transformieren, insbesondere das gemischte Moment zweiter Ordnung $m_{1,1}$:

$$m'_{1,1} = m_{1,1} + s_x m_{0,2} = 0 \rightarrow s_x = -\frac{m_{1,1}}{m_{0,2}}. \quad (19.30)$$

Dies gilt nur für Flächen- und Punktmomente, nicht für Liniennmomente. Mit diesem Scherungspараметer s_x transformieren wir nun alle Flächen- und/oder Punktmomente (in der Praxis aber nur bis zur vierten oder maximal fünften Ordnung) und erhalten die Momente $m'_{p,q}$ mit der Normierung $m'_{1,1} = 0$. Jetzt ist die Trägheitsellipse in achsenparalleler Lage. Nun müssen wir eine anisotrope Skalierung (1.28) so durchführen, dass beide Hauptachsen der Ellipse gleich werden. Dies erreichen wir, wenn

$$m''_{2,0} = 1, \quad m''_{0,2} = 1 \quad (19.31)$$

gilt. Durch die anisotrope Skalierung (1.28) transformieren sich die Momente folgendermaßen, wobei wir gleich die Normierung (19.31) nutzen:

- Flächenmomente

$$\begin{aligned} m''_{2,0} &= c^3 d \cdot m'_{2,0} = 1 \\ m''_{0,2} &= cd^3 \cdot m'_{0,2} = 1. \end{aligned} \quad (19.32)$$

Dieses Gleichungssystem lösen wir und erhalten:

$$c = \sqrt[8]{\frac{m'_{0,2}}{m'^3_{2,0}}}, \quad d = \sqrt[8]{\frac{m'_{2,0}}{m'^3_{0,2}}}. \quad (19.33)$$

- Punktmomente

$$\begin{aligned} m''_{2,0} &= c^2 \cdot m'_{2,0} = 1 \\ m''_{0,2} &= d^2 \cdot m'_{0,2} = 1. \end{aligned} \quad (19.34)$$

Dieses Gleichungssystem lösen wir und erhalten

$$c = \frac{1}{\sqrt{m'_{2,0}}}, \quad d = \frac{1}{\sqrt{m'_{0,2}}}. \quad (19.35)$$

Die Transformation von Linienmomenten führen wir nicht auf, weil sie uns analytisch nicht gelingt. Mit diesen Parametern c und d transformieren wir nun alle Momente $m'_{i,j}$ und erhalten die Momente $m''_{i,j}$. Wenn wir für die Momente $m_{i,j}$ bereits die zentralen Momente nutzen, dann sind alle transformierten Momente $m''_{i,j}$ **invariant** gegenüber Translationen, x -Scherungen und anisotropen Skalierungen. Wenn wir mit allen normierten Teil-Transformationen nicht nur die Momente transformieren, sondern auch das Bild/Objekt B , so befindet sich anschließend B in einer Standardlage B'' , auch *canonical frame* genannt. Die Trägheitsellipse in dieser Standardlage ist ein Trägheitskreis mit dem Radius eins.

Fitting mit der Trägheitsellipse Es sei ein „flächiges“ Binärobject B mit einer geschlossenen Kontur gegeben. Es liegt die Idee nahe, die Trägheitsellipse (19.3) zum Fitten einer Ellipse an B zu nutzen. Direkt und sofort geht das nicht, wir müssen zwei Bedingungen einhalten:

- Die eingehenden Momente in (19.3) sollten die zentralen Momente sein.
- Die Orientierung der Ellipse ist bei Verwendung der zentralen Momente korrekt, die Größe aber nicht. Daher müssen wir die Momente zweiter Ordnung auf die Fläche $m_{0,0}$ normieren. Das heißt, anstelle $m_{2,0}, m_{1,1}, m_{0,2}$ gehen die Momente $m'_{i,j} = m_{i,j}/m_{0,0}$ in (19.3) ein.

Dieses Trägheitsellipsen-Fitting ist sehr robust gegenüber Störungen, kann aber nur auf geschlossene Konturen angewendet werden.

19.4 Kovariante Merkmale, Verfahren und Umgebungen

Wenn man Bilder/Objekte B geometrisch mit T transformiert, erhält man Bilder/Objekte B' . Nun wäre es ideal, wenn sich bestimmte Merkmale M von Punkten, Linien, Kanten oder sonstigen Objekten in B bei der geometrischen Transformation $B' = TB$ ebenso verhalten, dass folglich $M' = TM$ gilt. Falls dies der Fall ist sagt man, dass sich die Merkmale bezüglich T kovariant transformieren. Dazu ein einfaches

Beispiel 1 Ein ideales und typisch affin kovariantes Merkmal ist der Schwerpunkt \mathbf{x}_c eines Bildes oder Objektes B . Wir betrachten dazu eine affine Transformation $\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{a}$, die B in B' überführt. Infolge der affinen Kovarianz des Schwerpunktes können wir folgendes tun: wir bestimmen in B den Schwerpunkt \mathbf{x}_c und völlig unabhängig davon in B' auch den Schwerpunkt \mathbf{x}'_c . Dann muss

$$\mathbf{x}'_c = \mathbf{A}\mathbf{x}_c + \mathbf{a} \quad (19.36)$$

gelten. Für praktische Anwendungen ist es wichtig zu wissen, dass der Schwerpunkt nur affin kovariant ist, leider nicht mehr projektiv kovariant. Ein Beispiel dazu sind Kalibriermuster mit Kreisen. Nimmt man diese mit einer Lochkamera auf, gehen die Kreise im Bild

in Ellipsen über. Die Kreismittelpunkte gehen aber nicht mehr in die Ellipsenmittelpunkte über. Eigentlich sind nur Merkmale kovariant bezüglich einer Transformation. Da aber Merkmale mit irgendwelchen Methoden bestimmt werden müssen und es oft sehr viele Methoden zur Bestimmung der gleichen Merkmale gibt, entsteht folgendes Problem: Wenn das Merkmal eindeutig ist, dann gibt es keinerlei Probleme, weil das Merkmal unabhängig von den Methoden ist. Wenn es nicht eindeutig ist, dann können verschiedene Methoden verschiedene Lösungen für das Merkmal berechnen. Ist das Merkmal aber für eine gewählte Methode eindeutig, dann kann man wieder untersuchen, ob diese auf die Methode bezogene eindeutige Lösung kovariant ist oder nicht. Im Falle der Kovarianz der berechneten Lösung für das Merkmal spricht man von einer kovarianten Methode. In diesem Falle beitzt die Methode die Eigenschaft der Kovarianz. Der Schwerpunkt ist ein schönes Beispiel dafür, dass er immer affin kovariant ist. Da er eindeutig ist, ist es auch uninteressant, welche Methode zu seiner Berechnung verwendet wurde. Für nicht eindeutige Merkmale folgt nun ein

Beispiel 2 Gegeben sei ein Binärobjekt oder eine endliche Punktmenge als Objekt. Dieses Objekt soll durch eine Ellipse beschrieben werden. Das Merkmal für das Objekt ist folglich eine Ellipse. Nun gibt es aber viele Möglichkeiten, Objekte durch Ellipsen zu beschreiben, dies hängt von der gewählten Fittingmethode ab. Dazu seien z. B. im Ausgangsbild n Punkte gegeben und an diese Punkte wollen wir mit einer Methode eine Ellipse E fitten. Als erster Schritt wird das Ausgangsbild mit einer affinen Transformation T transformiert und wir erhalten ein zweites Bild B' . Nun suchen wir im zweiten Bild die affinen Abbilder genau dieser n Punkte. Wir nehmen an, wir hätten sie exakt gefunden. Nun fitten wir mit der gleichen Methode an diese Punkte im Bild B' auch eine Ellipse E' . Wenn nun $E' = TE$ gilt, dann nennen wir die **Fitting-Methode** affin kovariant. Es gibt einige Fittingmethoden für Ellipsen, die nicht einmal kovariant bezüglich Translationen sind. Die Fitting-Methode, die die „Trägheitsellipse“ nutzt, ist affin kovariant. Die Kovarianz der Fittingmethoden von Ellipsen wird ausführlich im Abschn. 21.2.3 untersucht.

Kovariante Umgebungen eines Punktes Wenn wir einem Punkt in einem Grauwertbild signalbasierte Merkmale zuordnen, dann berechnen wir diese bezüglich einer gewählten Umgebung U . Wenn wir die Merkmale dieses Punktes z. B. für die Berechnung von Referenzpunkten nutzen, dann sollte diese Umgebung kovariant bezüglich der gewählten Transformation T sein, d. h. $U' = TU$. Dabei können wir drei grundsätzlich verschiedene Umgebungen U unterscheiden:

- Globale Umgebungen U_g , d. h. als Umgebung wählen wir das gesamte Bild oder Objekt B . Wir legen somit den Koordinatenursprung in den gewählten Punkt und berechnen vom gesamten Bild/Objekt die Merkmale. Dies ist sicher der stabilste und klarste Fall. Er geht aber dann nicht, falls Verdeckungen oder Überschneidungen des Bildes B auftreten. Das gesamte Bild/Objekt muss wieder sichtbar sein.

- Differentielle Umgebungen U_d , d. h. die Umgebung ist unendlich klein. Dies ist das extreme Gegenbeispiel zu den globalen Umgebungen. Wir berechnen Merkmale aus einer differentiell kleinen Umgebung, z. B. ordnen wir einem Punkt einer Kurve seine Krümmung zu. Dies scheint theoretisch immer möglich zu sein, die Merkmale sind aber nicht stabil berechenbar.
- Lokale Umgebungen endlicher Größe U_l , d. h. sie sind nicht global oder differentiell. Manchmal können aber auch globale Umgebungen lokal sein. Haben wir mehrere Objekte im Bild und betrachten einen Punkt eines Objektes, dann ist die globale Umgebung bezüglich dieses Objektes lokal bezüglich des gesamten Bildes. Die Kovarianz der lokalen Umgebungen bereitet erhebliche Probleme. So ist es z. B. einfach bezüglich Euklidischer Transformationen eine kovariante Umgebung zu wählen. Dies ist einfach ein **Kreis** mit einem fest vorgegebenen Radius. Schon bei Ähnlichkeitstransformationen oder gar affinen Transformationen geht dies nicht mehr. Die Umgebungsbestimmung muss sich dann irgendwie am Bildinhalt orientieren, was nicht immer möglich sein wird.

Im Folgenden soll kurz eine Methode angegeben werden, wie man lokale Objekte affin kovariant bestimmen kann. Die Methode heißt **MSER** (Maximally Stable Extremal Region, siehe [45]). Eigentlich wollten für einen vorgegebenen Punkt die affin kovariante Umgebung angeben. Nun gehen wir dual dazu vor: Wir bestimmen zuerst eine affin kovariante Umgebung, d. h. ein am Bildinhalt orientiertes Objekt und erst dann berechnen wir einen Punkt, z. B. den Schwerpunkt des Objektes. Dann ist dem Schwerpunkt als lokale Umgebung das Objekt zugeordnet. Wenn wir Objekte mit einem Konturfolgeverfahren bestimmen, nutzen wir eine globale Schwelle oder lokal adaptiv verschiedene Schwellen. Selbst bei einfachen Beleuchtungsänderungen des Bildes sind diese Methoden nicht mehr affin kovariant gegenüber diesen Beleuchtungsänderungen. Die MSER-Methode garantiert dies aber.

MSER-Methode

- Die Menge aller Pixel eines Bildes B bezeichnen wir mit P und die Grauwertfunktion mit $f(\mathbf{x})$. Die Menge $S(\mathbf{x})$ von Pixeln bezeichnen wir als „**level set**“ und ist

$$S(\mathbf{x}) = \{\mathbf{y} \in P : f(\mathbf{y}) \leq f(\mathbf{x})\}. \quad (19.37)$$

- Bezuglich der Vierer- oder Achternachbarschaft zerfällt die Menge $S(\mathbf{x})$ in verbundene Komponenten bzw. in maximal verbundene Komponenten. Oft heißen maximal verbundene Komponenten einfach nur Komponenten.
- Eine Extremalregion R ist eine maximal verbundene Komponente von $S(\mathbf{x})$.
- Mit $R(f)$ bezeichnen wir die Menge aller Extremalregionen und mit $f(R)$ den größten Grauwert der Extremalregion R

$$f(R) = \sup_{\mathbf{x} \in R} f(\mathbf{x}).$$

- Es sei $\Delta > 0$ ein Parameter des Verfahrens, dann bezeichne

$$R_{+\Delta} = \operatorname{argmin} \{|Q| : Q \in R(f), Q \supset R, f(Q) \geq f(R) + \Delta\},$$

$$R_{-\Delta} = \operatorname{argmax} \{|Q| : Q \in R(f), Q \subset R, f(Q) \leq f(R) - \Delta\}.$$

- Nun betrachten wir die relative Flächenänderung

$$\rho(R; \Delta) = \frac{|R_{+\Delta}| - |R_{-\Delta}|}{|R|}.$$

- Die Extremalregion R ist maximal stabil (*maximally stable*), wenn die relative Flächenänderung $\rho(R; \Delta)$ minimal ist. Das Minimum ist folgendermaßen zu verstehen. Es muss $\rho(R; \Delta)$ kleiner sein als $\rho(Q; \Delta)$ für jede Extremalregion Q , die unmittelbar in R enthalten ist. „Unmittelbar enthalten“ heißt: Für $Q \neq R, Q' \neq R$ und $Q \subset Q' \subset R$ muss $Q = Q'$ folgen. Wir suchen folglich alle lokalen Minima in einer Folge sich unmittelbar enthaltender Extremalregionen.

Auf die Implementierung gehen wir hier nicht ein.

19.5 Invarianten und Momente

19.5.1 Erweiterte Hu-Invarianten (Rotationen)

Zunächst wollen wir Invarianten bezüglich „reiner“ Rotationen herleiten. In der Literatur wird sich dabei stets auf die sieben Hu-Invarianten bezogen, siehe [32]. Bei „reinen“ Rotationen ist es egal ob wir Flächen-, Linien- oder Punktmomente verwenden, da sie sich alle gleichartig transformieren. Dazu betrachten wir (19.20) bis (19.25). Bei drei Ausdrücken tritt der Rotationswinkel nicht auf, somit sind diese Ausdrücke von vornherein Rotationsinvarianten. Ansonsten müssen wir „gesickt“ den Rotationswinkel eliminieren. Diese „gesickten“ Ausdrücke schreiben wir bis zur vierten Ordnung auf. Die Ausdrücke ab fünfter Ordnung kann sich der Leser selbst herleiten, diese werden aber recht umfangreich. Wir wählen die Bezeichnungen H_i derart, dass die Invarianten H_1 bis H_7 mit den bekannten sieben Hu-Invarianten aus der Literatur übereinstimmen, siehe ([32]). Die Invarianten $H_{-1}, H_0, H_8, H_9, H_{10}, H_{11}$ und H_{12} nennen wir jetzt erweiterte Hu-Invarianten:

$$H_{-1} = C'_{0,0} = m_{0,0}, \quad (19.38)$$

$$H_0 = C'_{1,0} \cdot C'_{0,1} = m_{1,0}^2 + m_{0,1}^2, \quad (19.39)$$

$$H_1 = C'_{1,1} = m_{2,0} + m_{0,2}, \quad (19.40)$$

$$H_2 = C'_{2,0} \cdot C'_{0,2} = (m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2, \quad (19.41)$$

$$H_3 = C'_{3,0} \cdot C'_{0,3} = (m_{3,0} - 3m_{1,2})^2 + (3m_{2,1} - m_{0,3})^2, \quad (19.42)$$

$$H_4 = C'_{2,1} \cdot C'_{1,2} = (m_{3,0} + m_{1,2})^2 + (m_{2,1} + m_{0,3})^2, \quad (19.43)$$

$$\begin{aligned}
H_5 &= \frac{1}{2} [C'_{3,0} \cdot (C'_{1,2})^3 + C'_{0,3} \cdot (C'_{2,1})^3] \\
&= (m_{3,0} - 3m_{1,2}) [(m_{3,0} + m_{1,2})^3 - 3(m_{2,1} + m_{0,3})^2(m_{3,0} + m_{1,2})] \\
&\quad + (m_{0,3} - 3m_{2,1}) [(m_{0,3} + m_{2,1})^3 - 3(m_{1,2} + m_{3,0})^2(m_{0,3} + m_{2,1})], \quad (19.44)
\end{aligned}$$

$$\begin{aligned}
H_6 &= \frac{1}{2} [C'_{2,0} \cdot (C'_{1,2})^2 + C'_{0,2} \cdot (C'_{2,1})^2] \\
&= (m_{2,0} - m_{0,2}) [(m_{3,0} + m_{1,2})^2 - (m_{2,1} + m_{0,3})^2] \\
&\quad + 4m_{1,1} [(m_{3,0} + m_{1,2})(m_{2,1} + m_{0,3})], \quad (19.45)
\end{aligned}$$

$$\begin{aligned}
H_7 &= \frac{i}{2} [C'_{3,0} \cdot (C'_{1,2})^3 - C'_{0,3} \cdot (C'_{2,1})^3] \\
&= (m_{3,0} - 3m_{1,2}) [(m_{0,3} + m_{2,1})^3 - 3(m_{1,2} + m_{3,0})^2(m_{0,3} + m_{2,1})] \\
&\quad - (m_{0,3} - 3m_{2,1}) [(m_{3,0} + m_{1,2})^3 - 3(m_{2,1} + m_{0,3})^2(m_{3,0} + m_{1,2})], \quad (19.46)
\end{aligned}$$

$$H_8 = C'_{3,1} \cdot C'_{1,3} = (m_{4,0} - m_{0,4})^2 + 4(m_{3,1} + m_{1,3})^2, \quad (19.47)$$

$$\begin{aligned}
H_9 &= C'_{4,0} \cdot C'_{0,4} \\
&= (m_{4,0} + m_{0,4})^2 + 16(m_{3,1} - m_{1,3})^2 - 12m_{2,2}(m_{4,0} - 3m_{2,2} + m_{0,4}), \quad (19.48)
\end{aligned}$$

$$\begin{aligned}
H_{10} &= \frac{1}{2} [C'_{3,1} \cdot (C'_{0,2}) + C'_{1,3} \cdot (C'_{2,0})] \\
&= (m_{4,0} - m_{0,4})(m_{2,0} - m_{0,2}) + 4m_{1,1}(m_{3,1} + m_{1,3}), \quad (19.49)
\end{aligned}$$

$$H_{11} = C'_{2,2} = m_{4,0} + 2m_{2,2} + m_{0,4}, \quad (19.50)$$

$$\begin{aligned}
H_{12} &= \frac{1}{2} [C'_{4,0} \cdot (C'_{0,2})^2 + C'_{0,4} \cdot (C'_{2,0})^2] \\
&= (m_{4,0} - 6m_{2,2} + m_{0,4}) [(m_{2,0} - m_{0,2})^2 - 4m_{1,1}^2] \\
&\quad - 16 [m_{1,1}(m_{2,0} - m_{0,2})(m_{3,1} - m_{1,3})]. \quad (19.51)
\end{aligned}$$

Es sei nochmals ausdrücklich betont, dass die Ausdrücke H_{-1} , H_0 , $H_1, \dots, H_7, \dots, H_{12}$ nur Invarianten bezüglich „reiner“ Rotationen sind. Da sich Flächen-, Linien- und Punkt-momente bei „reinen“ Rotationen gleichermaßen transformieren, sind diese Ausdrücke immer Invarianten, egal ob Flächen-, Linien- oder Punktmomente in die Invarianten eingehen. Die Invariante H_0 bezüglich der Momente erster Ordnung findet man kaum in der Literatur. Die Begründung liegt darin, dass eigentlich immer Translationen auftreten und diese mit dem Schwerpunkt eliminiert werden. Daher ist diese Invariante, wenn wir die zentralen Momente benutzen, immer Null und daher unbrauchbar. Dies ist aber nicht immer so, es gibt tatsächlich Anwendungen, wo wir die Translation nicht benötigen. Dann wird H_0 eine sehr wichtige Invariante, da sie aus den Momenten erster Ordnung besteht und daher numerisch sehr stabil ist, siehe Abschn. 19.8.4.

19.5.2 Invarianten für Euklidische Transformationen

„Reine“ Rotationen treten in der praktischen Bildverarbeitung kaum auf. Betrachten wir Euklidische Transformationen (1.22), dann kommen zu den Rotationen noch die Trans-

lationen hinzu. Jetzt kommt der Abschn. 19.3 ins Spiel. Wir normalisieren zunächst die Momente bezüglich Translationen, indem wir uns auf den Schwerpunkt beziehen. Folglich benutzen wir für die Rotationsinvarianten $H_{-1}, H_0, H_1, \dots, H_7, \dots, H_{12}$ nur zentrale Momente und schon sind alle Ausdrücke invariant gegenüber Euklidischen Transformationen. Sobald in die Invarianten die zentralen Momente eingehen, ist die Invariante H_0 nicht mehr benutzbar, da sie stets Null ist.

19.5.3 Invarianten für Ähnlichkeitstransformationen

Zu den Euklidischen Transformationen kommen nun noch isotrope Skalierungen $x' = s \cdot x, y' = s \cdot y$ hinzu, siehe auch (1.29). Folglich müssen wir die zentralen Momente noch gegenüber isotropen Skalierungen normalisieren. Welches Moment soll man normalisieren? Wir nehmen die Forderung $m'_{0,0} = 1$ und bestimmen daraus den Faktor s :

- $m'_{0,0} = 1 = s^2 \cdot m_{0,0} \rightarrow s = \frac{1}{\sqrt{m_{0,0}}}$ (Flächenmomente)
- $m'_{0,0} = 1 = s \cdot m_{0,0} \rightarrow s = \frac{1}{m_{0,0}}$ (Linienmomente)
- $m'_{0,0} = 1 = m_{0,0} \rightarrow s = \text{Widerspruch!}$ (Punktmomente) →
- $m'_{2,0} = 1 = s^2 \cdot m_{2,0} \rightarrow s = \frac{1}{\sqrt{m_{2,0}}}$ (Punktnmomente).

Nun transformieren wir alle Momente (bis vierter oder fünfter Ordnung) mit dem berechneten s :

- $m'_{p,q} = s^{p+q+2} m_{p,q}$ (Flächenmomente)
- $m'_{p,q} = s^{p+q+1} m_{p,q}$ (Linienmomente)
- $m'_{p,q} = s^{p+q} m_{p,q}$ (Punktmomente)

Mit diesen normalisierten Momenten berechnen wir nun die Rotationsinvarianten. Man muss beachten, dass Größen aus $H_{-1}, H_0, H_1, \dots, H_7, \dots, H_{12}$, die nur aus normalisierten Momenten bestehen, als Invarianten nicht mehr tauglich sind, da sie konstant über allen Objekten sind.

19.5.4 Invarianten für affine Transformationen

Nun ist es recht leicht affine Invarianten herzuleiten. Dazu nutzen wir die Eigenschaft, dass sich der lineare Anteil (Matrix \mathbf{A} , siehe (1.31)) zerlegen lässt in die Hintereinanderausführung einer:

- x -Scherung, siehe (1.26),
- anisotropen Skalierung, siehe (1.28),
- „reinen“ Rotation, siehe (1.24).

Daher müssen wir die zentralen Momente bezüglich einer x -Scherung und einer anisotropen Skalierung normalisieren und zwar so, dass bei einer folgenden Rotation diese Normalisierung sich nicht ändern darf. Dies gelingt indem wir wie im Abschn. 19.3 die Trägheitsellipse so transformieren, dass sie zum Trägheitskreis mit dem Radius eins wird. Wird nun anschließend das Bild/Objekt B rotiert, so ändert sich am Trägheitskreis nichts, der bleibt auch nach der Rotation noch ein Kreis mit dem Radius eins. Die auf den Trägheitskreis normierten zentralen Momente benutzen wir für die Berechnung der Rotationsinvarianten $H_{-1}, H_0, H_1, \dots, H_7, \dots, H_{12}$, die nun tatsächlich **affin invariant** sind. Allerdings können wir auf Grund der Normierungen die Invarianten H_{-1}, H_0, H_1, H_2 nicht verwenden, da sie für alle Objekte konstant sind. Folglich haben wir neun „echte“, affin invariante Größen, wenn wir die Momente bis vierten Ordnung benutzen. Wie im Abschn. 19.3 ausgeführt, gelingt uns für Linienmomente nicht die Normierung bezüglich einer anisotropen Skalierung auf den Trägheitskreis, dies gelang uns nur für Flächen- und Punktmomente. Daher erhalten wir affine Invarianten nur für „flächige“ Objekte und diskrete Punktmengen, nicht für Liniensegmente. Für Liniensegmente haben wir nur Invarianten bezüglich Ähnlichkeitstransformationen hergeleitet.

19.6 Gestaltsanalyse mit momentenbasiertem Fitting

Gegeben sei die geschlossene Kontur eines planaren Objektes als das Resultat einer Bildsegmentierung. Es sind nun die beiden Aufgaben zu lösen:

- Es sind vom Objekt grundlegende Gestaltsmerkmale zu bestimmen, wie z. B. Elliptizität, Rundheit, Dreieckigkeit, Rechteckigkeit, Konvexität, Kompaktheit … usw. Dabei sollen nur wesentliche Grundformen des Objektes ermittelt werden. In der Bildverarbeitung hat sich dafür der Begriff *Shape Analysis* etabliert, siehe [16]. Die Begriffe Rechteckigkeit, Dreieckigkeit usw. wurden von Rosin eingeführt, siehe z. B. [61].
- Es ist eine robuste, flächenbasierende (d. h. nicht auf der Kontur basierende) Fittingmethode zum anpassen von z. B. Ellipsen, Kreisen, Vierecken, Parallelogrammen, Rechtecken und Superquadriken zu entwickeln.

Dabei soll die Fittingmethode ein normiertes Maß liefern, mit dem die Grundform nach Aufgabe a) eingeschätzt werden kann. Die Fittingmethode soll robust sein und muss daher auf flächenbasierenden Merkmalen beruhen, wie den Flächenmomenten. Konturbasierende Merkmale sind zu störanfällig. Ein einfaches Beispiel dafür ist der Formfaktor, siehe (19.64).

Bereits existierende momentenbasierte Fittingmethoden Momentenbasierte Fittingmethoden für Objekte mit geschlossenen Konturen existieren bereits in der Literatur (siehe [72, 73, 83, 84]) für folgende geometrische Primitiva:

- Kreise und Kreissegmente
- Ellipsen und elliptische Segmente
- Superquadriken bzw. Superellipsen
- Gleichseitige Dreiecke
- Gleichschenklige Dreiecke
- Beliebige Dreiecke
- Quadrate
- Rechtecke
- Parallelogramme
- Beliebige Vierecke und Fünfecke.

Übliche Fittingmethoden, die auf ungeordneten Punktmenzen beruhen, gibt es natürlich für Kreise, Ellipsen und Superquadriken. Auch beliebige Dreiecke und Vierecke können konturbasiert angepasst werden. Eine diskrete Kontur ist weiter nichts als ein Polygon mit vielen Ecken. Es existieren Eckenreduktionsalgorithmen, die auf Split- und Merge Strategien beruhen. Wenn man die Anzahl der Ecken auf drei oder vier reduziert, hat man Dreiecke oder Vierecke angepasst. Flächenbasierte Methoden sind dagegen nicht nur robuster, sie können auch auf einfache Weise Restriktionen an die Dreiecke oder Vierecke realisieren. So können mit den flächenbasierten Momenten relativ einfach Fittingmethoden für Parallelogramme, Quadrate, Rechtecke, gleichseitige Dreiecke, gleichschenklige Dreiecke entwickelt werden, was mit den Eckenerreduktionsalgorithmen nicht ohne Weiteres möglich ist. Natürlich haben auch die flächenbasierten Methoden Nachteile, sie können nur auf Objekte mit geschlossenen Konturen angewendet werden.

In Abb. 19.1a ist die Kontur eines Verkehrszeichens zu sehen, die starke Störungen in der Segmentierung aufweist. In Abb. 19.1c ist eine konturbasierte Anpassung eines Viereckes und in Abb. 19.1b die momentenbasierte Anpassung zu sehen.

Eine momentenbasierte Fittingmethode ist schon im Abschn. 19.3 erläutert worden, nämlich das Fitten von Ellipsen basierend auf der Trägheitsellipse.

Im Folgenden soll die generelle Fittingidee basierend auf der Normierungsmethode und den Flächenmomenten erläutert werden. Zunächst muss zum geometrischen Primitivum solch eine geometrische Transformation gewählt werden, die zum Primitivum auch passt. Wollen wir z. B. Rechtecke fitten, dann sind sicher affine Transformationen ungeeignet, da ein Rechteck affin in ein beliebiges Parallelogramm überführt werden kann. Geeignete Kombinationen von geometrischem Primitivum und geometrischer Transformation (siehe auch Abschn. 1.4.3) sind die folgenden Beispiele:

Wahl: Geometrisches Primitivum ↔ Geometrische Transformation

- Dreieck (6 Parameter) – Affine Transformation (6 Parameter)
- Parallelogramm (6 Parameter) – Affine Transformationen (6 Parameter)
- Viereck (8 Parameter) – Projektive Transformation (8 Parameter)

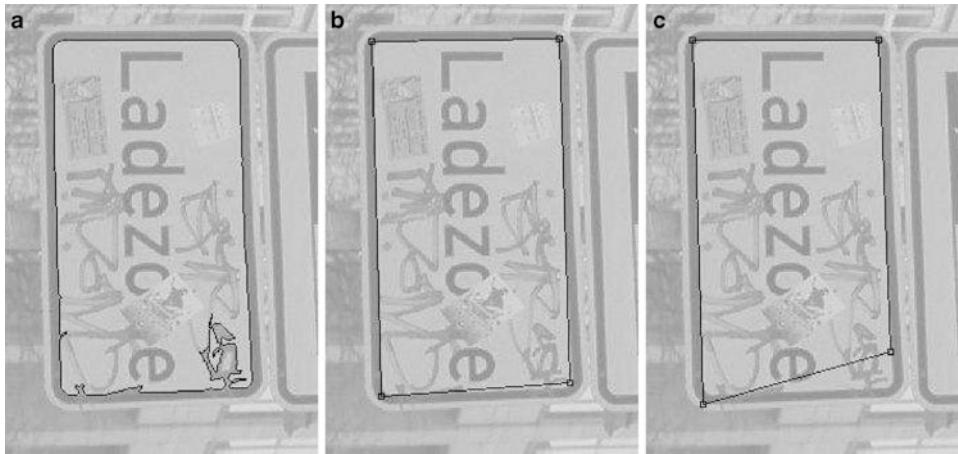


Abb. 19.1 a Kontur eines Verkehrszeichens mit Störungen, b Momentenbasiertes Fitting-Resultat, c Konturbasiertes Fitting-Resultat

- Rechteck (5 Parameter) – Ähnlichkeitstransformationen (4 Parameter)
- Superellipsen (7 Parameter) – Affine Transformationen (6 Parameter)
- Gleichseitiges Dreieck (4 Parameter) – Ähnlichkeitstransformationen (4 Parameter).

Wahl der Normierungsmethode

- Wähle eine geeignete Normierungsmethode bezüglich der Momente aus. Diese sollte auf einer Separation der Transformationsgruppe in Untergruppen beruhen. Für Ähnlichkeitstransformationen (1.29) ist die Separation in eine Translation, Rotation und isotrope Skalierung naheliegend. Für affine Transformationen kann man mehrere Separationen verwenden, siehe (1.32), (1.33) und (1.34).
- Für affine Transformationen wurden zu den Separationen (1.33) und (1.34) spezielle Normierungsstrategien entwickelt, siehe dazu [62], [83] und [84]. So wurden z. B. für die Separation (1.33) die Translation auf den Schwerpunkt normiert $m_{1,0} = m_{0,1} = 0$, die anschließende x -Scherung mit $m_{1,1} = 0$ normiert, die anschließende anisotrope Skalierung mit $m_{2,0} = m_{0,2} = 1$ normiert und die sich anschließende Rotation mit $m_{3,0} + m_{1,2} = 0$ normiert.
- Wichtig bezüglich der Separation ist: eine Untergruppe darf die Normierung der vorhergehenden Untergruppe nicht „zerstören“. Damit ist bei Separation (1.33) und Normierung die Gesamtnormierung

$$m_{1,0} = 0, \quad m_{0,1} = 0, \quad m_{2,0} = 1, \quad m_{1,1} = 0, \quad m_{0,2} = 1, \quad m_{3,0} + m_{1,2} = 0 \quad (19.52)$$

für die affine Transformation auch tatsächlich garantiert.

Nach der Wahl der Normierungsmethode

- Berechne möglichst „mit Bleistift und Papier“ und einmalig die Standardlage *canonical frame primitivum* des geometrischen Primitivums und dessen Momente. Für Rechtecke ist dies z. B. das Einheitsquadrat im Zentrum des Koordinatensystems.
- Das gegebene planare Objekt ist nun numerisch ebenfalls zu normalisieren und liefert eine Transformation T und die normalisierte Lage *canonical frame object*. Das *canonical frame primitivum* ist nun gegebenfalls an das *canonical frame object* anzupassen und liefert das benötigte Gütemaß. Das Anpassen ist nur nötig, wenn die Transformationsgruppe mehr Parameter als das geometrische Primitivum besitzt. Dies ist z. B. bei affinen Transformationen und Superquadriken der Fall. Dann ist ein freier Parameter durch ein eindimensionales Optimierungsproblem zu bestimmen, wobei die Momente im Sinne der kleinsten Quadrate bezüglich beider *canonical frames* zu optimieren sind.
- Nun wird mit der inversen Transformation T^{-1} das *canonical frame primitivum* transformiert und liefert das Fittingergebnis.

Die einzelnen Normierungsmethoden für die verschiedenen geometrischen Primitiva können in [62], [83] und [84] nachgelesen werden. Stellvertretend für alle soll im Folgenden das Fitting von Rechtecken beschrieben werden.

Fitting von Rechtecken Rechtecke besitzen fünf Parameter. Wählen wir Ähnlichkeitstransformationen (vier Parameter), müssen wir noch ein eindimensionales Optimierungsproblem lösen. Wir gehen aber geschickter vor: wenn wir eine Translation (2 Parameter), eine anisotrope Skalierung (2 Parameter) und eine Rotation (1 Parameter) ausführen, so haben wir auch fünf Parameter und brauchen nichts zu optimieren. Die Hintereinanderausführung dieser Transformationen bildet aber insgesamt keine Gruppe und ein Rechteck kann in bestimmten Lagen durch eine anisotrope Skalierung in ein Parallelogramm transformiert werden. Wenn wir aber garantieren können, dass die anisotrope Skalierung nur achsenparallel erfolgt, dann bleibt es auch ein Rechteck. Daher müssen wir die Reihenfolge der Transformationen sehr sorgfältig wählen. Wir wählen folgende Reihenfolge:

- Zuerst wird wie fast immer so translatiert, dass der Koordinatenursprung im Schwerpunkt liegt. Die nun zentralen Momente bezeichnen wir mit $m'_{i,j}$, wobei nun $m'_{1,0} = m'_{0,1} = 0$ gilt.
- Nun ist durch eine Rotationsnormierung das Objekt achsenparallel auszurichten. Dazu normieren wir $m''_{1,1} = 0$ und erhalten den Winkel zwischen einer Achse der Trägheitsellipse und der x -Achse nach (19.4) mit den Momenten $m'_{i,j}$, folglich $\varphi = \frac{1}{2} \arctan \frac{2m'_{1,1}}{m'_{0,2} - m'_{2,0}}$. Es gibt vier Lösungen modulo $\pi/2$, dabei ist es egal, welche Lösung wir nehmen. Mit diesem Winkel transformieren wir nun alle Momente $m'_{i,j}$ und erhalten die Momente $m''_{i,j}$, wobei $m''_{1,0} = m''_{0,1} = 0$, $m''_{1,1} = 0$ infolge der beiden Normierungen gilt.

- Nun normieren wir bezüglich einer anisotropen Skalierung. Durch die achsenparallele Lage kann nun ein Rechteck nicht mehr in ein Parallelogramm übergehen. Die Normierungsbedingung ist $m_{20}''' = m_{02}''' = \frac{1}{12}$. Die beiden anisotropen Skalierungsparameter α, β ergeben sich dann zu $\alpha = \frac{1}{\sqrt{2} \sqrt[4]{3}} \cdot \sqrt[8]{\frac{m_{20}}{m_{02}^2}}$, $\beta = \frac{1}{\sqrt{2} \sqrt[4]{3}} \cdot \sqrt[8]{\frac{m_{02}}{m_{20}^2}}$. Wir müssen nun alle Momente $m_{i,j}''$ mit den beiden Parametern α, β transformieren und erhalten die Momente $m_{i,j}'''$ mit den Normierungsb

$$m_{1,0}''' = m_{0,1}''' = 0, \quad m_{1,1}''' = 0, \quad m_{2,0}''' = m_{0,2}''' = \frac{1}{12}. \quad (19.53)$$

Damit haben wir in einer bestimmten Reihenfolge die Trägheitsellipse auf den Trägheitskreis normiert.

- Wie sieht nun das *canonical frame primitivum* für Rechtecke mit den Normierungsb

$$P_1 = \left(-\frac{1}{2}, +\frac{1}{2} \right), \quad P_2 = \left(+\frac{1}{2}, +\frac{1}{2} \right), \quad P_3 = \left(+\frac{1}{2}, -\frac{1}{2} \right), \quad P_4 = \left(-\frac{1}{2}, -\frac{1}{2} \right). \quad (19.54)$$

- Damit wird die Normierung sehr simpel und es reichen die Momente bis zweiter Ordnung völlig aus, die Momente ab dritter Ordnung benötigen wir nicht.
- Für ein gegebenes planares Objekt führen wir nun genau diese Normierung mit den drei Schritten Translation, Rotation und anisotrope Skalierung aus. Nun invertieren wir diese Transformationen, d. h. zuerst die inverse anisotrope Skalierung, dann die inverse Rotation und dann die inverse Translation und transformieren in dieser Reihefolge das *canonical frame primitivum* (19.54) und erhalten das an das Objekt angepasste Rechteck.
- Ein Problem entsteht nur dann, wenn von vornherein das Objekt nahezu ein Quadrat ist oder überhaupt, wenn die Trägheitsellipse von vornherein näherungsweise ein Trägheitskreis ist. Dann ist die Rotation mit den Momenten zweiter Ordnung nicht mehr stabil normierbar. Wie können wir das feststellen? Eine Möglichkeit ist bei der Normierung die Anwendung des Verhältnisses der beiden anisotropen Skalierungsparameter $ratio = \frac{\alpha}{\beta} = \sqrt{\frac{m_{2,0}''}{m_{0,2}''}}$. Praktische Experimente haben ergeben, dass bei einem Verhältnis $0,9 \leq ratio \leq 1,1$ eine stabile Normierung der Rotation nicht mehr möglich ist. Es ist zu beachten, dass *ratio* erst nach der Rotationsnormalisierung zu berechnen ist. Möchte man von vornherein mit den Momenten des Objektes eine Entscheidung treffen, so muss dieses Maß rotationsinvariant sein. Dazu bietet sich das relative Maß

$$0 \leq m_{\text{rechteck}} = \frac{\sqrt{(m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2}}{(m_{2,0} + m_{0,2})} \leq 1 \quad (19.55)$$

an, welches eine Funktion der beiden Hu-Invarianten H_1 (19.40) und H_2 (19.41) ist. Für Objekte, deren Trägheitsellipse ein Trägheitskreis ist, gilt $m_{\text{rechteck}} = 0$. Für Liniensegmente gilt dagegen $m_{\text{rechteck}} = 1$. Aus praktischen Bewertungen folgt $m_{\text{rechteck}} \leq 0,1$ als

Abb. 19.2 Robustes Rechteck-fitting



Abbruchkriterium. Damit kann man auch das Maß (19.55) anstelle des Formfaktors für das Kriterium „Rundheit“ eines Objektes benutzen. Damit sind allerdings alle Objekte, die einen Trägheitskreis besitzen, von der Grundform her als gleich „rund“ anzusehen.

- Ist das Fitting mit den zweiten Momenten nicht möglich, kann man ein Fittingverfahren mit den Momenten vierter Ordnung anwenden, siehe [72].

In Abb. 19.2 sind von einem Dokument durch den Bildrand Ecken verdeckt, trotzdem liefert das Fitting ein brauchbares Resultat.

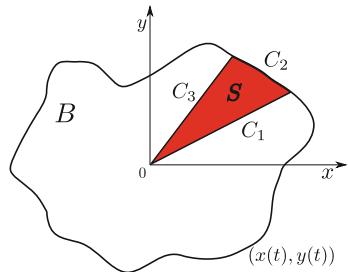
Bei der Auswahl eines geometrischen Primitivums, welches man an ein gegebenes Objekt fitten möchte, sollte man auf „Grundformen“ achten. Diese „Grundformen“ von planaren Objekten kann man grob in Dreier-Symmetrien und Vierer-Symmetrien einteilen, höhere Symmetriarten sind nicht mehr wesentlich. Dreier-Symmetrien sind durch allgemeine Dreiecke oder noch spezieller durch gleichseitige oder gleichschenklige Dreiecke zu beschreiben. Vierer-Symmetrien werden allgemein hervorragend durch Superellipsen beschrieben, spezielle Viersymmetrien sind dann Rechtecke, Parallelogramme usw. Etwas aus dem Rahmen dieser Symmetrieeinteilung fallen dann allerdings Kreissegmente und Ellipsensegmente.

19.7 Effiziente Berechnung von Konturmerkmalen

19.7.1 Elementare Konturmerkmale

Fläche und Umfang eines Objektes Im Folgenden nehmen wir an, es sind diskrete, geschlossene Konturen von Binärobjekten B gegeben, die wir mit dem Konturfolgeverfahren bestimmt haben, siehe Abschn. 10.2. In der Regel ist dann eine Kontur durch eine zyklische

Abb. 19.3 Sektor S mit Randkurve (C_1, C_2, C_3)



Punktfolge $(x_i, y_i), 0 \leq i \leq N - 1$ eindeutig beschrieben, wobei die Punkte bezüglich der Achternachbarschaft zusammenhängen. Die Zuwächse von benachbarten Punkten seien Δx_i bzw. Δy_i . Der Umfang U ist im einfachsten Fall die Anzahl der Randpunkte U_B des Objektes, welche nicht gleich sein muss mit der Anzahl der Konturpunkte. Ein leicht anderes Maß U_N ist die Anzahl aller Punkte der Kontur. Ist die Kontur nicht entartet, dann ist aber tatsächlich $U_N = U_B$. Beide Größen U_B und U_N stellen einen topologischen Umfang dar. Wollen wir einen geometrischen Umfang, müssen wir diese Größen korrigieren. Da die Kontur über die Achternachbarschaft zusammenhängt, sollten wir Diagonalen mit dem Faktor $\sqrt{2}$ korrigieren, sodass für U ein geometrisch brauchbares Maß U_D vorliegt. Alle Maße des Umfangs können simultan beim Konturfolgeverfahren berechnet werden. Wenn wir die Fläche des Objektes auch simultan beim Konturfolgeverfahren berechnen wollen, dann wird es etwas schwieriger. Dazu nehmen wir erst einmal theoretisch an, die Kontur C sei eine analoge, geschlossene Randkurve des Objektes B und durch eine Parameterdarstellung $x(t), y(t), 0 \leq t \leq T$ beschrieben. Weiterhin enthalte das Objekt B keine Löcher. Man kann nun die Fläche F_B des Objektes B durch ein Kurvenintegral bezüglich der Kontur C ausdrücken:

$$F_B = \frac{1}{2} \int_C x dy - y dx. \quad (19.56)$$

Diese Formel für die Fläche nennt man *Leibnizsche Sektorformel*. In der sehr einfachen Herleitung dieser Formel sieht man, dass auch $F_B = \int_C x dy$ und $F_B = -\int_C y dx$ Kurvenintegrale zur Berechnung der Fläche sind. Die Leibnizsche Sektorformel hat aber entscheidende Vorteile gegenüber den beiden anderen Integralen. Dazu legen wir den Koordinatenursprung in das Objekt B und betrachten als Objekt einmal nur den Sektor S mit der Randkurve $C = (C_1, C_2, C_3)$. Die Kurvensegmente C_1 und C_3 stellen Geradensegmente dar, während C_2 ein Segment der Randkurve von B ist, siehe Abb. 19.3. Um die Fläche von S auszurechnen, wenden wir die Sektorformel für die Randkurve (C_1, C_2, C_3) an. Dazu untersuchen wir, welchen Wert die Sektorformel für ein Geradensegment GS ergibt, das durch den Koordinatenursprung geht. Ein Geradensegment GS durch den Koordinatenursprung sei durch $x(t) = K_1 t, y(t) = K_2 t, 0 \leq t \leq T_1$ beschrieben. Damit ergibt sich

$$\frac{1}{2} \int_{GS} x dy - y dx = \frac{1}{2} \int_0^{T_1} (K_1 t K_2 dt - K_2 t K_1 dt) \equiv 0. \quad (19.57)$$

Damit bleibt für die Fläche eines Sektors S nur noch zu berechnen

$$F_S = \frac{1}{2} \int_{C_2} xdy - ydx. \quad (19.58)$$

Diese schöne Eigenschaft hat nur diese symmetrische Sektorformel, daher auch die gewählte Bezeichnung. Als nächstes berechnen wir die Sektorformel für ein beliebiges Geradensegment GS , welches vom Punkt (x_1, y_1) bis zum Punkt (x_2, y_2) geht. Die Zuwächse seien $\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$. Eine einfache Rechnung ergibt:

$$\int_{GS} ydx = y_1\Delta x + \frac{1}{2}\Delta x\Delta y, \quad \int_{GS} xdy = x_1\Delta y + \frac{1}{2}\Delta y\Delta x. \quad (19.59)$$

Damit ist

$$\frac{1}{2} \int_{GS} xdy - ydx = \frac{1}{2} (x_1\Delta y - y_1\Delta x) = \frac{1}{2} (x_1y_2 - y_1x_2). \quad (19.60)$$

Durch die Symmetrisierung der Sektorformel fallen die quadratischen Terme weg. Fassen wir daher eine diskrete Kontur als Polygon auf, wobei die Konturpunkte die Eckpunkte des Polygons sind, dann können wir exakt die Fläche dieses Polygons angeben:

$$\begin{aligned} F_B &= \frac{1}{2} \int_C xdy - ydx = \frac{1}{2} \sum_{i=0}^{N-1} (x_i\Delta y_i - y_i\Delta x_i) \\ &= \frac{1}{2} \sum_{i=0}^{N-1} (x_iy_{i+1} - y_ix_{i+1}). \end{aligned} \quad (19.61)$$

Dies ist der entscheidende Vorteil der symmetrischen Sektorformel gegenüber den nicht symmetrischen Kurvenintegralen bezüglich der Fläche. Wenn wir normalerweise ein Integral durch eine Summe ersetzen und die Differentiale durch Differenzen, dann sind das im allgemeinen nur Näherungen. Für Polygone erhalten wir aber mit der diskreten Sektorformel nicht nur Näherungen, sondern die exakte Fläche des Polygons. Ein weiterer Vorteil der Formel (19.61) ist, dass die Fläche simultan im Konturfolgealgorithmus berechnet werden kann. Als es noch keine digitale Bildverarbeitung gab, existierte eine „analoge Bildverarbeitung“. Eine Aufgabe war es z. B. auf Karten die Fläche von Freiformobjekten mit größtmöglicher Genauigkeit zu bestimmen. Dazu hat man ein mechanisches Gerät entwickelt, genannt **Planimeter**. Man umfährt mit einem Stift die Kontur des Objektes und in einem „Rechenwerk“ des Planimeters kann man die Fläche ablesen. Bezogen auf irgendeinen Koordinatenursprung misst das Planimeter ständig die x, y -Koordinaten inkrementell, d. h. die Zuwächse Δx_i und Δy_i werden mechanisch ermittelt. Sie Summation zur Berechnung der Fläche erfolgt mit der Sektorformel (19.61).

Picksche Formel Wenn wir mit dem Konturfolgeverfahren in Z^2 eine geschlossene Kontur berechnet haben, dann haben wir automatisch die Fläche F und verschiedene Maße für den Umfang zur Verfügung. So ist ein Maß für den topologischen Umfang einfach die Anzahl U_B der Randpunkte des Objektes B , die natürlich alle in Z^2 liegen und ganzzahlig sind. Kann man allein aus diesen Maßen die Anzahl der inneren Punkte I berechnen? Dies geht tatsächlich mit der sogenannten Pickschen Formel (Pick – österreichischer Mathematiker). Dazu müssen wir aber annehmen, dass die Kontur als Polygon aufgefasst kein entartetes Polygon darstellt. Eine geschlossene Kontur, die z. B. nur eine Linie als das eigentliche Objekt beschreibt, ist dann entartet. Die Picksche Formel lautet:

$$F = I + \frac{U_B}{2} - 1 \rightarrow I = F - \frac{U_B}{2} + 1. \quad (19.62)$$

Ein kleines Problem haben wir nun, da man erst feststellen müsste, ob die Kontur entartet ist. Daher müssen wir die Formel speziell für Konturen etwas modifizieren. Es gilt dann

$$F = I + U_B - \frac{U_N}{2} - 1 \rightarrow I = F - U_B + \frac{U_N}{2} + 1. \quad (19.63)$$

Die Anzahl der Randpunkte U_B reicht nun nicht mehr, wir benötigen zusätzlich U_N , dieses Umfangmaß ist die Länge der Kontur als Anzahl von Kanten, wobei die Kontur über die Achternachbarschaft zusammenhängt. Damit ist U_N auch gleichzeitig die Anzahl aller Konturpunkte N . Ist die Kontur nicht entartet, dann gilt $U_N = U_B$ und die Formeln (19.62) und (19.63) sind identisch. Wir können folglich mit sehr einfachen Mitteln stets mit der Pickschen Formel (19.63) die Anzahl der inneren Punkte berechnen. Dies gilt für beliebige geschlossene Konturen, diese können auch entartet sein.

Formfaktor Aus den einfachen Maßen Fläche und Umfang kann der **Formfaktor** f berechnet werden:

$$f = K \cdot \frac{U^2}{F_B}. \quad (19.64)$$

Dabei ist K ein zu wählender Normierungsfaktor. Der Formfaktor hat sein Minimum unter allen Objekten mit konstanter Fläche beim Kreis, im Raum ist dies die Kugel. Kleine Formfaktoren bedeuten demnach kreisförmige Objekte und große Formfaktoren längliche oder zerklüftete Objekte. Allerdings muss man die Konturen vorher glätten, da der Formfaktor sehr rauschanfällig ist. Dazu nehme man einen idealen Kreis, der Formfaktor hat hier sein Minimum. Nun verrausche man leicht die Kontur, wir würden das Objekt aber immer noch als kreisähnliches Objekt einstufen. Da die Fläche näherungsweise gleich bleibt, aber der Umfang steigt und dieser sogar quadratisch in den Formfaktor eingeht, steigt der Formfaktor drastisch an.

Bounding box Ein weitere elementare Beschreibung eines Objektes ist das umschreibende Rechteck, auch *bounding box* genannt. Diese *bounding box* kann simpel während des Konturfolgeverfahrens durch max und min Operationen bezüglich der Konturpunkte berechnet werden. Diese *bounding box* ist folglich achsenparallel und nicht zu verwechseln mit der *minimum bounding box*. Die *minimum bounding box* ist das flächenkleinste Rechteck, das das Objekt enthält und zwar unabhängig von der Rotation, es muss folglich nicht achsenparallel sein. Es ist schon relativ schwierig diese *minimum bounding box* zu berechnen. In der *computational geometry* gibt es dazu die entsprechenden Algorithmen. Da die konvexe Hülle des Objektes die gleiche *minimum bounding box* wie das Objekt selbst besitzt, beziehen sich Algorithmen oft auf die konvexe Hülle. Die *bounding box* wird oft zur Berechnung von Merkmalen des Objektes benutzt, wenn man sich nicht auf die Kontur beziehen kann. Dies kann darin begründet sein, dass ein Objekt aus mehreren Konturen besteht oder Konturen häufig nicht geschlossen sind, siehe auch Abschn. 1.4.5.

19.7.2 Berechnung der Momente

Die Flächenmomente sind Merkmale, die durch Flächenintegrale berechnet werden. Die Flächenintegrale kann man ähnlich der Sektorformel über die Kontur effektiv berechnen. Man bedient sich dabei aus der Analysis bestimmter Integralsätze, z. B. des Greenschen Integralsatzes, auch Gaußscher Integralsatz der Ebene genannt:

$$\begin{aligned} \iint_B P(x, y) dx dy &= \iint_B \left(\frac{\partial N(x, y)}{\partial x} - \frac{\partial M(x, y)}{\partial y} \right) dx dy \\ &= \int_C M(x, y) dx + N(x, y) dy. \end{aligned} \quad (19.65)$$

Die Funktionen $M(x, y)$ und $N(x, y)$ sind geeignet zu wählen. Die Sektorformel ist folglich auch nur ein Spezialfall dieser Greenschen Formel, denn: es sei die Fläche $m_{0,0} = \iint_B dx dy$ zu berechnen, damit ist $P(x, y) \equiv 1$, folglich wählen wir $N(x, y) = \frac{x}{2}$ und $M(x, y) = -\frac{y}{2}$ und es ergibt sich exakt die Leibnizsche Sektorformel (19.3).

Wollen wir nun die Flächenmomente $m_{p,q} = \iint_B x^p y^q dx dy$ berechnen, dann setzen wir mit $P(x, y) = x^p y^q$ die Funktionen M und N zu $M \equiv 0$ und $N(x, y) = \frac{x^{p+1}}{p+1} y^q$. Damit erhalten wir:

$$\iint_B x^p y^q dx dy = \frac{1}{p+1} \int_C x^{p+1} y^q dy. \quad (19.66)$$

Da eine Kontur ein Polygon darstellt, brauchen wir dieses Kurvenintegral nur für Polygone zu berechnen. Von Konturpunkt zu Konturpunkt brauchen wir also nur über ein Geraden-

segment zu integrieren, daher ergibt sich für ein Polygon:

$$\frac{1}{p+1} \int_C x^{p+1} y^q dy = \frac{1}{p+1} \sum_{i=0}^{N-1} \int_0^1 (x_i + t\Delta x_i)^{p+1} (y_i + t\Delta y_i)^q \Delta y_i dt. \quad (19.67)$$

Für Momente bis zur fünften Ordnung rechnen wir nun diese gewöhnlichen Integrale aus und erhalten explizite Formeln zur Berechnung der Momente. Wenn man diese Integrale geschickt auswertet, dann kann man sogar die Momente höherer Ordnung rekursiv auf die Berechnung der Momente niedrigerer Ordnung zurückführen.

Man kann auch noch anders und ganz elementar diese expliziten Formeln in Abhängigkeit von den Konturpunkten herleiten, und zwar durch Triangulation und Berechnung der Momenten-Flächenintegrale über diesen Dreiecken. Dazu betrachten wir ein Koordinatensystem, es ist völlig egal wo der Koordinatenursprung O liegt. Wir legen zunächst einen Umlaufsinn der Kontur fest, der eigentlich schon durch die Aufzählung der Konturpunkte gegeben ist. Nun betrachten wir das Dreieck, welches vom Koordinatenursprung ausgehend, durch die drei Punkte $\{O, (x_i, y_i), (x_{i+1}, y_{i+1})\}$ festgelegt ist. Über diesem Dreieck rechnen wir nun (mit Bleistift und Papier) die Flächenintegrale (19.2) bis zu den Momenten vierter Ordnung aus. Dann summieren wir über alle Dreiecke des Polygons und erhalten:

Nulltes Moment

$$m_{0,0} = \frac{1}{2} \sum_{i=0}^{N-1} (x_i y_{i+1} - x_{i+1} y_i) \quad (\text{identisch mit der Sektorformel}), \quad (19.68)$$

Erste Momente

$$\begin{aligned} m_{1,0} &= \frac{m_{0,0}}{3} \sum_{i=0}^{N-1} (x_i + x_{i+1}), \\ m_{0,1} &= \frac{m_{0,0}}{3} \sum_{i=0}^{N-1} (y_i + y_{i+1}), \end{aligned} \quad (19.69)$$

Zweite Momente

$$\begin{aligned} m_{2,0} &= \frac{m_{0,0}}{6} \sum_{i=0}^{N-1} (x_i^2 + x_i x_{i+1} + x_{i+1}^2), \\ m_{1,1} &= \frac{m_{0,0}}{12} \sum_{i=0}^{N-1} (2x_i y_i + x_{i+1} y_i + x_i y_{i+1} + 2x_{i+1} y_{i+1}), \\ m_{0,2} &= \frac{m_{0,0}}{6} \sum_{i=0}^{N-1} (y_i^2 + y_i y_{i+1} + y_{i+1}^2), \end{aligned} \quad (19.70)$$

Dritte Momente

$$\begin{aligned}
 m_{3,0} &= \frac{m_{0,0}}{10} \sum_{i=0}^{N-1} (x_i^3 + x_i^2 x_{i+1} + x_i x_{i+1}^2 + x_{i+1}^3), \\
 m_{2,1} &= \frac{m_{0,0}}{30} \sum_{i=0}^{N-1} (3x_i^2 + 2x_i x_{i+1} + x_{i+1}^2) y_i + (x_i^2 + 2x_i x_{i+1} + 3x_{i+1}^2) y_{i+1}, \\
 m_{1,2} &= \frac{m_{0,0}}{30} \sum_{i=0}^{N-1} (3y_i^2 + 2y_i y_{i+1} + y_{i+1}^2) x_i + (y_i^2 + 2y_i y_{i+1} + 3y_{i+1}^2) x_{i+1}, \\
 m_{0,3} &= \frac{m_{0,0}}{10} \sum_{i=0}^{N-1} (y_i^3 + y_i^2 y_{i+1} + y_i y_{i+1}^2 + y_{i+1}^3),
 \end{aligned} \tag{19.71}$$

Vierte Momente

$$m_{4,0} = \frac{m_{0,0}}{15} \sum_{i=0}^{N-1} (x_i^4 + x_i^3 x_{i+1} + x_i^2 x_{i+1}^2 + x_i x_{i+1}^3 + x_{i+1}^4), \tag{19.72}$$

$$\begin{aligned}
 m_{3,1} &= \frac{m_{0,0}}{60} \sum_{i=0}^{N-1} (y_i + 4y_{i+1}) x_{i+1}^3 + (2y_i + 3y_{i+1}) x_i x_{i+1}^2 + \\
 &\quad + \frac{m_{0,0}}{60} \sum_{i=0}^{N-1} (3y_i + 2y_{i+1}) x_i^2 x_{i+1} + (4y_i + y_{i+1}) x_i^3,
 \end{aligned} \tag{19.73}$$

$$\begin{aligned}
 m_{2,2} &= \frac{m_{0,0}}{180} \sum_{i=0}^{N-1} (2y_i^2 + 6y_i y_{i+1} + 12y_{i+1}^2) x_{i+1}^2 \\
 &\quad + \frac{m_{0,0}}{180} \sum_{i=0}^{N-1} (6y_i^2 + 8y_i y_{i+1} + 6y_{i+1}^2) x_i x_{i+1} \\
 &\quad + \frac{m_{0,0}}{180} \sum_{i=0}^{N-1} ((12y_i^2 + 6y_i y_{i+1} + 2y_{i+1}^2) x_i^2),
 \end{aligned} \tag{19.74}$$

$$\begin{aligned}
 m_{1,3} &= \frac{m_{0,0}}{60} \sum_{i=0}^{N-1} (x_i + 4x_{i+1}) y_{i+1}^3 + (2x_i + 3x_{i+1}) y_i y_{i+1}^2 \\
 &\quad + \frac{m_{0,0}}{60} \sum_{i=0}^{N-1} (3x_i + 2x_{i+1}) y_i^2 y_{i+1} + (4x_i + x_{i+1}) y_i^3,
 \end{aligned} \tag{19.75}$$

$$m_{0,4} = \frac{m_{0,0}}{15} \sum_{i=0}^{N-1} (y_i^4 + y_i^3 y_{i+1} + y_i^2 y_{i+1}^2 + y_i y_{i+1}^3 + y_{i+1}^4). \tag{19.76}$$

19.8 Matching

Matching kann man auch als Mustervergleich bezeichnen, d. h. es findet ein direkter Vergleich des Musters mit einem Referenzmuster statt. Für den Vergleich spielen somit das Matchingmaß und die geometrische Transformation die entscheidende Rolle. Weiterhin

kann dies lokal oder global im Bild erfolgen. Matching hängt auch direkt zusammen mit der sogenannten *Bildregistrierung*, die insbesondere in der medizinischen Bildverarbeitung große Bedeutung besitzt.

19.8.1 Matchingmaße

Da Matching immer auf einem Vergleich von Muster und Referenzmuster basiert, benötigt man Maße, die die Ähnlichkeit beider Muster numerisch quantifizieren. Diese Maße sind manchmal „echte“ Metriken, häufig aber nur Abstandsmaße oder völlig andere Maße wie Korrelationsmaße. Es bezeichne \mathbf{x} und \mathbf{y} Merkmalsvektoren von Referenzmuster und Muster. Die drei wichtigsten „echten“ Metriken sind dann:

$$\rho_1(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i| \quad (\text{city block metric}), \quad (19.77)$$

$$\rho_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2} \quad (\text{Euklidische Metrik}), \quad (19.78)$$

$$\rho_\infty(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i| \quad (\text{chessboard metric}). \quad (19.79)$$

Haben wir z. B. binäre Merkmalsvektoren (z. B. Objekte in Binärbildern), dann wird die *city block metric* (19.77) *Hammingdistanz* genannt. Die Hammingdistanz zählt die Anzahl der „Positionen“ (z. B. Pixel) an denen sich die Muster unterscheiden. Die Metriken können wir auch durch die Normen ausdrücken $\rho_i(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_i$. Insbesondere zur Euklidischen Metrik gibt es eine Reihe von gewichteten Modifikationen. Wenn z. B. die Merkmalsvektoren aus Merkmalen bestehen, die völlig verschiedenen Ursprungs sind, dann versagt die Euklidische Metrik als geeignetes Matchingmaß. Dann sollte man die *Mahalanobis-Metrik*

$$\rho_M^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \Sigma_{\mathbf{x}, \mathbf{y}}^{-1} (\mathbf{x} - \mathbf{y}) \quad (19.80)$$

verwenden. Dabei ist $\Sigma_{\mathbf{x}, \mathbf{y}}$ die auf den Erwartungswert zentrierte Kovarianzmatrix aus der Grundgesamtheit aller \mathbf{y} und \mathbf{x} . Sind die Merkmale dekorreliert, d. h. die Kovarianzen gleich Null, dann degeneriert die Kovarianzmatrix zur reinen Diagonalmatrix und es wird (19.80) zu:

$$\rho_M^2(\mathbf{x}, \mathbf{y}) = \sum_i \frac{(x_i - y_i)^2}{\sigma_y^2}. \quad (19.81)$$

Man sieht daran, Merkmale die mehr steuern, werden in der Metrik weniger gewichtet. In der Bildverarbeitung haben wir oft endliche Punktmengen zu vergleichen, man spricht dann vom *Point Pattern Matching* (PPM). Welche Maße sind dazu geeignet? Zu diesem Zwecke betrachten wir zwei endliche Punktmengen A und B , die unterschiedliche Anzahlen von Punkten haben können, $\text{card}(A) = N_A$ und $\text{card}(B) = N_B$. Wir verwenden eine

Norm und führen den gerichteten Abstand eines Punktes $a \in A$ von der Punktmenge B ein:

$$d(a, B) = \min_{b \in B} \|a - b\|_i, \quad (19.82)$$

wobei wir in der Regel die Euklidische Norm mit $i = 2$ benutzen. Als gerichteter Abstand der Menge A von der Menge B sind geeignet:

$$\begin{aligned} d_1(A, B) &= \min_{a \in A} d(a, B) \\ d_2(A, B) &= \text{Rangordnung}_p d(a, B) \\ d_3(A, B) &= \max_{a \in A} d(a, B) \\ d_4(A, B) &= \frac{1}{N_A} \sum_{a \in A} d(a, B). \end{aligned} \quad (19.83)$$

Diese gerichteten Maße werden typischerweise verwendet, wenn die Punktmenge A in der Punktmenge B enthalten ist. Beschreiben beide Punktmengen bis auf eine Transformation inhaltlich das gleiche Objekt, dann müssen aus den gerichteten Abstandsmaßen noch ungerichtete abgeleitet werden:

$$\begin{aligned} u_1(A, B) &= \min(d_i(A, B), d_i(B, A)) \\ u_2(A, B) &= \max(d_i(A, B), d_i(B, A)) \\ u_3(A, B) &= \frac{d_i(A, B) + d_i(B, A)}{2} \\ u_4(A, B) &= \frac{N_A d_i(A, B) + N_B d_i(B, A)}{N_A + N_B}. \end{aligned} \quad (19.84)$$

Die u_j können nun mit den d_i kombiniert werden. So ist die Kombination von u_2 mit d_3 die bekannte *Hausdorff-Metrik*, die auch tatsächlich eine „echte“ Metrik darstellt, alle anderen Kombinationen ergeben Ähnlichkeitsmaße, aber keine „echten“ Metriken. Die Hausdorff-Metrik ist für praktische Anwendungen jedoch nicht geeignet, da sie auf Grund der max-Operationen zu sensitiv auf Ausreißer reagiert. In praktischen Experimenten hat sich gezeigt, dass die Kombination von u_2 mit d_3 die beste Wahl ist:

$$d(A, B) = \max \left(\frac{1}{N_A} \sum_{a \in A} d(a, B), \frac{1}{N_B} \sum_{b \in B} d(b, A) \right). \quad (19.85)$$

Des Weiteren basieren Matchingmaße oft auf Korrelationsmaßen, wobei diese nur auf bestimmte Merkmalsvektoren anwendbar sind (z. B. zwei Grauwertblöcke gleicher Größe im Bild). Stochastisch betrachtet sind die Komponenten der Merkmalsvektoren Realisierungen einer einzigen Zufallsvariablen. Allgemeine Grundlage ist die *Cauchy-Schwarzsche*

Ungleichung. Es sei $\langle \cdot, \cdot \rangle$ ein Skalarprodukt und $\|\cdot\|$ die durch das Skalarprodukt induzierte Norm, dann gilt für reelle und komplexe Vektorräume:

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y}. \quad (19.86)$$

Damit folgt für reellwertige Vektorräume:

$$-1 \leq \rho_{\mathbf{x}, \mathbf{y}} = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} \leq +1 \quad \forall \mathbf{x}, \mathbf{y}. \quad (19.87)$$

Bei komplexen Vektorräumen müssen wir etwas aufpassen, dann können wir nur schreiben

$$0 \leq \rho_{\mathbf{x}, \mathbf{y}} = \frac{|\langle \mathbf{x}, \mathbf{y} \rangle|}{\|\mathbf{x}\| \|\mathbf{y}\|} \leq +1 \quad \forall \mathbf{x}, \mathbf{y}. \quad (19.88)$$

Die Korrelation komplexer Signale wird relativ selten benutzt, ein Beispiel ist die Korrelation zweier komplexer Spektren, siehe z. B. Abschn. 4.10. Im Folgenden beschränken wir uns auf den reellen Fall. Manchmal tritt auch noch eine „additive“ Normierung auf. Dazu bezeichne \bar{x} das arithmetische Mittel der Komponenten von \mathbf{x} und \mathbf{e} einen Vektor aus lauter Einsen, dann gilt auch:

$$-1 \leq \rho_{\mathbf{x}, \mathbf{y}} = \frac{\langle \mathbf{x} - \bar{x}\mathbf{e}, \mathbf{y} - \bar{y}\mathbf{e} \rangle}{\|\mathbf{x} - \bar{x}\mathbf{e}\| \cdot \|\mathbf{y} - \bar{y}\mathbf{e}\|} \leq +1 \quad \forall \mathbf{x}, \mathbf{y}. \quad (19.89)$$

Dieses Maß heißt nun der *normierte Korrelationskoeffizient* und besitzt einen entscheidenden Vorteil: Er ist invariant gegenüber linearen Transformationen der Komponenten von \mathbf{x} :

$$x'_i = c \cdot x_i + d. \quad (19.90)$$

Man sieht, beim Einsetzen in (19.89) kürzen sich die Konstanten c und d . Diese Konstanten c und d haben je nach Anwendung unterschiedliche Bedeutung. Nehmen wir einmal als Komponenten der Vektoren die Grautöne $x_{i,j}$ eines Bildes \mathbf{x} , dann lautet der normierte Korrelationskoeffizient:

$$\rho_{\mathbf{x}, \mathbf{y}} = \frac{\sum_{i,j} (x_{i,j} - \bar{x})(y_{i,j} - \bar{y})}{\sqrt{\sum_{i,j} (x_{i,j} - \bar{x})^2} \cdot \sqrt{\sum_{i,j} (y_{i,j} - \bar{y})^2}}. \quad (19.91)$$

Jetzt bedeutet die Invarianz gegenüber c und d die Invarianz gegenüber linearen Kontrast- und Helligkeitsänderungen der Grauwerte.

19.8.2 Histogrammbasierte Matchingmaße

In der Bildverarbeitung werden oft Histogramme zur Bewertung von Objekten herangezogen. Daher müssen diese Histogramme verglichen werden. Dazu setzen wir zunächst

voraus, dass die Histogramme die gleiche Klasseneinteilung besitzen. Diejenigen Abstandsmaße werden als *bin-by-bin* Maße bezeichnet, in deren Berechnung das Merkmal i des ersten Histogrammes nur in Relation zum Merkmal i des zweiten Histogrammes betrachtet wird. Es seien H und K zwei Histogramme mit den relativen Häufigkeiten h_i und k_i , dann sind

- Minkowski-Abstand:

$$d_r(H, K) = \left(\sum_i |h_i - k_i|^r \right)^{\frac{1}{r}}, \quad (19.92)$$

- Histogramm-Schnitt:

$$d_{\cap}(H, K) = 1 - \frac{\sum_i \min(h_i, k_i)}{\sum_i k_i}, \quad (19.93)$$

- Kullback-Leibler-Divergenz:

$$d_{\text{KL}}(H, K) = \sum_i h_i \log \frac{h_i}{k_i}, \quad (19.94)$$

- χ^2 -Statistik:

$$d_{\chi^2}(H, K) = \sum_i h_i \log \frac{(h_i - m_i)^2}{m_i}, m_i = \frac{(h_i + k_i)}{2} \quad (19.95)$$

bin-by-bin Maße. Im Gegensatz zu den *bin-by-bin* Maßen werden auch sogenannte *cross-bin* Maße benutzt. Diese nutzen auch die Informationen benachbarter Werte.

- Quadratische-Form-Abstand:

$$d_A(H, K) = \sqrt{(\mathbf{h} - \mathbf{k})^T \mathbf{A} (\mathbf{h} - \mathbf{k})}. \quad (19.96)$$

Die *cross-bin* Information ist in der Matrix \mathbf{A} enthalten, wobei $a_{i,j}$ die Ähnlichkeit zwischen den Merkmalen i und j ausdrückt. Ein einfaches Beispiel dafür ist $a_{i,j} = 1 - \frac{d_{i,j}}{d_{\max}}$. Dabei ist $d_{i,j}$ der Abstand der Merkmale i und j , und $d_{\max} = \max(d_{i,j})$.

- Match-Abstand:

$$d_M(H, K) = \sum_i |\hat{h}_i - \hat{k}_i|, \quad (19.97)$$

wobei $\hat{h}_i = \sum_{k \leq i} h_k$ die Summenfunktion, bzw. was dasselbe ist, die Verteilungsfunktion von H ist. Der Match-Abstand ist folglich der Minkowski-Abstand mit ($r = 1$), angewendet auf die Verteilungsfunktion. Weiterhin ist dieses Maß ein Spezialfall der Earth Mover's Distance (EMD) für eindimensionale Histogramme, siehe auch Abschn. 19.8.3.

- Kolmogorov-Smirnov-Abstand:

$$d_M(H, K) = \max_i |\hat{h}_i - \hat{k}_i|. \quad (19.98)$$

Dieser bezieht sich ebenfalls auf die Verteilungsfunktion, allerdings mit dem Minkowski-Abstand für ($r = \infty$).

19.8.3 Signaturen

Das Hauptproblem bei Histogrammen ist eine günstige Klasseneinteilung, dies ist eine Balance zwischen Informationsgehalt und Effektivität. Daher verallgemeinert man Histogramme zu Signaturen und betrachtet gleichzeitig den mehrdimensionalen Fall. Eine Signatur $P = [(\mathbf{m}_j, w_j)]_j$ repräsentiert eine Menge von Clustern mit Gewichten. Dabei ist der d -dimensionale Vektor \mathbf{m}_j ein Repräsentant des Clusters j und w_j beschreibt als ein dimensionales Maß die Ausdehnung des Clusters. Histogramme sind ein Spezialfall von Signaturen, weil dann \mathbf{m}_j der Mittelpunkt der Klasse j (*bin j*) ist und w_j die relative Häufigkeit in der Klasse j (*bin j*) darstellt. Wenn wir einen sinnvollen Abstand zwischen Signaturen angeben können, dann haben wir auch gleichzeitig einen sinnvollen Abstand zwischen mehrdimensionalen Histogrammen. Als brauchbarer Abstand gilt die *Earth Mover's Distance (EMD)*. Dabei stelle man sich zwei Signaturen P und Q vor. Die Cluster von P stellen wir uns als „Erdhaufen“ vor und die Cluster von Q sollen Löcher sein. Die Erdhaufen von P sollen nun in die Löcher von Q transportiert werden. Es darf nicht mehr transportiert werden als Erdhaufen da sind oder die Löcher von Q aufnehmen können. Dabei entstehen Transportkosten. Die minimalen Transportkosten stellen den Abstand zwischen P und Q dar. Folglich sind $P = [(\mathbf{p}_1, w_{\mathbf{p}_1}), \dots, (\mathbf{p}_m, w_{\mathbf{p}_m})]$ und $Q = [(\mathbf{q}_1, w_{\mathbf{q}_1}), \dots, (\mathbf{q}_n, w_{\mathbf{q}_n})]$ die Cluster. Weiterhin sei $\mathbf{D} = (d_{i,j})$ die Entfernungsmatrix der Cluster \mathbf{p}_i und \mathbf{q}_j . Analog beschreibe die Matrix $\mathbf{F} = (f_{i,j})$ die transportierte Menge zwischen den Clustern \mathbf{p}_i und \mathbf{q}_j . Dann lösen wir das Optimierungsproblem

$$\hat{z}(P, Q, \mathbf{F}) = \min z(P, Q, \mathbf{F}) = \min \sum_{i=1}^m \sum_{j=1}^n d_{i,j} f_{i,j}, \quad (19.99)$$

mit den Restriktionen

$$f_{i,j} \geq 0, \quad 1 \leq i \leq m, 1 \leq j \leq n, \quad (19.100)$$

$$\sum_{j=1}^n f_{i,j} \leq w_{\mathbf{p}_i}, \quad 1 \leq i \leq m, \quad (19.101)$$

$$\sum_{i=1}^m f_{i,j} \leq w_{\mathbf{q}_j}, \quad 1 \leq j \leq n, \quad (19.102)$$

$$\sum_{i=1}^m \sum_{j=1}^n f_{i,j} = \min \left(\sum_{i=1}^m w_{\mathbf{p}_i}, \sum_{j=1}^n w_{\mathbf{q}_j} \right). \quad (19.103)$$

Die Bedingung (19.103) berücksichtigt, dass die beiden Signaturen nicht normalisiert sein brauchen. Bei Histogrammen wäre das Minimum überflüssig. Als Abstand $EMD(P, Q)$ benutzen wir dann:

$$EMD(P, Q) = \frac{\sum_{i=1}^m \sum_{j=1}^n d_{i,j} f_{i,j}}{\sum_{i=1}^m \sum_{j=1}^n f_{i,j}}. \quad (19.104)$$

Die Normalisierung ist nur dann nötig, wenn die Signaturen selbst nicht normalisiert sind. Numerisch lösen wir dieses Transportproblem mit einer speziellen Version der Simplexmethode, die speziell diese Datenstruktur berücksichtigt, siehe auch Abschn. 21.2.5. Die EMD bietet eine Reihe von Vorteilen, insbesondere den Vorteil eines partiellen Matchings bei sich überlappenden Daten. Wenn die Abstände $d_{i,j}$ selbst Metriken sind und die Summen der Gewichte ausgeglichen sind, dann ist die EMD selbst eine „echte“ Metrik. Typische Anwendungen der EMD findet man in *image retrieval*-Systemen, in der Texturanalyse und beim Vergleich von Farbhistogrammen. Ein typisches Problem ist dabei die hohe Dimensionalität. Nehmen wir als Beispiel den Vergleich zweier Farbbilder. Die RGB-Farben werden dazu zunächst in den CIELAB-Farbraum umgerechnet, siehe auch Abschn. 11.3. Wenn wir z. B. jede Achse des dreidimensionalen CIELAB-Würfels in je 10 *bins* einteilen, so haben wir pro Würfel schon 1000 *bins*. Die Kostenmatrix $\mathbf{D} = (d_{i,j})$ ist dann immerhin eine $(1000, 1000)$ -dimensionale Matrix. Folglich müssen wir die Farbhistogramme zu Signaturen reduzieren. Dazu werden effiziente, bekannte Clusteralgorithmen benutzt. Für die Nutzung der EMD ist auch eine sorgfältige Wahl des Entfernungsabstandes $d_{i,j}$ notwendig. So wird bei den Farbhistogrammen oder Farbsignaturen häufig das Maß

$$d_{i,j} = 1 - e^{-\alpha \|\mathbf{p}_i - \mathbf{q}_j\|} \quad (19.105)$$

benutzt. Der Faktor $\alpha = \|(\sigma_1, \dots, \sigma_d)^T\|$ ist eine Norm des Vektors der Standardabweichungen über alle Farben und alle Klassen.

19.8.4 Matching und Registrierung

Mit den eingeführten Matchingmaßen kann nicht direkt entschieden werden, ob zwei Objekte/Bilder/Punktmengen usw. ähnlich sind oder nicht. Die Objekte sind in der Regel aus verschiedenen Positionen aufgenommen worden oder anderweitig deformiert worden. Daher müssen diese Objekte vorher mit Transformationen angepasst werden. Dies bezeichnet man als *Bildregistrierung* oder schlechthin als *Registrierung*. Dazu benötigt man aber A-priori-Informationen über die Deformation/Transformation. Dazu ein einfaches Beispiel, das sogenannte **Template-Matching** in Grautonbildern. Es soll überprüft werden, ob ein gegebener Bildblock B in einem „großen“ Bild I enthalten ist oder nicht. Weiterhin wissen wir a priori, der Bildblock B kann nur in verschobenen Positionen im Bild I vorhandensein. Er kommt nicht rotiert, skaliert und auch nicht gescherzt vor. Wir verwenden folglich als geometrische Transformation nur die Translation.

- Wir legen zuerst das Ähnlichkeitsmaß fest. Wir verwenden z. B. die normierte Korrelation (19.91), um Invarianz gegenüber linearen Beleuchtungsänderungen zu garantieren.
- Wir müssen nun über dem Parameterraum der Translationen diejenige Translation finden, die das beste Ähnlichkeitsmaß erzeugt. Wir haben folglich eine Optimierungsaufgabe über dem Raum der gewählten Transformationen zu lösen. Nach Möglichkeit sollte das globale Optimum gefunden werden.
- In der Praxis wird man aus Effizienzgründen den Suchraum einschränken, eventuell hierarchisch vorgehen und die Matchingmaß-Berechnung optimieren.
- Man könnte z. B. mit groben Translationsschritten den Suchraum abrastern und bei der Position des kleinsten Matchingmaßes eine lokale Suche starten, z. B. den koordinatenweisen Abstieg.
- Da die Norm des Templates B unabhängig von der Suche ist, braucht diese im Maß nicht berechnet zu werden. Dies steigert die Effizienz.
- Hat man den Ort des Templates B im Bild I lokalisiert, muss man zusätzlich prüfen, ob das Template auch tatsächlich an dieser Stelle ist. Dazu muss man eine Schwellwertentscheidung mit dem Matchingmaß durchführen oder zusätzliche Entscheidungen heranziehen.

Da die elementaren Transformationen, insbesondere die Translationen eine große Rolle spielen, soll kurz auf die signalbasierte Bestimmung dieser elementaren Transformationen eingegangen werden.

Elementare Transformationen In den folgenden Abschnitten soll auf rein signalbasierte Methoden eingegangen werden. Ein wichtiges Problem ist dabei folgendes: Die Bilder sind diskrete Funktionen, es gibt nur Funktionswerte an den Pixelpositionen. Bei Translationen dürfte man also nur ganzzahlige Verschiebungen zulassen und Rotationen wären überhaupt nicht definiert. Aber dennoch registrieren wir diese, auch im Subpixelbereich. Dazu wird oft die Interpolationsidee angewendet. Wenn ein Bild nicht unterabgetastet ist, dann kann die analoge Funktion durch Interpolation vollständig rekonstruiert werden und die Registrierung mittels Interpolation ist theoretisch in Ordnung. Liegt aber eine Unterabtastung vor, dann wird es problematisch. Im Abschn. 5.1 haben wir Bedingungen angegeben, wann dann noch etwas zu „retten“ ist. Wenn die Unterabtastung nicht zu „extrem“ ist, dann müssen wir die Bilder mit einem Tiefpass filtern und können trotzdem noch exakt im Subpixelbereich registrieren. Überschreitet die Unterabtastung aber eine gewisse Schranke, dann ist nichts mehr zu machen, dann kann man nicht mehr registrieren. Die wichtigste Transformation ist wohl die Translation zweier Bilder, Blöcke, Signale usw. Viele andere Transformationsbestimmungen werden sogar auf die Bestimmung von Translationen zurückgeführt, z. B. ist die Rotation eine Translation in Polarkoordinaten. Einige Methoden in diesem Abschnitt bestimmen theoretisch den verschobenen Einheitsimpuls. Da aber bei gestörten Bildern kein idealer verschobener Einheitsimpuls berechnet wird, sondern eine Funktion, die „hoffentlich“ ein ausgeprägtes deutliches Maximum besitzt, benötigen wir ein Maß, das die Abweichung vom idealen verschobenen Einheitsimpuls beschreibt.

Solch ein „Gütemaß“ des Maximums bestimmt folglich die „Sicherheit“ der Bestimmung der Verschiebung. Dieses Maximum wollen wir **Peak** nennen. Daher sollte man sich einen empirischen Algorithmus überlegen, der die Güte eines solchen Peaks berechnet. Dieses Gütemaß kann dann bei komplexeren Algorithmen als Gewicht für die berechnete Verschiebung benutzt werden. Ein einfaches Beispiel dazu ist das sogenannte **Blockmatching**. In jeder Blockposition wird eine Verschiebung und das zugehörige Gütemaß berechnet. Durch die Blöcke erhalten wir nun eine Liste von lokalen Verschiebungen mit den entsprechenden Gewichten. Aus dieser Liste können mit einer gewichteten Ausgleichsrechnung komplexere Transformationen für das gesamte Bild berechnet werden. Nun folgen einige signalbasierte Methoden zur Transformationsbestimmung:

Anwendung des Verschiebungstheorems Eine naheliegende Methode zur signalbasierten Translationsbestimmung ist die Anwendung des Verschiebungstheorems, siehe Abschn. 4.17. Wir schreiben dies für diskrete, zweidimensionale Bilder der Dimension M, N auf:

$$\alpha_{k,l}(S_{m,n}f) = \alpha_{k,l}(f) \cdot e^{-2\pi i(m\frac{k}{M} + l\frac{n}{N})}. \quad (19.106)$$

Da das Amplitudenspektrum translationsinvariant ist, können wir die Translation aus obiger Beziehung nur über die Phase ausrechnen. Dazu benötigen wir nur zwei Koeffizienten, um m und n auszurechnen. Wir müssen dazu zwei Gleichungen mit zwei Unbekannten lösen. Wählen wir speziell $k = 1, l = 0$ und $k = 0, l = 1$, so erhalten wir sogar zwei entkoppelte Gleichungen und können sofort nach den Verschiebungen m und n auflösen. Wir schreiben also das Verschiebungstheorem für diese beiden Koeffizienten bezüglich der Phase auf und beachten die Periodizität des komplexen e -Terms. Wenn wir mit $\varphi_{k,l}(f)$ die Phase von $\alpha_{k,l}(f)$ bezeichnen, so erhalten wir für die Verschiebungen:

$$\begin{aligned} m &= \frac{(\varphi_{1,0}(f) - \varphi_{1,0}(S_{m,n}f)) \cdot M}{2\pi} + M \cdot g_m, \\ n &= \frac{(\varphi_{0,1}(f) - \varphi_{0,1}(S_{m,n}f)) \cdot N}{2\pi} + N \cdot g_n. \end{aligned} \quad (19.107)$$

Die Notation der Periode ist sehr wichtig. Wählen wir z. B. $k = 2, l = 0$, dann ist m nur mit der Periode $\frac{M}{2}$ bestimmbar. Bei höheren Frequenzen könnten wir folglich nur kleine Verschiebungen berechnen. Daher ist die Nutzung der niederen Frequenzen doppelt vorteilhaft: die niederen Frequenzen beschreiben das globale Signal besser und robuster und gleichzeitig schränkt die Periode die Größe der Verschiebung nicht ein. Da wir nur zwei Fourierkoeffizienten benötigen, brauchen wir auch nicht die beiden Bilder vollständig zu transformieren.

Inverse Faltung Statt dem Verschiebungstheorem wenden wir die Faltung bezüglich der Verschiebung an:

$$f' = S_n f = S_n(f * \delta) = f * S_n \delta \rightarrow \alpha_k(S_n \delta) = C \cdot \frac{\alpha_k(f')}{\alpha_k(f)}. \quad (19.108)$$

Wir bestimmen also den verschobenen Einheitsimpuls mit dieser einfachen Faltungsinversion. Wir tun so, als ob es keine Fehler gäbe und es eine reine zyklische Verschiebung wäre. Bei praktischen Aufgaben (außer in Spezialfällen) ist die Verschiebung nicht zyklisch, im Gegenteil: Bildteile verschwinden aus dem Bild und neue Bildteile erscheinen. Daher muss es bei der inversen Faltung Fehler geben. Wir müssen deshalb die Berechnung des verschobenen Einheitsimpulses als *Restaurationsproblem* auffassen. In (19.108) haben wir als eine Restaurationsmethode das Invers Filter (siehe Abschn. 9.5.1) benutzt. Dabei werden Rauschen und andere Fehler bei der Restauration verstärkt und die Methode wird praktisch unbrauchbar. Nur wenn das Amplitudenspektrum von f nahezu konstant ist, können wir mit dem Invers-Filter stabil die Verschiebung bestimmen.

Kreuzkorrelation Die Kreuzkorrelation ist nur ein Synonym für das Wort Korrelation, das wir im Abschn. 2.1.2 eingeführt haben:

$$(f \circ g)_n = \sum_i f_{n+i} \overline{g_i}, (f \circ g)_{m,n} = \sum_i \sum_j f_{m+i, n+j} \overline{g_{i,j}} \quad (19.109)$$

$$(f \circ g)(t) = \int_B f(t + \xi) \overline{g(\xi)} d\xi, (f \circ g)(t_1, t_2) = \iint_B f(t_1 + \xi, t_2 + \eta) \overline{g(\xi, \eta)} d\xi d\eta. \quad (19.110)$$

Diese berechnen wir für zwei Bilder oder Bild-Blöcke, die sich nur durch eine Translation unterscheiden dürfen. Von der Kreuzkorrelation berechnen wir dann die Koordinaten des Maximums, die die Verschiebung repräsentieren. Dadurch erhalten wir die Verschiebung nur mit Pixelgenauigkeit, für subpixelgenaue Berechnungen wird auf Abschn. 19.8.4 verwiesen. Numerisch berechnen wir die Kreuzkorrelation mittels der FFT, da sich das Spektrum analog Abschn. 4.9 und (4.33) berechnet. Da wir gleichzeitig die Spektren zweier Bilder benötigen, können wir zusätzlich noch den Trick aus Abschn. 4.21.1 verwenden und somit die Kreuzkorrelation effektiv implementieren.

Phasenkorrelation Die Phasenkorrelation ist auch eine sehr robuste Methode zur Translationsbestimmung. Sie wurde bereits 1975 von Kuglin und Hines [37] eingeführt. Wir nehmen wieder an, dass $f' = S_n f$ die Verschiebung von f ist. Vom Verschiebungstheorem $\alpha_k(f') = \alpha_k(f) e^{-2\pi i n \frac{k}{N}}$ wissen wir, dass das Amplitudenspektrum invariant ist, d. h. $|\alpha_k(f')| = |\alpha_k(f)|$. Weiterhin nutzen wir wieder die Faltungsbeziehung bezüglich der Translation:

$$f' = S_n f = S_n(f * \delta) = f * S_n \delta \rightarrow \alpha_k(S_n \delta) = C \cdot \frac{\alpha_k(f')}{\alpha_k(f)}. \quad (19.111)$$

Nun erweitern wir Zähler und Nenner und nutzen die Invarianz des Amplitudenspektrums:

$$\begin{aligned}\alpha_k(S_n \delta) &= C \cdot \frac{\alpha_k(f')}{\alpha_k(f)} = C \cdot \frac{\alpha_k(f') \overline{\alpha_k(f)}}{\alpha_k(f) \overline{\alpha_k(f)}} = C \cdot \frac{\alpha_k(f') \overline{\alpha_k(f)}}{|\alpha_k(f)|^2} \\ &= C \cdot \frac{\alpha_k(f') \overline{\alpha_k(f)}}{|\alpha_k(f')| |\alpha_k(f)|}.\end{aligned}\quad (19.112)$$

Damit erhalten wir für die Phasenkorrelation:

$$\alpha_k(S_n \delta) = C \cdot \frac{\alpha_k(f') \overline{\alpha_k(f)}}{|\alpha_k(f')| |\alpha_k(f)|}. \quad (19.113)$$

Auf der linken Seite steht das Spektrum des verschobenen Einheitsimpulses. Der Zähler auf der rechten Seite ist das Spektrum der Kreuzkorrelation von f' mit f . Der Nenner ist der Betrag des Zählers, damit abstrahieren wir von der Amplitude der Kreuzkorrelation und nutzen nur noch die Phase, daher der Name Phasenkorrelation. Das Abstrahieren von der Amplitude in dieser Form hatten wir in Abschn. 4.20 als „whitening“, also als „weiß“-machen eines Bildes bezeichnet. Die Phasenkorrelation ist demzufolge nichts anderes als *whitening* der Kreuzkorrelation. Wenn wir noch zusätzlich photometrische Schwankungen als lineares Modell annehmen, d. h. $f'' = a \cdot f' + b$ mit $\alpha_k(f'') = a \cdot \alpha_k(f')$, $k \neq 0$, $a > 0$, so folgt:

$$\alpha_k(S_n \delta) = C \cdot \frac{\alpha_k(f') \overline{\alpha_k(f)}}{|\alpha_k(f')| |\alpha_k(f)|} = C \cdot \frac{\alpha_k(f'') \overline{\alpha_k(f)}}{|\alpha_k(f'')| |\alpha_k(f)|}, \quad k \neq 0. \quad (19.114)$$

Da der nullte Fourierkoeffizient nur für die Helligkeit des verschobenen Einheitsimpulses zuständig ist, ist die Phasenkorrelation invariant gegenüber linearen Kontrast- und Helligkeitsänderungen. In Abb. 19.4 ist links das Originalbild, in der Mitte das verschobene Bild und rechts das Ergebnis der Phasenkorrelation zu sehen. Der deutlich zu sehende Peak, der die Verschiebung bestimmt, wurde durch einen Kreis markiert. Trotz deutlicher Störungen bezüglich der Verschiebung kann der Peak sicher gefunden werden, man braucht dazu nicht einmal eine Fensterfunktion um die Randeffekte zu lindern.

SDR-Methode Die Methode *Shift Detection by Restoration* fasst die Inverse Faltung (19.108) als Restaurationsproblem auf:

$$f' = f * S_n \delta + N, \quad (19.115)$$

wobei N Rauschen und alle anderen Störungen und Fehler beschreiben soll. In der klassischen Restauration beschreibt N im Allgemeinen nur das klassische Rauschen. Als Restaurationsmethode nehmen wir die Methode *Restauration unter Zwang*, siehe Abschn. 9.5.2.

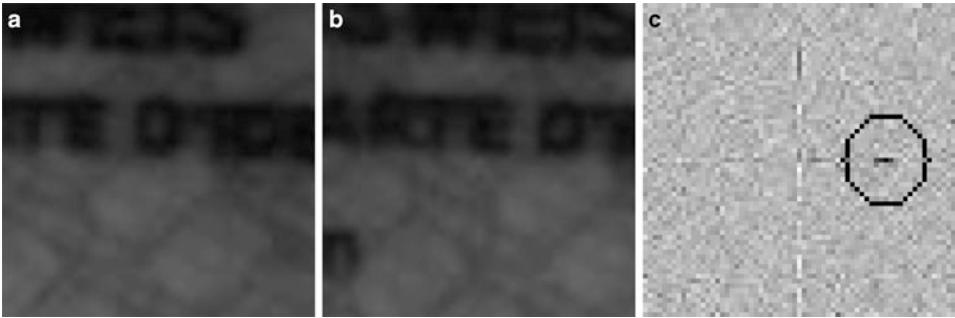


Abb. 19.4 **a** Originalbild, **b** Nach rechts verschobenes Bild, **c** Ergebnis der Phasenkorrelation

Daher können wir sofort für den restaurierten, verschobenen Einheitsimpuls (9.63) benutzen. Wir schreiben diese noch einmal für (19.115) auf:

$$\alpha_k(S_n\delta) \approx C \cdot \frac{\alpha_k(f') \cdot \overline{\alpha_k(f)}}{|\alpha_k(f)|^2 + \beta}. \quad (19.116)$$

Die Größe β beschreibt die „Stärke“ der Fehler und muss empirisch festgelegt werden, sie ist aber sehr robust und eine Größe von 0,01 hat sich in vielen Anwendungen als robust und stabil erwiesen. Allerdings hängt dieser Term β auch von der Bilddimension ab und sollte deshalb immer empirisch überprüft werden. Wir können β als eine Art „Stellschraube“ oder Regularisierungsparameter interpretieren. Für $\beta \rightarrow 0$ geht (19.116) in die inverse Faltung (19.108) über. Für $\beta \rightarrow \infty$ geht (19.116) in die Kreuzkorrelation über, siehe Abschn. 4.9. Daher ist die SDR-Methode auffassbar als regularisierte inverse Faltung und regularisierte Kreuzkorrelation. Man kann den verschobenen Einheitsimpuls aus (19.115) auch mit den anderen Restaurationsmethoden berechnen, siehe Abschn. 9.5.

Cepstrum-Methode Im Abschn. 4.22 wurde das komplexe *Cepstrum* $c(f)$ und das *power cepstrum* $pc(f)$ eingeführt. Wir gehen jetzt davon aus, dass nicht ein Bild f und dessen Verschiebung f' gegeben sind, sondern nur deren Überlagerung $g = f + f' = f' + f$. Und nur aus der Überlagerung g und sonst nichts wollen wir die Verschiebung bestimmen. Dazu bilden wir

$$g = f + f' = f + S_n f = f * \delta + S_n f * \delta = f * (\delta + S_n \delta). \quad (19.117)$$

Wir können nun keine inverse Faltung anwenden, da wir f nicht kennen. Wenn wir aber auf der rechten Seite eine additive Überlagerung von f und $\delta + S_n \delta$ erreichen würden, dann würde der verschobene Einheitsimpuls bestimmbar sein, egal wie f aussieht, da dieser sich in jedem Bild abheben müsste. Also müssen wir „irgendwie“ aus der Faltung eine Addition machen. Mit dem Faltungstheorem wird die Faltung eine Multiplikation, logarithmieren

wir diese nun, dann wird aus der Multiplikation tatsächlich eine Addition. Wir beschränken uns in der Herleitung wieder auf das Bildmodell D1[N]. Folglich gilt nach Anwendung des Faltungstheorems auf (19.117):

$$\alpha_k(g) = \sqrt{N} \alpha_k(f) \cdot \left(\frac{1}{\sqrt{N}} + \frac{1}{\sqrt{N}} e^{-2\pi i n \frac{k}{N}} \right). \quad (19.118)$$

Jetzt müssen wir die Multiplikation beseitigen indem wir logarithmieren:

$$\log \alpha_k(g) = \log \alpha_k(f) + \log \left(1 + e^{-2\pi i n \frac{k}{N}} \right). \quad (19.119)$$

Man beachte: die Logarithmen beziehen sich auf komplexe Größen. Das logarithmierte Spektrum in (19.119) lässt sich nicht so einfach zurücktransformieren. Es fällt aber auf, dass der zweite Logarithmus die Form $\log(1+z)$ hat. Dafür benutzen wir nun die klassische Entwicklung in eine komplexe Potenzreihe:

$$\log(1+z) = \sum_{m=1}^{\infty} (-1)^{m+1} \frac{z^m}{m}, \quad (19.120)$$

die für $|z| \leq 1$ konvergiert. Damit ist:

$$\begin{aligned} \alpha_l^{-1}(\log(1 + e^{-2\pi i n \frac{k}{N}})) &= \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi i n m \frac{j}{N}} \cdot e^{+2\pi i l \frac{j}{N}} \\ &= \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{+2\pi i (l-n \cdot m) \frac{j}{N}}. \end{aligned} \quad (19.121)$$

Bevor wir diese Reihe genauer betrachten erinnern wir uns an:

$$\begin{aligned} \alpha_l(\delta) &= \frac{1}{\sqrt{N}}, \quad \delta_l = \frac{1}{N} \sum_{j=0}^{N-1} e^{+2\pi i l \frac{j}{N}} \\ \rightarrow (S_n \delta)_l &= \delta_{l-n} = \frac{1}{N} \sum_{j=0}^{N-1} e^{+2\pi i (l-n) \frac{j}{N}}. \end{aligned} \quad (19.122)$$

Daraus folgt nun:

$$c(g) = \alpha_l^{-1}(\log \alpha(g)) = \alpha_l^{-1}(\log \alpha(f)) + \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \sqrt{N} \delta_{l-n \cdot m}. \quad (19.123)$$

Nun gilt aber ebenso:

$$g = S_{-n} f' + f' = f' * (\delta + S_{-n} \delta) \quad (19.124)$$

und damit:

$$c(g) = \alpha_l^{-1}(\log \alpha(g)) = \alpha_l^{-1}(\log \alpha(f')) + \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \sqrt{N} \delta_{l+n \cdot m}. \quad (19.125)$$

Wenn wir also das komplexe Cepstrum $c(g)$ der überlagerten Bilder berechnen, erhalten wir „irgendein“ Bild, das mit einer abfallenden Folge von verschobenen Einheitsimpulsen überlagert ist. Diese liegen symmetrisch zum Koordinatenursprung. Wir bestimmen also den „ersten“ Peak nahe des Koordinatenursprungs, dessen Koordinaten beschreiben die absolute Verschiebung. Da wir komplexe Logarithmen berechnen müssen, interessiert die Frage, ob dies auch mit dem *power cepstrum* $pc(g)$ funktioniert? Dazu quadrieren wir die Gleichung:

$$\alpha_k(g) = \alpha_k(f) \cdot (1 + e^{-2\pi i n \frac{k}{N}}) \quad (19.126)$$

und erhalten:

$$\log |\alpha_k(g)|^2 = \log |\alpha_k(f)|^2 + \log |(1 + e^{-2\pi i n \frac{k}{N}})|^2. \quad (19.127)$$

Jetzt benutzen wir die einfache Beziehung:

$$\log |1+z|^2 = \log (1+z)(1+\bar{z}) = \log (1+z) + \log (1+\bar{z}), \quad (19.128)$$

womit wir diese Form auf die obige des komplexen Cepstrums zurückgeführt haben. Damit erhalten wir für die Rücktransformation die Summe zweier Reihen

$$\begin{aligned} \alpha_l^{-1}(\log |\alpha(g)|^2) &= \alpha_l^{-1}(\log |\alpha(f)|^2) + \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \sqrt{N} \delta_{l-n \cdot m} \\ &\quad + \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \sqrt{N} \delta_{l+n \cdot m}. \end{aligned} \quad (19.129)$$

Damit könnten wir eigentlich schon aufhören, da wir tatsächlich ein ähnliches Ergebnis erhalten haben. Wir können aber auch formal noch das endgültige *power cepstrum* bilden und die linke und rechte Seite quadrieren und erhalten formal

$$\begin{aligned} pc(g) &= \left| \alpha_l^{-1}(\log |\alpha(g)|^2) \right|^2 = \left| \alpha_l^{-1}(\log |\alpha(f)|^2) \right|^2 \\ &\quad + \sum_{m=1}^{\infty} a_m \delta_{l-n \cdot m} + \sum_{m=1}^{\infty} b_m \delta_{l+n \cdot m}, \quad a_m, b_m \in R. \end{aligned} \quad (19.130)$$

Wir können also auch das „vollständige“ *power cepstrum* benutzen.

Translationen und Subpixel-Genauigkeit Bei vielen Methoden zur Translationsbestimmung haben wir zum Schluss das Spektrum zurücktransformiert und das Maximum bestimmt. Die Koordinaten des Maximums nehmen wir als Koordinaten der Verschiebung. Da wir dies aber im Quadratgitter der Pixel tun, kann das Maximum nur im Quadratgitter bestimmt werden und ist damit immer ganzzahlig. Häufig wird dann in der Praxis, z. B. bei der Kreuzkorrelation, an die Punkte in der Nähe des Maximums eine Funktion gefittet (z. B. eine Parabel). Anschließend wird das Maximum dieser Funktion bestimmt und diese Koordinaten sind nicht mehr ganzzahlig. Dies liefert oft gute Ergebnisse, ist aber nur ein heuristisch begründetes Herangehen. Wir können nun signaltheoretisch ein anderes, aber theoretisch begründetes Vorgehen angeben.

Wenn zwei Bilder mit je N Pixeln je so bandbegrenzt sind, sodass wir je aus den N Pixeln das Original rekonstruieren können, also das trigonometrische Interpolationspolynom, dann ist das Ergebnisbild, z. B. die Kreuzkorrelation, auch bandbegrenzt und kann auch vollständig rekonstruiert werden. Wir müssten dann das Maximum dieses trigonometrischen Interpolationspolynoms bestimmen und erhalten die Koordinaten mit Subpixel-Genauigkeit. Dies ist relativ aufwendig. Eine gröbere, aber schnellere Methode ist das *Zero padding*. Bevor wir das Ergebnisspektrum zurücktransformieren, füllen wir Nullen auf. Füllen wir Nullen auf die doppelte Größe auf, erhalten wir die Genauigkeit mit 1/2 Pixel, füllen wir z. B. auf die zehnfache Größe auf, erhalten wir die Genauigkeit mit 1/10 Pixel usw. Es entstehen sehr große Bilder mit viel Speicherplatz, was man aber durch geschickte Verwaltung umgehen kann.

In [25] wurden verschiedene Interpolations-Methoden zur subpixelgenauen Registrierung verglichen. Die Fourierbasierte trigonometrische Interpolation lieferte die besten Ergebnisse, allerdings steht dieser der hohe Rechenaufwand entgegen. Möchte man unterabgetastete Bilder mit Subpixelgenauigkeit registrieren, dann muss man vorher ideale Tiefpässe anwenden, siehe dazu Abschn. 5.1.

Reine Rotationen Liegen nur Rotationen vor, dann gibt es ein besonders einfaches signalorientiertes Verfahren. Wann eine reine Rotation vorliegen könnte, werden wir später klären. Die Idee ist jedenfalls sehr simpel, wir führen durch Polarkoordinaten die Rotation auf eine Verschiebung zurück und bestimmen die Verschiebung. Wir transformieren folglich jedes Bild in Polarkoordinaten r, φ . Damit wird die analoge Grauwertfunktion $f(x, y)$ in die analoge Funktion $f^{\text{Pol}}(r, \varphi) = f(r \cos \varphi, r \sin \varphi)$ transformiert. Bezuglich des Polarwinkels φ unterscheiden sich die Bilder nur durch eine 1D-Translation, obwohl die Polarbilder 2D-Bilder sind. Um die Rotation und damit diese 1D-Translation effizienter zu bestimmen, bilden wir aus jedem Polarbild (praktisch aber direkt aus den Ausgangsbildern) die 1D-Funktion:

$$h(\alpha) = \int_{\alpha - \frac{\delta\alpha}{2}}^{\alpha + \frac{\delta\alpha}{2}} \int_0^\infty f^{\text{Pol}}(r, \varphi) dr d\varphi. \quad (19.131)$$

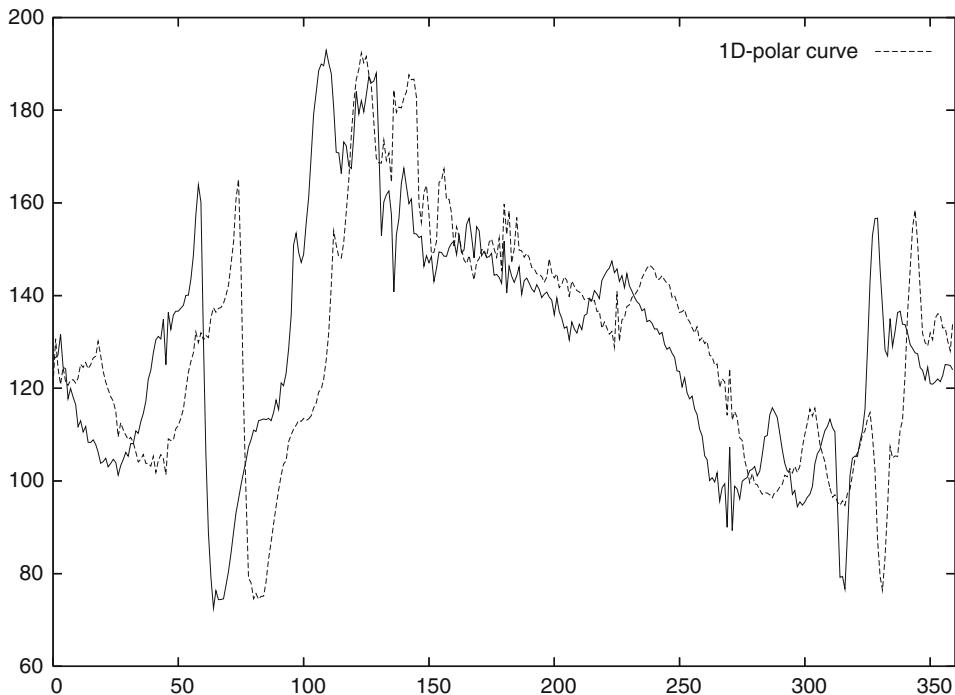


Abb. 19.5 Verschobene 1D-Polarfunktionen

Jetzt brauchen wir nur noch die Verschiebung dieser 1D-Bilder zu berechnen. Da die Bilder aber alle diskret sind und durch den Rand „hart“ beschnitten sind, müssen wir ein paar Überlegungen zur praktischen Realisierung anstellen. Wir müssen also in einem Sektor alle Grauwerte addieren, bzw. besser den Mittelwert berechnen, um unterschiedliche Anzahlen von Grauwerten im Sektor zu normieren. Da nach einer Rotation der Rand Strukturen „wegschneidet“ oder es kommen neue hinzu, müssen wir diese Strukturen eliminieren. Wir legen vom Bildmittelpunkt aus gedanklich einen Kreis mit einem geeigneten Durchmesser ρ für alle Bilder und beziehen uns nur noch auf diese Kreise. Nun müssen wir α diskretisieren, z. B. auf eine Genauigkeit von 0,1 Grad. Da aber in der Nähe des Koordinatenursprungs die Diskretisierung zu grob ist, legen wir noch einen kleinen Kreis fest, bei dem wir die Grauwerte ausschließen. Der Durchmesser des kleinen Kreises könnte z. B. bei $\frac{\rho}{10}$ liegen. So wurde z. B. das Lenna-Bild aus Abb. 1.8a und das um ca. 11 Grad rotierte Lenna-Bild in diese 1D-Polarform überführt. In der Abb. 19.5 sieht man deutlich die Verschiebung der beiden Kurven zueinander. In den meisten praktischen Anwendungen reicht diese 1D-Methode aus. Man muss sich aber im Klaren sein, dass wir das 2D-Polarbild in ein 1D-Polarbild projizieren, also in einen Unterraum und dies führt zu einem Informationsverlust. Man kann also Bilder finden, wo wir die Translation auf den 2D-Polarbildern bestimmen müssen, weil dies auf den 1D-Polarbildern nicht mehr funktioniert.

Euklidische Transformationen Eine sehr häufig vorkommende Transformation ist die Rotation verbunden mit Translationen (Spiegelungen schließen wir hier einmal aus). Dann wenden wir folgendes Standardverfahren an:

- Wir versuchen zunächst, die beiden Bilder so zu transformieren, dass die Translation zwischen den beiden verschwindet, die Rotation aber erhalten bleibt. Im Abschn. 4.26 hatten wir hergeleitet: Wenn zwei Bilder $f_1(x, y)$ und $f_2(x, y)$ sich nur durch eine Rotation und eine Translation unterscheiden, dann unterscheiden sich die Amplitudenspektren $|\alpha_1(u, v)|$ und $|\alpha_2(u, v)|$ der beiden Bilder nur noch durch die Rotation, die Translation ist eliminiert. Damit liegt tatsächlich ein praktisch relevanter Fall vor, wann sich zwei Bilder (wenn es auch nur die Amplitudenspektren sind) durch eine „reine“ Rotation unterscheiden.
- Wir bestimmen die „reine“ Rotation der beiden Amplitudenspektren mit der Methode aus dem Teil „Reine Rotationen“ des Abschn. 19.8.4. Eine Kleinigkeit ist noch zu bedenken. Da bei reellen Funktionen die Fourierkoeffizienten konjugiert komplex bezüglich des Koordinatenursprungs auftreten, ist das Amplitudenspektrum am Koordinatenursprung gespiegelt, also spiegelsymmetrisch. Daher können wir Rotationen nur noch von -90 Grad bis $+90$ Grad bestimmen, dies reicht aber in den meisten Fällen völlig aus.
- Wir rotieren nun mit dem ermittelten Rotationswinkel eines der beiden Bilder, so dass sich das eine Bild und das transformierte Bild nur noch durch die Translation unterscheiden.
- Jetzt können wir die Translation bestimmen.

Die nächsthöhere Stufe sind Ähnlichkeitstransformationen.

Ähnlichkeitstransformationen Kommen zu den Euklidischen Transformationen noch isotrope Skalierungen hinzu, sprechen wir von Ähnlichkeitstransformationen, die in der Ebene aus vier zu bestimmenden Parametern bestehen: der Translation (2 Parameter), der Rotation (1 Parameter) und der isotropen Skalierung (1 Parameter). Nun nutzen wir die gleiche Idee wie im Teil „Euklidische Transformationen“ des Abschn. 19.8.4 zur Eliminierung der Translation. Wir bilden von beiden Bildern die Amplitudenspektren $|\alpha_1(u, v)|$ und $|\alpha_2(u, v)|$, welche translations-invariant sind. Nun wollen wir wieder die Bestimmung der „restlichen“ Parameter (Rotation und Skalierung) auf die Bestimmung einer 2D-Translation zurückführen. Dazu nutzen wir die Idee der Fourier-Mellin-Transformation (siehe Abschn. 3.6) oder auch nur der *log-polar* Transformation. Wir überführen folglich die beiden Amplitudenspektren in ihre *log-polar*-Darstellung, die sich nur noch durch eine 2D-Translation unterscheiden. Diese berechnen wir nun, damit haben wir die Rotation und die Skalierung bestimmt. Anschließend transformieren wir die beiden Originalbilder mit diesen beiden Parametern und bestimmen aus diesem Bildpaar die eigentliche Translation.

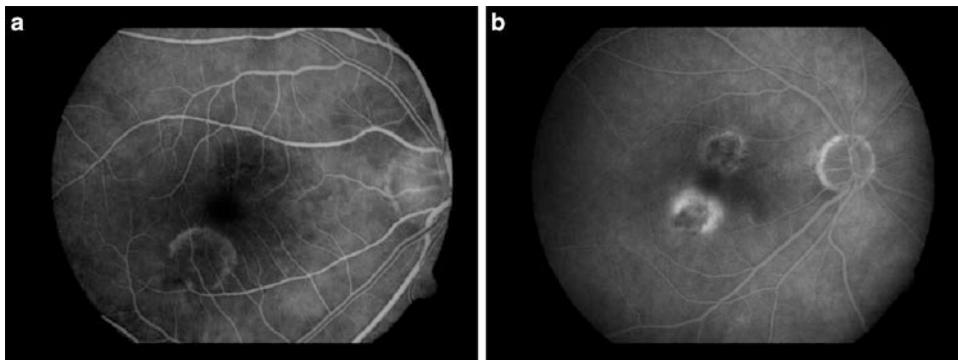


Abb. 19.6 Fundus oculi: Aufnahme des Augenhintergrundes. **a** Referenzbild 1, **b** Referenzbild 2

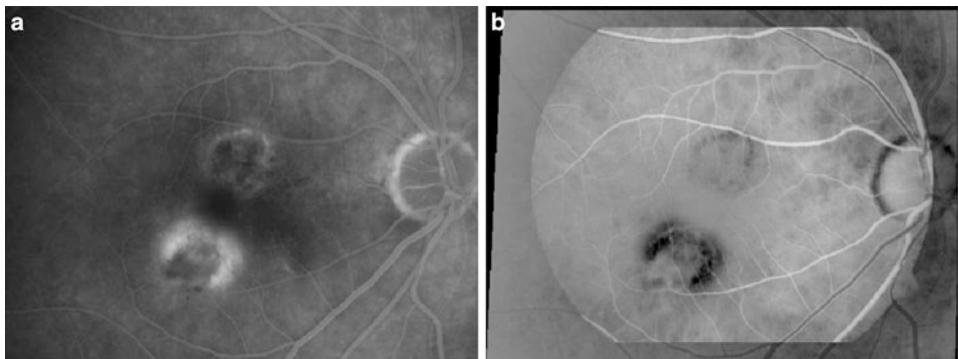


Abb. 19.7 **a** Referenzbild 2 aus Abb. 19.6b grob affin transformiert und zur Deckung mit Referenzbild 1 aus Abb. 19.6a gebracht, **b** Bild aus **a** durch Blockmatching genauer zur Deckung mit Bild aus Abb. 19.6a gebracht: Im Differenzbild sind gut die krankhaften Veränderungen zu erkennen

Affine Transformationen Die signalbasierte affine Registrierung zweier Bilder ist schon ein recht schwieriges Problem. Eine möglich Idee ist das Blockmatching, siehe auch Abschn. 4.21.1. Wir verteilen gleichmäßig über beide Bilder eine Anzahl von Blöcken einer bestimmten Größe. Nun bestimmen wir für alle Blockpaare des Quell- und Zielbildes die Verschiebung und zwar mit einer der bisher behandelten Methoden. Als Resultat erhalten wir eine Liste von Quellpunkten und den verschobenen Zielpunkten. An diese Liste passen wir mit der LSE-Methode eine affine Transformation an. Diese Methode funktioniert nur dann, wenn wir lokal und näherungsweise die affine Transformation als Verschiebung modellieren können. Eine typische Anwendung dazu ist in den Abb. 19.6 und 19.7 zu sehen. In Abb. 19.6a ist ein Fundus-Bild zu sehen. In Abb. 19.6b ist eine spätere Aufnahme mit krankhaften Veränderungen zu sehen. Aufgabe ist nun, diese zu registrieren, um die krankhaften Veränderungen besser zu beurteilen. Die Blockmatching-Methode lässt sich nicht sofort anwenden, da die Veränderungen von Abb. 19.6a zu 19.6b zu groß sind. Daher

wird zunächst durch interaktive Auswahl von Referenzpunkten eine grobe affine Transformation bestimmt, das Ergebnis der Transformation von Abb. 19.6a ist in Abb. 19.7a zu sehen. Nun wird die Blockmatching-Methode auf die Abb. 19.7a und 19.6a angewendet, eine affine Transformation bestimmt und das Abb. 19.6a affin transformiert. Die Differenz des Ergebnisbildes zum Referenzbild 1 (Abb. 19.6a) ist in Abb. 19.7b zu sehen. Die Registrierung ist perfekt gelungen.

Geordnete Punktmengen Nun betrachten wir die affine Registrierung von Objekten mit geschlossenen Konturen. Die geschlossenen Konturen zweier (flächiger) Objekte sollen verglichen werden bzw. registriert werden. Diese sollen durch eine affine Transformation T ineinander überführbar sein. Da die Konturpunkte einer Ordnung unterliegen und durch Diskretisierungseffekte Stauchungen und Streckungen auftreten, bietet sich als Matchingmethode das *time warping* an, allerdings über der Gruppe der affinen Transformationen. Da eine Optimierung über dem Raum der affinen Transformationen schon sehr aufwendig ist, verwenden wir eine andere Idee. Wir ordnen jedem Konturpunkt Merkmale zu, die affin invariant sein müssen, dann können wir das *time warping* direkt benutzen. Für jeden Konturpunkt ist nun zu berechnen:

- Wir legen gedanklich in den ausgewählten Konturpunkt den Koordinatenursprung, d. h. der Koordinatenursprung wechselt immer, wenn wir einen neuen Konturpunkt auswählen. Nun legen wir eine affin kovariante Umgebung U des Konturpunktes fest. Für affine Transformationen ist dies schwierig, deshalb wählen wir als Umgebung das gesamte Objekt. Für Euklidische Transformationen wählen wir einen Kreis mit einem festen Radius.
- Bezuglich des gewählten Koordinatenursprungs und der Umgebung U berechnen wir jetzt die affinen Momentinvarianten $H_{-1}, H_0, H_1, \dots, H_{12}$, siehe Abschn. 19.5. Der wesentliche Unterschied ist aber nun, es dürfen nicht die zentralen Momente eingehen, sondern die Momente sind bezüglich des festgelegten Koordinatenursprungs und der Umgebung U zu berechnen. Die Momente werden also nur bezüglich der Trägheitsellipse nomalisiert und dann die Rotationsinvarianten berechnet. Die Invarianten sind nun invariant gegen den linearen Anteil A einer affinen Transformation. Daher können wir auch die Invariante H_0 benutzen, da die Momente erster Ordnung nicht normalisiert werden. Folglich benutzen wir die Momente bis vierter Ordnung und demzufolge die 11 Invarianten $H_0, H_3, H_4, \dots, H_{12}$. Die Invarianten H_1 und H_2 sind nicht verwendbar, da sie durch die Normalisierung der Trägheitsellipse stets konstant sind. Bei Euklidischen Transformationen normalisieren wir nichts, es werden die „reinen“ Rotationsinvarianten verwendet.
- Nun haben wir jedem Punkt aus der geordneten Konturpunktfolge einen 11-dimensionalen Merkmalsvektor aus den Invarianten zugeordnet. Dies tun wir für das erste Objekt B mit (x_i, y_i) , $i = 1, \dots, m$ und für das zweite Objekt B' mit (x'_j, y'_j) , $j = 1, \dots, n$. In der Regel ist $m \neq n$. Die Nummerierung der Punkte beginnt jeweils mit einem willkürlich gewählten Punkt, d. h. es liegt eine zyklische Ordnung vor.

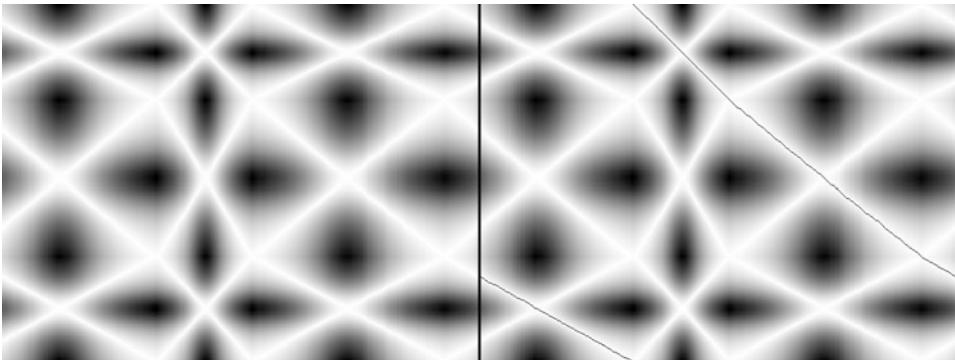


Abb. 19.8 Distanz-Funktion und detektierter, optimaler Pfad

- Jetzt ist das Referenzproblem zu lösen. Es sind die Punkte aus B den Punkten aus B' zuzuordnen, die die gleichen Invarianten besitzen, und dies möglichst eindeutig. Dazu benutzen wir ein Matchingmaß, z. B. die Euklidische Metrik und ordnen mit dem *time warping* die Punkte einander zu, da Stauchungen und Streckungen auftreten müssen und wir die gegebene Ordnung der Konturpunkte ausnutzen müssen.

Zuordnung mittels Dynamischer Programmierung Zunächst berechnen wir eine Kostenfunktion $\text{dist}(i, j)$ zwischen dem Merkmalsvektor des i -ten Punktes aus B und dem Merkmalsvektor des j -ten Punktes aus B' . Wenn die Merkmalsvektoren ähnlich sind, dann sollte es geringe Kosten geben und wenn sie unähnlich sind, dann sollte es große Kosten geben. Die einfachste Kostenfunktion ist der Euklidische Abstand zwischen den Merkmalsvektoren, den wir auch im Folgenden benutzen werden. In der Abb. 19.8 ist eine typische Distanzmatrix oder Distanzfunktion im linken Teil zu sehen. Wir suchen nun einen Pfad mit minimalen Kosten, d. h. eine Folge von Referenzpunktpaaren (i, j) von der ersten Spalte bis zur letzten Spalte, wobei wieder gilt:

$$\sum_{(i,j) \in \text{Pfad}} d(i, j) \rightarrow \text{Minimum.} \quad (19.132)$$

Wir wollen dabei das Grundprinzip der Dynamischen Programmierung nutzen, siehe Abschn. 10.7.1. Da wir von geordneten, aber zyklischen Punktfolgen ausgehen, müssen wir die Periodizität unbedingt beachten. Endet also ein Pfad an der unteren oder oberen Matrixgrenze, dann ist der Pfad periodisch fortzusetzen. In Abb. 19.8 ist beim optimalen Pfad die periodische Fortsetzung deutlich zu sehen. Nun wollen wir den optimalen Pfad berechnen. Dazu nehmen wir an, dass es in horizontaler Richtung (Index i) mehr Punkte als in vertikaler Richtung (Index j) gibt, notfalls vertauschen wir einfach die Punktmengen. Damit haben wir eine ähnliche Situation wie beim Bresenham-Algorithmus zum Bestimmen eines digitalen Geradensegmentes: in i -Richtung wird immer um eins inkrementiert, während in j -Richtung entweder um eins inkrementiert wird oder nicht inkrementiert wird.

Der Unterschied zum Bresenham-Algorithmus besteht nur in der Entscheidung, wann bei j inkrementiert wird, somit entsteht natürlich keine digitale Gerade. Damit können einem j -ten Punkt mehrere i -te Punkte zugeordnet werden, aber nicht umgekehrt. Der Pfad ist also monoton nicht fallend. In Abb. 19.8 ist der Koordinatenursprung „links oben“.

Vorwärtssrechnung Nun legen wir eine (Summen)-Matrix/Funktion $c(i, j)$ der gleichen Dimension wie $d(i, j)$ an und übertragen die erste Spalte von $d(i, j)$ nach $c(i, j)$. Jetzt wird die (Summen)-Funktion/Matrix $c(i, j)$ ab der zweiten Spalte durch eine sogenannte *Vorwärtsrechnung* gefüllt:

$$c(i, j) = \min(c(i - 1, j), c(i - 1, j - 1)) + d(i, j). \quad (19.133)$$

Falls die affine Transformation eine Spiegelung enthält, dann können sich die Ordnungen in den Punktmengen „umdrehen“. In diesem Falle müssen wir eine zweite (Summen)-Funktion/Matrix $c(i, j)$ anlegen und füllen:

$$c(i, j) = \min(c(i - 1, j), c(i - 1, j + 1)) + d(i, j). \quad (19.134)$$

Im Falle von Spiegelungen führen wir also zwei (Summen)-Funktionen/Matrizen, ansonsten nur eine. Nun entnehmen wir aus der letzten Spalte der (Summen)-Funktion/Matrix die beiden Größen

$$\text{minimale_kosten} = \min_{j=1, \dots, n}(c(m, j)), \quad j^* = \operatorname{argmin}_{j=1, \dots, n}(c(m, j)), \quad (19.135)$$

und legen nun diejenige von den beiden (Summen)-Funktionen/Matrizen fest, die die kleinere Größe von *minimale_kosten* besitzt. Nun haben wir die minimale Kostensumme des optimalen Pfades berechnet, aber den optimalen Pfad selbst noch nicht. Dazu führen wir jetzt eine *Rückwärtsrechnung* durch.

Rückwärtsrechnung

- Entsprechend der Wahl der (Summen)-Funktion/Matrix wissen wir, ob die Vorgänger eines Punktes (i, j) in der Funktion/Matrix nach (19.133) oder (19.134) zu wählen sind.
- Wir beginnen beim Punkt (m, j^*) in der letzten Spalte.
- Da wir nun die zulässigen Vorgänger dieses Punktes kennen, wählen wir denjenigen aus, der in der (Summen)-Funktion/Matrix den kleineren Wert c besitzt.
- Dies wiederholen wir nun solange, bis wir in der ersten Spalte angekommen sind. Dabei müssen wir bezüglich der Vorgänger in der obersten und untersten Matrixzeile die Periodizität unbedingt beachten.
- In der Abb. 19.8 ist so der optimale Pfad ermittelt worden.
- Die Komplexität des gesamten Algorithmus beträgt $O(m \cdot n)$.

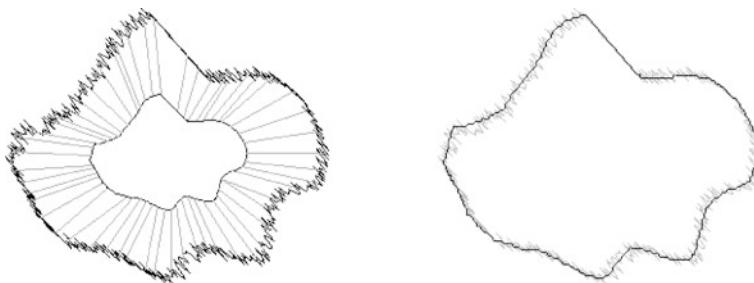


Abb. 19.9 Referenzpunkte und affin transformiertes Objekt

- Eigentlich müssten wir berücksichtigen, dass der Nachfolger des Punktes (m, j^*) zyklisch der Startpunkt in der ersten Spalte sein muss. Tun wir dies, erhöht sich sofort die Komplexität des Algorithmus. Praktische Erfahrungen haben gezeigt, dass dies nicht nötig ist.

Bestimmung der affinen Transformation

- Durch den optimalen Pfad haben wir eine Liste von Punktreferenzen bestimmt, wobei es für einen Punkt durchaus mehrere Referenzpunkte geben kann. Da die mehrdeutigen Referenzpunkte Nachbarpunkte sein müssen, stören sie nicht und können in der Liste verbleiben.
- Für jedes Referenzpaar können wir noch ein Gewicht eintragen, z. B. die Kosten $d(i, j)$.
- Nun bestimmen wir mit der LSE-Methode oder noch besser mit der LAD-Methode die affine Transformation T und transformieren z. B. das Objekt B mit $A = TB$. Mit der LSE-Methode führt dies auf die Gaußschen Normalengleichungen mit den 6 Unbekannten der affinen Transformation. Benutzen wir dagegen die LAD-Methode, dann ist dies numerisch aufwendiger und wir benutzen wieder die Lineare Programmierung. Zum Schluß vergleichen wir die Ähnlichkeit von A und B' mit einem ungerichteten Matchingmaß $d(A, B')$.
- Das Verfahren kann auch zur affinen Registrierung benutzt werden. Die Momente eines Binärobjektes können auch durch Grauwertmomente ersetzt werden, wenn im Innern der Objekte die Grauwerte zur Verfügung stehen. Allerdings sollte man dann auf Beleuchtungsschwankungen achten oder die Invarianten sollten zusätzlich noch invariant gegenüber Beleuchtungsänderungen sein.

In der Abb. 19.9 wird die innere Kontur affin transformiert und mit Rauschen überlagert. Man sieht die Referenzen und das Ergebnis des Matchings. In der Abb. 19.10 wird die innere Kontur transformiert und mit einer starken Störung versehen. Das Matchingresultat ist trotzdem korrekt, was die große Robustheit dieses Verfahrens zeigt.

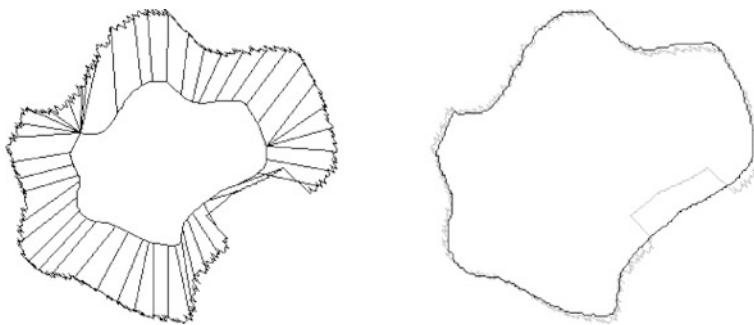


Abb. 19.10 Referenzpunkte und affin transformiertes Objekt

Ungeordnete Mengen von Referenzpunkten Sind die Punkte nicht geordnet, dann ändert sich nur das Zuordnungsverfahren, z. B. ist dann die dynamische Programmierung nicht anwendbar, sondern wir nutzen z. B. den ICP-Algorithmus oder die Ungarische Methode usw. Haben wir die Umgebung eines Punktes festgelegt, dann berechnen wir von der Umgebung des Punktes die signalbasierten Momentinvarianten. Betrachten wir z. B. nur Euklidische Transformationen, dann können wir als Umgebung der ausgewählten Punkte einen Kreis mit festem Radius wählen und dann berechnen wir als Merkmale natürlich nur noch Momentinvarianten bezüglich Euklidischer Transformationen. Diese Idee wurde sogar zur Registrierung von 3D-Punktwolken benutzt, siehe [76]. Dabei ist die Umgebung eine Kugel mit festem Radius, die Momente sind Oberflächenmomente bezüglich der Dreiecke der Oberfläche und die Momentinvarianten sind 3D-Rotations-Momentinvarianten.

Eine völlig andere Frage ist, wo bekommt man eigentlich die Referenzpunkte her, diese müssen ja selbst kovariant sein?

- Wir rastern das Bild/Objekt/Kontur mit Punkten „dicht genug“ ab, dann ist der Fehler bezüglich der Kovarianz nur noch durch die „Dichte“ der Rasterung bestimmt. Ein Nachteil ist, wir erzeugen somit sehr viele Referenzpunkte. Im Abschn. 19.8.4 hatten wir dies für die Punkte der Kontur getan, daher sind die Konturpunkte näherungsweise kovariant.
- Wir berechnen aus dem Bild/Objekt nur Punkte als Referenzpunkte, die kovariant sind. Dazu müssen wir natürlich Eigenschaften nutzen, die uns dies garantieren. Berechnen wir z. B. aus dem Bild/Objekt die MSER-Bereiche und anschließend von diesen die Schwerpunkte, so haben wir tatsächlich affin-kovariante Referenzpunkte ermittelt.
- Besonders einfach wird die Wahl der Referenzpunkte, wenn wir eine endliche, diskrete Punktmenge P und deren Transformierte Q von vornherein zur Verfügung haben. Dann spricht man auch von *Point pattern matching*. Wir nehmen einmal die Gleichmächtigkeit der Mengen an, d. h. $|P| = |Q| = n$. Dann können wir einfach jeden Punkt $p \in P$ und $q \in Q$ als Referenzpunkt wählen. Jetzt legen wir wieder für jeden Referenz-

punkt eine kovariante Umgebung fest, z. B. einen Kreis bei Euklidischen Transformationen oder die gesamte diskrete Punktmenge P bzw. Q als globale Umgebung. Bezuglich der gewählten Umgebung, die jetzt aus einer endlichen, diskreten Punktmenge besteht, berechnen wir die Momentinvarianten. In die Momentinvarianten gehen aber nun die Punktmomente (19.6) ein. Bei der Normalisierung der Punktmomente bezüglich der Trägheitsellipse ist der Unterschied zur Normalisierung der Linien- und Flächenmomente zu beachten, siehe Abschn. 19.3. Die Normalisierung bez. der Translation darf nicht erfolgen, der Koordinatenursprung ist wieder der jeweilige Referenzpunkt. Die Rotationsinvarianten $H_0, H_3, H_4, \dots, H_{12}$ können nach der Normalisierung wieder uningeschränkt verwendet werden, so dass jedem Referenzpunkt in P als auch in Q ein 11-dimensionalmer Merkmalsvektor zugeordnet ist, der invariant gegenüber dem linearen Anteil A einer affinen Transformation ist. Nun stellen wir wieder die Kostenmatrix auf (Euklidischer Abstand zwischen den Merkmalsvektoren der Mengen P und Q) und berechnen (z. B. mit der Ungarischen Methode) die eineindeutigen Zuordnungen der Punkte aus P zu den Punkten aus Q . Wir erhalten demzufolge als Endergebnis eine Permutationsmatrix, siehe [85].

Kann man eigentlich eine Kostenfunktion aufstellen ohne die Wahl von Umgebungen und die Berechnung von Merkmalen aus den Umgebungen? Dies geht tatsächlich. Wir gehen direkt vom Zuordnungsproblem und einer Kostenfunktion $c(p, q)$ aus und wollen die optimale Permutation der Zuordnungsindizes berechnen. Man formuliert dies oft als Matching in bipartiten Graphen. Wir bezeichnen mit $(k_1, k_2, \dots, k_i, \dots, k_n)$ eine Permutation der Indizes der Punkte $\mathbf{p}_1, \dots, \mathbf{p}_n$ bzw. der Punkte $\mathbf{q}_1, \dots, \mathbf{q}_n$ und nehmen die Gleichmächtigkeit der Punktmengen P und Q an. Wir müssen nun eine Permutation finden, die die Aufgabe

$$S = \sum_{i=1}^n c(\mathbf{p}_i, \mathbf{q}_{k_i}) \rightarrow \text{Minimum} \quad (19.136)$$

löst. Das wesentliche Problem ist nun: wie kann man die Kostenfunktion $c(p, q)$ wählen, so dass wir die richtige Zuordnung der Punkte bezüglich einer affinen Transformation erhalten? Dazu wählen wir als Transformation zunächst eine einfache Translation

$$\mathbf{q}_i = \mathbf{p}_i + \mathbf{a}, \quad i = 1, \dots, n, \quad (19.137)$$

wobei $\mathbf{a}^T = (a_{10}, a_{20})$ der Translationsvektor ist. Wenn wir nun als Kosten die Euklidischen Abstände $d(\mathbf{p}_i, \mathbf{q}_{k_i}) = |\mathbf{p}_i - \mathbf{q}_{k_i}|$ wählen, dann kann man mit einem einfachen Beispiel zeigen, dass das Optimum nicht die richtige Permutation liefert. Daher versuchen wir es mit den Quadraten der Euklidischen Abstände:

$$S = \sum_{i=1}^n c(\mathbf{p}_i, \mathbf{q}_{k_i}) = \sum_{i=1}^n |\mathbf{p}_i - \mathbf{q}_{k_i}|^2 \rightarrow \text{Minimum.} \quad (19.138)$$

Wir setzen die Translation ein und erhalten

$$\begin{aligned} S &= \sum_{i=1}^n (\mathbf{p}_i - (\mathbf{p}_{k_i} + \mathbf{a}))^2 \\ &= \sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_i - 2 \sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_{k_i} - 2 \sum_{i=1}^n \mathbf{p}_i^T \mathbf{a} + \sum_{i=1}^n (\mathbf{p}_{k_i} + \mathbf{a})^2 \rightarrow \text{Minimum.} \end{aligned} \quad (19.139)$$

Gesucht ist die richtige Permutation bezüglich der Translation der Punktmenge. Die Translation \mathbf{a} ist zwar unbekannt, aber für das konkrete Problem als Konstante aufzufassen. Wir sehen, dass die Terme $\sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_i$, $\sum_{i=1}^n \mathbf{p}_i^T \mathbf{a}$ nicht von der gesuchten Permutation abhängen. Da

$$\sum_{i=1}^n (\mathbf{p}_{k_i} + \mathbf{a})^2 = \sum_{i=1}^n \mathbf{p}_{k_i}^2 + 2 \sum_{i=1}^n \mathbf{p}_{k_i}^T \mathbf{a} + \sum_{i=1}^n \mathbf{a}^2 = \sum_{i=1}^n \mathbf{p}_i^2 + 2 \sum_{i=1}^n \mathbf{p}_i^T \mathbf{a} + \sum_{i=1}^n \mathbf{a}^2 \quad (19.140)$$

gilt, hängt der Term $\sum_{i=1}^n (\mathbf{p}_{k_i} + \mathbf{a})^2$ auch nicht von der gesuchten Permutation ab. Deshalb können wir das Optimierungsproblem reduzieren auf:

$$\sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_{k_i} \rightarrow \text{Maximum.} \quad (19.141)$$

Wir schätzen nun diese Summe $\sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_{k_i}$ ab. Mit

$$\mathbf{p}_i^T = (x_i, y_i) \quad \text{und} \quad \mathbf{x}^T = (x_1, x_2, \dots, x_n), \quad \mathbf{x}_{\text{perm}}^T = (x_{k_1}, x_{k_2}, \dots, x_{k_n})$$

(analog für $\mathbf{y}, \mathbf{y}_{\text{perm}}$) erhalten wir

$$\sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_{k_i} = \sum_{i=1}^n x_i x_{k_i} + \sum_{i=1}^n y_i y_{k_i} = \mathbf{x}^T \mathbf{x}_{\text{perm}} + \mathbf{y}^T \mathbf{y}_{\text{perm}}. \quad (19.142)$$

Wir benutzen die Cauchy-Schwarzsche Ungleichung und erhalten

$$(\mathbf{x}^T \mathbf{x}_{\text{perm}})^2 \leq \|\mathbf{x}\|^2 \|\mathbf{x}_{\text{perm}}\|^2 = \|\mathbf{x}\|^2 \|\mathbf{x}\|^2 = \|\mathbf{x}\|^4. \quad (19.143)$$

Dies bedeutet $\mathbf{x}^T \mathbf{x}_{\text{perm}} \leq \|\mathbf{x}\|^2$. Daraus folgt

$$\mathbf{x}^T \mathbf{x}_{\text{perm}} + \mathbf{y}^T \mathbf{y}_{\text{perm}} \leq \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 = \sum_{i=1}^n \|\mathbf{p}_i\|^2. \quad (19.144)$$

Nochmals zusammengefasst ist dann:

$$\sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_{k_i} \leq \sum_{i=1}^n \|\mathbf{p}_i\|^2. \quad (19.145)$$

Das Maximum der linken Seite und damit die Gleichheit in der Schwarzschen Ungleichung wird genau dann angenommen, wenn beide Vektoren Linearkombinationen voneinander sind. Dies bedeutet, dass $\mathbf{x}_{k_i} = \mathbf{x}_i, \mathbf{y}_{k_i} = \mathbf{y}_i$ gilt und deshalb folgt $\mathbf{p}_{k_i} = \mathbf{p}_i, i = 1, 2, \dots, n$. Dies bedeutet, wir erhalten für Translationen als optimale Lösung von (19.138) tatsächlich die korrekte Permutation und damit die korrekten Korrespondenzen. Wie ist es aber nun mit allgemeinen affinen Transformationen:

$$\mathbf{q}_i = \mathbf{A}\mathbf{p}_i + \mathbf{a} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} a_{10} \\ a_{20} \end{pmatrix} ? \quad (19.146)$$

Dazu setzen wir die affine Transformation in (19.138) ein:

$$S = \sum_{i=1}^n (\mathbf{p}_i - (\mathbf{A}\mathbf{p}_{k_i} + \mathbf{a}))^2 \quad (19.147)$$

$$= \sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_i - 2 \sum_{i=1}^n \mathbf{p}_i^T \mathbf{A} \mathbf{p}_{k_i} - 2 \sum_{i=1}^n \mathbf{p}_i^T \mathbf{a} + \sum_{i=1}^n (\mathbf{A} \mathbf{p}_{k_i} + \mathbf{a})^2 \rightarrow \text{Minimum.} \quad (19.148)$$

Es ist leicht zu sehen, dass die Terme $\sum_{i=1}^n \mathbf{p}_i^T \mathbf{p}_i$, $\sum_{i=1}^n \mathbf{p}_i^T \mathbf{a}$ und $\sum_{i=1}^n (\mathbf{A} \mathbf{p}_{k_i} + \mathbf{a})^2$ nicht von der Permutation abhängen. Daher bleibt nur noch:

$$\sum_{i=1}^n \mathbf{p}_i^T \mathbf{A} \mathbf{p}_{k_i} \rightarrow \text{Maximum.} \quad (19.149)$$

Die Idee ist nun das Problem (19.149) auf die Aufgabe (19.141) zurückzuführen. Deshalb führen wir eine Matrix \mathbf{C} ein mit $\mathbf{p}'_i = \mathbf{C}\mathbf{p}_i$ und

$$\sum_{i=1}^n \mathbf{p}'_i{}^T \mathbf{p}'_{k_i} = \sum_{i=1}^n (\mathbf{C}\mathbf{p}_i)^T (\mathbf{C}\mathbf{p}_{k_i}) = \sum_{i=1}^n \mathbf{p}_i^T \mathbf{C}^T \mathbf{C} \mathbf{p}_{k_i} \rightarrow \text{Maximum.} \quad (19.150)$$

Wenn wir nun eine Matrix \mathbf{C} mit der Beziehung $\mathbf{A} = \mathbf{C}^T \mathbf{C}$ finden können, dann ist die optimale Permutation die korrekte Lösung von (19.149) bzw. (19.147). So eine Matrix \mathbf{C} existiert, wenn die affine Matrix \mathbf{A} symmetrisch und positiv definit ist. Die geforderte Faktorisierung bedeutet: „wir können die Wurzel aus \mathbf{A} ziehen“. Wenn dies möglich ist, dann folgt $\mathbf{p}'_{k_i} = \mathbf{p}'_i$, d.h. $\mathbf{C}\mathbf{p}_{k_i} = \mathbf{C}\mathbf{p}_i$ und somit $p_{k_i} = p_i$. Damit haben wir bisher gezeigt: wenn die Punktmengen durch die spezielle affine Transformation

$$q_i = \mathbf{S}\mathbf{p}_i + \mathbf{a}, \quad \mathbf{S} \text{ ist symmetrisch und positiv definit} \quad (19.151)$$

ineinander überführbar sind, dann können wir die korrekte Permutation durch Minimierung der Quadrate der Euklidischen Punktabstände berechnen. Anders ausgedrückt:

Die optimale Permutation ist invariant gegenüber diesen speziellen affinen Transformationen. Überraschend ist jedoch, dass dies

$$S = \sum_{i=1}^n |\mathbf{p}_i - \mathbf{q}_{k_i}|^n \rightarrow \text{Minimum} \quad (19.152)$$

mit $n \neq 2$ nicht gilt. Was fehlt nun noch zur allgemeinen affinen Transformation? Dazu formulieren wir eine leicht beweisbares

Theorem Eine beliebige Matrix \mathbf{A} mit $\det(\mathbf{A}) > 0$ kann zerlegt werden in eine symmetrische, positiv definite Matrix \mathbf{S} und in eine Rotationsmatrix \mathbf{R} ($\mathbf{R}^{-1} = \mathbf{R}^T$, $\det(\mathbf{R}) = +1$), d. h. $\mathbf{A} = \mathbf{S} \cdot \mathbf{R}$. Die Einschränkung $\det(\mathbf{A}) > 0$ bedeutet lediglich, dass die affine Transformation keine Spiegelungen enthält.

Das heißt, uns fehlt nur noch die Rotation, d. h. nur ein einziger Transformationsparameter. Daher gehen wir praktisch folgendermaßen vor und nennen den Algorithmus DAPPM-Algorithmus.

Distance based affine point pattern matching algorithm (DAPPM)

- Wir rotieren die Punktmenge P (oder alternativ die Punktmenge Q , aber stets nur eine Punktmenge) von 0 bis 360 Grad in Schritten von 5 bis 10 Grad. In den meisten Anwendungen reichen schon 10 Grad-Schritte. Dabei kann ein beliebiges Rotationszentrum gewählt werden, z. B. der Schwerpunkt der Punktmenge.
- Für jede Rotation lösen wir das Optimierungsproblem (19.138) mit Hilfe der Ungarischen Methode (*Hungarian Method*) und erhalten für jeden Rotationsschritt eine optimale Permutation.
- Mit den berechneten Referenzen $p_i \rightarrow q_{k_i}$ berechnen wir die affine Transformation mittels der LSE-Methode oder der LAD-Methode, siehe Abschn. 21.2. Ausgleichsrechnung. Mit dieser affinen Abbildung transformieren wir die Punktmenge P und erhalten die Punktmenge P' . Die Bewertung der Referenzen $p'_i \rightarrow q_{k_i}$ ist sicher ein Kriterium für die Korrektheit der berechneten affinen Transformation.
- Deshalb leiten wir aus den Referenzen $p'_i \rightarrow q_{k_i}$ das Maß $d(P', Q)$ ab:

$$d(P', Q) = \sum_{i=1}^n |p'_i - q_{k_i}|. \quad (19.153)$$

- Wir wählen denjenigen Rotationswinkel für den das Maß $d(P', Q)$ minimal ist. Die Permutation für diesen Winkel ist die korrekte Permutation.
- Bei Rotationsschritten von z. B. 5 Grad müssen wir 72 eindimensionale Optimierungsprobleme mit der Ungarischen Methode lösen. Die Ungarische Methode hat die Komplexität $O(n^3)$.
- Es wurde die Gleichmächtigkeit der beiden Punktmengen angenommen. Ist dies nicht der Fall, wird die Kostenmatrix einfach mit Nullen aufgefüllt, so dass eine quadratische Kostenmatrix entsteht. Die Unterschiede dürfen allerdings nicht zu groß sein, und es dürfen nicht zu viele Ausreißer vorhanden sein, dann versagt die Methode.
- Besonders robust ist die Methode gegenüber Rauschen und der affinen Transformation selbst. Wenn z. B. Bilder nicht mehr näherungsweise mit affinen Transformationen

beschreibbar sind, sondern nur mit projektiven Transformationen, dann liefert die Methode immer noch die richtigen Referenzen.

- In den Abb. 15.4 und 15.5 wurde die Entzerrung des Kalibriermusters mit dieser Methode durchgeführt, obwohl die Aufnahmen nicht mehr näherungsweise affin beschreibbar sind.

Nichtelementare Transformationen Matchingalgorithmen laufen immer nach diesem Grundprinzip ab. Die Transformationen können globale oder lokale geometrische Transformationen sein, sie können nichtlinear sein oder sogar Signaltransformationen sein. Das Matchingmaß muss der Anwendung genau angepasst werden und das Optimierungsverfahren muss effizient entworfen sein. Die gewählten Transformationen müssen geeignet und passend zur Aufgabe sein, sie dürfen nicht zu starr, aber auch nicht zu elastisch und universell sein. Gibt man eine gewisse „Steifigkeit“ auf, dann kann man schließlich jedes Objekt in jedes andere überführen, wie z. B. beim *Morphing*, was aber nicht mehr Sinn des Matchings ist.

Beispiel – ICP-Algorithmus Als typisches Beispiel zum Matchen von zwei Punktmenge A und B (2D als auch 3D) soll der populär gewordene ICP-Algorithmus (Iterative Closest Point) (Besl und McKay, 1992) angegeben werden:

- Grundvoraussetzung a): Man wähle eine geeignete geometrische Transformation T, die als Matchinggrundlage dient. Dies hängt natürlich von der konkreten Aufgabe ab, von Translationen bis zu projektiven Transformationen ist alles möglich.
- Grundvoraussetzung b): Als Startlösung muss man die beiden Mengen wenigstens „grob“ bezüglich der gewählten geometrischen Transformation zu Deckung bringen. Ist A nur ein Teilstück von B, dann muss es grob in diese Lage gebracht werden. Dies kann mit den verschiedensten Methoden und Merkmalen erfolgen. Liegen z. B. nur Translationen vor, dann kann man die Schwerpunkte der Punktmengen benutzen und verschiebt eine Punktmenge (dies geht aber nicht, wenn A Teilstück von B ist).
- Für jeden Punkt $a \in A$ wird der Punkt $b \in B$ mit dem gerichteten Abstand $d(a, B)$ bestimmt, d. h. für jedes $a \in A$ wird der nächste Nachbar $b \in B$ bestimmt. Alle diese Pseudo-Referenzpaare werden in eine Liste L aufgenommen.
- Falls die Punktmengen A und B als gleiche Objekte angesehen werden, tun wir dies auch mit allen Punkten aus B, d. h. für jedes $b \in B$ wird der nächste Nachbar $a \in A$ bestimmt. Diese Pseudo-Referenzpaare werden auch in die Liste L übernommen. Wird das Objekt A nur als Teilstück von B angesehen, dann entfällt dieser Schritt.
- Mit dieser Liste L der Pseudo-Referenzpaare wird mittels eines Schätzers die geometrische Transformation T berechnet. Als Schätzer kann man die LSE-Methode oder die weniger gegen Ausreißer sensitive LAD-Methode benutzen, siehe Abschn. 21.2.5. Das gerichtete bzw. das ungerichtete Abstandsmaß $d(A, B)$ beurteilt die Güte der Übereinstimmung von A mit B bzw. mit einer Teilmenge von B.
- Man wiederholt die Schritte solange bis sich $d(A, B)$ nicht mehr wesentlich ändert.

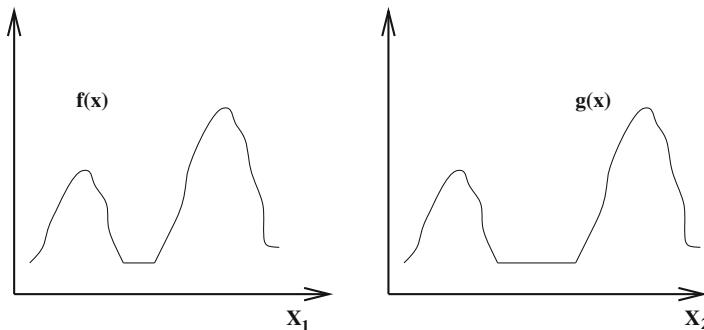


Abb. 19.11 Gedehnte Muster

Die Qualität des Matchingergebnisses hängt entscheidend von der Anfangslösung ab. Ist diese Anfangslösung nicht „gut genug“, dann erhält man womöglich ein nicht brauchbares „Nebenminimum“. Der ICP-Algorithmus wird erfolgreich bei vielen Registrierungsaufgaben eingesetzt.

Beispiel – time warping Eine typische nichtlineare Transformation liegt beim *time warping* vor, welches aus der Sprachverarbeitung stammt. Da der Zeit eine Ordnung zugrunde liegt, können wir diese Idee in der Bildverarbeitung nur nutzen, wenn dem Muster eine Ordnung zu Grunde liegt. Dies ist z. B. bei Handschriften, Unterschriften der Fall, aber auch Konturen, Polygone usw. unterliegen einer Ordnung. Bei Unterschriften, Handschriften usw. schreiben wir immer von links kontinuierlich nach rechts. Für jede horizontale Position der Schrift berechnen wir pro Spalte einen Merkmalsvektor. Folglich erhalten wir für zwei zu vergleichende Unterschriften zwei Vektorfunktionen:

$$\text{Muster}_1 = \mathbf{f}(x) \text{ für } 0 \leq x \leq X_1, \quad \text{Muster}_2 = \mathbf{g}(x) \text{ für } 0 \leq x \leq X_2. \quad (19.154)$$

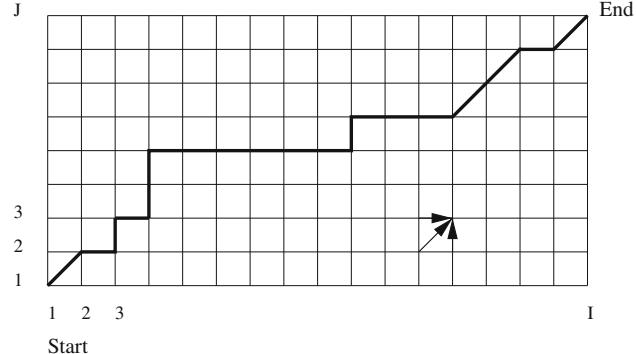
Die beiden Muster können bezüglich x stellenweise gedehnt oder gestaucht sein, z. B. erhöht oder erniedrigt man beim Unterschreiben die Geschwindigkeit. Je nach Aufgabenstellung kann man die beider Muster in Abb. 19.11 als gleich oder unterschiedlich ansehen. Bei Gleichheit würde ein direktes Matching zu völlig falschen Ergebnissen führen, auch lineare geometrische Transformationen können diese Dehnungen und Stauchungen nicht kompensieren. Wir müssen also eine spezielle nichtlineare Transformation finden, die dies kann. Gesucht ist folglich eine nichtlineare Funktion $h(x)$, so dass

$$\rho(\mathbf{f}(x), \mathbf{g}(h(x))) \rightarrow \text{Minimum} \quad (19.155)$$

gilt. Ohne weitere Einschränkungen an h ist die Aufgabe vollkommen unsinnig gestellt, deshalb müssen wir jetzt nur Stauchungen und Dehnungen beschreiben. Dazu fordern wir:

$$h(0) = 0, \quad h(X_2) = X_1, \quad 0 \leq x \leq X_1, \quad h(x) \text{ monoton und stetig}. \quad (19.156)$$

Abb. 19.12 Dynamische Programmierung



Die Funktion $h(x)$ zu finden, ist ein Problem der Variationsrechnung. Da wir in der Praxis sowieso nur diskrete Stützstellen vorfinden, formulieren wir jetzt diese Aufgabe sofort diskret. Dazu seien die Funktionen als Folgen von Merkmalsvektoren gegeben:

$$\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_I; \quad \mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_J. \quad (19.157)$$

Statt einer Funktion $h(x)$ suchen wir jetzt eine Folge von Indexpaaren $w_1 = (i_1, j_1), \dots, w_K = (i_K, j_K)$, sodass

$$\sum_{k=1}^K \rho(\mathbf{f}_{i_k}, \mathbf{g}_{j_k}) \rightarrow \text{Minimum} \quad (19.158)$$

gilt. Sowohl die Größe von K als auch die Folge der Indexpaare müssen Bedingungen bezüglich der Forderung nach Monotonie erfüllen, dies ist in Abb. 19.12 deutlich zu erkennen. Übersetzt heißt dies, wir suchen in einem Graphen einen minimalen Weg vom Startknoten zum Endknoten. Jeder Knoten bedeutet ein Indexpaar (i, j) und jedem Knoten wird ein Matchingmaß $d(i, j)$ zugeordnet, nämlich das Abstandsmaß zwischen dem i -ten Merkmalsvektor \mathbf{f}_i und dem j -ten Merkmalsvektor \mathbf{g}_j . Die Anzahl K entspricht demnach der Länge des Weges. Die Monotoniebedingung schränkt die zulässigen Wege ein. Den optimalen Weg finden wir mit der allgemeinen Methode der *Dynamischen Programmierung* (Bellman 1957). Dazu betrachten wir die simple Rekursionsgleichung

$$D(i, j) = d(i, j) + \min [D(i - 1, j), D(i - 1, j - 1), D(i, j - 1)]. \quad (19.159)$$

Die Rekursion beginnt mit $D(1, 1) = d(1, 1)$. Die Rekursion (19.159) berücksichtigt, dass ein Knoten nur genau drei bestimmte Vorgänger haben darf. Beim Startknoten beginnend berechnet man nun in einer *Vorwärtsrechnung* für jeden Knoten den Wert $D(i, j)$. Dann ist das Abstandsmaß beider Muster der letzte Wert, nämlich $D(I, J)$. Möchte man zusätzlich noch die Folge der Indexpaare haben, also die Koordinaten der Knoten bezüglich des optimalen Weges, dann muss man eine sogenannte *Rückwärtsrechnung* durchführen. Man

beginnt beim „letzten“ Knoten (I, J) und sucht den Vorgängerknoten, der von den drei zulässigen Vorgängerknoten den kleinsten Wert $D(i, j)$ hat. Dies wird solange fortgesetzt bis man beim Startknoten $(1, 1)$ angelangt ist. Mit dieser Folge von Indexpaaren kann nun eines der beiden Muster „entzerrt“ werden. Sind z. B. die beiden Unterschriften von vornherein nicht in Größe und Lage aneinander angepasst, so muss man dies natürlich entweder vorher tun oder in die dynamische Programmierung einbinden, d. h. man muss ein Muster transformieren und anschließend dynamisch entzerrn. Wir suchen dann das beste Maß $D(I, J)$ über dem Raum der Transformationen. Eine mögliche Idee dazu wurde beim Beispiel „Konturmatching“ gezeigt.

19.9 Lokale Merkmale und Häufigkeitshistogramme

Wenn wir aus einer lokalen Umgebung $U_{i,j}$ eines Pixels (i, j) rotationsinvariante Merkmale berechnen wollen, dann ist dies recht einfach: wir legen eine rotationsinvariante Umgebung eines Pixels fest und berechnen von den Grauwerten aus dieser Umgebung rotationsinvariante Merkmale. Diese Umgebung ist im einfachsten Fall ein Kreis, wobei der Radius r des Kreises a priori fest vorgegeben werden muss und sich nach einem A-priori-Wissen richten sollte. Die Umgebung als Kreis mit einem Radius kann auch durch einen unscharfen Kreis, z. B. die symmetrische 2D-Gauß-Funktion, ersetzt werden. Die Rolle des Radius übernimmt dann die Standardabweichung σ der symmetrischen 2D-Gauß-Funktion. Wollen wir aber zusätzlich noch skalierungsinvariante, lokale Merkmale berechnen, so fangen die Schwierigkeiten an. Dann kann die Standardabweichung der Gauß-Funktion nicht mehr vorgeben werden, sie muss sich aus der konkreten Skalierung ergeben, die wir aber nicht kennen. Folglich muss sich die Wahl der Standardabweichung σ oder des Radius r am Bildinhalt orientieren, denn wir benötigen eine Bezugsgröße. Wenn sich diese aber am Bildinhalt orientieren sollen, dann muss in der Umgebung $U_{i,j}$ auch eine Bildinformation tatsächlich vorhanden sein. Eine Umgebung mit einem nahezu konstanten Grauwertverlauf ist dazu sicher ungeeignet. Daher reduziert sich das Problem fast immer auf:

- Selektiere die Pixel in deren Umgebung auch tatsächlich geeignete Bildinformationen vorhanden sind. Die Umgebung muss dabei selbst mit berechnet werden. Die Selektion der Pixel nennt man auch den **Detektor**. Dabei spielt die geometrische Transformation und die photometrische Änderung der Grauwerte eine wichtige Rolle für die Wahl der Umgebung. So wählt man für Ähnlichkeitstransformationen Kreise oder symmetrische Gauß-Funktionen, während man für affine Transformationen Ellipsen oder beliebige Gauß-Funktionen benutzt. Die Berechnung der Umgebung sollte kovariant bezüglich der gewählten geometrischen und photometrischen Transformation sein.
- Berechne aus der bestimmten Umgebung Merkmale, die gegenüber der gewählten geometrischen und photometrischen Transformation invariant sind. Weiterhin sollten diese Invarianten robust gegenüber Störungen und gleichzeitig aber diskriminativ sein. Die Berechnung der Merkmale aus der Umgebung nennt man den **Deskriptor**.

19.9.1 Detektor nach Kadir und Brady

Ein einfaches Beispiel ist der Detektor nach Kadir und Brady [33]. Man betrachte Ähnlichkeitstransformationen und wähle als Umgebung einen Kreis mit dem Radius r . Die Aufgabe ist nun die kovariante Berechnung des Radius r für diejenigen Pixel, in deren Umgebung relevante Informationen vorhanden sind. Als Maß für die Information in der Umgebung benutzen wir die Entropie. Je größer die Entropie, umso mehr Information trägt die Umgebung. Die benötigten Wahrscheinlichkeiten $p_i(x, y, r)$ zur Berechnung der Entropie beziehen wir im einfachsten Fall auf die Grauwerte g_i , man kann aber auch die Wahrscheinlichkeiten von den Grauwerten abgeleiteter Größen benutzen. Die Wahrscheinlichkeiten $p_i(x, y, r)$ hängen vom gewählten Pixel (x, y) und dem Radius r des Kreises als gewählte Umgebung ab. Daher ergibt sich die Entropie zu

$$H(x, y, r) = - \sum_i p_i(x, y, r) \log p_i(x, y, r). \quad (19.160)$$

Pro Pixel ist nun H eine Funktion von r . Man sucht nun ein typisches lokales Maximum dieser Funktion bezüglich r , wir nennen das Argument an dieser Stelle r_{\max} . Zu besseren Bewertung und zum Vergleich der Maxima zwischen den Pixeln berechnen wir an dieser Stelle die Größe

$$W(x, y, r_{\max}) = \sum_i \left| \frac{\partial}{\partial r} p_i(x, y, r_{\max}) \right|. \quad (19.161)$$

Als finales Bewertungsmaß berechnen wir $Y(x, y, r_{\max}) = W(x, y, r_{\max}) \cdot H(x, y, r_{\max})$. Für alle Pixel führt der Detektor nun mit diesem Maß ein Ranking aus und wählt die typischsten Pixel aus. Eine Erweiterung des Detektors auf affin kovariante Umgebungen ist leicht möglich. Dazu wählen wir pro Pixel eine beliebige Ellipse als Umgebung und berechnen wie oben die gleichen Maße. Pro Pixel hängt dann die Ellipse von drei Parametern ab, was den Suchraum und damit den Aufwand erheblich steigert.

Generell wird der *saliency-detector* erfolgreich in der Objekterkennung eingesetzt, weniger erfolgreich wird er zur Korrespondenzfindung in der 3D-Rekonstruktion benutzt.

19.9.2 SIFT-Merkmale

Die SIFT-Merkmale (Scale Invariant Feature Transform) gehen auf D.G. Lowe [42] zurück. Im Folgenden soll nicht die genaue effiziente Implementierung beschrieben werden (siehe dazu D.G. Lowe [42]), sondern es ist unser Ziel die Grundidee zu erläutern. Generell lässt sich der SIFT-Algorithmus in zwei Teile aufteilen: einen Detektor, welcher für ein gegebenes Bild eine Anzahl von Schlüsselpunkten (*keypoints*) auswählt und ein Verfahren welches die Umgebung um jeden dieser Punkte mit einem mehrdimensionalen Deskriptor beschreibt.

SIFT-Detektor Wir wenden uns zunächst der Bestimmung der Schlüsselpunkte zu. Gegeben sei ein Grauwertbild $f_{i,j}$, wobei Ähnlichkeitstransformationen die eigentlichen geometrischen Transformationen darstellen. Als Umgebung wählen wir die symmetrische 2D

Gauß-Funktion mit dem Parameter σ . Folglich muss sich die Wahl der Standardabweichung σ am Bildinhalt orientieren, denn wir benötigen eine Bezugsgröße. Für die Bezugsgröße bieten sich oft Extremwerte von Funktionen an, die sich auf den lokalen Bildinhalt beziehen und skalierungs kovariant sind. Da aber die Umgebung eines beliebigen Pixels nicht unbedingt „Informationen“ beinhaltet (z. B. konstante Grauwertbereiche in denen keine „Struktur“ vorhanden ist), kann man nicht alle Pixel benutzen, sondern nur diejenigen, in deren Umgebung diese Information vorhanden ist. Genau hier setzt die Idee der SIFT-Merkmale an. Es werden also nur diejenigen Pixel (i, j) ermittelt, die bezüglich einer Funktion $S_{i,j}(\sigma)$ ein Extremwertverhalten bezüglich σ zeigen. Dies hängt folglich ganz stark von der gewählten Funktion $S_{i,j}(\sigma)$ ab. Die Funktion $S_{i,j}(\sigma)$ muss Informationen bezüglich der Skalierungen σ widerspiegeln. D.G. Lowe [42] schlägt dazu die Funktion

$$S_{x,y}(\sigma) = \sigma^2 (\Delta G(x, y; 0, \sigma) * f(x, y)) \quad (19.162)$$

vor. Wir falten die Originalfunktion (Bild) $f(x, y)$ mit dem normierten LoG-Filter $\sigma^2 \Delta G$. Die Wahl des LoG-Filters $\sigma^2 \Delta G$ geht auf experimentelle Erfahrungen zurück. Andererseits muss der Filteroutput (zu mindestens theoretisch) translations- und rotationsinvariant sein, was beim LoG-Filter erfüllt ist. Die Normierung von ΔG mit dem Faktor σ^2 beruht auf der *Scale-Space*-Theorie von Lindeberg [41] und bedeutet simpel Skalen-Kovarianz. Skalen-Kovarianz bedeutet konkret: Man filtere ein Bild mit einem Filter einer bestimmten Skala σ an einer Stelle (x, y) . Nun skaliere man das Bild isotrop mit einem Faktor s . Jetzt filtern wir das skalierte Bild mit dem Filter der Skala $\sigma' = s\sigma$ an der skalierten Stelle (sx, sy) . Wenn die beiden Filterergebnisse übereinstimmen, dann liegt Skalierungs-Kovarianz vor. Für den normierten Filter $\sigma^2 \Delta G$ wollen wir die Skalierungs-Kovarianz zeigen. Dazu bezeichne $f(x, y)$ die Grauwertfunktion des Ausgangsbildes und $f'(x', y') = f'(sx, sy) = f(x, y)$ mit $x' = sx, y' = sy$ bezeichne die Skalierung. Die Flächendifferentiale verhalten sich dann zu $dx'dy' = s^2 dxdy$. Entsprechend (2.43) lautet der normalisierte Filter (NLoG)

$$\sigma^2 \Delta G = NLoG(x, y; 0; \sigma) = \sigma^2 LoG = -\frac{1}{\pi\sigma^2} \left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad (19.163)$$

Nun gilt mit $\sigma' = s \cdot \sigma$:

$$\begin{aligned} S'_{sx,sy}(\sigma') &= (NLoG(\cdot; 0, \sigma') * f')(sx, sy) \\ &= \iint_B NLoG(sx - \xi_x, sy - \xi_y; 0, \sigma') \cdot f'(\xi_x, \xi_y) d\xi_x d\xi_y \\ &= -\frac{s^2}{\pi\sigma'^2} \iint_B \left(1 - \frac{(sx - s\xi_x)^2 + (sy - s\xi_y)^2}{2\sigma'^2}\right) \cdot e^{-\frac{(sx - s\xi_x)^2 + (sy - s\xi_y)^2}{2\sigma'^2}} f(\xi_x, \xi_y) d\xi_x d\xi_y \\ &= -\frac{1}{\pi\sigma^2} \iint_B \left(1 - \frac{(x - \xi_x)^2 + (y - \xi_y)^2}{2\sigma^2}\right) \cdot e^{-\frac{(x - \xi_x)^2 + (y - \xi_y)^2}{2\sigma^2}} f(\xi_x, \xi_y) d\xi_x d\xi_y \\ &= (NLoG(\cdot; 0, \sigma) * f)(x, y) = S_{x,y}(\sigma). \end{aligned} \quad (19.164)$$

Skalen-Kovarianz bedeutet folglich

$$S'_{sx,sy}(\sigma') = S_{x,y}(\sigma). \quad (19.165)$$

Bezüglich der Referenzpunkte $(x, y) \leftrightarrow (sx, sy)$ unterscheiden sich die beiden Funktionen in Abhängigkeit von σ nur durch die Skalierung s , d. h. die eine Funktion ist die skalierte Version der anderen Funktion, die Amplituden bleiben gleich. Man sieht nun leicht, dass auch direkt für die symmetrische 2D Gauß-Funktion $G(x, y; 0, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2}\sigma^2}$ die Skalierungs-Kovarianz bezüglich

$$L_{x,y}(\sigma) = G(x, y; 0, \sigma) * f(x, y) \quad (19.166)$$

gilt. Hat in einem Pixel (x, y) die Funktion $S_{x,y}(\sigma)$ einen typischen lokalen Extremwert (Minimum oder Maximum), so muss die skalierte Funktion in dem Referenzpixel (sx, sy) auch solch einen typischen lokalen Extremwert besitzen und beide Extremwerte lassen sich durch die Skalierung s ineinander überführen, sie sind selbst skalierungs kovariant. Da her ermittelt man zunächst alle diejenigen Pixel (x, y) , deren Funktionen $S_{x,y}(\sigma)$ einen typischen Extremwert besitzen. Das ist wieder der Detektor. Natürlich muss bei der praktischen Implementierung noch geklärt werden, was ein typischer lokaler Extremwert ist. Alle Pixel mit diesem typischen Extremwertverhalten werden in beiden Bildern bestimmt, markiert und wir nennen sie Schlüsselpunkte (*keypoints*). Die Liste aller *keypoints* besteht aus Tripeln (x^*, y^*, σ^*) , wobei (x^*, y^*) die Pixelkoordinaten der *keypoints* sind und σ^* die Standardabweichung ist, bei der die Funktion $S_{x^*,y^*}(\sigma)$ das typische Extremum aufweist. Auf die praktische und insbesondere diskrete Implementierung der Filter zur Bestimmung der Funktionen $S_{x,y}(\sigma)$ und auf die Beurteilung der typischen lokalen Extrema gehen wir hier nicht ein, siehe dazu [42]. Wichtig ist nur (2.45) aus Abschn. 2.1.3:

$$G(x, y, 0, k\sigma) - G(x, y, 0, \sigma) \approx (k-1)\sigma^2 \Delta G = (k-1)NLoG. \quad (19.167)$$

Diese besagt, dass sich der *NLoG*-Filter näherungsweise auf die Differenz zweier Gauß-Filter zurückführen lässt, womit eine effiziente Implementierung mit Gauß-Pyramiden möglich ist. Dabei hängt der Faktor k nicht von σ ab und braucht deshalb bei der Extremwertbestimmung nicht berücksichtigt zu werden. Man berechnet somit praktisch die Funktion $L_{x,y}(\sigma)$ und nicht direkt die Funktion $S_{x,y}(\sigma)$. *Keypoints*, die typisch für ver rauschte Umgebungen, homogene Bereiche oder auch Kanten sind, müssen noch eliminiert werden.

SIFT-Deskriptor Nun müssen für die ermittelten *keypoints* Merkmale (Deskriptoren) berechnet werden, die translations-, skalierungs- und rotationsinvariant sind. Dazu benutzen wir im Folgenden die skalierungs kovariante Funktion $L_{x,y}(\sigma)$, da diese sowieso bei der praktischen Implementierung berechnet werden muss. Diese wird berechnet für alle Tripel von *keypoints* (x^*, y^*, σ^*) , d. h. für jeden *keypoint* kann es eine andere Standardabweichung σ^* geben. Damit ist die Skalierungsinvianranz tatsächlich gesichert. Natürlich

müssen auch im Diskreten für die Nachbarpunkte (x, y) eines *keypoints* (x^*, y^*) mit der Standardabweichung σ^* die Funktionswerte $L_{x,y}(\sigma^*)$ berechnet werden, da wir diese zur Berechnung des Gradienten im *keypoint* und dessen Nachbarpunkten benötigen.

Nun müssen aus einer Umgebung U_{x^*,y^*} der *keypoints* mit den Funktionswerten $L_{x,y}(\sigma^*)$ rotationsinvariante Merkmale berechnet werden. Dies geschieht durch Richtungshistogramme, die natürlich auf einer passenden Quantisierung der Winkel von 0 bis 360 Grad beruhen. Da diese nicht rotationsinvariant sind, bestimmt man eine „Hauptrichtung“ und bezieht alle anderen Richtungen relativ (Differenzbildung) auf diese Hauptrichtung. Diese Hauptrichtung könnte man mit dem Strukturtensor bestimmen. In der Originalarbeit Lowe [42] wird diese aus einem gewichteten Richtungshistogramm bestimmt, indem man die häufigste Richtung wählt, d. h. dort wo der maximale *Peak* im Histogramm vorliegt. Dies hat aber den Nachteil, dass ein *Peak* im Richtungshistogramm nicht eindeutig im Sinne von kleinen Schwankungen sein muss. Daher kann es sein, dass man mehrere Hauptrichtungen pro *keypoint* berechnen muss.

Nun teilt man die Umgebung des *keypoints* in mehrere Regionen ein, berechnet pro Region das Richtungshistogramm, bezieht dieses auf die berechnete Hauptrichtung, und fügt alle Richtungshistogramme der Regionen zu einem einzigen Merkmalsvektor pro *keypoint* zusammen. Dieser Merkmalsvektor wird noch auf die Länge Eins normiert und wird SIFT-Deskriptor genannt.

Bemerkung Wie zu Beginn dieses Abschnittes festgestellt wurde, spielt bei lokalen Merkmalen die Wahl der Umgebung die entscheidende Rolle. Durch die Berechnung der *keypoints* und deren Umgebung σ^* ist dies bezüglich Ähnlichkeitstransformationen bei den SIFT-Merkmalen recht gut gelungen. Die Wahl der invarianten Merkmale durch Richtungshistogramme erscheint bei den SIFT-Merkmalen doch recht willkürlich und heuristisch zu sein. Im Prinzip kann man nun bezüglich Ähnlichkeitstransformationen alle invarianten Merkmale benutzen, man muss sie nur bezüglich der ermittelten Gauß-Umgebung (σ) berechnen, sie müssen robust sein und gleichzeitig diskriminativ. Die SIFT-Deskriptoren werden besonders erfolgreich in der Korrespondenzfindung bezüglich der 3D-Rekonstruktion aus mehreren Ansichten eingesetzt.

19.9.3 HOG-Merkmale

Die sogenannten HOG-Merkmale (*Histogram of Oriented Gradients*) sind weiter nichts als Richtungshistogramme. Sie wurden z. B. von Dalal und Triggs [18] erfolgreich zur Detektion von Personen in Videos oder Bildern benutzt. Sie besitzen eine gewisse Ähnlichkeit mit den SIFT-Richtungshistogrammen, unterscheiden sich aber im Wesentlichen durch den Detektor. Die HOG-Merkmale werden nicht nur einzelnen *keypoints* zugeordnet, sondern das gesamte Bild (oder eine Bildregion) wird gleichmäßig gerastert in kleine Zellen, diese können kleine Quadrate oder Kreise sein. Für jede Zelle wird ein Richtungshistogramm berechnet und alle Richtungshistogramme ergeben zusammen gehängt den HOG-

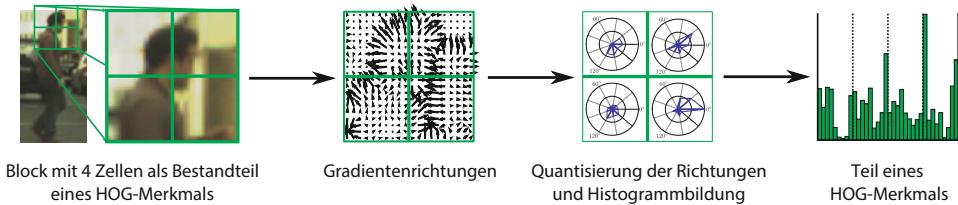


Abb. 19.13 Berechnung der Richtungshistogramme für ein Block des HOG-Merkmalss

Merkmalsvektor. Um noch eine gewisse Invarianz gegenüber photometrischen Einflüssen zu erzielen, werden einige Zellen zu Blöcken zusammengefasst, wobei sich die Blöcke überlappen können. Pro Block werden dann die Richtungshistogramme aller Zellen dieses Blockes auf die Länge Eins normiert. In Abb. 19.13 ist die Berechnung des Histogramms eines einzelnen Blockes dargestellt.

Die drei Basisparameter des Verfahrens sind die Größe einer Zelle, die Größe eines Blockes und die Quantisierung der Richtungen in den Richtungshistogrammen. Für bestimmte Klassifikationsaufgaben muss dann in Experimenten die optimale Parameterauswahl erfolgen. Je nachdem die Blöcke/Zellen Quadrate oder Kreise sind, spricht man von R-HOG-Features oder C-HOG-Features. In [18] werden noch viele kleinere Implementierungsdetails besprochen, wie etwa der Durchführung einer linearen Interpolation zwischen den Zellen.

19.9.4 Local Binary Patterns (LBP)

Die Anwendung der LBP ist ähnlich der Anwendung der HOG-Merkmalss. Die LBP gehen auf Ojala et. al.(1994) zurück [52] und werden erfolgreich bei der Texturklassifikation und Personenerkennung eingesetzt. Zunächst berechnen wir pro Pixel einen neuen Merkmalswert mit Hilfe der Pixel-Umgebung. Im einfachsten Fall wählen wir für ein Pixel (i, j) die Achternachbarschaft (1.5) und legen eine Orientierung im Uhrzeigersinn fest. Beginnend bei irgendeinem der acht Nachbarn zählen wir die acht Grauwerte in dieser Reihenfolge auf und schreiben sie in einen Vektor. Dieser Vektor wird nun folgendermaßen binarisiert: wenn der Grauwert $f_{i,j}$ im zentralen Pixel größer als ein Grauwert in diesem Vektor ist, dann ersetzen wir den Grauwert durch eine Eins, ansonsten durch eine Null. Damit erhalten wir in diesem einfachen Falle eine Folge von acht Bits und interpretieren diese als eine Achtbitzahl von 0 bis 255. Das Ergebnisbild f' können wir als Ergebnis einer nichtlinearen Filterung $f' = Tf$ von f betrachten. Das Ergebnisbild f' wird nun genau wie bei den HOG-Merkmalss in Zellen eingeteilt und pro Zelle direkt die Histogramme der Funktionswerte f' berechnet (nicht das Histogramm der Gradientenrichtungen). Probleme bereitet natürlich die Rotationsinvarianz von f' bezüglich Rotationen von f , diese ist so nicht gegeben. Dazu müssen wir die gewöhnliche Achternachbarschaft eines Pixels (i, j) durch eine

kreisförmige Umgebung ersetzen. Wir betrachten dazu im Bild f in der Umgebung des Pixels (i, j) einen Kreis eines bestimmten Radius und berechnen in diskreten Abständen auf dem Kreis durch Interpolation die gewünschten Funktionswerte. Nun werden diese Funktionswerte wie oben beschrieben in eine Binärzahl transformiert, wobei wir die zirkulare Aufzählung bei einer normierten Hauptrichtung, ähnlich den HOG-Merkmalen, beginnen. Damit haben wir eine Rotationsinvarianz erreicht. Weiterhin können wir die Aufzählung der interpolierten Werte auf dem Kreis als eine zyklische, diskrete Funktion interpretieren und mittels Verschiebungsinvarianten neue Merkmale berechnen, die gegenüber Rotationen von f invariant sind, siehe dazu z. B. die Fouriertransformation (eindimensionale DFT).

Was sind nun Vorteile gegenüber den HOG-Merkmalen? Die Transformation T von f in f' ist trivialerweise invariant gegenüber monotonen, photometrischen Änderungen des Bildes f . Diese Änderungen dürfen auch nichtlinear sein, Hauptsache sie sind monoton. Damit sind die LBP-Merkmale robuster gegenüber photometrischen Schwankungen. Weiterhin ist durch die einfache Berechnung der LBP eine effiziente Implementierung möglich. Durch die Binarisierung haben wir allerdings einen gewissen Informationsverlust hinzunehmen.

Eine völlig andere Nutzung der einem Pixel zugeordneten Folge von Bits besteht in der Ermittlung der Anzahl der Übergänge von 1 zu 0 und umgekehrt, wobei dies zyklisch zu interpretieren ist. So hat (0001110) zwei Übergänge, (00000000) keinen Übergang, (0101010101) zehn Übergänge usw. LBP werden als uniforme LBP bezeichnet, wenn das LBP höchstens zwei Übergänge besitzt. Bei einem 8-Bit-Muster gibt es genau 58 verschiedene uniforme LBP, alle anderen sind nicht uniform. In der Texturklassifikation vergibt man daher bei 8-Bit-Mustern 59 Label, 58 für die uniformen LBP und 1 gemeinsames Label für alle nicht uniformen LBP. Die uniformen LBP kann man als Mikro-Texton einer Textur auffassen und folglich zur Texturklassifikation nutzen.

19.9.5 Statistiken von lokalen Merkmalen

Lokale Merkmale, wie etwa die in den vorherigen Abschnitten vorgestellten Verfahren LBP und SIFT, eignen sich besonders gut für die Berechnung von exakten Korrespondenzen zwischen zwei Bildern. Dabei versucht man die gegebenen Punktmengen mit ihren Deskriptoren mit einem geeigneten Matchingverfahren zu registrieren. Bei Erkennungsaufgaben, wie wir sie im Abschn. 20.2 noch näher kennenlernen werden, ist aber oft eine exakte Korrespondenzfindung zwischen zwei Bildern weder notwendig noch möglich. Dies liegt oft daran, dass man es hier mit Transformationen zu tun hat, die sich nicht genau geometrisch modellieren lassen, wie etwa die Transformation zwischen unterschiedlichen Tassen oder anderen Gegenständen. Ziel ist es daher nicht einen lokalen Bereich um einen einzelnen Schlüsselpunkt mit einem Merkmal zu beschreiben sondern einen globalen Merkmalsvektor für das ganze Bild oder eine Bildregion zu bestimmen. Wie bereits erläutert lässt sich dies mit HOG oder LBP Merkmalen realisieren. Durch deren feste git-

terförmige Einteilung sind diese jedoch oft zu starr um eine gewisse Invarianz auch bei starken Deformationen zu gewährleisten. Einen Ausweg bietet hier eine Repräsentation des Bildes, welche die Positionen lokaler Merkmale komplett ignoriert und nur eine reine Statistik aufgrund der auftretenden Deskriptoren erstellt. Diese Idee wird als *Bag-of-Features*- oder *Bag-of-Words*-Ansatz bezeichnet.

Der *Bag-of-Words* (BoW) Ansatz kommt ursprünglich aus dem Bereich der Textkategorisierung. Gegeben ist dort ein Text, welcher einer Rubrik (Klasse) zugeordnet werden soll, zum Beispiel um Werbetexte von persönlichen Briefen zu trennen, eine Anwendung welche in Zeiten der Überflutung mit Spammnachrichten eine hohe Aufmerksamkeit geschenkt wird. Ein BoW-Merkmal ist dabei ein einfaches Histogramm der Häufigkeiten von bestimmten Schlüsselworten, wie etwa „Angebot“, „günstig“ oder „bestellen“. Im Bereich der visuellen Erkennung möchte man die Häufigkeit von visuellen Worten bestimmen. Wenn wir uns nun eine Menge von Bildern der Kategorie „Auto“ vorstellen, so lassen sich hypothetisch einzelne Objektteile als Cluster von lokalen Merkmalen identifizieren. Theoretisches Ziel der BoW-Merkmale in der Bildverarbeitung ist es daher ein Bild als Menge von interessanter Teile zu beschreiben.

Wir bezeichnen im Folgenden die Menge der Deskriptoren (Merkmalsvektoren der Schlüsselpunkte) eines Bildes mit $\mathcal{L} = \{\mathbf{l}^{(k)}\}_{k=1}^W$, wobei jeder einzelne Deskriptor Element der Menge \mathcal{U} ist. Bei SIFT-Deskriptoren ist dies zum Beispiel der \mathbb{R}^{128} . Die Berechnung von BoW-Merkmalen erfolgt nun in zwei Schritten: (1) Quantisierung von lokalen Merkmalen und (2) Berechnung eines Histogramms. Im ersten Schritt muss die Menge \mathcal{U} in Regionen eingeteilt werden, zum Beispiel mit dem k -Means-Algorithmus (Abschn. 18.7.1). Diese Einteilung kann dann für eine Quantisierung (Abschn. 1.1) verwendet werden, d. h. man erhält eine Funktion $q : \mathcal{U} \rightarrow \{1, \dots, n_q\}$, welche jeden möglichen Deskriptor einen Cluster zuweist. Im Zusammenhang mit dem BoW-Ansatz spricht man hier auch oft von einem Kodebuch.

Die Berechnung des Histogramms $\mathbf{h} = (h_1, \dots, h_{n_q})^T$ eines Bildes ist bei gegebener Quantisierung ein einfacher Schritt, welcher aber viele Möglichkeiten aufgrund von unterschiedlichen Normierungen bietet. Ein einfacher Ansatz ist es die, absoluten Häufigkeiten c_j direkt zu verwenden:

$$\tilde{h}_j = c_j = |\{k \mid q(\mathbf{l}^{(k)}) = j\}|. \quad (19.168)$$

Diese sind natürlich anfällig bei unterschiedlichen Bildgrößen und einer unterschiedlichen Anzahl W von lokalen Merkmalen. Daher werden oft relative Häufigkeiten verwendet:

$$h_j = \frac{1}{W} \cdot c_j. \quad (19.169)$$

Durch die einfache Umsetzung dieses Ansatzes und der großen Flexibilität und Robustheit bezüglich starker Objektvariationen, hat sich die BoW-Methode als eines der Hauptwerkzeuge in der Merkmalsberechnung etabliert. Mit der steigenden Popularität sind natürlich auch eine Vielzahl von Erweiterungen und Modifikationen entstanden. So gibt es noch zahlreiche andere Normierungsmöglichkeiten der Histogramme (z. B. Binarisierung)

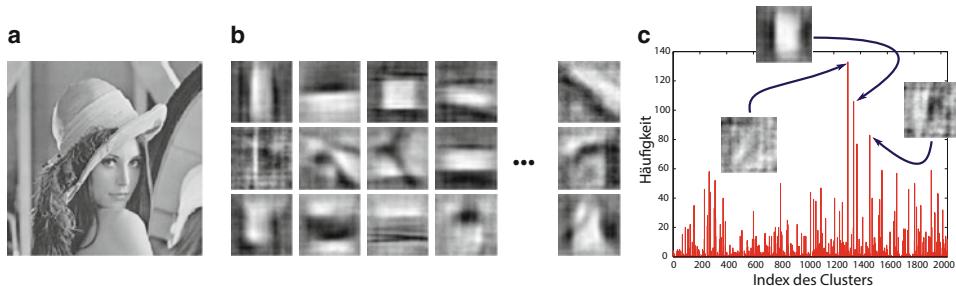


Abb. 19.14 Darstellung der Berechnung eines BoW-Merkmales für ein gegebenes Eingabebild. Als lokale Merkmale dienen in diesem Beispiel einfache Richtungshistogramme und Prototypen werden anhand der Visualisierungsidee von [82] als Grauwertbild dargestellt. **a** Eingabebild, **b** Prototypen eines Kodebuchs, **c** BoW-Histogramm nach Quantisierung

sowie Clusterverfahren, welche gezielt versuchen diskriminative visuelle Worte zu finden. Eine Verwendung dieser Merkmale in der Objekterkennung werden wir in Kap. 20 sehen. Abbildung 19.14 zeigt anhand eines Beispiels die Berechnung von BoW-Merkmalen.

Viele Studien mit Erkennungssystemen welche auf diesen Merkmalen basieren zeigen, dass die Wahl des Quantisierungsverfahrens selbst eine eher geringe Auswirkung auf die Erkennungsleistung hat. Besondere Bedeutung hat hingegen die Art der Histogrammberechnung, welcher oft als Kodierungsschritt bezeichnet wird. So kann zum Beispiel ein lokaler Deskriptor seinen k nächsten Clustern zugewiesen werden und dies auch gewichtet relativ zur Distanz zu den jeweiligen Prototypen. Dem interessierten Leser sei die Arbeit von Coates und Ng empfohlen [15], welche eine experimentelle Untersuchung dieser Aspekte bietet.

19.9.6 Schnelle Berechnungen mit Integralbildern

Zur Berechnung von Merkmalen aus Bildern werden häufig Summen von Grauwerten bzw. Summen anderer Merkmale über einen Ausschnitt des Bildes, im einfachsten Fall über einen rechteckigen Block, benötigt. Dies kann man effizient mit konstanter Komplexität mit Hilfe der Integralbilder (engl.: *integral images*) realisieren. Die Idee ist recht simpel und ist ähnlich der Idee aus der Wahrscheinlichkeitsrechnung zur Berechnung der Wahrscheinlichkeit über ein Rechteck. Diese wird einfach auf simple Additionen und Differenzen an den Eckpunkten des Rechteckes bezüglich der Verteilungsfunktion (Integralbild) zurückgeführt. Es sei $f_{i,j}$ die Grauwertfunktion eines diskreten Bildes. Das *integral image* $I_{i,j}$ ist dann:

$$I_{i,j} = \sum_{k \leq i} \sum_{l \leq j} f_{k,l}. \quad (19.170)$$

Dieses kann in einem einzigen Bilddurchlauf rekursiv berechnet werden:

$$I_{i,j} = I_{i-1,j} + S_{i,j}, \quad S_{i,j} = S_{i,j-1} + f_{i,j}, \quad I_{-1,j} = 0, \quad S_{i,-1} = 0. \quad (19.171)$$

Es seien nun die vier Punkte $(m_1, n_1), (m_2, n_1), (m_2, n_2), (m_1, n_2)$ die Eckpunkte eines Rechteckes. Dann kann die Summe aller Werte $f_{i,j}$ über dem Rechteck simpel durch

$$P_{\text{Rechteck}} = \sum_{m_1 \leq k \leq m_2} \sum_{n_1 \leq l \leq n_2} f_{k,l} = I_{m_1, n_1} + I_{m_2, n_2} - I_{m_1, n_1} - I_{m_2, n_1} \quad (19.172)$$

berechnet werden. Wie man leicht sieht, ist die Komplexität konstant bezüglich der Größe des Rechteckes. In der Arbeit von Viola und Jones [81] werden Merkmale von fünf Rechtecken berechnet, die sich mit einem *integral image* mit konstanter Komplexität berechnen lassen. Auch die Implementierung von DoB-Filtern lässt sich mit konstanter Komplexität durch die *integral images* realisieren, siehe (7.12) in Abschn. 7.5. Die *integral images* lassen sich leicht erweitern auf um 45 Grad gedrehte Rechtecke, die durch den Rand des Bildes beschnitten sein können, sowie auf 3D-*integral images*.

Eine wichtige Anwendung finden Integralbilder auch bei der schnellen Berechnung von Histogrammen von bestimmten Regionen des Bildes. Dies ist besonders wichtig bei der Berechnung der HOG oder auch BoW-Merkmale, welche wir in den letzten Abschnitten kennengelernt haben. Bei HOG-Merkmalen müssen Gradientenrichtungshistogramme für verschiedene, oft auch überlappende, Regionen im Bild berechnet werden. Die Bestimmung der Gradientenrichtung $d_{k,l}$ an einem einzelnen Pixel (k, l) wurde in Abschn. 8.2 besprochen. Diese kann auch in quantisierter Form vorliegen und in K Richtungen eingeteilt sein $g_{k,l} \in \{1, \dots, K\}$. Ausgangspunkt für die Berechnung von Histogrammen von d ist dann ein Bild f , welches für jedes Pixel einen K -dimensionalen Vektor enthält, welcher nur eine 1 an der Position von $d_{k,l}$ enthält und ansonsten null ist. Das Integralbild dieses mehrdimensionalen Bildes (K Grauwertbilder) kann direkt für die effiziente Berechnung von Teilhistogrammen genutzt werden. Dabei sind die Berechnungen analog zur Beschreibung für Grauwertbilder.

19.10 Transformation von Merkmalen

Oft sind Merkmale in ihrer ursprünglichen Form nicht anwendbar. Der Merkmalsraum ist oft zu hochdimensional oder die Merkmale sind nicht konsistent zueinander. Weiterhin möchte man redundante Merkmale ermitteln oder in der Mustererkennung Merkmale, die nicht diskriminativ sind. Dazu werden die Merkmale oft einer Transformation unterworfen und dann analysiert.

19.10.1 Principal Component Analysis (PCA)

Die bekannteste, einfachste und zugleich lineare Transformation ist die Principal Component Analysis (Hauptkomponentenanalyse). Sie wird auch als Karhunen-Loeve-Transformation bezeichnet.

Der Merkmalsraum werde durch n -dimensionale Merkmalsvektoren $\mathbf{x}^T = (x_1, \dots, x_n)$ beschrieben. Die Zugehörigkeit von \mathbf{x} zu einer Klasse soll keine Rolle spielen. Wir betrachten nun eine lineare, orthogonale Koordinatentransformation. Wenn die Matrix quadratisch ist, dann ist dies speziell eine Rotation:

$$\mathbf{x} = \mathbf{Ay}, \quad \mathbf{AA}^T = \mathbf{A}^T \mathbf{A} = \mathbf{I}. \quad (19.173)$$

Es sei dabei \mathbf{I} die Einheitsmatrix, folglich bilden die Spaltenvektoren (aber auch die Zeilenvektoren) $\mathbf{a}_i, i = 1, \dots, n$ ein Orthonormalsystem, sind also Basisvektoren, die eine Orthonormalsbasis bilden. Equivalent zur Schreibweise (19.173) ist die Entwicklung von \mathbf{x} nach der Basis:

$$\mathbf{x} = y_1 \mathbf{a}_1 + y_2 \mathbf{a}_2 + \dots + y_n \mathbf{a}_n. \quad (19.174)$$

Wir wollen im Sinne einer Orthogonalprojektion auf einen m -dimensionalen Unterraum den Merkmalsvektor \mathbf{x} durch eine Linearkombination der „ersten“ m Basisvektoren $\mathbf{a}_i, i = 1, \dots, m$ ersetzen. Wir wissen sofort: sind diese „ersten“ Basisvektoren gegeben, dann können wir die Koeffizienten y_i vor den Basisvektoren über das Skalarprodukt $y_i = \langle \mathbf{x}, \mathbf{a}_i \rangle$ ausrechnen. Diese sogenannten verallgemeinerten Fourierkoeffizienten hängen aber vom konkreten Merkmalsvektor \mathbf{x} ab, wir wollen dagegen eine beste Approximation für alle Merkmalsvektoren haben. Daher gehen wir zu Zufallsvariablen über. Der Merkmalsvektor \mathbf{x} wird zum Zufallsvektor \mathbf{X} , analog geht \mathbf{y} in \mathbf{Y} über. Da wir das Koordinatensystem rotieren, müssen wir vorher die Translation eliminieren. Dies tun wir wie üblich mit der Annahme $E(\mathbf{X}) = \mathbf{0}$. Ist dies nicht der Fall, dann normieren wir $Z = \mathbf{X} - E(\mathbf{X})$, damit $E(Z) = 0$ gilt. Im Folgenden gehen wir immer von $E(\mathbf{X}) = \mathbf{0}$ aus. Bei fest vorgegebenem m müssen wir denjenigen Unterraum bestimmen, der im Mittel alle Merkmalsvektoren am besten approximiert:

$$R_m(\mathbf{A}) = E \left(\left[\mathbf{X} - \sum_{i=1}^m a_i Y_i \right]^2 \right) = E \left(\left[\sum_{i=m+1}^n a_i Y_i \right]^2 \right) \rightarrow \text{Minimum.} \quad (19.175)$$

mit

$$g(\mathbf{a}_i) = \mathbf{a}_i^T \mathbf{a}_i - 1 = 0. \quad (19.176)$$

Die Basisvektoren \mathbf{a}_i sind jetzt selbst die gesuchten Größen. Wir wollen nun die Zielfunktion $R_m(\mathbf{A})$ etwas anders schreiben. Auf Grund der Orthonormalität gilt $Y_i = \mathbf{a}_i^T \mathbf{X} = \mathbf{X}^T \mathbf{a}_i$. Weiterhin ist

$$(\mathbf{a}_i \mathbf{X}^T \mathbf{a}_i)^2 = (\mathbf{a}_i \mathbf{X}^T \mathbf{a}_i)^T (\mathbf{a}_i \mathbf{X}^T \mathbf{a}_i) = \mathbf{a}_i^T \mathbf{X} \mathbf{a}_i^T \mathbf{a}_i \mathbf{X}^T \mathbf{a}_i = \mathbf{a}_i^T \mathbf{X} \mathbf{X}^T \mathbf{a}_i. \quad (19.177)$$

Damit wird (19.175) zu:

$$\begin{aligned} R_m(\mathbf{A}) &= E \left(\left[\sum_{i=m+1}^n a_i Y_i \right]^2 \right) = \sum_{i=m+1}^n \mathbf{a}_i^T E(\mathbf{X}\mathbf{X}^T) \mathbf{a}_i \\ &= \sum_{i=m+1}^n \mathbf{a}_i^T \Sigma_{\mathbf{X}} \mathbf{a}_i \rightarrow \text{Minimum.} \end{aligned} \quad (19.178)$$

mit

$$g(\mathbf{a}_i) = \mathbf{a}_i^T \mathbf{a}_i - 1 = 0. \quad (19.179)$$

Dabei ist $\Sigma_{\mathbf{X}}$ die Kovarianzmatrix bezüglich \mathbf{X} . Alle Summanden in (19.178) sind bezüglich der unbekannten und gesuchten Basisvektoren \mathbf{a}_i entkoppelt. Damit wird die Zielfunktion genau dann minimal, wenn jeder einzelne Summand minimal wird:

$$z = \mathbf{a}_i^T \Sigma_{\mathbf{X}} \mathbf{a}_i \rightarrow \text{Minimum bei } \mathbf{a}_i^T \mathbf{a}_i - 1 = 0, \quad i = 1, \dots, n. \quad (19.180)$$

Wir bilden die Lagrange-Funktionen:

$$L_i = \mathbf{a}_i^T \Sigma_{\mathbf{X}} \mathbf{a}_i - \lambda_i (\mathbf{a}_i^T \mathbf{a}_i - 1), \quad i = 1, \dots, n. \quad (19.181)$$

Von diesen bestimmen wir die Extremwerte:

$$\nabla_{\mathbf{a}_i} L_i = 2 \Sigma_{\mathbf{X}} \mathbf{a}_i - 2 \lambda_i \mathbf{a}_i = 0, \quad i = 1, \dots, n. \quad (19.182)$$

Daraus ergeben sich offensichtlich die Eigenwertprobleme:

$$\Sigma_{\mathbf{X}} \mathbf{a}_i = \lambda_i \mathbf{a}_i, \quad i = 1, \dots, n. \quad (19.183)$$

Da die Kovarianzmatrix $\Sigma_{\mathbf{X}}$ symmetrisch und positiv semidefinit ist, gilt $\lambda_i \geq 0$, $i = 1, \dots, n$ für die Eigenwerte. Setzt man die rechte Seite von (19.183) in das Zielfunktional ein, so ergibt sich

$$R_m(\mathbf{A}) = \sum_{i=m+1}^n \mathbf{a}_i^T \Sigma_{\mathbf{X}} \mathbf{a}_i = \sum_{i=m+1}^n \mathbf{a}_i^T \lambda_i \mathbf{a}_i = \sum_{i=m+1}^n \lambda_i. \quad (19.184)$$

Ordnet man nun diese nichtnegativen Eigenwerte der Größe nach und wir bezeichnen sie mit

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n, \quad (19.185)$$

dann ist die Lösung des Extremalproblems (19.175) offensichtlich:

- Die Spaltenvektoren der Transformationsmatrix \mathbf{A} sind die orthonormierten Eigenvektoren der Kovarianzmatrix $\Sigma_{\mathbf{X}}$ des Zufallsvektors aller Merkmale.
- Der Unterraum der vorgegebenen Dimension m mit der im Mittel besten Approximation aller Merkmale wird durch die m Eigenvektoren aufgespannt, die zu den m größten Eigenwerten der Kovarianzmatrix gehören.
- Nun soll die Kovarianzmatrix der transformierten Merkmale Y_i berechnet werden. Es sei $\mathbf{A}_m = (\mathbf{a}_1, \dots, \mathbf{a}_m)$ und $\mathbf{X} = \mathbf{A}_m \mathbf{Y}$ bzw. $\mathbf{Y} = \mathbf{A}_m^T \mathbf{X}$. Damit ist $E(\mathbf{Y}\mathbf{Y}^T) = \mathbf{A}_m^T E(\mathbf{X}\mathbf{X}^T) \mathbf{A}_m$ und folglich

$$E(\mathbf{Y}\mathbf{Y}^T) = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_m \end{pmatrix}. \quad (19.186)$$

- Die Komponenten von \mathbf{Y} sind somit dekorreliert, die Varianzen der Komponenten sind gerade die berechneten Eigenwerte.

Ist $m = 0$ oder $m = n$, dann hat die Optimierung eigentlich keinen Sinn mehr. Dennoch ist der Fall $m = n$ interessant, wenn man lediglich fordert, dass die Merkmale im transformierten Raum dekorreliert sein sollen. Die PCA könnte man auch als „statistische Hauptachsentransformation“ bezeichnen. Wir rotieren so, dass die Kovarianzmatrix zur Diagonalmatrix wird. Anschaulich betrachten wir Realisierungen \mathbf{x}_i des Zufallsvektors \mathbf{X} . Diese Realisierungen bilden im Merkmalsraum eine Punktwolke. Von dieser diskreten Punktmenge berechnen wir die diskreten Momente bis zweiter Ordnung, dies ist die Kovarianzmatrix. Mit diesen Momenten können wir nun die Trägheitsellipse dieses Punkteobjektes berechnen, siehe auch Abschn. 19.3. Mit der PCA rotieren wir nun die Punktmenge so, dass die zugehörige Trägheitsellipse in Hauptachsenlage gerät. Für die PCA benötigen wir also immer eine Punktwolke, ein einzelner Punkt genügt nicht, denn dann ist $E(\mathbf{X}\mathbf{X}^T) = \mathbf{x}\mathbf{x}^T$. Folglich wird das Eigenwertproblem trivial zu:

$$(\mathbf{x}\mathbf{x}^T)\mathbf{x} = \mathbf{x}(\mathbf{x}^T\mathbf{x}) = \lambda\mathbf{x} \rightarrow \lambda = |\mathbf{x}|^2. \quad (19.187)$$

Der Vektor \mathbf{x} ist damit selbst Basisvektor und das Problem ist unsinnig gestellt. Im Folgenden sollen noch einige typische Anwendungsbeispiele der PCA angegeben werden.

- **Beispiel 1** (Datenkompression):

Da wir bei der Herleitung der PCA den Unterraum mit der besten Approximation gewählt haben, eignet sich die PCA auch zur Datenkompression von Bildern. Die optimale Basis für ein einziges Bild zu berechnen, ist nach (19.187) Unsinn. Wir müssen dies für eine Punktwolke tun, dies ist folglich bei Bildern ein Menge von Bildern oder eine bestimmte Grundgesamtheit von Bildern. Ein Merkmalsvektor \mathbf{x} ist dann ein vollständiges Bild, der Zufallsvektor \mathbf{X} wird dann auch als stochastisches Feld bezeichnet. Die Kovarianzmatrix besteht dann aus den Kovarianzen $E(\mathbf{X}_i\mathbf{X}_j)$ und wird als Kovarianzfunktion

oder Autokorrelationsfunktion bezeichnet. Diese Autokovarianzfunktion muss nun für eine Grundgesamtheit von Bildern ermittelt werden. Bei zweidimensionalen Bildern hängt diese Autokovarianzfunktion von vier Variablen ab. Bei kleinen (256,256)-Bildern hätte die Kovarianzmatrix die Dimension $(2^{16}, 2^{16})$). Dasher müsste man wie bei der DCT das Bild in kleine Blöcke einteilen und nur diese Blöcke betrachten. Haben wir nun einen Koder und Dekoder, so müssen neben den berechneten y_i auch die Basisvektoren \mathbf{a}_i übermittelt werden. Der Aufwand wird also immer günstiger, wenn die Basisvektoren bezüglich einer sehr großen Klasse ermittelt werden, anderseits sinkt dann der Kompressionsgrad.

- **Beispiel 2** (Merkmalsbewertung):

Für diskriminatorische Aussagen bezüglich der Merkmale in der Mustererkennung ist die PCA nicht geeignet. Dennoch wird sie häufig zur Merkmalsbewertung herangezogen. Dann sollte man aber wenigstens beachten, dass Merkmalsbewertungen völlig unabhängig von Translationen und anisotropen Skalierungen sind. Deshalb normieren wir die Merkmale wie üblich durch:

$$Z_i = \frac{X_i - E(X_i)}{\sigma(X_i)}. \quad (19.188)$$

Die Kovarianzmatrix von Z entspricht dann der Korrelationsmatrix von X . Benutzt man z. B. auf den Originalmerkmalsvektoren x_i zur Klassifikation (z. B. als Matchingmaß) die Euklidische Metrik, obwohl die Merkmale zueinander nicht „passen“, dann kann es sehr schlechte Klassifikationsergebnisse geben. In diesem Falle sollte man besser die Mahalanobismetrik (19.80) benutzen. Wenn die Merkmale unabhängig sind oder nicht sehr redundant, kann man auch statt dessen die Originalmerkmale durch (19.188) transformieren und anschließend benutzt man zur Klassifikation die Euklidische Metrik bezüglich der transformierten Merkmale.

- **Beispiel 3** (Whitening Transformation):

Unter einem Zufallsvektor einer bestimmten Verteilung kann man sich immer als Realisierungen eine Punktwolke einer bestimmten Struktur vorstellen. Für Testzwecke gibt man sich die theoretische Verteilung vor und möchte die Punktwolke erzeugen, folglich Pseudozufallszahlen erzeugen. Oft sind in Softwaresystemen eindimensionale Zufallszahlengeneratoren eingebaut und diese beschränken sich meist auf Gleichverteilungen und Normalverteilungen. Für Zufallsvektoren sind Zufallszahlengeneratoren viel schwieriger zu implementieren. Wollen wir nun mehrdimensionale, normalverteilte Zufallszahlen erzeugen, so können wir die PCA benutzen. Eine mehrdimensionale Normalverteilung ist eindeutig durch den Erwartungswertvektor $E(\mathbf{X})$ und die Kovarianzmatrix $\Sigma_{\mathbf{X}}$ bestimmt. Den Erwartungswertvektor $E(\mathbf{X})$ eliminieren wir leicht durch die übliche Normierung $\mathbf{X}' = \mathbf{X} - E(\mathbf{X})$, damit ist $E(\mathbf{X}') = \mathbf{0}$. Die Transformation $\mathbf{Y} = \mathbf{A}^T \mathbf{X}'$ (PCA) führt auf dekorrelierte Zufallsvariablen \mathbf{Y} . Wenn \mathbf{X}' normalverteilt ist, dann ist auch \mathbf{Y} normalverteilt. Bei normalverteilten Zufallsvariablen ist Dekorrelation und Unabhängigkeit dasselbe. Die Kovarianzmatrix von \mathbf{Y} ist eine Diagonalmatrix Λ , wobei in der Hauptdiagonalen die Eigenwerte (Varianzen) der Größe nach geordnet stehen. Die-

se Diagonalmatrix transformieren wir noch zur Einheitsmatrix durch eine anisotrope Skalierung, damit erhalten wir als Gesamttransformation:

$$\mathbf{Y} = \Lambda^{-\frac{1}{2}} \mathbf{A}^T (\mathbf{X} - E(\mathbf{X})). \quad (19.189)$$

Diese Transformation (19.189) wird als *whitening transformation* bezeichnet. Die Komponenten von \mathbf{Y} bestehen folglich alle aus unabhängigen, $N(0, 1)$ -verteilten Zufallsvariablen. Damit lassen sich unabhängig für alle Komponenten sehr leicht eine Probe ziehen, das Ergebnis sei der Probenvektor \mathbf{y} . Nun transformieren wir diesen durch

$$\mathbf{x} = \mathbf{A} \Lambda^{+\frac{1}{2}} \mathbf{y} + E(\mathbf{X}) \quad (19.190)$$

zurück und erhalten die gewünschte Probe \mathbf{x} .

- **Beispiel 4** (Matching):

Diese Anwendung hat eigentlich wieder etwas mit der Datenkompression zu tun. Wir wollen in einer Szene ein 3D-Objekt erkennen, wir wissen aber nicht, welche Ansicht des Objektes vorliegt. Wir nehmen deshalb ein Objekt aus vielen, verschiedenen Positionen auf und speichern diese pro Objekt in einer Datenbank. Bei vielen Objekten mit vielen Ansichten ist eine Fülle von Daten zu verwalten. Das Bild eines unbekannten Objektes wird dann mit allen Ansichten in der Datenbank verglichen (*matching*). Auf Grund der ungeheueren Fülle von Daten ist dies fast unmöglich zu realisieren. Deshalb benutzt man oft die sogenannte *Eigenspace*-Methode oder Eigenraummethode, die 1991 Turk und Pentland [77] einführten. Ein Bild – eine Ansicht – wird zeilenweise als ein Gesamtvektor aufgefasst. Nun bilden wir den Mittelwert $\bar{\mathbf{x}}$ über alle Ansichten und Objekte aus der Datenbank und betrachten die Matrix:

$$\mathbf{X} = ((\mathbf{x}_1 - \bar{\mathbf{x}}), (\mathbf{x}_2 - \bar{\mathbf{x}}), \dots, (\mathbf{x}_n - \bar{\mathbf{x}})). \quad (19.191)$$

Die Matrix $\mathbf{Q} = \mathbf{X}\mathbf{X}^T$ ist dann die Gesamtkovarianzmatrix. Diese unterziehen wir einer PCA und wählen eine geeignete Anzahl von Basisvektoren aus. In der Datenbank werden nun diese Basis und für jedes Bild die Koeffizienten y_i bezüglich dieser Basis abgespeichert. Ein unbekanntes Bild wird nun auch bezüglich dieser Basis dargestellt und es erfolgt ein Matching bezüglich der Basiskoeffizienten. Das erfolgreiche Matching setzt aber einiges voraus, z. B. keine starken projektiven Verzerrungen, nur ein Objekt im Bild, Normierungen bezüglich Helligkeit und Größe des Objektes. Bei der Gesichtserkennung kann man sich dazu leicht vorstellen, dass jedes Gesicht ungefähr die gleiche Größe hat, dass nur ein Gesicht im Bild ist, dass das Gesicht zentriert in der Bildmitte ist usw. Bei der Gesichtserkennung werden die Eigenvektoren auch *eigenfaces* genannt.

19.10.2 Faktorenanalyse

Da die Faktorenanalyse der PCA sehr ähnlich ist und häufig mit dieser verwechselt wird, bzw. die Begriffe durcheinandergehen, soll hier kurz das Grundprinzip erläutert werden.

Die Faktorenanalyse wird häufig in den Sozialwissenschaften und der Psychologie angewendet. Gegeben sei wie bei der PCA ein n -dimensionaler Zufallsvektor \mathbf{X} , der irgendwelche Merkmale beschreibt. Als Realisierungen \mathbf{x} können wir uns auch eine Punktfolge im n -dimensionalen Merkmalsraum vorstellen. Bei der PCA haben wir diese Punktfolge durch Rotation des Koordinatensystems durch eine Folge von Unterräumen approximiert, wobei die Dimension m des Unterraumes vom Anwender geeignet zu wählen ist. In der Notation der PCA hatten wir geschrieben:

$$\mathbf{X} = \sum_{k=1}^m \mathbf{a}_k Y_k + \sum_{k=m+1}^n \mathbf{a}_k Y_k = \sum_{k=1}^m \mathbf{a}_k Y_k + \mathbf{E}. \quad (19.192)$$

Folglich erhalten wir die Basisvektoren durch Varianzminimierung des Zufallsvektors \mathbf{E} aus der Gleichung:

$$\mathbf{X} = \sum_{k=1}^m \mathbf{a}_k Y_k + \mathbf{E} = \mathbf{A}_m \mathbf{Y} + \mathbf{E} \quad (19.193)$$

Wir haben also bei gegebenem \mathbf{X} oder der Punktfolge ein Approximationsproblem gelöst. Dadurch ist die Approximation eines Punktes \mathbf{x} im Merkmalsraum immer seine orthogonale Projektion auf den berechneten Unterraum. Daher sind die Fehler \mathbf{e} „total“ korreliert bzw. die Kovarianzmatrix des Fehler-Zufallsvektors \mathbf{E} ist keine Diagonalmatrix. Für Approximationsaufgaben (Datenkompression) ist dies auch uninteressant und die Aufgabe ist bestens von der PCA gelöst. Nun drehen wir aber den Spieß um und stellen uns vor, bei gegebenem Unterraum, also den Basisvektoren \mathbf{a}_i , $i = 1, \dots, m$, würfeln wir einen Punkt \mathbf{y} aus dem Unterraum, würfeln einen Fehlervektor \mathbf{e} und erzeugen mit (19.193) den Punkt \mathbf{x} im Merkmalsraum. Wir erzeugen somit mit dem Modell (19.193) eine Punktfolge im Merkmalsraum. Die Fehlervektoren \mathbf{e} würfeln wir natürlich unabhängig voneinander, damit sind sie dekorreliert und die Kovarianzmatrix von \mathbf{E} muss eine Diagonalmatrix sein. Das ist der wesentliche Unterschied der PCA zur Faktorenanalyse. Wir geben folglich bei der Faktorenanalyse a priori vor, dass durch ein Erzeugungsmodell bezüglich eines Unterraumes im statistischen Sinne der Merkmalsraum generiert wurde. Die wichtigste Aufgabe der Faktorenanalyse wird es nun sein, aus (19.193) die Basisvektoren zu berechnen, wobei diese im Gegensatz zur PCA nicht orthogonal sein müssen. Dies geht aber nicht so trivial, dazu müssen wir erst einmal alle statistischen Voraussetzungen aufzählen, die sinnvoll sind:

- a) $E(\mathbf{X}) = \mathbf{0}$,
- b) $E(Y_k) = 0$, $\sigma^2(Y_k) = 1$, $\text{cov}(Y_k, Y_l) = 0 \forall k \neq l$,
- c) $E(E_j) = 0$, $\sigma^2(E_j) = \sigma_j^2$, $\text{cov}(E_i, E_j) = 0 \forall i \neq j$,
- d) $\text{cov}(E_j, Y_k) = 0 \forall j \neq k$.

Mit den gegebenen Kovarianzmatrix $\Sigma_{\mathbf{X}}$ und obigen statistischen Annahmen, z. B. ist auf Grund von c) die Kovarianzmatrix $\Sigma_{\mathbf{E}}$ des Fehlervektors \mathbf{E} eine Diagonalmatrix, kann man

leicht folgende Varianzzerlegung herleiten. Diese wird oft als Hauptsatz der Faktorenanalyse bezeichnet:

$$\Sigma_X = \mathbf{A}_m \mathbf{A}_m^T + \Sigma_E. \quad (19.194)$$

Wenn \mathbf{R} eine Rotationsmatrix bezüglich des m -dimensionalen Unterraumes ist, dann gilt für das Modell (19.193) auch

$$\mathbf{X} = \mathbf{A}_m \mathbf{Y} + \mathbf{E} = \mathbf{A}_m \mathbf{R} \mathbf{R}^T \mathbf{Y} + \mathbf{E} = \tilde{\mathbf{A}}_m \tilde{\mathbf{Y}} + \mathbf{E}. \quad (19.195)$$

Für $\tilde{\mathbf{Y}}$ gelten die gleichen statistischen Aussagen wie für \mathbf{Y} . Daher ist das Faktoremodell (19.193) nur bis auf eine Rotation des Unterraumes eindeutig. Daher wählt man zur Schätzung der Matrix \mathbf{A}_m der Basisvektoren des Unterraumes irgendeine Rotationslage aus. Noch ein paar Bemerkungen zur Faktorenanalyse:

- In der Faktorenanalyse haben sich eigenständige Begriffe und Bezeichnungen entwickelt. Die Matrix der Basisvektoren \mathbf{A}_m heißt z. B. Matrix der Faktorladungen oder Ladungsmatrix, die Komponenten Y_i des Vektors \mathbf{Y} bezüglich des Unterraumes heißen Faktoren. Weiterhin gibt es den Begriff Kommunalität für Varianz differenzen.
- Zur Schätzung der Parameter des Faktoremodells, insbesondere die Faktormatrix, wird die Varianzzerlegung (19.194) benutzt.
- Das Faktoremodell kann unabhängig von der Rotation des Unterraumes, keine Lösung, eine eindeutige Lösung oder unendlich viele Lösungen besitzen.
- Zur Vereinfachung von Interpretationen wird oft mit normierten Zufallsvariablen gearbeitet, siehe (19.188), dann wird die Kovarianzmatrix zur Korrelationsmatrix.
- Da man die Faktoren interpretieren möchte und die Ladungsmatrix nur bis auf eine Rotation des Unterraumes eindeutig ist, rotiert man nach der Schätzung der Parameter den Unterraum nach bestimmten Kriterien. So nennt sich ein Verfahren die Varimax-Rotation.
- Die offensichtliche Wahl eines Unterraumes ist mit der PCA nicht so einfach möglich. Sind bei der PCA z. B. die ersten m Eigenwerte der Kovarianzmatrix groß und alle annähernd gleich, die restlichen $n - m$ Eigenwerte aber Null, dann ist die Wahl des Unterraumes tatsächlich naheliegend.
- Unter der Annahme $\sigma_j^2 \approx 0$ ähnelt (19.194) der unvollständigen Cholesky-Faktorisierung (*incomplete Cholesky decomposition*).

Im nachfolgenden Kapitel werden wir beispielhaft ein paar Verfahren der visuellen Objekterkennung kennenlernen. Dabei handelt es sich oft um eine Kombination einer Merkmalsberechnung und eines Klassifikationsverfahrens. Das Gebiet der Objekterkennung zählt zu den aktivsten Forschungsbereichen in der Bildverarbeitung. Daher werden im Rahmen dieses Buches auch nur wichtige Grundprinzipien erläutert, welche in aktuellen Methoden Verwendung finden.

20.1 Herausforderungen der visuellen Objekterkennung

Es ist wichtig, die einzelnen Aufgabenstellungen der Objekterkennung korrekt zu definieren. Der Begriff der „Objekterkennung“ ist ein sehr weit gefasster Begriff, welcher eher für ein komplettes Gebiet als für eine konkrete Problemstellung steht. Wir hatten in Abschn. 18.1 eine Unterscheidung beim maschinellen Lernen aufgrund der Art der Ausgabe vorgenommen (Regression, binäre Klassifikation, Mehrklassenklassifikation). Diese Unterscheidung lässt sich nun für die visuelle Objekterkennung erweitern. Als Eingabe erhalten die Verfahren immer ein Testbild, welches beschriftet werden soll. Anstatt allgemein von der Ausgabe des Verfahrens zu sprechen, werden wir in diesem Kapitel oft den Begriff *Beschriftung* wählen. In Abb. 20.1 sind zunächst die drei Hauptabstraktionsebenen der visuellen Objekterkennung dargestellt. Diese richten sich nach der Art der gewünschten Ausgabe des Systems und des Detaillierungsgrades:

1. Bildklassifikation (Beschriftung des gesamten Bildes): Für ein gegebenes Bild muss entschieden werden, ob es ein Beispiel einer gelernten Klasse enthält.
2. Objektlokalisation (Beschriftung mit umschreibenden Rechtecken): Ziel ist es, Objekte in einem Bild zu finden und deren Position mit umschreibenden Rechtecken anzugeben.
3. Semantische Segmentierung (Beschriftung jedes Pixels): Jeder Pixel eines gegebenen Bildes soll mit einer der gelernten Klassen beschriftet werden.



Abb. 20.1 Unterschiedliche Aufgabenstellungen der visuellen Objekterkennung: Bildklassifikation/kategorisierung (Beschriftung des gesamten Bildes), Objektlokalisation (Beschriftete umschreibende Rechtecke) und Semantische Segmentierung (Beschriftung jedes einzelnen Bildpixels)

Jede dieser Aufgabenstellungen hat ihre eigenen Besonderheiten, welche wir in diesem Kapitel behandeln werden. Weiterhin lässt sich ganz klar eine Steigerung der Komplexität und der Anforderungen an die Ausgaben des Erkennungssystems beobachten. Die Liste ist natürlich nicht als vollständig zu betrachten, so gibt es viele weitere Aufgabenstellungen, welche aber im weiteren Verlauf des Kapitels nicht besprochen werden und über die Einteilung in semantische Klassen hinausgehen. Beispielhaft sei hier die Posenschätzung genannt, welche zum Beispiel bei der Microsoft Kinect notwendig ist, um die aktuelle Körperposition des Benutzers zu ermitteln.

Der Begriff der „Klasse“ ist bisher auch nur ein abstraktes Konzept, welches wir weiter konkretisieren müssen. Bei der visuellen Objekterkennung gibt es nicht nur mehrere Ebenen der Beschriftung sondern auch der Klassenspezifikation. So unterscheidet man im Allgemeinen zwischen der Erkennung auf *Instanzebene* und auf *Kategorieebene*. Auf der Instanzebene unterscheidet man zwischen spezifischen Objekten, ein klassisches Beispiel ist hier die Gesichtserkennung für die Authentifizierung, wo zwischen Gesichtsaufnahmen verschiedener Personen unterschieden wird. Eine Erkennung auf Kategorieebene hat das Ziel allgemeine semantische Konzepte zu erkennen und zum Beispiel zwischen unterschiedlichen Tierarten zu unterscheiden.

In Abschn. 18.2 hatten wir uns bereits mit den Herausforderungen beim maschinellen Lernen beschäftigt und etliche Aspekte herausgearbeitet. Bei den Beispielbildern in Abb. 20.1 lassen sich auch die Schwierigkeiten des automatischen Lernens von visuellen Klassen gut erkennen und folgendermaßen zusammenfassen:

1. Verdeckungen: Objekte können sich gegenseitig verdecken (zu sehen in der StraßenSzene unten links) und 3D-Informationen sind bei Einzelaufnahmen nicht direkt verfügbar.

2. Hintergrund: Die Darstellung von Objekten und Bildelementen, welche nicht erkannt werden sollen, erschwert die Erkennung zusätzlich (z. B. die Fahrradfahrer im Bild in der Mitte).
3. Intraklassenabstand: Die Erscheinungen der Objekte variieren sehr stark durch unterschiedliche Rotationen, Skalierungen, andere Perspektiven, nicht-starre Deformationen, farbliche Gestaltung, Unterkategorien anderer Ausprägung (z. B. verschiedene Arten von Vegetation in den Bildern auf der rechten Seite).
4. Interklassendistanz: Bestimmte Objekte sind ähnlich zueinander und lassen sich schwierig voneinander trennen (z. B. Kategorie Fenster und Tür auf der rechten Seite).

Es ist dabei zu beachten, dass das Problem einer geringen Interklassendistanz besonders bei der Erkennung auf Instanzebene auftritt. Ein hoher Intraklassenabstand und eine hohe Variabilität in der Klasse ist vor allem ein typisches Problem der Erkennung auf Kategorieebene. Die große Variabilität der Objekte einer Kategorie lässt sich meist nur durch die Angabe vieler beschrifteter und repräsentativer Beispiele erlernen. So werden zum Beispiel zum Anlernen von Fußgängerdetektoren oft mehrere tausende Beispiele benötigt [20].

20.2 Bildklassifikation

Im Folgenden werden wir uns mit der Aufgabenstellung der Bildklassifikation beschäftigen, d. h. der Zuweisung von Bildern zu Objektkategorien. Dabei werden wir nur die Spitze des Eisberges in diesem Bereich diskutieren können, da hier im Laufe des letzten Jahrzehnts hunderte von Algorithmen und Erweiterungen entstanden sind. So werden wir uns zum Beispiel auf komplett überwachte Verfahren beschränken, welche davon ausgehen, dass wir bereits eine Anzahl von beschrifteten Bildern für jede Kategorie zur Verfügung haben.

20.2.1 Kategorisierung mit globalen Merkmalen

In den Abschnitten zur Merkmalsberechnung (Abschn. 19.9) und zum maschinellen Lernen (Kap. 18) haben wir schon die notwendigen Werkzeuge für die Bildklassifikation kennengelernt. Wir extrahieren mittels eines gewählten Merkmalsextraktionsverfahrens einfach einen Merkmalsvektor fester Größe für ein gegebenes Bild und erhalten so ein klassisches Klassifikationsproblem, welches mit Standardalgorithmen des maschinellen Lernens, wie etwa SVM- oder NN-Klassifikatoren (Abschn. 18.4.6 und 18.4.1) angegangen werden kann.

Der Fokus bei der Entwicklung eines Verfahrens zur Bildklassifikation liegt daher oft auf der Merkmalsberechnung, welche mit den Herausforderungen die wir im vorherigen Abschnitt diskutiert haben zurechtkommen muss. Im Abschn. 19.9.5 hatten wir bereits den sogenannten *Bag-of-Words* Ansatz (BoW) kennengelernt, welcher Statistiken von lokalen

Merkmale berechnet. Bei der folgenden Beschreibung gehen wir von den üblicherweise verwendeten SIFT-Merkmalen (Abschn. 19.9.2) aus, das Verfahren der globalen Merkmalsbestimmung ist aber prinzipiell unabhängig von der Art der lokalen Merkmale.

Ausgehend von einer Quantisierung von SIFT-Merkmalen (Abschn. 19.9.2) mittels Gruppierungsverfahren wie etwa k -Means (Abschn. 18.7.1), wird die Verteilung der lokalen Merkmale in einem Bild mit einem Histogramm beschrieben. Das Histogramm beschreibt dadurch die Häufigkeit von typischen Bildstrukturen. Ein wichtiger Aspekt ist hierbei, dass bei der Histogrammbestimmung, so wie wir sie im Abschn. 19.9.5 kennengelernt haben, keine Positionsinformationen der SIFT-Merkmale verwendet werden. Ein BoW-Histogramm enthält nur die Häufigkeit einzelner Bildstrukturen, nicht aber deren Position oder Anordnung im Bild. Wie bereits in Abschn. 19.9.5 angesprochen, ist ein großer Vorteil dieser Darstellung, die Robustheit bezüglich starker Objektveränderungen. Der Nachteil ist hingegen, dass durch die Missachtung der Position der lokalen Merkmale Information verloren gehen kann, welche für die Unterscheidung von Objektkategorien notwendig wäre. Dies ist besonders bei Aufgabenstellungen der Fall, welche sich durch eine niedrige Interklassendistanz auszeichnen.

Einbeziehung von Positionsinformationen Eine Möglichkeit die Positionen von lokalen Merkmale wenigstens teilweise in die Merkmalsberechnung zu integrieren, ist das sogenannte *spatial pooling*. Hinter diesem Schlagwort, verbirgt sich nichts anderes als die Berechnung von BoW-Histogrammen in mehreren Bereichen des Bildes und die anschließende Konkatenation dieser Merkmalsvektoren. Dieses Prinzip ist sehr ähnlich zur Bestimmung von Histogrammen in einzelnen Blöcken beim HOG-Merkmal (Abschn. 19.9.3). Die Bereiche, in denen die Histogramme extrahiert werden, müssen entweder vorher fest definiert oder als Parameter während des Lernens optimiert werden. Typischerweise wird einfach das Bild mit einem regelmäßigen Gitter unterschiedlicher Größe in einzelne Blöcke eingeteilt. In Abb. 20.2 ist dies einmal für ein Beispiel dargestellt, als lokales Merkmal wird einfach die Gradientenrichtung gewählt, welche in 8 unterschiedliche Richtungen quantisiert ist ähnlich zu HOG Merkmalen ohne Normierung.

Wie in der Abbildung zu erkennen ist, erhält man für jede Wahl des Gitters ein Merkmalsvektor $\mathbf{h}^{(y)} \in \mathbb{R}^{d_y}$, welcher wiederum durch Zusammenfügen der Histogramme in den einzelnen Blöcken entstanden ist. Bei der Kombination der Merkmalsvektoren aus jeder Stufe $y \in \{1, \dots, m\}$ ($y = 1$ korrespondiert zu einem groben Gitter mit wenigen Blöcken und $y = m$ ist das Gitter mit der kleinsten Blockgröße) gibt es nun mehrere Strategien. Eine einfache Strategie ist es, diese Merkmalsvektoren wieder ebenfalls zu konkatenieren um einen globalen Merkmalsvektor $\mathbf{h} = (\mathbf{h}^1, \dots, \mathbf{h}^m) \in \mathbb{R}^D$ des Bildes zu erhalten. Diese Methode hat aber einen entscheidenden Nachteil, welchen wir im Folgenden erläutern wollen. In Abschn. 18.5 hatten wir uns bereits mit Kernelfunktionen beschäftigt und gesehen, dass kernelbasierte Klassifikatoren nur auf paarweise Ähnlichkeiten der Lernbeispiele für das Lernen zurückgreifen. Diese Ähnlichkeiten werden mittels einer Kernelfunktion berechnet. Wenn wir den Histogrammschnitt (Abschn. 18.5.2, (18.92)) als Kernelfunktion zum

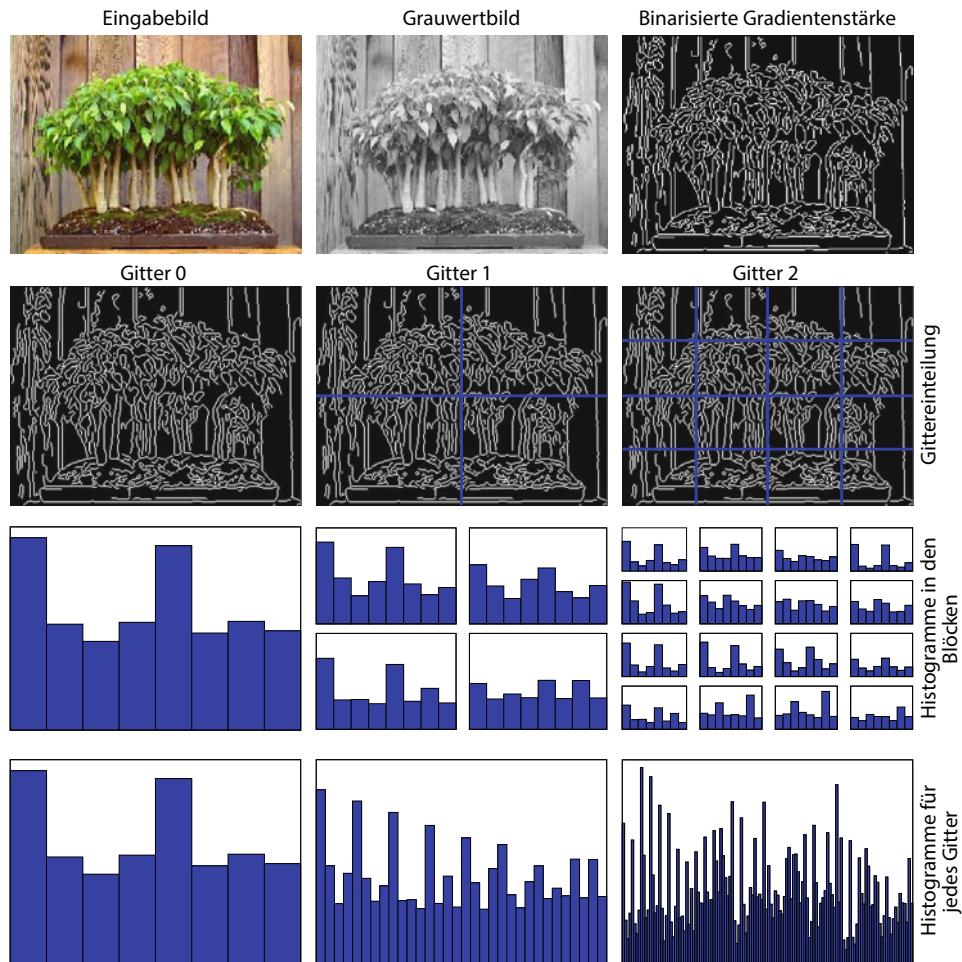


Abb. 20.2 Bestimmung von BoW-Histogrammen in unterschiedlichen Bereichen des Bildes: Im Beispiel wurden die Gradientenrichtungen im Bild in 8 Richtungen quantisiert

Vergleich von zwei Merkmalsvektoren \mathbf{h} und \mathbf{h}' heranziehen, so erhalten wir:

$$K(\mathbf{h}, \mathbf{h}') = \sum_{i=1}^D \min(h_i, h'_i) = \sum_{y=1}^m \sum_{j=1}^{d_y} \min(h_j^{(y)}, (h')_j^{(y)}) \quad (20.1)$$

Es lässt sich erkennen, dass die Unterschiede in den Histogrammeinträgen bei größeren Gittern den gleichen Einfluss auf die Ähnlichkeitsbestimmung haben wie Unterschiede in den Histogrammeinträgen bei feineren Gittern, obwohl Bildbereiche unterschiedlicher Größe zur Bestimmung der Einträge verwendet wurden. Um dies zu vermeiden, führt man

üblicherweise Gewichte β_y ein, welche entweder als Parameter während des Lernens bestimmt oder anhand der Dimensionen d_y als Normierungsfaktoren gewählt werden:

$$K(\mathbf{h}, \mathbf{h}') = \sum_{y=1}^m \beta_y \left(\sum_{j=1}^{d_y} \min(h_j^{(y)}, (h')_j^{(y)}) \right). \quad (20.2)$$

Als alternative Kernelfunktion kann auch der Chi-Quadrat-Kern verwendet (Abschn. 18.5.2).

Einfache randomisierte Bestimmung der Quantisierung Die Bestimmung der Quantisierung oder der Kodebücher von lokalen Merkmalen mit dem k -Means Algorithmus ist durch die quadratische Komplexität des Algorithmus sehr aufwendig, daher wird oft nur eine kleine zufällige Auswahl von lokalen Deskriptoren verwendet. In vielen Untersuchungen hat sich gezeigt, dass sogar eine zufällige Wahl der Prototypen für die Quantisierung durchgeführt werden kann ohne deutliche Unterschiede in den Erkennungsraten zu beobachten. In dem Fall wählt man einfach zufällig Merkmalsvektoren aus den lokalen Deskriptoren aller Lernbeispiele aus und verwendet diese als Prototypen.

Bestimmung überwachter Quantisierungen Bei der Bestimmung der Quantisierung der lokalen Merkmale wird oft der k -Means Algorithmus oder eine randomisierte Auswahl von Prototypen verwendet. Ein Nachteil dieser Gruppierungsverfahren ist es, dass die Klassenzugehörigkeiten, der den lokalen Merkmalen zugehörigen Bilder, nicht beachtet werden. Ein Verfahren welches dies berücksichtigt, könnte Cluster von lokalen Merkmalen bestimmen, welche charakteristisch für spezifische Klassen sind. Es gibt in der Literatur viele Verfahren, welche den k -Means Algorithmus entsprechend erweitern um die Homogenität bezüglich der Klassenzugehörigkeit in den Gruppen zu fördern. Diese Verfahren werden oft dem Schlagwort *discriminative codebook learning* zugeordnet. Im Gegensatz dazu ist es aber auch möglich ein bewährtes Konzept bei der Klassifikation, auch für die überwachte Gruppierung zu verwenden: randomisierte Entscheidungsbäume (Abschn. 18.4.4).

Wir gehen einmal davon aus, dass wir in allen für das Lernen zur Verfügung gestellten Bildern, lokale Merkmalsdeskriptoren extrahiert haben. Jedem dieser lokalen Deskriptoren ist die Klassenbeschriftung des jeweiligen Bildes zugeordnet. Wir lernen nun einen randomisierten Entscheidungsbäum mit diesen Daten an, d. h. wir versuchen einen Klassifikator zu bestimmen, welcher ausgehend von einem lokalen Merkmal die Klassenbeschriftung schätzt. Dies ist natürlich eine schwierige Aufgabenstellung, da der Bildbereich, welcher für die Berechnung der lokalen Deskriptoren verwendet wird, oft sehr klein ist und nur wenig Informationen enthält um Objekte einzelner Kategorien zu erkennen. Wir werden dieses Problem in Abschn. 20.4 im Rahmen der semantischen Segmentierung noch näher diskutieren. Eine Klassifikation ist aber im aktuellen Schritt der Merkmalsberechnung auch noch gar nicht gefordert. Viel mehr erhalten wir durch den gelernten Entscheidungsbäum (oder Entscheidungswald) eine indirekte Einteilung des Merkmalsraumes. Die einzelnen Blätter des Baumes lassen sich dabei jeweils als Teil des Merkmalsraumes interpretieren und durch

den gesamten Baum ergibt sich eine Zerlegung, sowie sie auch durch den k -Means Algorithmus mit der Voronoi-Zerlegung prinzipiell erzielt wird. Vorteil bei der Verwendung eines Entscheidungsbaumes ist es aber, dass beim Lernen versucht wird Blätter zu erhalten, welche Lernbeispiele nur einer Klasse enthalten. Dies ist genau das, was wir bei der ursprünglichen Motivation der überwachten Gruppierung gefordert hatten.

Um ein BoW-Histogramm für ein Bild mit diesem Verfahren zu bestimmen, berechnen wir zunächst alle lokalen Deskriptoren $\{\mathbf{l}^{(k)}\}_{k=1}^W$ in diesem Bild. Dabei ist es vorteilhaft keinen Detektor zu verwenden, sondern in einem festen Gitter Deskriptoren zu berechnen. Danach wenden wir den gelernten Entscheidungsbaum auf jeden dieser Deskriptoren an und zählen die Häufigkeit der Deskriptoren in den Blättern $\{\vartheta_j\}_{j=1}^D$:

$$h_j = |\{\mathbf{l}^{(k)} \mid \vartheta(\mathbf{l}^{(k)}) = \vartheta_j\}| \cdot W^{-1}. \quad (20.3)$$

Die Größe der BoW-Histogramme \mathbf{h} wird dabei durch die Anzahl der Blätter des Entscheidungsbaumes angegeben. Die beim Lernen ermittelten empirische Klassenwahrscheinlichkeiten in den Blättern sind für die Merkmalsberechnung nicht relevant.

Earth Mover's Distance als Abstandsmaß Wir hatten in Abschn. 19.8.3 bereits die *Earth Mover's Distance* (EMD) als Abstandsmaß für zwei Signaturen kennengelernt. Als Signatur hatten wir Mengen $P = [(\mathbf{m}_j, w_j)]_j$ mit Prototypen $\mathbf{m}_j \in \mathbb{R}^d$ und Gewichten $w_j \in \mathbb{R}$ definiert. Unsere verwendeten BoW-Histogramme in diesem Abschnitt lassen sich als Signaturen $P = [(\mathbf{z}_j, h_j)]_j$ mit fest definierten Clustern \mathbf{z}_j (Elemente des Kodebuchs) auffassen. Die EMD-Distanz lässt sich aber nun auch dafür einsetzen, Signaturen mit unterschiedlichen Clustern zu vergleichen. Daher können wir die lokalen Merkmale jedes Bildes individuell mit dem k -Means Algorithmus gruppieren („clustern“) und die dadurch entstandenen Signaturen miteinander mit der EMD-Distanz vergleichen (siehe (19.104)). Ausgehend von dieser paarweisen Distanz der Bilder, lässt sich dann eine Kernelfunktion konstruieren:

$$K_{\text{EMD}}(P, Q) = \exp(-\gamma \cdot \text{EMD}(P, Q)), \quad (20.4)$$

wobei P und Q die Signaturen der Bilder sind. Die Variable $\gamma > 0$ ist ein Hyperparameter der Kernelfunktion. Ein Vorteil dieser Methode ist die Unabhängigkeit von einem festen Kodebuch. Diesem Vorteil steht der Nachteil des hohen Rechenaufwandes bei der Berechnung der paarweisen Distanzen gegenüber.

20.2.2 Kernelbasierte Kombination von mehreren Merkmalen

Für viele Aufgabenstellungen ist es notwendig, mehrere Merkmalsrepräsentationen zu berechnen und für die Erkennung zu nutzen. Zum Beispiel beinhalten BoW-Histogramme auf der Basis von lokalen SIFT Deskriptoren nur Informationen über lokale Grauwertstrukturen ohne Farbinformationen zu berücksichtigen. Daher werden oft mehrere BoW-

Histogramme mit unterschiedlichen lokalen Deskriptoren berechnet, welche dann kombiniert werden. Eine einfache Möglichkeit der Kombination ist die Konkatenation der Merkmale. Ein Nachteil dieser Methode ist es, dass alle Merkmale gleichberechtigt beim Klassifikator vorliegen. Dies ist besonders bei kernelbasierte Methoden nachteilig, da dann Unterschiede in den Merkmalen den gleichen Einfluss auf die Distanzberechnung haben.

Dieser Nachteil kann mit *multiple kernel* Methoden überwunden werden. Wir gehen einmal davon aus, dass wir für ein gegebenes Bild M BoW-Histogramme $(\mathbf{h}^{(k)})_{k=1}^M$ mit unterschiedlichen Verfahren der lokalen Merkmalsextraktion bestimmt haben. Weiterhin sei für jede dieser M Merkmalstypen eine Kernelfunktion $K^{(k)}$ definiert, welche die Histogramme zweier unterschiedlicher Beispiele des gleichen Merkmalstyps vergleichen kann. Für die kernelbasierte Klassifikation benötigen wir eine gesamte Kernelfunktion K , welche zwei Histogrammrepräsentationen $\mathcal{H}_1 = (\mathbf{h}_1^{(k)})_{k=1}^M$ und $\mathcal{H}_2 = (\mathbf{h}_2^{(k)})_{k=1}^M$ vergleicht. Eine Möglichkeit Kernelfunktionen zu kombinieren ist eine einfache gewichtete Linearkombination:

$$K(\mathcal{H}_1, \mathcal{H}_2) = \sum_{k=1}^M \alpha_k \cdot K^{(k)}(\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}). \quad (20.5)$$

Die Gewichte $\alpha_k \in \mathbb{R}$ sollten natürlich aus den gegebenen Lerndaten ermittelt werden. Eine Möglichkeit hatten wir bereits bei der Klassifikation mit Gaußprozessen kennengelernt. Dort ist es möglich (Hyper-)Parameter der Kernelfunktion aufgrund einer Optimierung zu ermitteln (Abschn. 18.5.11). Da sich die Gewichte α_k als Parameter der Kernelfunktion K auffassen lassen, kann man die dort beschriebenen Techniken direkt anwenden.

20.2.3 Kombinierte Merkmalsberechnung und Klassifikation

Wir haben schon zahlreiche Merkmalstypen kennengelernt, wie zum Beispiel Richtungshistogramme und ausgehend von diesen sogenannte globale BoW-Merkmale. Im Folgenden werden wir uns den Gesamtlauf der Merkmalsberechnung noch einmal detailliert ansehen und davon allgemeine Prinzipien ableiten.

Für die Berechnung von Richtungshistogrammen, sei es nun konkret für SIFT oder HOG Merkmale, ist die Bestimmung der einzelnen Gradienten im Bild notwendig. Wie in Abschn. 2.3 erläutert, geschieht dies meist durch die Anwendung von linearen Filteroperationen, wie etwa dem Sobel-Filter. Eine einfache aber ineffiziente Möglichkeit für die Berechnung einer in K Bereiche quantisierten Gradientenrichtung, ist zum Beispiel die Verwendung von K rotierten Gradientenfiltern (siehe analog hierzu die Richtungskantenfilter in Abschn. 8.2.2). Die (normierten) Ausgaben der einzelnen Filter in jedem Pixel können dann als eine Art Histogramm über die Gradientenrichtungen interpretiert werden. Bei der Berechnung von SIFT oder HOG Merkmalen werden diese Histogramme lokal zusammengefasst, zum Beispiel innerhalb von 8×8 Nachbarschaften. Ausgehend von den lokalen Merkmalen erfolgt wieder eine Quantisierung, diesmal mittels eines unüberwacht

erlernten BoW Kodebuches (siehe Abschn. 19.9.5). Die quantisierten lokalen Merkmalen werden danach wieder in bestimmten Bildregionen zu einem Histogramm zusammengefasst wie im vorherigen Abschnitt gezeigt.

Betrachtet man diesen Ablauf, so lassen sich zwei fundamentale Grundkonzepte erkennen: Anwendung von Filteroperationen und Zusammenfassen von Merkmalen. Die alternierende Anwendung dieser zwei Konzepte ist daher typisch für Erkennungsverfahren in der Bildverarbeitung und es ergeben sich mehrere miteinander verbundene Ebenen der Merkmalsberechnung, welche oft als Netzwerk dargestellt werden. Grundidee ist es, in dieses Modell direkt auch das Modell eines Klassifikators zu integrieren, zum Beispiel durch mehrere lineare Operationen, welche für jede einzelne Klasse eine kontinuierliche Ausgabe liefern.

Die Idee neuer Verfahren, welche oft mit dem Schlagwort *deep learning* assoziiert werden, ist es nicht wie ursprünglich jede einzelne Ebene getrennt voneinander zu betrachten und zu optimieren, sondern möglichst viele Parameter anhand von Lerndaten zu bestimmen. Die Beschränkung auf SIFT Merkmale ganz am Anfang der üblichen Ansätze ist zum Beispiel in vielen Fällen viel zu restriktiv und anwendungsspezifisch. Die Frage stellt sich nun aber, wie es möglich ist, die schier immense Anzahl von Parametern (Filtermasken, Regionen für die Zusammenfassung, etc.) effizient und robust zu erlernen, und vor allem welches Optimierungskriterium anwendbar ist. Als grundlegendes Kriterium beim maschinellen Lernen haben wir bereits die Minimierung einer Fehlerfunktion auf den Lerndaten kennengelernt (siehe zum Beispiel Abschn. 18.4.5). Diese Kriterien sind intuitiv verständlich und kommen dem finalen Kriterium, die Fehlerrate bei der Klassifikation zu verringern, am nächsten. Um eine Optimierung der Parameter der Schichten der Merkmalsextraktion mittels einer Fehlerfunktion auf den Lerndaten zu optimieren, werden häufig Gradientenabstiegsverfahren verwendet, ähnlich zum *back propagation* bei neuronalen Netzwerken. Ein großes Problem bei diesem Ansatz ist die große Anzahl zu bestimmender Parameter, daher ist es oft notwendig immense Mengen von Lernbeispielen dem Verfahren zur Verfügung zu stellen, um eine Überanpassung zu vermeiden. Beim Verfahren von Krizhevsky et al. [36] wird zum Beispiel ein Teil des *ImageNet*-Datensatzes verwendet mit über 1 Million Bildern eingeteilt in 1000 Objektkategorien¹.

Eine genaue und ausführliche Beschreibung dieser Verfahren ist nicht Bestandteil dieses Buches und wir verweisen den interessierten Leser auf die Arbeit von Krizhevsky et al. [36] und den dortigen Literaturzitaten.

20.2.4 Erkennung auf Instanzebene

Bisher hatten wir kurz und knapp ein paar Aspekte der Bildklassifikation auf Kategorieebene betrachtet. Wie schon in Abschn. 20.1 besprochen, muss bei der Erkennung auf Instanzebene hingegen ein spezifisches Objekt aus zum Beispiel unterschiedlichen Blickwinkeln

¹ <http://www.image-net.org/>

und in unterschiedlichen Beleuchtungen erkannt werden. Im Vergleich zur Kategorisierung haben wir es also in dem Fall mit ganz anderen Anforderungen an die Invarianzeigenschaften der Merkmale zu tun, da die zu betrachtende Klasse eine wesentlich kleinere Intraklassenvarianz besitzt.

Bei der Instanzerkennung liegt einer Klasse immer das gleiche 3D-Objekt zu Grunde und für die Erkennung müssen wir daher hauptsächlich mit den Auswirkungen der Projektion auf ein 2D-Bild zurechtkommen. In Abschn. 16.4 hatten wir uns mit der Bestimmung von Punktkorrespondenzen für die 3D Rekonstruktion beschäftigt, mit einer fast identischen Ausgangslage. Aus diesem Grund lassen sich auch bei der Instanzerkennung nahezu die gleichen Algorithmen verwenden.

Üblicherweise werden also auch SIFT-Deskriptoren an interessanten Punkten bestimmt (vollständige SIFT Merkmale mit Detektor) um dann diese mit Matchingverfahren, wie der ungarischen Methode, zu vergleichen (Abschn. 19.8). Dabei ist es sinnvoll iterativ das Matchingverfahren mit einer Bestimmung einer Menge von Punktkorrespondenzen zu verbinden, welche einem gemeinsamen geometrischen Modell unterliegen. Das Matching erfolgt immer zwischen einem Testbild und einem der Lernbeispiele. Die Kosten beim Matchingverfahren können dann als Distanzmaß beim Nächster-Nachbar-Klassifikator eingesetzt werden. Bei vielen Lernbildern ist dieses Verfahren sehr aufwendig. Eine Möglichkeit der Reduktion der notwendigen Rechenzeit ist die Verwendung von globalen Merkmalen, wie etwa BoW-Merkmale mit *spatial pooling* (Abschn. 20.2.1), um zunächst eine kleine Menge von k Kandidaten mittels des k -Nächster-Nachbar-Klassifikators zu erhalten. Dieser zweistufige Prozess ist auch als *query expansion* bekannt.

Ein anderer Ansatz basiert auf der verallgemeinerten Hough-Transformation (Abschn. 10.8.5). Für jeden Deskriptor merken wir uns zusätzlich im Lernschritt die relative Position zum Mittelpunkt des Objektes, die Orientierung und die Skalierung. Bei der Erkennung versuchen wir dann mit einem Voting den Mittelpunkt eines Objektes oder mehrerer Objekte zu bestimmen. Zusätzlich kann dann anschliessend ein geometrischer Validierungsschritt erfolgen, indem wir versuchen eine affine Transformation an die gegebenen Punktmengen der lokalen Merkmale anzupassen. Die vorherige Hough-Transformation kann auch zusätzlich zur Auswahl einer kleinen Menge von möglichen Objektpunkten verwendet werden. Weitere Details zu dieser Anwendung von SIFT-Deskriptoren befinden sich in der Originalarbeit von David Lowe [42].

20.3 Objektlokalisation

Im Folgenden werden wir uns mit der Objektlokalisation, so wie sie im einführenden Abschn. 20.1 dargestellt wurde, beschäftigen. Diese Aufgabenstellung wird auch oft als Objektdetektion bezeichnet, da sich besonders in der englischen Literatur der Begriff *object detection* eingebürgert hat. Ziel ist es nicht nur eine globale Beschriftung für ein Testbild zu erhalten, sondern auch ein umschreibendes Rechteck, welches die Position des Objektes angibt.

20.3.1 Sliding-Window Klassifikation

Die meisten Verfahren der Objektlokalisierung basieren auf dem allgemeinen Konzept der *Sliding-Window Klassifikation*. Wie der Name schon vermuten lässt, findet bei diesem Ansatz eine Klassifikation von einzelnen Bildausschnitten statt, welche wir im Folgenden auch als Fenster bezeichnen wollen. Genauer gesagt wählen wir uns ein Fenster fester Größe und schieben dieses Rechteck Pixel für Pixel über das Eingabebild. An jeder Position des Fensters wenden wir nun einen Klassifikator an, welcher zur Merkmalsberechnung nur die Bildinformation innerhalb des Fensters verwendet. Der Klassifikator trifft für die aktuelle Position eine binäre Klassifikationsentscheidung, ob das Fenster gerade eine Instanz der Objektkategorie zeigt.

Wir werden dieses Konzept nun ein wenig exakter formulieren und dabei auch mathematische Notationen einführen, welche wir später noch benötigen werden. Wir bezeichnen zunächst ein Fenster mit $\mathcal{R} = (p_x, p_y, m_{\mathcal{R}}, n_{\mathcal{R}})$, definiert durch die linke obere Ecke $\mathbf{p} = (p_x, p_y)$ und den Seitenlängen $m_{\mathcal{R}}$ und $n_{\mathcal{R}}$. Weiterhin nehmen wir an, dass ein Verfahren zur Merkmalsextraktion gegeben ist, welches für eine gegebene Fenstergröße \mathcal{R} und ein Bild g einen Merkmalsvektor $\mathbf{x}(\mathcal{R}; g) \in \mathbb{R}^D$ bestimmt. Bei einer gegebenen Bewertungsfunktion $f : \mathbb{R}^D \rightarrow \mathbb{R}$ eines Klassifikators bestimmen wir eine Menge von Objekthypothesen, d. h. Fenster die höchstwahrscheinlich ein Objekt einer vorher festgelegten Kategorie enthalten, indem wir die Fenster betrachten, bei denen der Wert von f einen bestimmten Schwellwert ζ_{det} überschreitet:

$$\mathcal{D}(g) = \{\mathcal{R} \mid f(\mathbf{x}(\mathcal{R}; g)) > \zeta_{\text{det}}\}. \quad (20.6)$$

Da der Klassifikator mit Beschriftungen $y = 1$ für positive Beispiele und $y = -1$ für negative Beispiele angelernt wurde, sind Beispiele (Fenster) mit hohen Werten der Entscheidungsfunktion mögliche Objekte der gelernten Kategorie.

Die Anzahl aller möglichen Fenster in einem Bild ist natürlich riesig und bei $M \times M$ großen Bildern im Bereich von $\mathcal{O}(M^4)$. Aus diesem Grund werden in der Praxis oft Fenster mit festem Seitenverhältnis $\beta \in \mathbb{R}$ gewählt und auch nur mit einer geringen Anzahl von Skalierungsstufen. Ein Fenster ist in diesem Fall definiert durch seinen Mittelpunkt sowie einem Skalierungsfaktor $\sigma \in \{\sigma_1, \dots, \sigma_S\}$. Um die Menge der Hypothesen zu bestimmen, können wir daher S Bilder für jede Skalierung betrachten, welche die Ausgabewerte der Entscheidungsfunktion in jedem Pixel speichern:

$$s(\mathbf{p}, \sigma; g) = f(\mathbf{x}(p_x, p_y, \sigma \cdot m_{\mathcal{R}}, \sigma \cdot n_{\mathcal{R}}; g)) \quad (20.7)$$

$$= f(\mathbf{x}(p_x, p_y, m_{\mathcal{R}}, n_{\mathcal{R}}; g_{\sigma})). \quad (20.8)$$

Dabei können wir anstelle der Skalierung der Fenster auch ein entsprechend skaliertes Bild g_{σ} verwenden. Für die Betrachtung mehrerer Skalierungen können auch Gaußsche Bildpyramiden (Abschn. 5.4.5) genutzt werden, d. h. die betrachtenden Fenster haben immer eine feste absolute Größe und durch die Skalierung des Bildes selbst, ist es möglich Objekte unterschiedlicher Größen in einem Bild zu detektieren.

Das Lernen des Klassifikator f erfolgt sowohl mit positiven Beispielen, d. h. Fenster welche ein Beispiel der Objektkategorie zeigen, als auch mit negativen Bildausschnitten, d. h. Fenster welche kein komplettes Beispiel der Objektkategorie darstellen. Wichtig ist noch einmal die Abgrenzung zur Bildklassifikation, ein Beispiel ist in unserem aktuellen Fall immer ein Bildausschnitt nicht ein komplettes Bild. Daher ergeben sich im Falle der Objektlokalisation eine große Anzahl möglicher negativer Beispiele. In der Praxis wird daher oft ein *Bootstrapping*-Ansatz gewählt, welcher zunächst zufällige negative Bildausschnitte verwendet um einen initialen Klassifikator zu erstellen und dann in folgenden Schritten immer neue zufällige negative Bildausschnitte hinzufügt, welche einen hohen Wert der Bewertungsfunktion f des bisherigen Klassifikators erhalten haben. Ein hoher Wert der Bewertung für ein negatives Beispiel signalisiert eine Falschklassifikation, daher wird bei diesem Verfahren versucht den Klassifikator schrittweise zu verbessern indem besonders schwierige Beispiele beim Lernen zur Verfügung gestellt werden.

Im vorherigen Abschnitt hatten wir uns mögliche Merkmale angesehen, welche sich für die Bildklassifikation eignen. Ein Beispiel waren Bag-of-Words-Merkmale, welche Statistiken von lokalen Merkmalen berechnen, ohne deren Positionen zu berücksichtigen. Bei der Objektlokalisation sind diese nur bedingt geeignet, da in diesem Fall die berechneten Merkmale nicht translationsinvariant sein sollten. Ein Klassifikator welcher translationsinvariante Merkmale verwendet, würde schon höchstwahrscheinlich einen hohen Wert der Entscheidungsfunktion zurück liefern, obwohl das Objekt im aktuell betrachteten Fenster nur teilweise zu sehen ist. Jedoch gibt es einen gewissen Kompromiss an dieser Stelle, zum Beispiel ist eine Invarianz bezüglich kleinerer Verschiebungen von wenigen Pixeln sinnvoll, besonders wenn der Klassifikator nicht an allen Stellen des Bildes ausgewertet wird sondern nur alle paar Pixel.

Auf Basis der in diesem Abschnitt erfolgten Überlegungen, fassen wir noch einmal die Anforderungen an die Bestandteile eines Verfahrens der Objektlokalisation zusammen:

1. Schnelle Merkmalsberechnung für ein gegebenes Fenster
2. Schnelle Auswertung des Klassifikators
3. Effizientes Lernen eines Klassifikators mit vielen Lernbeispielen
4. Die berechneten Merkmale dürfen nicht invariant bezüglich signifikanter Translationen sein um eine ausreichende Lokalisationsgenauigkeit zu gewährleisten.

20.3.2 Viola-und-Jones-Verfahren

Das erste Verfahren, welches wir uns näher ansehen wollen, ist das Verfahren nach Viola und Jones [81]. Das Verfahren wurde ursprünglich für die Gesichtsdetektion entwickelt und hat sich auch schon bereits in vielen Anwendungen bewährt. Es ist davon auszugehen, dass viele Algorithmen zur Gesichtsdetektion, welche sich in nahezu allen heutigen Digitalkameras befinden, auf diesem Verfahren basieren.

Das Viola-und-Jones-Verfahren (VJ-Verfahren) ist ein typischer Vertreter der Sliding-Window-Klassifikation wie wir sie im vorherigen Abschnitt kennengelernt haben.

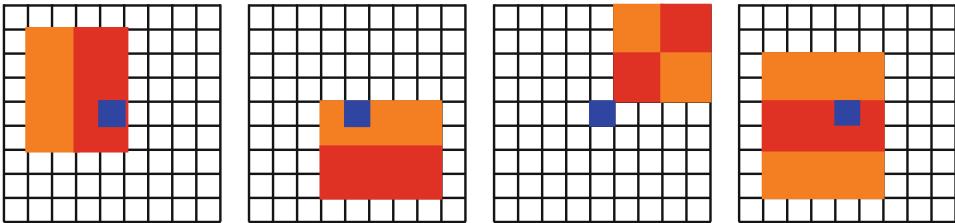


Abb. 20.3 Einfache Rechteckmerkmale beim Viola-und-Jones-Verfahren: In der Grafik sind die verwendeten Rechtecke für die Berechnung der Grauwertsummen markiert, diese können sowohl von der Position als auch von der Größe innerhalb des Fensters variieren

Einfache Rechteckmerkmale Wir hatten bereits erwähnt, dass bei der Objektlokalisierung ein schnelles Verfahren zur Berechnung von Merkmalen für einen gegebenen Bildausschnitt notwendig ist. Als Merkmale verwendet das VJ-Verfahren daher einfache Differenzen von Grauwertsummen in Rechtecken innerhalb des aktuell betrachteten Bildausschnitts. Allgemein lassen sich diese Merkmale wie folgt berechnen:

$$x_{\text{Rechteck}}(\mathbf{p}, g; (R_i)_{i=1}^r, (\gamma_i)_{i=1}^r) = \sum_{i=1}^r \gamma_i \left(\sum_{\mathbf{p}' \in R_i} g(\mathbf{p} + \mathbf{p}') \right) \quad (20.9)$$

Die Regionen R_i sind dabei Mengen von relativen Positionsangaben. In der Praxis ist es oft ausreichend, höchstens $r = 3$ Rechtecke zu verwenden und diese auch in speziellen Anordnungen, welche in Abb. 20.3 dargestellt sind. Die einzelnen Werte der Merkmale lassen sich wieder sehr leicht mittels Integralbildern (Abschn. 19.6) berechnen. Für den ersten Merkmalstyp in Abb. 20.3 sind zum Beispiel nur 6 Zugriffe auf ein Integralbilder notwendig. Dieser Merkmalstyp ist eine Art breiter Gradientenfilter in x -Richtung, welcher allein durch die linke obere Ecke und die Größe der Rechtecke bestimmt wird. Aufgrund ihrer Ähnlichkeit zu Haar-Waveletfunktionen werden diese Merkmale oft als *haar-like features* bezeichnet.

Die direkte Verwendung der Grauwerte des Eingabebildes führt dazu, dass die berechneten Merkmale bezüglich einfacher Grauwerttransformationen nicht invariant sind. Um dies zu beheben, können wir die Grauwerte innerhalb eines Fensters normieren:

$$g'(\mathbf{p}) = \frac{g(\mathbf{p}) - \mu_{\mathcal{R}}}{\sigma_{\mathcal{R}}}, \quad (20.10)$$

dabei sind $\mu_{\mathcal{R}}$ der Mittelwert und $\sigma_{\mathcal{R}}$ die Standardabweichung der Grauwerte innerhalb eines Fensters. Verwendet man diese normierten Grauwerte anstelle der ursprünglichen Grauwerte bei der Berechnung der Merkmale in (20.9) erhalten wir die Invarianz bezüglich linearer Grauwerttransformationen. Die effiziente Bestimmung des Mittelwertes $\mu_{\mathcal{R}}$ und der Standardabweichung $\sigma_{\mathcal{R}}$ kann wieder mittels Integralbildern von g und dem Bild g^2 der quadrierten Grauwerte erfolgen.

Klassifikator Betrachten wir die Merkmalstypen in Abb. 20.3, so ergeben sich selbst bei kleineren Fenstern wie zum Beispiel mit einer Größe von 24×24 Pixeln, tausende mögliche Merkmale welche für die Klassifikation verwendet werden können. Es ist daher notwendig einen Klassifikator einzusetzen welcher effizient wichtige Merkmale für die Klassifikation auswählen kann.

Als Klassifikator kommt bei diesem Verfahren ein Boosting-Klassifikator zum Einsatz. Diese Art des Klassifikators hatten wir in Abschn. 18.4.5 kennengelernt und die Bewertungsfunktion f entsteht durch Linearkombination von T Basisklassifikatoren:

$$f(\mathbf{x}; \mathbf{a}, (\boldsymbol{\theta}_t)_{t=1}^T) = \sum_{t=1}^T \alpha_t \cdot h_t(\mathbf{x}; \boldsymbol{\theta}_t). \quad (20.11)$$

Als Basisklassifikatoren wählen wir nun sehr einfache Schwellwertoperationen mit dem Index r des Merkmals, dem Schwellwert ζ und einer Parität $b_t \in \{-1, 1\}$ als Parameter, d. h. $\boldsymbol{\theta}_t = (r(t), \zeta_t, b_t)$:

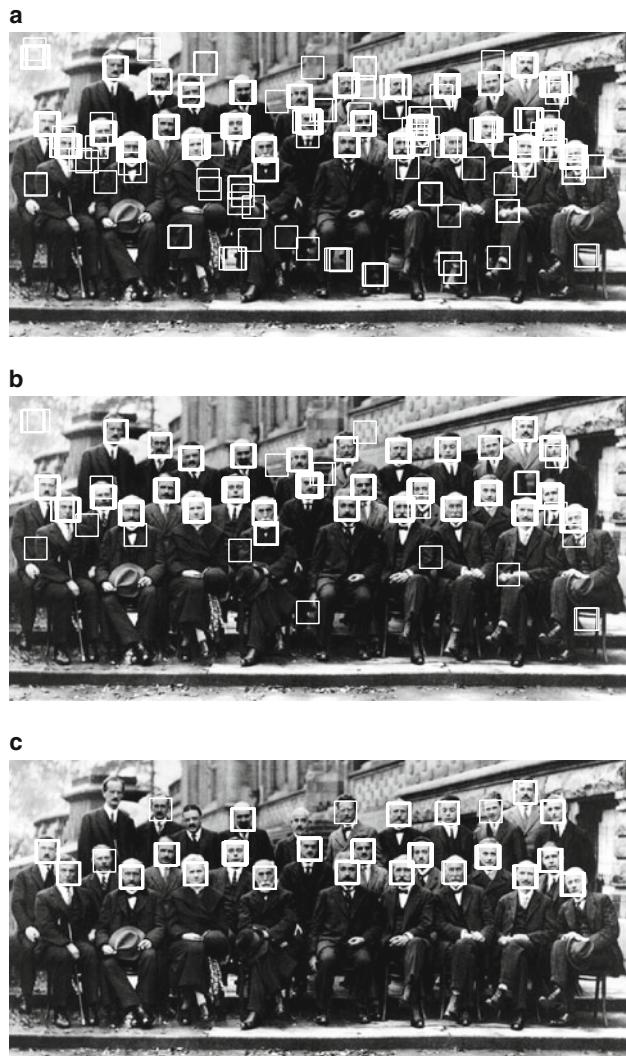
$$h_t(\mathbf{x}; \boldsymbol{\theta}_t) = \begin{cases} 1 & \text{wenn } b_t \cdot x_{r(t)} < b_t \cdot \zeta_t \\ -1 & \text{sonst.} \end{cases} \quad (20.12)$$

Das Lernen des Klassifikators kann dann wie in Abschn. 18.4.5 realisiert werden. Dem Verfahren müssen nur entsprechende positive und negative Lernbeispiele zur Verfügung gestellt werden.

Verwendung einer Kaskade Bei der Standardanwendung des VJ-Verfahrens, der Gesichtsdetektion, wird eine geringe Laufzeit des Verfahrens benötigt, dies gilt natürlich insbesondere bei der Verwendung des Verfahrens in Geräten mit geringerer Rechenleistung (z. B. Digitalkameras oder Smartphones). Das bisher beschriebene Verfahren verwendet bereits schon Merkmale welche mit wenigen Rechenoperationen berechnet werden können sowie eine Kombination einfacher Schwellwertklassifikatoren durch Boosting.

Potential für eine Verbesserung bietet die Tatsache, dass für die Auswertung eines Fensters bisher alle Basisklassifikatoren ausgewertet werden müssen um zu einem finalen Klassifikationsergebnis zu gelangen. Es lassen aber sich bestimmte Fenster schnell, d. h. unter Verwendung weniger Merkmale, als negative Beispiele zurückweisen. Im Falle der Gesichtsdetektion wären dies zum Beispiel homogene Gebiete. Aus dieser Grundidee lässt sich das Konzept einer Kaskade von (Boosting-)Klassifikatoren entwickeln. Eine Kaskade ist eine Menge $\mathcal{C} = (f_i, t_i)_{i=1}^C$ von C Klassifikatoren f_i mit zugehörigen Schwellwerten t_i . Bei der Auswertung der Kaskade werden die Klassifikatoren der Reihe nach durchlaufen und ist der Wert $f_i(\mathbf{x}_*)$ des aktuellen Klassifikators in Stufe i kleiner als der Schwellwert t_i , so erfolgt der Abbruch der Auswertung und das Beispiel \mathbf{x}_* wird der negativen Klasse

Abb. 20.4 Ergebnisse des Viola-und-Jones-Verfahrens für die Gesichtsdetektion mit einer unterschiedlichen Anzahl von Boosting-Klassifikatoren in der Kaskade. **a** Kaskade mit 8 Boosting-Klassifikatoren, **b** Kaskade mit 10 Boosting-Klassifikatoren, **c** Vollständige Kaskade



zugewiesen. Eine Kaskade ist im Vergleich zu Klassifikatoren welche auf Entscheidungsbäumen basieren, ein entarteter Baum mit Trennknoten, welche jeweils mindestens ein Blatt als Kindknoten besitzen.

Bei der Objektlokalisierung lässt sich eine Kaskade lernen, welche mit schnell auszuwertenden Boosting-Klassifikatoren bereits einen großen Teil aller negativen Bildausschnitte „zurückweist“, d. h. als negativ deklariert. Dadurch kann die benötigte Rechenzeit bei der Auswertung drastisch reduziert werden. Das Lernen einer Kaskade erfolgt oft unter

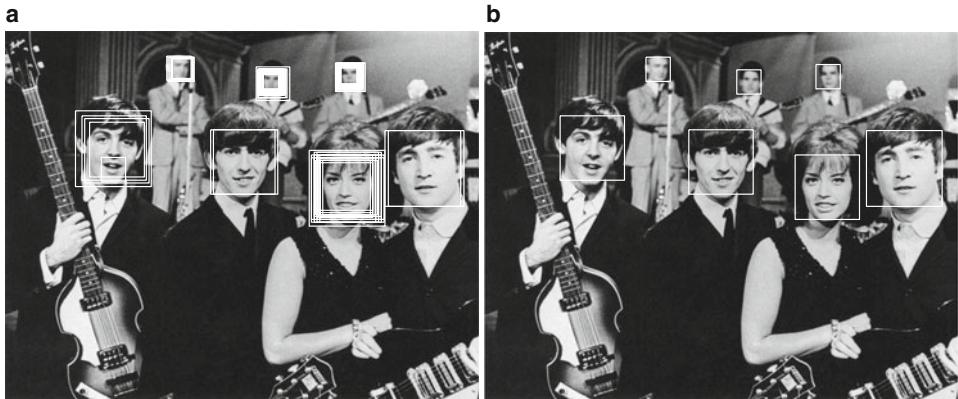


Abb. 20.5 Beispieldarstellung der Gesichtsdetektion **a** ohne und **b** mit Fusion der Objekthypothesen

Verwendung eines zusätzlichen Validierungsdatensatzes, welcher genutzt wird um die Falschpositivrate und die Richtigpositivrate der aktuellen Kaskade zu bestimmen und die Schwellwerte t_i anzupassen. Dabei wird von jeder Stufe der Kaskade eine minimale Richtigpositivrate gefordert. Der Schwellwert t_i kann dann so eingestellt werden, dass diese Rate auf jeden Fall erreicht wird. Für einen konkreten Algorithmus zur Bestimmung einer Kaskade verweisen wir an dieser Stelle auf [81].

In Abb. 20.4 sind Beispiele der Lokalisation mit Teilberechnungen einer Kaskade dargestellt, dabei ist sehr schön die schrittweise Zurückweisung von negativen Beispielen zu erkennen. Jedoch wird zum Beispiel auch die korrekte Detektion des Gesichtes in der linken oberen Ecke von Klassifikatoren am Ende der Kaskade falsch zurückgewiesen.

Das Prinzip der kaskadierten Klassifikation lässt sich bei beliebigen binären Klassifikatoren anwenden und ist besonders sinnvoll zur Beschleunigung der Auswertung einsetzbar, wenn (1) im Testschritt a priori viele negative Beispiele auftreten und (2) wenn sich durch einen Parameter des Klassifikators ein Kompromiss zwischen Auswertungszeit und Genauigkeit einstellen lässt.

20.3.3 Fusion von Objekthypothesen

Ein Beispiel für ein Detektionsergebnis ist in Abb. 20.5 zu sehen. Wie zu erkennen ist, entsprechen oft mehrere Objekthypothesen einer korrekten Detektion. Dies ist ein typisches Problem bei der Objektlokalisierung und abhängig von der Robustheit der Merkmale bezüglich kleinerer Translationen. Für praktische Anwendungen ist eine Fusion der Objekthypothesen notwendig. Sei $\mathcal{D}(g)$ die Menge der ermittelten Objekthypothesen für ein Bild g . Ein einfaches Verfahren zur Fusion dieser Hypothesen ist es, Gruppen von Hypothesen zu betrachten, bei denen die Fenster \mathcal{R} signifikant überlappen. Die Überlappung oder

Ähnlichkeit von zwei Fenstern wird dabei oft durch die relative Schnittfläche bestimmt:

$$A(\mathcal{R}, \mathcal{R}') = \frac{|\mathcal{R} \cap \mathcal{R}'|}{|\mathcal{R} \cup \mathcal{R}'|}. \quad (20.13)$$

Als Repräsentant der Gruppe wird oft die Hypothese mit dem größten Ausgabewert $f(\mathbf{x}(\mathcal{R}; g))$ des Klassifikators gewählt, da sich der Klassifikator für diese Hypothese am „sichersten“ ist. Dieses Konzept wird auch oft als *non-maximum suppression* bezeichnet und kann bei vielen Methoden angewendet werden, bei denen ungewünschte Nebenmaxima auftreten, wie zum Beispiel bei der Hough-Transformation (Abschn. 10.8). Das Verfahren durchläuft alle Hypothesen absteigend nach den Ausgabewerten des Klassifikators. Ist die Überlappung der aktuellen Hypothese zu einem Repräsentanten der bisherigen Gruppen größer als ein gewisser Schwellwert so wird diese der Gruppe hinzugefügt, ansonsten wird die aktuelle Hypothese zu einem Repräsentanten einer neuen Gruppe.

Alternativ kann natürlich auch eine Fusion der Objekthypothesen einer Gruppe erfolgen, zum Beispiel mittels der Bestimmung eines gewichteten Mittelwertes der Koordinaten. Als Gewichte dienen dann normierte Ausgabewerte des Klassifikators.

20.3.4 HOG Detektor

Wie wir gesehen haben, eignet sich das Verfahren von Viola und Jones sehr gut für die Gesichtsdetektion. Dennoch hat sich gezeigt, dass die verwendeten Merkmale sich nur bedingt für die Lokalisation von Objekten anderer Kategorien mit höherer Intraklassenvarianz eignen. In Abschn. 19.9.3 hatten wir Gradientenrichtungshistogramme und sogenannte HOG-Merkmale kennengelernt, welche invariant nicht nur bezüglich monotoner Grauwerttransformationen sind. Diese wurden ursprünglich für die Fußgägerdetektion entwickelt und sind aktuell die erste Wahl für die Merkmalsberechnung bei der Lokalisation. Als Klassifikator wird oft in diesem Fall ein linearer SVM-Klassifikator verwendet sowie wir ihn in Abschn. 18.4.6 kennengelernt haben. Das Lernen dieses Klassifikators erfolgt oft mit dem Bootstrapping-Ansatz zur iterativen Bestimmung und Verwendung von schwierigen negativen Beispielen.

Im Gegensatz zum VJ-Verfahren werden meistens komplett HOG-Merkmale bestimmt und für die Klassifikation verwendet. Für die Anwendung bei der Sliding-Window-Klassifikation ist die effiziente Bestimmung dieser Merkmale für jeden Bildausschnitt entscheidend. HOG-Merkmale setzen sich aus normierten Richtungshistogrammen einzelner Zellen des aktuellen Fensters zusammen. Für die effiziente Berechnung der Merkmale, können wir daher zunächst die Histogramme für jede (nicht-überlappenden) Zelle des Eingabebildes bestimmen. Bei einem $M \times N$ großen Bild und einer Zellengröße von $c \times c$ Pixeln (üblicherweise ist $c = 8$) ergibt sich dadurch ein Merkmalsbild $\mathbf{x} : \{1, \dots, \tilde{M}\} \times \{1, \dots, \tilde{N}\} \rightarrow \mathbb{R}^d$, wobei d die Anzahl der Quantisierungsstufen der Gradientenrichtungen ist und wir davon ausgehen, dass M und N Vielfache von c sind mit

$\tilde{M} = \frac{M}{c}$ und $\tilde{N} = \frac{N}{c}$. Wenn wir uns nun auf Fenster beschränken, welche nur komplette Zellen enthalten, so reduziert sich die Bestimmung eines HOG-Merkmales für einen gegebenen Bildausschnitt auf eine reine Konkatenation der Histogramme der enthaltenden Zellen.

Wir gehen im Folgenden von einer festen Größe $m_{\mathcal{R}} \times n_{\mathcal{R}}$ der Fenster aus, wobei $m_{\mathcal{R}}$ und $n_{\mathcal{R}}$ wieder Vielfache der Zellengröße c sein sollen. Bei der Anwendung eines linearen Klassifikator $f(\mathbf{x}(\mathcal{R}; g)) = \mathbf{w}^T(\mathbf{x}(\mathcal{R}; g))$ ergibt sich für das Bild der Ausgabewerte des Klassifikators bei Betrachtung einer festen Skalierung:

$$s(\mathbf{p}; g) = f(\mathbf{x}(p_x, p_y, m_{\mathcal{R}}, n_{\mathcal{R}}; g)) \quad (20.14)$$

$$= \mathbf{w}^T(\mathbf{x}(p_x, p_y, m_{\mathcal{R}}, n_{\mathcal{R}}; g)) \quad (20.15)$$

$$= \sum_{z=1}^d \sum_{j=1}^{\tilde{M}} \sum_{k=1}^{\tilde{N}} w(j, k, z) \cdot x(\tilde{p}_x + j, \tilde{p}_y + k, z) \quad (20.16)$$

$$= \sum_{z=1}^d (w_z^R * x_z)(\tilde{p}_x, \tilde{p}_y). \quad (20.17)$$

Im Falle eines linearen Klassifikators lassen sich also die kompletten Ausgabewerte des Klassifikators mittels d Faltungen berechnen, in dem wir die Hyperebene des Klassifikators in mehrere (gespiegelte) Filtermasken w_z^R zerlegen. Dies beschleunigt die Auswertung des Klassifikators für jedes Fenster erheblich, da wir nicht explizit HOG-Merkmale für jeden Ausschnitt bestimmen müssen sondern lediglich einzelne normierte Richtungshistogramme für jede Zelle im gesamten Bild. Eine weitere Beschleunigung ist durch die Durchführung der Faltungen im Frequenzraum möglich.

20.3.5 Teilebasierte Verfahren

Die beiden vorgestellten Verfahren zur Objektlokalisierung, die Methode nach Viola und Jones und der HOG-Detektor, sind Basismethoden aktueller Ansätze. Im Laufe der Jahre wurden viele Erweiterungen entwickelt, welche den Rahmen dieses Buches bei weitem sprengen würden. Ein Bereich dieser Erweiterungen beschäftigt sich zum Beispiel mit der Entwicklung teilebasierter Verfahren. Der HOG-Detektor lernt zum Beispiel immer nur eine feste starre Filtermaske für HOG-Merkmale und kann daher zum Beispiel weder mit signifikant verschiedenen Perspektiven oder starken Deformationen der Objekte umgehen. Ein möglicher Ausweg ist daher die Repräsentation von Objekten als Kollektion einzelner Objektteile. Die Lokalisation wird dann durch eine Maximierung einer Zielfunktion durchgeführt, welche die Ausgabe einzelner Detektoren für die Objektteile enthält sowie einen Term für die Bewertung der Konstellation der Positionen dieser Objektteile. Dem interessierten Leser sei an dieser Stelle die Lektüre der Originalarbeit von [23] zu sogenannten *deformable part models* empfohlen.

20.4 Semantische Segmentierung

Im Folgenden wollen wir uns mit der semantischen Segmentierung beschäftigen und wir beschränken uns dabei auf zwei klassische und sehr einfache Verfahren.

20.4.1 Klassifikation lokaler Deskriptoren

Bei der semantischen Segmentierung soll jedem Pixel eine Klasse zugewiesen werden. So wollen wir zum Beispiel in einem Bild automatisch pixelgenau erkennen, wo sich die Straße befindet oder ein anderes Auto. Bei industriellen Bildverarbeitungsaufgaben handelt es sich oft um binäre Aufgabenstellungen, bei denen man sich für jeden Pixel zwischen Vordergrund und Hintergrund entscheiden muss. In Abschn. 10.1 haben wir uns bereits mit Binarisierungsverfahren beschäftigt, welche das Histogramm der Grauwerte in einem Bild analysieren um daraus einen geeigneten Schwellwert zu ermitteln. Die Analyse des Histogramms lässt sich als Lernschritt interpretieren und im Testschritt wird aufgrund des Grauwertes des aktuellen Pixels eine Klassifikationsentscheidung getroffen.

Die Betrachtung nur des Grauwertes des aktuellen Pixels für die Entscheidung ist aber nur in sehr einfachen Sonderfällen sinnvoll. Stellen Sie sich einfach an der Stelle einmal vor, Sie müssten aufgrund eines einzelnen Grauwertes die Entscheidung treffen, ob dieser Pixel zu einem Auto gehört oder nicht. Im Abschn. 20.2 hatten wir globale Merkmale betrachtet, welche aufgrund des gesamten Bildes Merkmale berechnen und zur Klassifikation verwenden. Bei der semantischen Segmentierung ist ein Mittelweg wichtig, schließlich soll sich die Klassifikationsentscheidung auf den lokalen Bereich eines Pixels beziehen, dieser Bereich soll aber auch groß genug sein um eine vernünftige Entscheidung zu treffen. Ein Schritt in diese Richtung ist die Berechnung von lokalen Deskriptoren, so wie wir sie in Abschn. 19.9.2 kennengelernt haben. Da wir für jeden einzelnen Pixel einen Merkmalsvektor benötigen, ist die Verwendung von Detektoren um interessante Punkte in Bildern auszuwählen nicht sinnvoll.

Ein einfaches Verfahren auf Basis der Klassifikation von lokalen Deskriptoren lässt sich demnach folgendermaßen skizzieren:

1. Berechnung von lokalen Deskriptoren für jeden Pixel,
2. Klassifikation der lokalen Deskriptoren in jedem Pixel und Einteilung in die jeweiligen Klassen,
3. Nachbearbeitung mit einer klassischen unüberwachten Segmentierung.

Im Folgenden wollen wir auf diese Schritte näher eingehen. Weitere Details lassen sich auch in der Publikation von Fröhlich et al. [26] finden.

Berechnung lokaler Deskriptoren Als lokaler Deskriptor wird oft ein SIFT-Deskriptor eingesetzt. Wie in Abschn. 19.9.2 dargestellt handelt es sich bei diesem um ein Histogramm von Gradientenrichtungen ähnlich zu HOG-Merkmalen, sowie sie in Abschn. 19.9.3 vorgestellt wurden. Unterschiede bestehen in der Art der Normierung und der Wahl eines räumlichen Glättung der Histogramme. Bei der semantischen Segmentierung eignen sich SIFT-Deskriptoren zur robusten Beschreibung von Kanten, Ecken und Konturelementen. Oft wird in diesem Zusammenhang auch auf die Rotationsinvarianz des Deskriptors verzichtet. Da wir nun diese Deskriptoren an jeder Stelle des Bildes berechnen müssen, ist eine effiziente Bestimmung der Gradientenrichtungshistogramme notwendig, sowie sie in Abschn. 19.9.6 anhand der Integralbilder vorgestellt wurde. Eine weitere Möglichkeit ist die Berechnung der Deskriptoren nicht in jedem Pixel aber in einem festen Abstand zueinander auf einem festen Gitter.

Um auch Objekte in unterschiedlichen Größen und Skalen zu erkennen, werden oft mehrere lokale Deskriptoren auf unterschiedlichen Ebenen einer Bildpyramide bestimmt, d. h. mit unterschiedlichen Größen der lokalen Nachbarschaft.

Lernen eines Klassifikators für Deskriptoren Nach der Berechnung der lokalen Deskriptoren kann im Lernschritt ein Klassifikator zur Unterscheidung von K Klassen gelernt werden. Anders als bei der Bildklassifikation erhält man nun nicht nur ein einziges Lernbeispiel für jedes Bild, sondern theoretisch ein Lernbeispiel für jeden Pixel in den gegebenen Bildern des Lerndatensatzes. Dadurch entsteht schon bereit bei wenigen Bildern im Lernschritt ein gewaltiger Satz von Merkmalsvektoren. Daher ist zu beachten, dass sich für die Aufgabenstellung der semantischen Segmentierung nicht jeder Klassifikator eignet, sondern besonders darauf geachtet werden muss, dass dieser mit vielen Lerndaten effizient arbeiten kann. Wir hatten in Abschn. 18.4.4 bereits randomisierte Entscheidungsäume kennengelernt. Dieses Klassifikationsverfahren wurde in vielen Arbeiten der semantischen Segmentierung eingesetzt und besitzt weitere interessante Erweiterungsmöglichkeiten, welche wir im nächsten Abschnitt noch näher betrachten werden.

Erfolgt die Berechnung der lokalen Deskriptoren nicht in jedem Pixel, so kann auch nicht in jedem Pixel ein Klassifikationsergebnis ermittelt werden. Eine einfache Idee um die Klassifikationsentscheidungen zu extrapolieren ist es, Wahrscheinlichkeitskarten für jede Klasse mit einem randomisierten Entscheidungsbaum zu berechnen. Diese Wahrscheinlichkeitskarten, an denen nur an Pixeln eine Schätzung eingetragen wird, an denen auch eine Berechnung eines lokalen Deskriptors erfolgt ist, werden anschließend mit Gaußfiltern geglättet um eine Schätzung der Klassenwahrscheinlichkeiten für jeden Pixel zu erhalten.

Fusion lokaler Entscheidungen Am Anfang des Abschnittes hatten wir uns bereits den Unterschied der semantischen Segmentierung zur unüberwachten Segmentierung näher angesehen. Oft ist es vorteilhaft, Techniken der unüberwachten Regionensegmentierung auch bei der semantischen Segmentierung einzusetzen. Eine Regionensegmentierung liefert eine Einteilung des Bildes in sogenannte Superpixel, Bildregionen die einem Ho-

mogenitätskriterium genügen (siehe zum Beispiel Abschn. 10.5). Grundannahme ist nun, dass diese Pixel innerhalb dieser Superpixel auch oft ein und derselben Objektklasse zuzuordnen sind. Eine Klassifikationsentscheidung aufgrund der berechneten Wahrscheinlichkeitskarten aus dem vorherigen Schritt kann demnach einheitlich für einen Superpixel mittels Mehrheitsentscheidung getroffen werden. Dem Superpixel wird die Klasse zugeordnet, welche insgesamt die größte Wahrscheinlichkeitsmasse für diese Bildregion besitzt.

Viele andere Verfahren versuchen direkt Superpixel aufgrund von berechneten Merkmalen zu klassifizieren ohne den Umweg über lokale Deskriptoren innerhalb des Superpixels. Die lokalen Entscheidungen werden dann mit Markovfeldern zu einem konsistenten Gesamtergebnis kombiniert. Die dazu notwendigen Techniken sprengen aber bei weitem den Rahmen dieses Buches und wir verweisen den interessierten Leser auf das Buch von Blake, Kohli und Rother [4]. Die Grundzüge der Modellierung mit paarweisen Abhängigkeiten von Pixeln hatten wir in Abschn. 9.4 vorgestellt.

20.4.2 Klassifikation mit effizienten Pixelmerkmalen

Das im vorherigen Abschnitt beschriebene Verfahren besitzt zwei wesentliche Nachteile: (1) Die Klassifikation basiert zunächst nur auf lokalen Merkmalen, welche oft wenig Information zur Unterscheidung von Objektkategorien enthalten und (2) vor der Klassifikation müssen zunächst alle lokalen Merkmale berechnet werden. Der zweite Nachteil ist vor allem entscheidend für Anwendungen, bei denen nur wenige Hardwareressourcen zur Verfügung stehen, sowie zum Beispiel bei Fahrerassistenzsystemen. Wir werden daher im Folgenden ein Verfahren vorstellen welches sehr wenige einfache lokale Merkmale während der Klassifikation bestimmen muss. In Abschn. 20.4.3 werden wir dieses Verfahren dann erweitern um den Nachteil von reinen lokalen Merkmalen bei der Klassifikation zu vermeiden.

Das Verfahren basiert auf randomisierten Entscheidungsbäumen, welche wir bereits in Abschn. 18.4.4 kennengelernt haben, und wurde ursprünglich von [67] entwickelt und dann von [27] erweitert. Die Idee ist es zunächst wieder diesen Klassifikatorotyp einzusetzen, um lokale Merkmale zu klassifizieren. Im Gegensatz zum vorherigen Verfahren verwenden wir aber jetzt Merkmale, welche sich mit wenigen Operationen direkt anhand der gegebenen Werte des Eingabebildes berechnen lassen. Wir betrachten im Folgenden die Merkmalsberechnung für einen einzelnen Pixel $\mathbf{p} \in \mathbb{Z}^2$ mit einer vordefinierten Nachbarschaft. Ein Beispiel für einfache Pixelmerkmale sind Differenzen der Grauwerte zweier Nachbarpixeln:

$$x_{\text{Differenz}}(\mathbf{p}, g; \mathbf{v}_1, \mathbf{v}_2) = g(\mathbf{p} + \mathbf{v}_1) - g(\mathbf{p} + \mathbf{v}_2), \quad (20.18)$$

dabei ist g das aktuelle Eingabebild und die Positionen \mathbf{v}_1 und \mathbf{v}_2 der Nachbarpixel sind Parameter der Merkmalsfunktion x_{diff} . Hohe Differenzen in den Grauwerten benachbarter

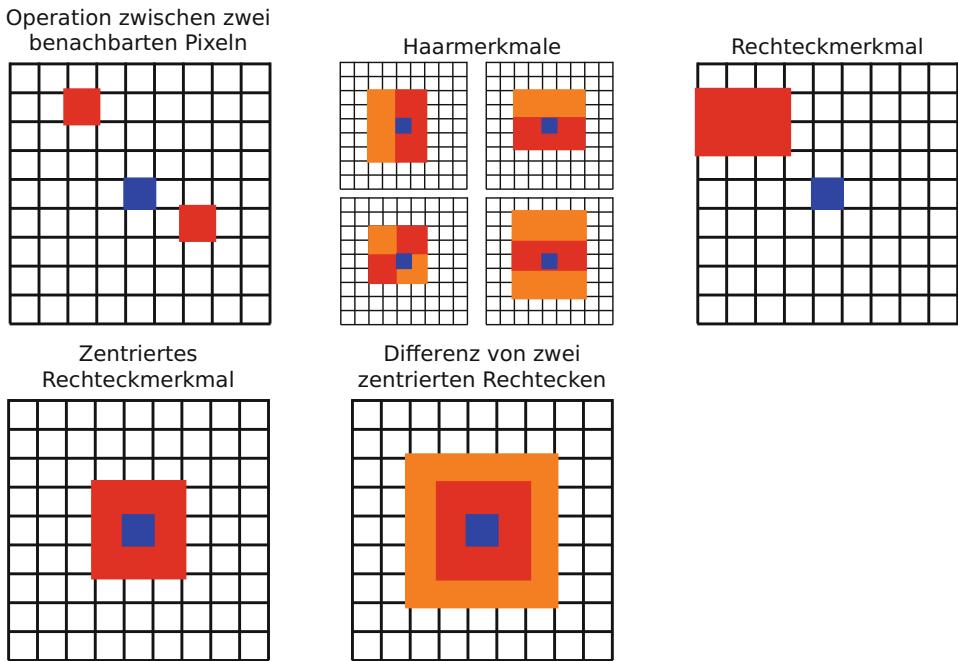


Abb. 20.6 Einfache Pixelmerkmale bei der semantischen Segmentierung mit randomisierten Entscheidungsbäumen

Pixel können zum Beispiel auf eine Kante hinweisen. Generell können in (20.18) beliebige Operationen von zwei Grauwerten verwendet werden.

Zusätzlich zu Operationen mit jeweils zwei Nachbarn des aktuellen Pixels können wir auch Regionen in der Nachbarschaft betrachten und Merkmale aufgrund der Summe der Grauwerte in diesen Regionen entwerfen. Diese Art der Merkmale hatten wir schon in Abschn. 20.3.2 kennengelernt. In Abb. 20.6 sind typische Pixelmerkmale dargestellt, welche sich natürlich auch bei RGB Bildern für jeden Kanal berechnen lassen.

Einzelne dieser Pixelmerkmale können effizient berechnet werden, jedoch ist zu bedenken, dass jeder Merkmalstyp eine riesige Anzahl von möglichen Merkmalen erzeugt. Betrachtet man zum Beispiel allein das Differenzmerkmal aus (20.18), so gibt es bei der Nachbarschaft der Größe $n \times n$ insgesamt $(n^4 - n^2)$ Möglichkeiten für die sinnvolle Belegung der Parameter v_1 und v_2 (der Fall $v_1 = v_2$ wird nicht betrachtet). Dies sind bei der üblichen Nachbarschaftsgröße von $n = 21$ bereits schon 194.040 mögliche Merkmale. Alle Merkmale für jeden Pixel zu berechnen und dann damit zu lernen, wäre viel zu aufwendig und würde selbst bei kleinen Datensätzen an die Grenze des verfügbaren Speicherplatzes stoßen. An dieser Stelle kommen randomisierte Entscheidungsbäume ins Spiel. Bei der Bestimmung von Basisklassifikatoren in den einzelnen Trennknoten untersuchen randomisierte Entscheidungsbäume nicht jedes einzelne Merkmal sondern nur eine zu-

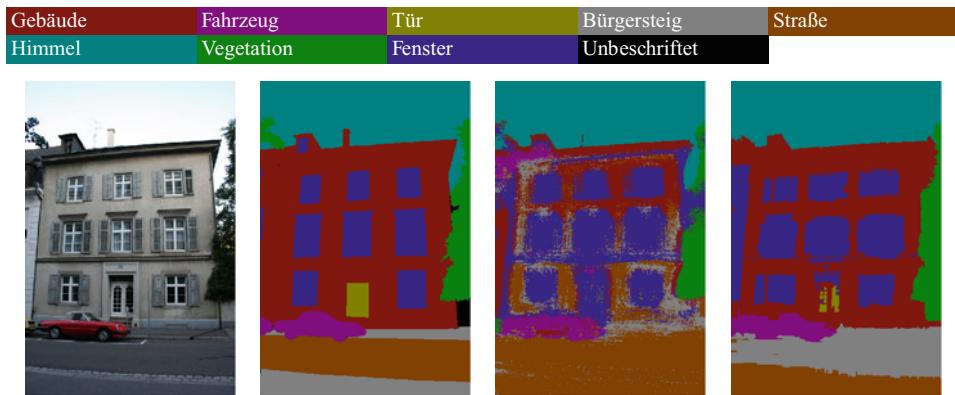


Abb. 20.7 Ergebnisse der semantischen Segmentierung bei der visuellen Szenenanalyse: **a** Eingabebild, **b** manuelle Beschriftung, **c** Klassifikation nur mittels lokaler Merkmale, **d** Verwendung zusätzlicher automatisch ausgewählter Kontextmerkmale. Die Farben in den Ergebnisbildern stellen die einzelnen Objektkategorien dar und sind oben aufgelistet. Das Eingabebild entstammt der eTRIMS-Datenbank (http://www.ipb.uni-bonn.de/projects/etrimis_db/)

fällige Auswahl fester Größe. Bei der semantischen Segmentierung mit effizienten Pixelmerkmalen werden daher einfach Merkmalstyp und Parameter der Merkmalsfunktionen ausgewürfelt um nur einen sehr geringen Anteil aller möglichen Merkmale zu erhalten. Die Anzahl der zufällig gezogenen Merkmale sowie die Anzahl der zufällig gezogenen Beispiele sind dabei die entscheidenden Faktoren für die Zeit, welche zum Lernen benötigt wird.

20.4.3 Iterative Verwendung von Kontextmerkmalen

Bisher sind wir bei Verfahren der semantischen Segmentierung immer von einer Klassifikation rein lokaler Merkmale ausgegangen, Klassifikationsentscheidungen für einzelne Pixel basieren daher immer nur auf Informationen aus einer eingeschränkten festgelegten Nachbarschaft um den Pixel. Dies ist aber oft nicht ausreichend, wie in Abb. 20.7c beispielhaft dargestellt. Dort werden zum Beispiel Pixel im oberen Bereich des Bildes als Fahrzeug klassifiziert.

Dieses Problem lässt sich natürlich mit einer größeren Nachbarschaft beheben, dennoch fehlen bei der Klassifikationsentscheidung für einen Pixel wichtige Kontextinformationen anderer Objekte. Eine Frage stellt sich in diesem Zusammenhang sofort: Was sind überhaupt Kontextinformationen bei der semantischen Segmentierung? Mit Kontextinformationen wollen wir im Folgenden Informationen über das Auftreten und die Lage von anderen Objekten bezeichnen. Abstrakt gesehen sind dies Antworten auf Fragen wie etwa „Befindet sich über dem aktuellen Pixel ein Bereich der Kategorie Himmel?“ oder „Sind in diesem Bild Fahrzeuge zu erkennen“. Diese Informationen sind besonders dann entschei-

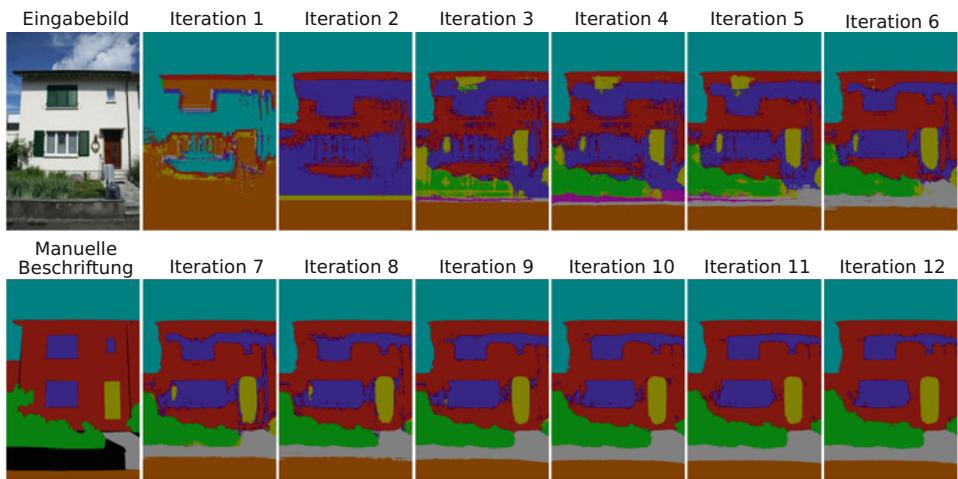


Abb. 20.8 Iterative semantische Segmentierung mit randomisierten Entscheidungsbäumen: Dargestellt ist das Klassificationsergebnis für eine unterschiedliche Anzahl von verwendeten Tiefen der Bäume, d. h. Iterationen des Verfahrens aus Abschn. 20.4.3. Das verwendete Eingabebild ist ebenfalls aus der eTRIMS-Datenbank und die Bedeutung der Farben lässt sich Abb. 20.7 entnehmen

dend, wenn ansonsten nur lokale Merkmale vorliegen. In Abb. 20.7c ist ein Beispiel eines Ergebnisses einer semantischen Segmentierung zu sehen, welche nur einfache Pixelmerkmale verwendet, sowie sie im vorherigen Abschnitt vorgestellt wurden. Es lassen sich im Beispiel viele Inkonsistenzen bezüglich der Anordnung von Objekten erkennen, so ist etwa die Positionierung eines Fahrzeugs auf dem Dach eines Hauses eher als unwahrscheinlich anzusehen. Ziel von Kontextmerkmalen ist es vor allem diese relative Anordnung von Objekten aus den Daten zu lernen.

Beim Entwurf von Kontextmerkmalen, steht man zunächst vor einem typischen Henne-Ei-Problem: um Merkmale zu berechnen, welche Hinweise auf die Positionen von Objekten beinhalten, ist es zunächst erst einmal notwendig diese Objekte auch in den Bildern zu lokalisieren. Dies ist aber wiederum die Hauptaufgabe des Verfahrens der semantischen Segmentierung. Einen Ausweg bietet hier wieder einmal der randomisierte Entscheidungsbaum und eine zyklische Schätzung. Betrachten wir zunächst die Klassifikation: Grundidee ist es, dass wir bei der Entscheidung für jeden Pixel nicht den Baum vollständig bis zu den Blättern durchlaufen sondern nur bis zu einer festen Tiefe. Während des Lernens der Entscheidungsbäume merken wir uns, wie bei den Blattknoten, eine Schätzung der empirischen Wahrscheinlichkeitsverteilung \hat{p} der Kategorien, d. h. wie viele Lernbeispiele einer Kategorie durchlaufen diesen Trennknoten beim Lernen. Dadurch erhalten wir zunächst einmal für jeden Pixel eine grobe Schätzung der Zugehörigkeiten zu den einzelnen Kategorien, d. h. eine vorläufige Wahrscheinlichkeitskarte für jede Kategorie. Dies ist beispielhaft in Abb. 20.8 dargestellt.

Auf Grundlage dieser Wahrscheinlichkeitskarte ist es nun möglich Kontextmerkmale zu bestimmen. Dabei verwenden wir wieder einmal einfache Rechteckmerkmale, diesmal aber um geschätzte Klassenwahrscheinlichkeiten \hat{p} in einem bestimmten Bereich R des Bildes zu summieren:

$$x_{\text{Kontext}}(\mathbf{p}, g; R, \kappa) = \sum_{\mathbf{v} \in R} \hat{p}(y(\mathbf{p} + \mathbf{v}) = \kappa) \quad (20.19)$$

Bei obiger Formel haben wir die Zufallsvariable der Beschriftung im Pixel $\mathbf{p} + \mathbf{v}$ mit $y(\mathbf{p} + \mathbf{v})$ bezeichnet. Auch die Summation von Klassenwahrscheinlichkeiten kann mittels Integralbildern schnell berechnet werden. In diesem Fall wird einfach für jede Klasse κ ein Integralbild bestimmt. Im Lernschritt werden sowohl die Region als auch die Klasse mehrfach zufällig gezogen um eine Teilmenge von Merkmalen zu erhalten. Dies kann aber natürlich erst ab Tiefe zwei der Bäume durchgeführt werden, da auch im Lernschritt Wahrscheinlichkeitskarten für jedes Lernbeispiel bestimmt werden müssen.

Abbildung 20.8 zeigt das Ergebnis der semantischen Segmentierung mit diesen iterativ bestimmten Kontextmerkmalen. Ein Vorteil dieses Verfahrens ist es, dass selbst zu einem frühen Zeitpunkt ein grobes Klassifikationsergebnis zur Verfügung steht. Dies ist besonders in zeitkritischen Anwendungen, so wie sie in der Automobilindustrie häufig anzutreffen sind, wichtig. Für weitere Details verweisen wir den interessierten Leser auf die Veröffentlichung von [27].

Teil IV

Mathematische Hilfsmittel

Auf Grund der diskreten Natur digitaler Bilder extrahiert man stets diskrete Strukturen aus den Bildern. Diese diskreten Strukturen müssen aber oft durch analoge Modelle beschrieben werden, d. h. aus den diskreten Werten muss man Parameter, Transformationen, geometrische Objekte usw. schätzen bzw. berechnen. Dies geschieht mit der Ausgleichsrechnung, zu der die verschiedensten sogenannten *Schätzer* gehören. Wir hatten im Abschn. 18.3 schon Schätzer aus der Perspektive der Stochastik und Statistik betrachtet. Im Folgenden werden wir uns mit weiteren Aspekten der Ausgleichsrechnung beschäftigen, da sie ein fundamentales Basiswerkzeug der 2D- und 3D-Bildverarbeitung ist.

21.1 Quadratische Formen und Eigenwertprobleme

Ausgleichsprobleme führen oft auf Eigenwertprobleme und diese stehen in direktem Zusammenhang zu quadratischen Formen. Es seien \mathbf{x}, \mathbf{y} Vektoren und \mathbf{A} eine dazu passende Matrix. Wir betrachten zunächst eine lineare Form in \mathbf{x}

$$z = \mathbf{x}^T \mathbf{A} \mathbf{y} \quad (21.1)$$

und wollen den Gradienten von z bilden. Durch leichtes Nachrechnen finden wir:

$$\nabla_{\mathbf{x}} z = \mathbf{A} \mathbf{y}. \quad (21.2)$$

Nun bilden wir eine zweite lineare Form, indem wir \mathbf{x} und \mathbf{y} vertauschen:

$$w = \mathbf{y}^T \mathbf{A} \mathbf{x} \quad (21.3)$$

Wir können nun obige Ableitungsregel verwenden, indem wir die Form transponieren (w ist nur eine Zahl):

$$w = \mathbf{x}^T \mathbf{A}^T \mathbf{y} \quad (21.4)$$

Nach obiger Ableitungsregel ergibt sich dann:

$$\nabla_{\mathbf{x}} w = \mathbf{A}^T \mathbf{y}. \quad (21.5)$$

Betrachten wir nun eine quadratische Form

$$u = \mathbf{x}^T \mathbf{A} \mathbf{x}, \quad (21.6)$$

so können wir diese nach der Produktregel ableiten. Dabei nutzen wir obige beide Ableitungsregeln, also ergibt sich:

$$\nabla_{\mathbf{x}} u = (\mathbf{A} + \mathbf{A}^T) \mathbf{x}. \quad (21.7)$$

Nur wenn \mathbf{A} symmetrisch ist, gilt:

$$\nabla_{\mathbf{x}} u = 2\mathbf{A} \mathbf{x}. \quad (21.8)$$

Jetzt betrachten wir eine andere quadratische Form:

$$z = (\mathbf{A} \mathbf{x})^T (\mathbf{A} \mathbf{x}) = \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x}. \quad (21.9)$$

Wir bilden jetzt nicht die Ableitung nach \mathbf{x} , sondern nach der Matrix \mathbf{A} , es ergibt sich:

$$\nabla_{\mathbf{A}} z = 2\mathbf{A}(\mathbf{x} \mathbf{x}^T). \quad (21.10)$$

Ist \mathbf{x} ein Zufallsvektor und wir bilden den Erwartungswert, so ist:

$$E(\nabla_{\mathbf{A}} z) = 2\mathbf{A}\Sigma_{\mathbf{x}}. \quad (21.11)$$

Dabei ist wie üblich $\Sigma_{\mathbf{x}}$ die Kovarianzmatrix. Man kann das alles noch um eine weitere Matrix ergänzen, dazu betrachten wir die quadratische Form:

$$z = (\mathbf{A} \mathbf{B} \mathbf{x})^T (\mathbf{A} \mathbf{B} \mathbf{x}) = \mathbf{x}^T \mathbf{B}^T \mathbf{A}^T \mathbf{A} \mathbf{B} \mathbf{x}. \quad (21.12)$$

Die Ableitung ist nun:

$$\nabla_{\mathbf{A}} z = 2\mathbf{A} \mathbf{B} (\mathbf{x} \mathbf{x}^T) \mathbf{B}^T. \quad (21.13)$$

Im stochastischen Fall ist:

$$E(\nabla_{\mathbf{A}} z) = 2\mathbf{A} \mathbf{B} \Sigma_{\mathbf{x}} \mathbf{B}^T. \quad (21.14)$$

Jetzt betrachten wir wie zu Beginn wieder die einfache quadratische Form:

$$z = \mathbf{x}^T \mathbf{A} \mathbf{y}. \quad (21.15)$$

Wir bilden nicht die Ableitung nach \mathbf{x} , sondern nach der Matrix \mathbf{A} , dann ergibt sich:

$$\nabla_{\mathbf{A}} z = \mathbf{x} \mathbf{y}^T. \quad (21.16)$$

Wenn wir nun ein lineares Ausgleichsproblem der Form

$$z = \|\mathbf{Ax} - \mathbf{b}\|^2 \rightarrow \text{Minimum} \quad (21.17)$$

betrachten, dann können wir z auch so darstellen:

$$z = (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) = \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{b} - 2\mathbf{b}^T \mathbf{A} \mathbf{x}. \quad (21.18)$$

Mit unseren Ableitungsregeln ergibt sich dann:

$$\nabla_{\mathbf{A}} z = 2(\mathbf{Ax} - \mathbf{b}) \mathbf{x}^T = 0. \quad (21.19)$$

Im Zusammenhang mit Ableitungen von quadratischen Formen erhält man oft Eigenwertprobleme. Gegeben sei eine reelle, quadratische Matrix \mathbf{B} der Dimension (n, n) . Die Lösung des Eigenwertproblems

$$\mathbf{Bx} = \lambda \mathbf{x} \quad (21.20)$$

liefert Eigenwerte λ zu den zugehörigen Eigenvektoren \mathbf{x} . Ist die Matrix \mathbf{B} symmetrisch, dann existieren n reelle Eigenwerte $\lambda_1, \lambda_2, \dots, \lambda_n$ und es existiert ein zugehöriges Orthonormalsystem von Eigenvektoren $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$. Ist die Matrix \mathbf{B} noch zusätzlich positiv semidefinit, d. h. es gilt $\mathbf{x}^T \mathbf{Bx} \geq 0 \forall \mathbf{x}$, so sind auch alle Eigenwerte nicht negativ. Fassen wir für symmetrische Matrizen \mathbf{B} die Eigenwerte zur Diagonalmatrix Λ und die Eigenvektoren \mathbf{u}_i (als Spaltenvektoren) zur Orthonormalmatrix \mathbf{U} zusammen, so können wir schreiben:

$$\mathbf{BU} = \mathbf{U}\Lambda. \quad (21.21)$$

Damit ist:

$$\mathbf{B} = \mathbf{U}\Lambda\mathbf{U}^T. \quad (21.22)$$

Das „gewöhnliche“ Eigenwertproblem (21.20) kann noch erweitert werden. Dazu sei zusätzlich zur Matrix \mathbf{B} noch eine positiv definite Matrix \mathbf{C} gegeben. Das verallgemeinerte Eigenwertproblem lautet dann:

$$\mathbf{Bx} = \lambda \mathbf{Cx}. \quad (21.23)$$

Ist \mathbf{B} symmetrisch, dann sind wieder alle Eigenwerte reell und es existiert ein Orthonormalsystem von Eigenvektoren. Diese sind aber nun orthonormal bezüglich des verallgemeinerten Skalarproduktes:

$$\mathbf{u}_i^T \mathbf{C} \mathbf{u}_j = \delta_{i-j}. \quad (21.24)$$

Ist \mathbf{B} zusätzlich positiv semidefinit, so sind auch alle Eigenwerte wieder nicht negativ.

21.2 Ausgleichsrechnung

Die grundlegende Methode der Ausgleichsrechnung ist sicher die *Methode der kleinsten Quadrate*, die breits von C.F. Gauß und Legendre entwickelt wurde. Sie ist ein wichtiges Hilfsmittel in der Bildverarbeitung, um Parameterschätzungen durchzuführen. Warum heißt die Überschrift dieses Abschnittes nicht Methode der kleinsten Quadrate, sondern Ausgleichsrechnung? Die Methode der kleinsten Quadrate ist sehr wichtig, aber eben doch nicht die einzige Methode um eine Ausgleichsrechnung durchzuführen. Häufig wird diese Methode mit LSE-Methode (Least Squares Estimation) bezeichnet. Formuliert man diese Methode mit den Mitteln der Stochastik, dann nennt man sie MMSE-Methode (Minimum Mean Squares Estimation). Ein konkretes Ausgleichsproblem kann entweder deterministisch (LSE-Methode) oder stochastisch (MMSE-Methode) formuliert werden. Fundamental ist auf jeden Fall immer die sogenannte **Modellgleichung**. Das ist dasjenige Modell für das wir die unbekannten Parameter suchen.

21.2.1 Lineare Ausgleichsrechnung

In diesem Falle muss die Modellgleichung linear in den gesuchten Parametern sein. In der deterministischen Schreibweise lautet die allgemeine Modellgleichung:

$$\sum_{k=1}^n a_k f_i = f_0. \quad (21.25)$$

Die $a_k, k = 1, \dots, n$ sind die gesuchten Parameter und die Funktionen $f_i, i = 0, 1, \dots, n$ sind die sogenannten *Messfunktionen*. In der Bildverarbeitung sind die Messfunktionen häufig Funktionen der zwei- oder dreidimensionalen Pixel. Schreibt man z. B. die Modellgleichung einer Ellipse in der Form

$$a_1 x^2 + a_2 xy + a_3 y^2 + a_4 x + a_5 y = 1, \quad (21.26)$$

so kann man die Messfunktionen deutlich ablesen, wobei $f_0(x, y) \equiv 1$ ist. Es müssen nun Punkte $\mathbf{x}_i, i = 1, \dots, K$ gegeben sein, für die die Funktionswerte der Messfunktionen $f_i^{(k)} :=$

$f_i(\mathbf{x}_k)$ berechnet werden. Wir suchen nun diejenigen Parameter $a_i, i = 1, \dots, n$, für die

$$S(a_1, \dots, a_n) = \sum_{k=1}^K \left(\sum_{i=1}^n a_i f_i^{(k)} - f_0^{(k)} \right)^2 \rightarrow \text{Minimum} \quad (21.27)$$

gilt. Nun müssen wir die partiellen Ableitungen nach den a_k bilden, diese Null setzen und nach den a_k auflösen. Offensichtlich entsteht ein lineares Gleichungssystem, das wir lösen müssen. Da dies aber relativ unübersichtlich ist, gehen wir zur Matrix-Vektor Notation über und fassen zu Vektoren zusammen: $\mathbf{a}^T = (a_1, \dots, a_n)$, $\mathbf{f}^T = (f_1, \dots, f_n)$. Die Modellgleichung lautet nun:

$$\mathbf{a}^T \mathbf{f} = \mathbf{f}^T \mathbf{a} = f_0. \quad (21.28)$$

Bevor wir das Optimierungsproblem vektoriell aufschreiben, müssen wir erst die *Messgleichungen* in Matrixform aufschreiben. Dazu setzen wir einfach die Punkte \mathbf{x}_k in die Modellgleichung ein und schreiben diese so oft auf, wie wir Punkte haben:

$$\begin{array}{ccccccccc} f_1^{(1)} a_1 & + & \cdots & + & f_n^{(1)} a_n & = & f_0^{(1)} \\ \vdots & + & \vdots & + & \vdots & = & \vdots \\ f_1^{(K)} a_1 & + & \cdots & + & f_n^{(K)} a_n & = & f_0^{(K)} \end{array} \leftrightarrow \mathbf{M} \cdot \mathbf{a} = \mathbf{f}_0. \quad (21.29)$$

Zunächst betrachten wir den Defektvektor $\mathbf{d} = \mathbf{M} \cdot \mathbf{a} - \mathbf{f}_0$. Nun können wir das Optimierungsproblem in Matrix-Vektorschreibweise formulieren:

$$S(\mathbf{a}) = \mathbf{d}^T \mathbf{d} = \mathbf{a}^T \mathbf{M}^T \mathbf{M} \mathbf{a} - \mathbf{a}^T \mathbf{M}^T \mathbf{f}_0 - \mathbf{f}_0^T \mathbf{M} \mathbf{a} + \mathbf{f}_0^T \mathbf{f}_0 \rightarrow \text{Minimum}. \quad (21.30)$$

Nun bilden wir den Gradienten nach \mathbf{a} (siehe Abschnitt 21.1):

$$\nabla_{\mathbf{a}} S(\mathbf{a}) = 2\mathbf{M}^T \mathbf{M} \mathbf{a} - 2\mathbf{M}^T \mathbf{f}_0 = \mathbf{0} \quad (21.31)$$

und erhalten damit die *Gaußschen Normalengleichungen*:

$$\mathbf{M}^T \mathbf{M} \mathbf{a} = \mathbf{M}^T \mathbf{f}_0. \quad (21.32)$$

Einfache Merkregel

- Lineare Ausgleichsrechnung bedeutet: Es liegt eine lineare Modellgleichung vor.
- Wir schreiben die Messgleichungen $\mathbf{M} \mathbf{a} = \mathbf{f}_0$ auf. Diese werden häufig mit $\mathbf{A} \mathbf{x} = \mathbf{b}$ bezeichnet. Die Koeffizientenmatrix \mathbf{M} besitzt in der Regel mehr Zeilen als Spalten und damit hat dieses lineare Gleichungssystem in der Regel keine Lösung im Sinne der exakten Gleichheit. Deshalb suchen wir eine Lösung im Sinne der kleinsten Quadrate.
- Nachdem wir die Messgleichungen aufgestellt haben, multiplizieren wir einfach die Messgleichungen (linke und rechte Seite) mit der transponierten Messmatrix \mathbf{M} und erhalten die Gaußschen Normalengleichungen. Dieses Gleichungssystem besitzt nun immer und zwar mindestens eine Lösung im Sinne der echten Gleichheit.

- Wie wir in Abschn. 22.3.2 zeigen werden, kann die Lösung der Gaußschen Normalengleichungen numerisch instabil werden. Daher wird oft die SVD benutzt, siehe auch Abschn. 22.3.2. Eine andere, aber recht einfach zu verstehende Methode ist die Nutzung eines Standardalgorithmus der numerischen Mathematik, genannt QR-Zerlegung einer beliebigen Matrix \mathbf{A} . Dabei gehen wir in der Bezeichnung einmal davon aus, dass $\mathbf{Ax} = \mathbf{b}$ die Messgleichungen sind und im Sinne der kleinsten Quadrate zu lösen sind. Die QR-Zerlegung lautet $\mathbf{A} = \mathbf{QR}$. Dabei ist \mathbf{Q} eine quadratische (K, K), orthogonale Matrix. Die rechteckige (K, n) Matrix \mathbf{R} ist eine obere Dreiecksmatrix, d. h. genaugenommen ist der obere quadratische (n, n) Teil \mathbf{R}_1 eine obere Dreiecksmatrix und der untere Teil ist eine Nullmatrix. Weiter sei $\tilde{\mathbf{b}} = \mathbf{Q}^T \mathbf{b}$, wobei die ersten n -Komponenten mit $\tilde{\mathbf{b}}_1$ und die restlichen mit $\tilde{\mathbf{b}}_2$ bezeichnet werden. Dann gilt

$$\|\mathbf{Ax} - \mathbf{b}\|_2^2 = \|\mathbf{Q}^T(\mathbf{Rx} - \mathbf{b})\|_2^2 = \|\mathbf{Rx} - \tilde{\mathbf{b}}\|_2^2 = \|\mathbf{R}_1 \mathbf{x} - \tilde{\mathbf{b}}_1\|_2^2 + \|\tilde{\mathbf{b}}_2\|_2^2. \quad (21.33)$$

Das Funktional in (21.33) wird folglich genau dann minimal, wenn

$$\mathbf{R}_1 \mathbf{x} = \tilde{\mathbf{b}}_1 \quad (21.34)$$

ist. Folglich brauchen wir nur das lineare Gleichungssystem (21.34) lösen, was infolge der Dreiecksstruktur durch Rückwärtseinsetzen trivial zu erledigen ist. Wenn wir davon ausgehen, dass die QR-Zerlegung einer Matrix numerisch stabil erfolgt, dann können wir auch das lineare Ausgleichsproblem numerisch stabil lösen.

Wollen wir eine gewichtete Ausgleichsrechnung durchführen, d. h. jede Messgleichung kann ein anderes Gewicht erhalten, dann schreiben wir diese Gewichte in die quadratische Diagonalmatrix \mathbf{G} . Die Messgleichungen lauten dann:

$$\mathbf{G} \cdot \mathbf{Ma} = \mathbf{G} \cdot \mathbf{f}_0. \quad (21.35)$$

Wir multiplizieren nun mit der transponierten erweiterten Messmatrix $\mathbf{G} \cdot \mathbf{M}$ durch und erhalten die folgenden Gaußschen Normalengleichungen:

$$\mathbf{M}^T \mathbf{G}^2 \mathbf{Ma} = \mathbf{M}^T \mathbf{G}^2 \mathbf{f}_0. \quad (21.36)$$

Die Gaußsche Normalenmatrix $\mathbf{N} = \mathbf{M}^T \mathbf{M}$ hat generell sehr schöne Eigenschaften:

- Sie ist symmetrisch.
- Sie ist mindestens positiv semidefinit, d. h. für beliebige \mathbf{a} gilt:

$$\mathbf{a}^T \mathbf{Na} = \mathbf{a}^T \mathbf{M}^T \mathbf{Ma} = (\mathbf{Ma})^T \cdot \mathbf{Ma} \geq 0. \quad (21.37)$$

- Aus der positiven Semidefinitheit folgt sofort, dass alle Eigenwerte von \mathbf{N} nichtnegativ sind, denn:

$$0 \leq \mathbf{a}^T \mathbf{Na} = \lambda \mathbf{a}^T \mathbf{a} \rightarrow \lambda \geq 0. \quad (21.38)$$

Eine ähnliche Interpretation erhalten wir, wenn wir das Ausgleichsproblem mit den Mitteln der Stochastik beschreiben. Dazu fassen wir einfach die Funktionswerte der Messfunktionen $f_i^{(k)}$ als Realisierungen von Zufallsgrößen F_i auf. Dann lautet die Optimierungsaufgabe (E bedeutet der Erwartungswert):

$$S(\mathbf{a}) = E((\mathbf{F}^T \mathbf{a} - F_0)^2) \rightarrow \text{Minimum.} \quad (21.39)$$

Die Hyperebene $F_0 = a_1 F_1 + a_2 F_2 + \dots + a_n F_n$ mit obiger Forderung nennt man in der Stochastik *Regressionshyperebene zweiter Art* der Zufallsvarianlen F_0 bezüglich der Zufallsvariablen F_1, \dots, F_n . Wir bilden wieder den Gradienten

$$\nabla_{\mathbf{a}} S = 2E(\mathbf{F}(\mathbf{F}^T \mathbf{a} - F_0)) = \mathbf{0}. \quad (21.40)$$

Mit der Abkürzung $\Sigma_{\mathbf{F}} = E(\mathbf{F}\mathbf{F}^T)$ erhalten wir als Analogon zu den Gaußschen Normalengleichungen

$$\Sigma_{\mathbf{F}} \cdot \mathbf{a} = E(F_0 \cdot \mathbf{F}). \quad (21.41)$$

Die Gaußsche Normalenmatrix \mathbf{N} ist folglich als Realisierung der Kovarianzmatrix $\Sigma_{\mathbf{F}}$ aufzufassen.

Fitting von Geraden Nun wollen wir an eine gegebene Menge von Punkten $\mathbf{x}_k, k = 1, \dots, K$ im Bild eine Gerade anpassen. Dazu müssen wir zuerst die Modellgleichung einer Geraden aufschreiben:

$$ax + by + c = 0. \quad (21.42)$$

Dabei ist nun $\mathbf{a}^T = (a, b, c)$, $f_1(x, y) = x$, $f_2(x, y) = y$, $f_3(x, y) = 1$, $f_0(x, y) = 0$. Die Modellgleichung ist tatsächlich linear in \mathbf{a} , die Funktion $f_0(x, y) = 0$ sorgt allerdings für Probleme. Die Gaußschen Normalengleichungen sind dann ein homogenes lineares Gleichungssystem mit unendlich vielen Lösungen, bzw. das Optimierungsproblem wird mit der trivialen Lösung $\mathbf{a} = 0$ minimiert. Wir benötigen also in der Modellgleichung ein $f_0(x, y)$ ungleich Null. Welche einfachen Modellgleichungen, die auch weiterhin linear sein sollen, kommen in Frage:

- Modell G1: $ax + b = y \rightarrow f_0(x, y) = y$
 - Diese Form entspricht der in der Stochastik verwendeten Regression von y bezüglich x .
 - Wenn die Punkte auf einer vertikalen Geraden liegen, dann versagt diese Modellgleichung.
- Modell G2: $ay + b = x \rightarrow f_0(x, y) = x$
 - Diese Form entspricht der in der Stochastik verwendeten Regression von x bezüglich y .
 - Wenn die Punkte auf einer horizontalen Geraden liegen, dann versagt diese Modellgleichung.

- Modell G3: $ax + by = 1 \rightarrow f_0(x, y) = 1$
 - Wenn die Punkte auf einer Geraden liegen, die durch den Koordinatenursprung geht, dann versagt diese Modellgleichung.
- Modell G4: $ax + by + c = 0$ und lineare Kombinationen zwischen den a, b, c als Bedingung, z. B. $a + b = 1 \rightarrow (1 - b)x + by + c = 0 \rightarrow f_0(x, y) = -x$
 - Wenn die Punkte z. B. auf der Geraden $y = x$ liegen, dann versagt diese Modellgleichung.
- Man kann sich beliebige lineare Kombinationen der Parameter mit Beschränkungen ausdenken, es gibt immer wieder Fälle für die die Modellfunktion falsch ist.

Wir sehen den wichtigen unterschied zur linearen Regression in der Stochastik: dort geht man immer von funktionalen Zusammenhängen aus. In einem Bild wollen wir dagegen generell eine Gerade anpassen, die muss nicht unbedingt eine Funktion darstellen. Entweder wir haben A-priori-Wissen, dass solche Fälle nicht vorkommen oder wir müssen nichtlineare Restriktionen an die Modellparameter stellen, dann haben wir aber keine lineare Ausgleichsrechnung mehr, siehe Abschn. 21.2.2. Es gibt aber noch eine „unschöne“ Lösung: wir fitten zwei Geraden, z. B. nach Modell G1 und Modell G2 und nehmen die Lösung, für die der Fittingfehler $S(\mathbf{a})$ kleiner ist.

Generell minimieren wir die Summe der quadratischen Abweichungen von der Modellgleichung, genannt Defekte: $d_k = \sum_i a_i \cdot f_i^{(k)} - f_0^{(k)}$. Sind die Defekte d_k irgendwie geometrisch deutbar? Im allgemeinen nicht und deshalb nennt man sie auch **algebraische Distanz** des Punktes \mathbf{x}_k zur Kurve. Auch bei den obigen Modellgleichungen für eine Gerade haben sie bei jedem Modell G1, G2, G3 oder G4 eine andere geometrische Bedeutung. Die Größe dieser Defekte hängt sogar von der Rotationslage der Geraden ab, was natürlich schlecht ist. Deshalb wäre es günstig, wenn dieses Maß den Abstand eines Punktes zur Kurve beschreiben würde, da Abstände rotationsinvariant sind. Dies lässt sich aber nur durch nichtlineare Restriktionen an die Geradenparameter erfüllen. Dies Anpassung heißt dann in der Stochastik *Orthogonale Regression*.

Fitting von Kreisen Die bekannte Kreisgleichung $(x - x_m)^2 + (y - y_m)^2 = r^2$ überführen wir in eine lineare Modellgleichung:

$$a(x^2 + y^2) + bx + cy = 1. \quad (21.43)$$

Dabei haben wir $d = -1$ gesetzt. Daher ist nun $f_1 = x^2 + y^2$, $f_2 = x$, $f_3 = y$ und $f_0 = 1$. Da wir wieder eine linare Normierung benutzt haben, darf der Kreis nicht durch den Koordinatenursprung gehen.

Fitting von Ellipsen Benutzen wir die gleiche lineare Normierung wie bei den Kreisen, dann haben wir die Modellgleichung

$$ax^2 + bxy + cy^2 + dx + ey = 1. \quad (21.44)$$

Nun ist wieder $f_1 = x^2$, $f_2 = xy$, $f_3 = y^2$, $f_4 = x$, $f_5 = y$ und $f_0 = 1$. Man beachte: dies ist die Modellgleichung einer Kurve zweiter Ordnung. Wir können so nicht garantieren, dass eine Ellipse angepasst wird, es könnte auch eine Hyperbel oder Parabel berechnet werden. Wenn die Punkte jedoch typisch eine Ellipse beschreiben, dann werden wir auch vermutlich eine Ellipse erhalten. Die geometrischen Parameter der Ellipse lassen sich aus den algebraischen dann wie folgt berechnen:

- Mittelpunkt:

$$x_m = \frac{\frac{be}{4} - \frac{cd}{2}}{ac - \frac{b^2}{4}}, \quad x_m = \frac{\frac{bd}{4} - \frac{ae}{2}}{ac - \frac{b^2}{4}}. \quad (21.45)$$

- Längen der beiden Halbachsen A und B :

$$h = 1 - ax_m^2 - bx_my_m - cy_m^2 - dx_m - ey_m, \quad (21.46)$$

$$\lambda_1 = \frac{1}{2} \left(a + c + \sqrt{(a - c)^2 + b^2} \right) \quad (21.47)$$

$$\lambda_2 = \frac{1}{2} \left(a + c - \sqrt{(a - c)^2 + b^2} \right) \quad (21.48)$$

$$A = \sqrt{\frac{h}{\lambda_1}}, \quad B = \sqrt{\frac{h}{\lambda_2}}, \quad (21.49)$$

- Winkel der großen Achse gegen die x -Achse:

$$\alpha = \arctan \left(\frac{c - a + \sqrt{(a - c)^2 + b^2}}{b} \right). \quad (21.50)$$

21.2.2 Nichtlineare Ausgleichsrechnung

Modellgleichung ist linear, aber die Restriktion ist nichtlinear Wir wollen die nichtlineare Ausgleichsrechnung in die beiden Fälle

- a) Die Modellgleichung ist linear, aber die Restriktion ist eine quadratische Form.
- b) Die Modellgleichung und die Restriktion sind nichtlinear.

einteilen. Bevor wir auf den allgemeinen Fall b) eingehen, behandeln wir den Fall a). Die Nichtlinearität der Restriktion wird insbesondere dann interessant, wenn sie eine quadratische Form darstellt. In diesem Falle beherrschen wir noch die Nichtlinearität, die Lösung führt dann auf ein Eigenwertproblem.

Wir nehmen nun an, es liege eine lineare Modellgleichung der Form (21.28) vor und es sei $f_0 = 0$. Da wir die triviale Lösung $\mathbf{a} = \mathbf{0}$ ausschließen müssen, bietet sich eine nichtlineare Restriktion der Form

$$\sum_{i=1}^n a_i^2 = \mathbf{a}^T \mathbf{a} = 1 \quad (21.51)$$

an. In der Literatur wird dies als *Principal Axis Curve Fit* bezeichnet. Wir müssen folglich $S(\mathbf{a})$ mit dieser Restriktion minimieren:

$$S(\mathbf{a}) = E[(\mathbf{a}^T \mathbf{F})^2] = \mathbf{a}^T \Sigma_F \mathbf{a} \rightarrow \text{Minimum bei } \mathbf{a}^T \mathbf{a} = 1. \quad (21.52)$$

Dazu bietet sich die Methode der Lagrangeschen Multiplikatoren an. Unter Benutzung der stochastischen Formulierung des Ausgleichsproblems lautet das Ausgleichsproblem mit der Lagrange-Funktion:

$$L(\mathbf{a}, \lambda) = E[(\mathbf{a}^T \mathbf{F})^2] - \lambda(\mathbf{a}^T \mathbf{a} - 1) \rightarrow \text{Minimum}. \quad (21.53)$$

Wir bilden von L den Gradienten nach \mathbf{a} und setzen gleich Null, dann erhalten wir das Gleichungssystem:

$$\Sigma_F \mathbf{a} = \lambda \mathbf{a} \quad \text{mit } \mathbf{a}^T \mathbf{a} = 1. \quad (21.54)$$

Die Kovarianzmatrix Σ_F ist wieder die Gaußsche Normalenmatrix \mathbf{N} . Wir sehen, dieses Gleichungssystem ist ein typisches Eigenwertproblem. Wir bestimmen folglich von \mathbf{N} alle Eigenwerte und alle zugehörigen Eigenvektoren. Da \mathbf{N} positiv semidefinit und symmetrisch ist, sind alle Eigenwerte nichtnegativ und die Eigenvektoren bilden ein Orthonormalsystem. Welchen Eigenvektor nehmen wir nun als Lösung für \mathbf{a} ? Es gilt offensichtlich mit $f_0 = 0$:

$$S(\mathbf{a}) = E[(\mathbf{a}^T \mathbf{F})^2] = \mathbf{a}^T \Sigma_F \mathbf{a} = \lambda \mathbf{a}^T \mathbf{a} = \lambda. \quad (21.55)$$

$S(\mathbf{a})$ wird dann minimiert, wenn wir den Eigenvektor zum kleinsten Eigenwert nehmen.

Mit der nichtlinearen Restriktion (21.51) können wir nun ohne Schwierigkeiten Geraden, Ebenen, Kreise, Ellipsen usw. fitten. Zu beachten ist nur, dass bei der Modellgleichung $f_0 = 0$ gelten muss. Also lautet z. B. die algebraische Modellgleichung einer Ellipse

$$a_1 x^2 + a_2 x y + a_3 y^2 + a_4 x + a_5 y + a_6 = 0. \quad (21.56)$$

Wir können aber immer noch nicht garantieren, dass tatsächlich eine Ellipse gefittet wird, es könnte auch eine Hyperbel werden. An dieser Ellipsen-Modellgleichung mit $f_0 = 0$ sieht man, dass es einen Term (hier Koeffizient vor a_6) gibt mit $f_i = 1$. Wir spalten diesen gesonderten Term einmal ab und nennen die Unbekannte a_{n+1} . Bei der Ellipsengleichung ist

dann $n = 5$. In den Vektor \mathbf{a} gehen bei der Ellipsengleichung somit nur die Unbekannten a_1, a_2, a_3, a_4, a_5 ein, a_6 zählen wir gesondert auf. Da wir a_6 auf der rechten Seite haben möchten, bezeichnen wir a_6 mit $-a_6$. Dann wird unser Ausgleichsproblem im allgemeinen Fall zu:

$$S(\mathbf{a}, a_{n+1}) = E \left[(\mathbf{a}^T \mathbf{F} - a_{n+1})^2 \right] \rightarrow \text{Minimum bei } \mathbf{a}^T \mathbf{a} = 1. \quad (21.57)$$

Die Lagrange-Funktion

$$L(\mathbf{a}, a_{n+1}, \lambda) = S(\mathbf{a}, a_{n+1}) - \lambda(\mathbf{a}^T \mathbf{a} - 1) \quad (21.58)$$

leiten wir einmal nur nach a_{n+1} ab und setzen Null:

$$\frac{\partial L}{\partial a_{n+1}} = -2\mathbf{a}^T E[\mathbf{F}] + 2a_{n+1} = 0. \quad (21.59)$$

Wir erhalten folglich:

$$a_{n+1} = \mathbf{a}^T E[\mathbf{F}]. \quad (21.60)$$

Wir sehen, wenn das optimale \mathbf{a} bereits berechnet wurde, können wir daraus simpel a_{n+1} ausrechnen. Weiterhin: wenn $E[\mathbf{F}] = \mathbf{0}$ wäre, dann ist $a_{n+1} = 0$ und wir hätten das gleiche Problem wie in (21.52) zu lösen. Dies können wir aber mit einem „Trick“ erreichen: wir transformieren \mathbf{F} so, dass für die transformierten Werte der Erwartungswert Null ist. Diese Normierung auf den Erwartungswert (Schwerpunkt) ist in der Stochastik üblich und erfolgt durch:

$$\mathbf{Z} = \mathbf{F} - E[\mathbf{F}] \rightarrow E[\mathbf{Z}] = \mathbf{0}. \quad (21.61)$$

Damit haben wir das gleiche Optimierungsproblem wie in (21.52), allerdings müssen wir anstatt \mathbf{F} die normierte Größe \mathbf{Z} verwenden:

$$S(\mathbf{a}) = E \left[(\mathbf{a}^T \mathbf{Z})^2 \right] = \mathbf{a}^T \Sigma_Z \mathbf{a} \rightarrow \text{Minimum bei } \mathbf{a}^T \mathbf{a} = 1. \quad (21.62)$$

Praktisches Vorgehen

- Wenn wir Punkte im Bild gegeben haben, dann muss von jedem Punkt der Vektor \mathbf{f} berechnet werden.
- Von allen Vektoren \mathbf{f} berechnen wir den Schwerpunkt und transformieren alle Vektoren so, dass sie auf den Schwerpunkt normiert sind. Die neuen Vektoren heißen nun \mathbf{z} .
- Von diesen Vektoren \mathbf{z} berechnen wir die Gaußsche Normalenmatrix und berechnen von dieser den Eigenvektor zum kleinsten Eigenwert. Damit ist \mathbf{a} berechnet. Nun berechnen wir noch a_{n+1} mittels (21.60).

Fitting von Geraden und Ebenen Mit diesem Vorgehen haben wir die Dimension des Lösungsvektors um Eins erniedrigt, aber zum Preis, dass die Menge von Vektoren \mathbf{f} auf deren Schwerpunkt transformiert werden müssen. Was ist nun der eigentliche Vorteil? Dazu betrachten wir das Fitten von Geraden, Ebenen, Hyperebenen usw.. Wir wählen z. B. das Fitten einer Ebene an eine gegebene 3D-Punktwolke. Die allgemeine Gleichung einer Ebene lautet dann

$$a_1x + a_2y + a_3z - a_4 = 0. \quad (21.63)$$

Damit ist die Dimension von \mathbf{a} beziehungsweise \mathbf{f} gleich drei und es ist $f_1 = x$, $f_2 = y$, $f_3 = z$. Folglich ist nicht nur die Dimension gleich drei, sondern die Vektoren sind im Falle von Geraden, Ebenen und Hyperebenen identisch mit den gegebenen Punkten, was sonst (z. B. bei Ellipsen, Kreisen usw.) nicht der Fall ist. Damit ist der Schwerpunkt geometrisch deutbar: Es ist jetzt tatsächlich der Schwerpunkt der Punktmenge. In nichtstochastischer Schreibweise müssen wir jetzt lösen:

$$S(\mathbf{a}) = \sum_{k=1}^K (a_1x_k + a_2y_k + a_3z_k - a_4)^2 \rightarrow \text{Minimum bei } a_1^2 + a_2^2 + a_3^2 = 1. \quad (21.64)$$

Jetzt sehen wir den entscheidenden Vorteil: die Normierung $a_1^2 + a_2^2 + a_3^2 = 1$ bedeutet, dass die Ebenengleichung in der Hesseschen Normalform formuliert wurde. Wenn wir a_4 mit in die Normierung aufgenommen hätten, dann ist dies nicht mehr die Hessesche Normalform. Setzt man irgendeinen Punkt in die Hessesche Normalform ein, dann ist $d_k = a_1x_k + a_2y_k + a_3z_k - a_4$ der Abstand des Punktes zur Ebene (Gerade). Damit minimieren wir in (21.64) die Summe der quadratischen Abstände der Punkte zur Ebene. Da die Abstände rotationsinvariant sind, haben wir keinerlei Probleme mehr, ob das Fitten in speziellen Lagen versagt oder nicht. Damit ist die Aufgabe (21.64) die **Standardmethode** zum Fitten von Geraden und Ebenen.

Nichtlineare Modellgleichungen Wir betrachten wieder die Modellgleichung (kann auch eine Vektorgleichung sein)

$$F(\mathbf{a}, \mathbf{x}) = 0. \quad (21.65)$$

Diese darf nun in den Unbekannten des Parametervektors \mathbf{a} nichtlinear sein. Es seien nun wieder K Messtupel $\mathbf{x}^i, i = 1, \dots, K$ gegeben und wir schreiben die Messgleichungen auf:

$$\begin{aligned} F(\mathbf{a}, \mathbf{x}^1) &= d_1(\mathbf{a}) \\ F(\mathbf{a}, \mathbf{x}^2) &= d_2(\mathbf{a}) \\ &\vdots \\ F(\mathbf{a}, \mathbf{x}^K) &= d_K(\mathbf{a}). \end{aligned} \quad (21.66)$$

Mit der Abkürzung $d(\mathbf{a})^T = (d_1(\mathbf{a}), \dots, d_K(\mathbf{a}))$ haben wir dann wie üblich zu lösen:

$$\|d(\mathbf{a})\|^2 = d^T d = \text{Minimum.} \quad (21.67)$$

Dazu betrachten wir irgendeine, allerdings geeignete Startlösung \mathbf{a}_0 und linearisieren in diesem Punkt die Defektfunktion $d(\mathbf{a})$ zu

$$d^*(\mathbf{a}) = d(\mathbf{a}_0) + \mathbf{J} \cdot (\mathbf{a} - \mathbf{a}_0). \quad (21.68)$$

Dabei ist wie üblich bei der Taylorentwicklung von Vektorfunktionen \mathbf{J} die Matrix der ersten partiellen Ableitungen, folglich die Jakobimatrix. Da nun $d^*(\mathbf{a})$ linear in \mathbf{a} ist, lösen wir dieses inhomogene, lineare Ausgleichsproblem mit den Gaußschen Normalengleichungen:

$$\mathbf{J}^T \mathbf{J}(\mathbf{a} - \mathbf{a}_0) = -\mathbf{J}^T d(\mathbf{a}_0). \quad (21.69)$$

Wir rechnen nun \mathbf{a} aus und nennen die Lösung \mathbf{a}_1 . Damit ist:

$$\mathbf{a}_1 - \mathbf{a}_0 = \Delta \mathbf{a}_1, \mathbf{a}_1 = \mathbf{a}_0 + \Delta \mathbf{a}_1. \quad (21.70)$$

Daraus leiten wir nun formal eine Iteration ab und hoffen, dass diese gegen die Lösung konvergiert. Also ist

$$\Delta \mathbf{a}_i = -(\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T d(\mathbf{a}_i). \quad (21.71)$$

Führen wir noch eine Schrittweitensteuerung ein, dann erhalten wir die Iteration:

$$\mathbf{a}_{i+1} = \mathbf{a}_i - \alpha_i (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T d(\mathbf{a}_i), \quad i = 0, 1, \dots \quad (21.72)$$

Dies ist die sogenannte *Gauß-Newton-Methode*. Wir bilden nun eine leichte Modifikation der Gauß-Newton-Methode:

- $\alpha_i = 1 \forall i$.
- Iteriere mit

$$\mathbf{a}_{i+1} = \mathbf{a}_i - (\lambda \mathbf{I} + \mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T d(\mathbf{a}_i), \quad i = 0, 1, \dots \quad (21.73)$$

(\mathbf{I} ist die Einheitsmatrix)

- Wähle zu Beginn λ klein, so ca. $\lambda = 10^{-3}$. Fällt nach einer Iteration der Fehler, d. h. $\|d(\mathbf{a}_{i+1})\| < \|d(\mathbf{a}_i)\|$, setze $\lambda_{i+1} = \lambda_i / 10$. Steigt der Fehler, d. h. $\|d(\mathbf{a}_{i+1})\| > \|d(\mathbf{a}_i)\|$, so verwerfen wir \mathbf{a}_{i+1} , setzen $\lambda_i := \lambda_i * 10$ und zwar solange, bis der Fehler wieder fällt.

Diese modifizierte Gauß-Newton-Methode nennt man *Levenberg-Marquardt-Methode*. Die Modifikation beinhaltet eine sogenannte *Tichonov-Arsenin-Regularisierung*, siehe Abschn. 22.2 bzw. (22.39).

21.2.3 Kovariantes Fitting

Fitting ist ein sogenannter Schätzer, der verschiedene Eigenschaften haben kann. So sollte ein Schätzer erwartungstreu (*unbiased*) und konsistent sein. Wir wollen hier eine weitere wichtige Eigenschaft untersuchen, die Invarianz oder besser die Kovarianz des Schätzers. Wir stellen uns eine Punktmenge in einem Bild vor und fitten an diese nach irgendeiner Fittingmethode eine Gerade. Nun nehmen wir genau die gleiche Punktmenge, verschieben diese Punktmenge und fitten mit der gleichen Fittingmethode wieder eine Gerade. Dann sollte doch die zweite Gerade eigentlich eine Verschiebung der ersten Geraden sein. Dies ist aber oft nicht der Fall. Diese Eigenschaft sollte außerdem nicht nur für Verschiebungen gelten, sondern für möglichst viele geometrische Transformationen. Auf diese Eigenschaft wird im Abschn. 19.4 ausführlich eingegangen. Die Standardmethode zum Fitten von Geraden und Ebenen (orthogonale Regression, (21.64)) beruht auf der Minimierung der quadratischen Abstände der Punkte von den Geraden bzw. Ebenen. Da Abstände translations- und rotationsinvariant sind, ist die Standardmethode kovariant gegenüber Euklidischen Transformationen, ja sogar kovariant gegenüber Ähnlichkeitstransformationen. Nun wollen wir das Fitten von Kurven zweiter Ordnung untersuchen. Dazu benutzen wir die Darstellung einer Kurve zweiter Ordnung nach (21.56). Diese Form wollen wir in Matrix-Vektor-Notation aufschreiben. Dazu führen wir die Matrix \mathbf{E} und den Vektor \mathbf{x} ein:

$$\mathbf{E} = \begin{pmatrix} a_1 & \frac{a_2}{2} & \frac{a_4}{2} \\ \frac{a_2}{2} & a_3 & \frac{a_5}{2} \\ \frac{a_4}{2} & \frac{a_5}{2} & a_6 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \quad (21.74)$$

Dann lässt sich (21.56) durch

$$\mathbf{x}^T \mathbf{E} \mathbf{x} = 0 \quad (21.75)$$

angeben. Für einen beliebigen Punkt \mathbf{x}_k ergibt sich dann als Defekt oder algebraische Distanz $d_k = \mathbf{x}_k^T \mathbf{E} \mathbf{x}_k$. Nun betrachten wir als geometrische Transformation eine affine Transformation und schreiben diese auch in dieser Notation

$$\mathbf{x}' = \mathbf{A} \mathbf{x}, \quad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{10} \\ a_{21} & a_{22} & a_{20} \\ 0 & 0 & 1 \end{pmatrix}. \quad (21.76)$$

Es beschreibe nun $\mathbf{x}^T \mathbf{E} \mathbf{x} = 0$ eine Kurve zweiter Ordnung vor und $\mathbf{x}'^T \mathbf{E}' \mathbf{x}' = 0$ die Kurve zweiter Ordnung nach einer affinen Transformation. Wir setzen nun die affine Transformation ein und erhalten:

$$\mathbf{x}^T \mathbf{A}^T \mathbf{E}' \mathbf{A} \mathbf{x} = 0 \rightarrow \alpha \mathbf{E} = \mathbf{A}^T \mathbf{E}' \mathbf{A}, \quad \alpha \in \mathbb{R}. \quad (21.77)$$

Damit wissen wir nun, wie sich die Ellipsenparameter transformieren, diese sind bis auf einen Faktor α eindeutig bestimmt. Damit ergibt sich aber für einen beliebigen Punkt \mathbf{x}_k ,

eagl ob er auf der Kurve liegt oder nicht, die Beziehung:

$$d'_k = \mathbf{x}'_k^T \mathbf{E}' \mathbf{x}'_k = \alpha \mathbf{x}_k^T \mathbf{E} \mathbf{x}_k = \alpha d_k \quad \forall k. \quad (21.78)$$

Wir sehen, die Defekte oder algebraischen Distanzen sind alle affin invariant bis auf einen gemeinsamen Faktor α . Nun bilden wir die Determinanten:

$$\det(\mathbf{A}^T \mathbf{E}' \mathbf{A}) = \det(\mathbf{A})^2 \det(\mathbf{E}') = \alpha^3 \det(\mathbf{E}) \rightarrow \alpha^3 = \frac{\det(\mathbf{A})^2 \det(\mathbf{E}')}{\det(\mathbf{E})}. \quad (21.79)$$

Daraus folgt:

$$\alpha^3 = \det(\mathbf{A})^2, \quad \text{falls } \det(\mathbf{E}') = \det(\mathbf{E}) = 1. \quad (21.80)$$

Wenn wir folglich die Normierung $\det(\mathbf{E}') = \det(\mathbf{E}) = 1$ einhalten, dann hängt α nur noch von der affinen Transformation \mathbf{A} ab. Dann gilt

$$S'(\mathbf{a}') = \sum_{k=1}^K d'^2_k = \alpha^2 \sum_{k=1}^K d_k^2 = \alpha^2 S(\mathbf{a}) \rightarrow \text{Minimum}. \quad (21.81)$$

Wenn \mathbf{a} die Zielfunktion $S(\mathbf{a})$ minimiert, dann minimiert folglich auch \mathbf{a}' die Zielfunktion $S'(\mathbf{a}')$, weil der positive Faktor α^2 das Minimum nicht beeinträchtigt (da dieser Faktor durch die Normierungsbedingungen nicht mehr von den gesuchten Ellipsenparametern abhängt). Wenn folglich die beiden Ausgleichsprobleme (vor der affinen Transformation und nach der affinen Transformation) eindeutig lösbar sind, dann sind die Lösungen die affinen Abbilder voneinander. Lösen wir folglich das Ausgleichsproblem

$$S(\mathbf{a}) = \sum_{k=1}^K (\mathbf{x}_k^T \mathbf{E} \mathbf{x}_k)^2 \rightarrow \text{Minimum bei } \det(\mathbf{E}) = 1, \quad (21.82)$$

dann ist dieses Ellipsen-Fitting affin kovariant. Wir haben nach wie vor eine lineare Modellgleichung, aber die Restriktion $\det(\mathbf{E}) = 1$ ist nichtlinear und kompliziert. Daher zerlegen wir jetzt die Beziehung (21.77). Zu diesem Zwecke unterteilen wir jetzt die Matrizen \mathbf{E} bzw. \mathbf{E}' und \mathbf{A} :

$$\mathbf{E} = \begin{pmatrix} \mathbf{D} & \mathbf{e} \\ \mathbf{e}^T & f \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{B} & \mathbf{a}_0 \\ \mathbf{0} & 1 \end{pmatrix}, \quad (21.83)$$

dann zerfällt (21.77) in die drei Beziehungen:

$$\mathbf{B}^T \mathbf{D}' \mathbf{B} = \alpha \mathbf{D}, \quad (21.84)$$

$$\mathbf{B}^T \mathbf{D}' \mathbf{a}_0 + \mathbf{B}^T \mathbf{e}' = \alpha \mathbf{e}, \quad (21.85)$$

$$\mathbf{a}_0^T \mathbf{D}' \mathbf{a}_0 + 2 \mathbf{a}_0^T \mathbf{e}' + f' = \alpha f. \quad (21.86)$$

Jetzt können wir α von einer dieser drei Gleichungen eliminieren. Die erste Beziehung (21.84) ist der Bedingung (21.77) sehr ähnlich, deshalb wenden wir jetzt die gleiche Determinantenregel an:

$$\det(\mathbf{B}^T \mathbf{D}' \mathbf{B}) = \det(\mathbf{B})^2 \det(\mathbf{D}') = \alpha^2 \det(\mathbf{D}). \quad (21.87)$$

Folglich ist

$$\det(\mathbf{D}) = ac - \frac{b^2}{4} = 1 \quad (21.88)$$

eine weitere Bedingung für ein affin kovariantes Fitting von Kurven zweiter Ordnung. Überraschend ist zusätzlich, dass wir damit das Fitten einer Ellipse erzwingen, Hyperbeln und Parabeln werden damit ausgeschlossen. Setzen wir nun eine spezielle affine Transformation an, nämlich eine Ähnlichkeitstransformation mit $\mathbf{B} = s \cdot \mathbf{R}$, wobei \mathbf{R} eine Rotationsmatrix und s ein positiver Skalierungsfaktor ist, dann können wir α aus der gleichen Beziehung (21.77) auch anders eliminieren: wir wenden auf beide Seiten die Frobenius-Norm an:

$$\|s^2 \mathbf{R}^T \mathbf{D}' \mathbf{R}\|_F^2 = s^2 \|\mathbf{D}'\|_F^2 = \alpha^2 \|\mathbf{D}\|_F^2, \quad \|\mathbf{D}\|_F^2 = a^2 + \frac{b^2}{2} + c^2. \quad (21.89)$$

Damit ist

$$a^2 + \frac{b^2}{2} + c^2 = 1 \quad (21.90)$$

eine Fittingbedingung, die Kovarianz von Kurven zweiter Ordnung bezüglich Ähnlichkeitstransformationen erzwingt. Dies ist in der Literatur als *Bookstein-Fit* bekannt. Wir können für Ähnlichkeitstransformationen sogar auf (21.77) noch eine weitere Rechenregel anwenden:

$$s^2 \cdot \text{Spur}(\mathbf{D}') = \text{Spur}(\alpha \mathbf{D}) = \alpha \cdot \text{Spur}(\mathbf{D}), \quad \text{Spur}(\mathbf{D}) = a + c. \quad (21.91)$$

Damit ist

$$a + c = 1 \quad (21.92)$$

noch eine Bedingung für kovariantes Ellipsenfitting bezüglich Ähnlichkeitstransformationen. Diese hat den Vorteil, dass sie tatsächlich linear ist. Damit haben wir zusammen mit der linearen Modellgleichung ein lineares Ausgleichsproblem, was sehr einfach zu lösen ist. Der Nachteil ist wieder: Wir erzwingen damit nicht ein Ellipsenfitting. Sehen wir uns einmal die Beziehung (21.84) an, so folgt, dass die Bedingung $f = 1$ nicht einmal die Kovarianz bezüglich Translationen garantiert. Wir können aber wieder einen Normierungstrick anwenden:

Wir legen den Koordinatenursprung der Punkte in ihren Schwerpunkt, damit haben wir die Translation durch $\mathbf{a}_0 = 0$ eliminiert. Dann folgt aber aus (21.84), dass die Bedingung

$$f = 1, \quad \text{Punkte vorher auf den Schwerpunkt transformieren} \quad (21.93)$$

sogar die affine Kovarianz bedingt. Diese Bedingung ist wieder linear, allerdings kann wiederum kein Ellipsenfitting erzwungen werden und die Ellipse darf nicht durch den Schwerpunkt der Punktmenge gehen. In der Regel ist diese Bedingung eigentlich immer erfüllt, dass die Ellipse nicht durch den Schwerpunkt der Punktmenge geht. Da die Bedingung (21.93) formal auch auf Geradenfitting angewendet werden kann, erhalten wir einen Widerspruch: liegen z. B. alle Punkte auf einer Geraden, dann geht die Gerade immer durch den Schwerpunkt, was aber ein Widerspruch zu $f = 1$ ist. Betrachten wir noch einmal das Geradenfitting, dann ist $\mathbf{D} = \mathbf{D}' = 0$, damit fällt die Beziehung (21.77) weg und die anderen beiden reduzieren sich. Für Ähnlichkeitstransformationen können wir aus (21.78) α mit der Bedingung $\mathbf{e} = \mathbf{e}' = 1$ eliminieren. Dies ist aber genau die orthogonale Regression, die kovariant bezüglich Ähnlichkeitstransformationen ist.

Zusammenfassung

- Für Kurven zweiter Ordnung gibt es Bedingungen, die sogar die affine Kovarianz implizieren.
- Für Geraden und Ebenen gibt es nur Bedingungen, die die Kovarianz bezüglich Ähnlichkeitstransformationen garantieren.
- Für die Beweise sind zwei Voraussetzungen entscheidend:
 - Die Punkte \mathbf{x}'_k müssen auch tatsächlich die Abbilder von \mathbf{x}_k bezüglich der geometrischen Transformation sein. Dies wird in der Praxis nur näherungsweise der Fall sein.
 - Die Anzahl K der Punkte muss vor und nach der geometrischen Transformation die gleiche sein. Dies wird wohl in der Praxis nicht sehr oft der Fall sein.

21.2.4 Ausreißer-Probleme (Outlier)

Wenn man Punkte im Bild detektiert, dann detektiert man häufig auch völlig falsche Punkte, *Ausreißer* genannt. Wir wollen nun untersuchen, wie die Fittingergebnisse auf Ausreißer reagieren und wie man die Ergebnisse verbessern kann. In der Regel gehen wir von einer linearen Modellgleichung aus und minimieren die Quadrate der algebraischen Distanzen, d. h. für das Zielfunktional $S(\mathbf{a})$ benutzen wir die L_2 -Norm (zum Quadrat). Warum benutzen wir eigentlich die L_2 -Norm? Dazu nehmen wir die Modellgleichung einmal linear an. Durch die Ableitung einer quadratischen Zielfunktion ergibt sich dann ein lineares Gleichungssystem, weil die Modellgleichungen linear sind. Wenn wir eine Norm L_p mit $p > 2$ benutzen würden, dann ergeben die Ableitungen ein nichtlineares Gleichungssystem, obwohl die Modellgleichungen linear sind, daher ergibt die L_2 -Norm die einfachsten

numerischen Gleichungen. Die L_2 -Norm bietet somit eine Reihe entscheidender Vorteile, ist aber gegenüber Ausreißern sehr sensitiv. Wie können wir nun trotz L_2 -Norm eine gegenüber Ausreißern robuste Lösung finden. Dazu betrachten wir wieder die Modellgleichungen in der Form $F(\mathbf{a}, \mathbf{x}) = 0$, lösen dieses Ausgleichsproblem, erhalten eine Lösung $\mathbf{a}^{(1)}$ und betrachten die *Defekte* (auch Residuen oder algebraische Distanzen genannt):

$$\delta_k = F(\mathbf{a}^{(1)}, \mathbf{x}_k), \quad k = 1, \dots, K. \quad (21.94)$$

Nun können wir ein einfaches iteratives Verfahren durchführen:

- Für die gegebenen Bildpunkte $\mathbf{x}_k, k = 1, \dots, K$ löse man das Ausgleichsproblem und bestimme die Defekte $\delta_k, k = 1, \dots, K$. Die Menge der Indizes $\{1, \dots, K\}$ bezeichnen wir mit $I^{(1)}$.
- Nun bestimmen wir

$$k^* = \operatorname{argmax}_k (\delta_k) \quad (21.95)$$

und interpretieren den Bildpunkt \mathbf{x}_{k^*} als Ausreißer.

- Wir entfernen den Bildpunkt \mathbf{x}_{k^*} und entfernen aus der Indexmenge $I^{(1)}$ den Index k^* , diese sei nun $I^{(2)}$.
- Nun wiederholen wir die gesamte Ausgleichsrechnung bezüglich der Indexmenge $I^{(2)}$.
- Der Abbruch der Iteration ist aus der konkreten Anwendung heraus zu wählen, z. B. wenn ca. 10 Prozent der Punkte gestrichen wurden.

Dieses einfache iterative Verfahren kann auf alle Ausgleichsprobleme angewendet werden. Wenn allerdings die Anfangslösung „zu schlecht“ ist, ist es möglich, dass wir anstatt Ausreißer „korrekte“ Punkte eliminieren. Damit diese aber wieder eine Chance haben in die Ausgleichsrechnung einzugehen, kann man auch eine andere Strategie wählen. Diese Strategie nennt man *adaptives Fitting mit Reparametrisierung der Gewichte*:

- Für die gegebenen Bildpunkte $\mathbf{x}_k, k = 1, \dots, K$ löse man eine gewichtetes Ausgleichsproblem und bestimme die Defekte $\delta_k, k = 1, \dots, K$. Die Gewichte werden zu Beginn auf $g_k = 1, k = 1, \dots, K$ gesetzt.
- Nun berechnen wir aus den Defekten $\delta_k, k = 1, \dots, K$ die neuen Gewichte $g_k, k = 1, \dots, K$. Dazu gibt es viele Möglichkeiten. Man kann z. B. an die Defekte die Dichte einer Normalverteilung anpassen und die Gewichte sind dann die Funktionswerte der Dichtefunktion.
- Mit den neuen Gewichten wiederholen wir wieder die gewichtete Ausgleichsrechnung.
- Die Anzahl der Iterationen ist wieder heuristisch zu wählen und richtet sich nach der konkreten Anwendung.

In den Abb. 21.1 und 21.2 ist das adaptive Fitting mit Reparametrisierung der Gewichte für eine Geraden- und Kreisanpassung demonstriert.

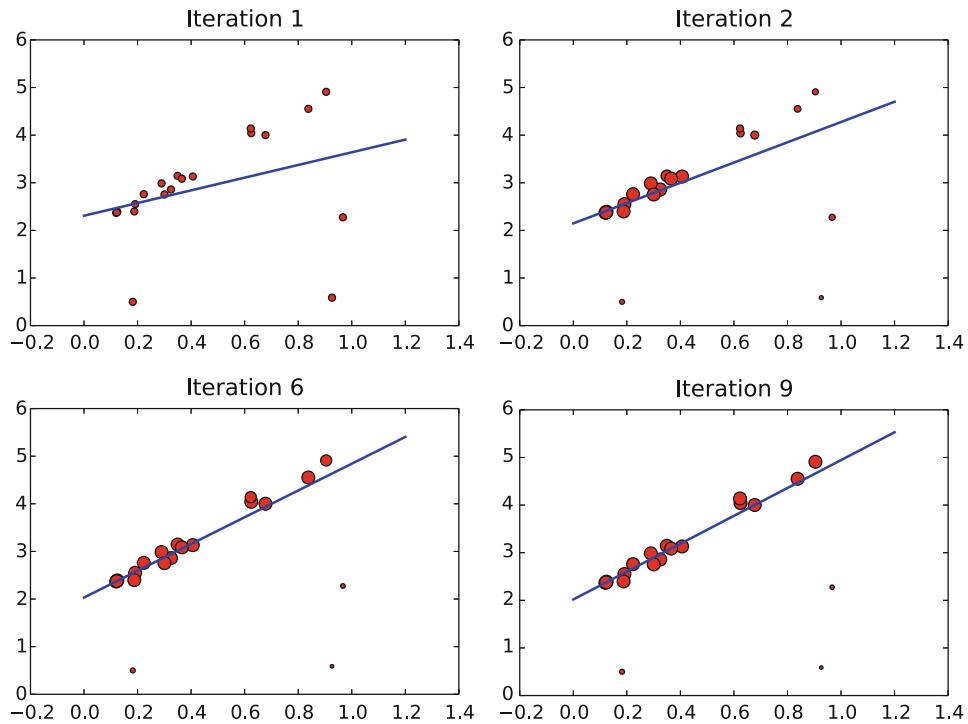


Abb. 21.1 Adaptives Geradenfitting: Schätzungen einer Geraden in mehreren Iterationen des Verfahrens. In der Abbildung ist die Größe der Punkte proportional zu den adaptiven Gewichten, welche aus den Defekten ermittelt werden

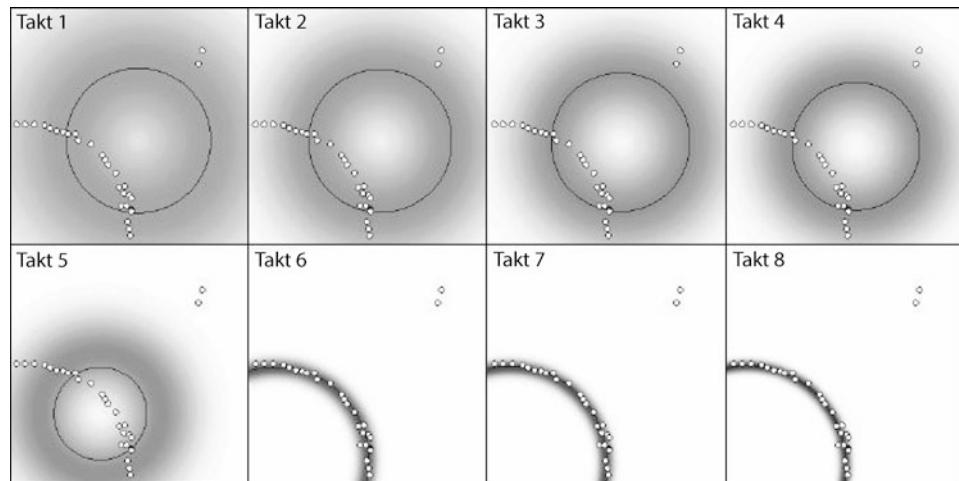


Abb. 21.2 Adaptives Kreisfitting

21.2.5 Andere Schätzer (LAD)

Bisher hatten wir uns sowohl adaptiv als auch nicht, mit oder ohne Gewichten, immer auf die L_2 -Norm bezogen. Natürlich gibt es noch eine Reihe anderer Schätzer, insbesondere *Median*-Schätzer. Diese sind in der Regel robuster gegenüber Ausreißern, bedingen aber einen höheren numerischen Aufwand. Diese Eigenschaft wurde z. B. beim Medianfilter ausgenutzt, siehe Abschn. 8.2.5. Die Robustheit wird oft mit dem *break-down-Index* charakterisiert. Dieser besagt: wieviel Prozent Ausreißer reichen aus, damit das Fittingergebnis völlig verfälscht wird. Mehr als 50 Prozent ist theoretisch gar nicht möglich. Wir wollen hier stellvertretend einmal die L_1 -Norm untersuchen, die auch tatsächlich etwas mit dem Median zu tun hat. Nutzen wir diese Norm, dann nennt man den Schätzer einen LAD-Schätzer (**Least Absolute Differences**). Dazu betrachten wir einmal eindimensionale Messwerte $x_k, k = 1, \dots, K$ und betrachten das L_2 -Ausgleichsproblem:

$$S(a) = \sum_{k=1}^K (x_k - a)^2 \rightarrow \text{Minimum} \rightarrow a = \frac{1}{K} \sum_{k=1}^K x_k. \quad (21.96)$$

Die Summe der Abweichungsquadrate minimiert also gerade der Mittelwert. Bei einem großen Ausreißer wird daher sofort auch der Mittelwert verfälscht. Nehmen wir dagegen die L_1 -Norm

$$S(a) = \sum_{k=1}^K |x_k - a| \rightarrow \text{Minimum} \rightarrow a = \text{median}(x_1, \dots, x_K), \quad (21.97)$$

so minimiert gerade der Median die Summe der Betragsabweichungen. Die Robustheit ist sofort zu erkennen, haben wir einen extremen Ausreißer, so hat das keinen Einfluss auf den Median. Da es z. B. für Vektoren keinen Median gibt, bezeichnet man oft als „Pseudomedian“ den Vektor, der die L_1 -Norm minimiert und nutzt diesen auch.

Um die L_1 -Norm für die Ausgleichsrechnung zu untersuchen, setzen wir lineare Modellgleichungen voraus. Wir nehmen an, dass

$$F(\mathbf{a}, \mathbf{x}) = \sum_{i=1}^n a_i f_i(\mathbf{x}) - f_0(\mathbf{x}) = 0 \quad (21.98)$$

die Modellgleichung ist. Dann ist das Extremalproblem

$$S(\mathbf{a}) = \sum_{k=1}^K |F(\mathbf{a}, \mathbf{x}_k)| \rightarrow \text{Minimum} \quad (21.99)$$

zu lösen. Wir führen nun die Lösung von (21.99) auf die Lösung einer linearen Optimierungsaufgabe zurück. Diese Idee geht bereits auf W.M. Wagner (1959) zurück. Dazu schreiben wir erst einmal die Standardform eines LO-Problems auf (die Bezeichnungen

haben nichts mit den bisherigen Bezeichnungen zu tun):

$$z = \mathbf{c}^T \mathbf{x} \rightarrow \text{Minimum bei } \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}. \quad (21.100)$$

Ist $\dim(\mathbf{x}) = n$, so ist die Restriktionsmatrix \mathbf{A} vom Typ (m, n) , wobei in der Regel $m < n$ ist. Gegeben sind also $\mathbf{A}, \mathbf{b}, \mathbf{c}$ und gesucht ist das optimale \mathbf{x} . Der zulässige Bereich ist die Menge $Z = \{\mathbf{x} | \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. Wir nehmen an, dieser ist nichtleer und beschränkt. Aus der Optimierung ist bekannt, dass dieser zulässige Bereich ein konkaves Polyeder beschreibt und dass das Minimum in einem Eckpunkt dieses Polyeders angenommen wird. Als Lösungsmethode wurde bereits von Dantzig (1947) die *Simplexmethode* entwickelt, die es heute in vielen Varianten gibt, z. B. primale Simplexmethode, duale Simplexmethode, revidierte Simplexmethode. Es gibt auch noch andere Lösungsmethoden als die Simplexmethode. Wichtig ist, eine Lösungsmethode steht in vielen Softwaresystemen als *Standardwerkzeug* zur Verfügung. Das LAD-Problem (21.99) ist nun der LO-Aufgabe

$$z = \sum_{k=1}^K (z_k^+ + z_k^-) \rightarrow \text{Minimum} \quad (21.101)$$

$$\text{bei } F(\mathbf{a}, \mathbf{x}_k) + z_k^+ - z_k^- = 0, z_k^+, z_k^- \geq 0, k = 1, \dots, K \quad (21.102)$$

äquivalent. Man sieht, die Defekte $\delta_k = z_k^- - z_k^+$ wurden als Differenz von nichtnegativen Variablen modelliert.

Wichtig Man kann sich leicht überlegen, bei der optimalen Lösung muss immer mindestens eine der beiden Variablen z_k^-, z_k^+ Null sein. Dadurch werden tatsächlich die Beträge der Defekte minimiert.

Da die Variablen $a_i, i = 1, \dots, n$ auch negativ werden können, modellieren wir sie auch als Differenz von nichtnegativen Variablen, folglich $a_i = a_i^+ - a_i^-, a_i^+, a_i^- \geq 0, i = 1, \dots, n$. Diese gehen allerdings nicht in die Zielfunktion z ein. Folglich haben wir insgesamt K Restriktionen und $2(n + K)$ Variable. Ist allerdings $f_0(\mathbf{x}) = 0$, so kommt noch eine lineare Restriktion für die Parameter a_i hinzu. Da wir Differenzen von nichtnegativen Variablen eingeführt haben, ist der zulässige Bereich Z unbeschränkt. Kommt eine Softwareroutine damit nicht zurecht, führen wir eine künstliche Beschränkung

$$\sum_{i=1}^n (a_i^+ + a_i^-) + \sum_{k=1}^K (z_k^+ + z_k^-) + s = M \quad (21.103)$$

ein, wobei $s \geq 0$ gilt und M „hinreichend groß“ zu wählen ist. Dazu ein

Beispiel Wir wollen an eine Punktmenge eine Ellipse mit der L_1 -Norm fitten. Die lineare Modellgleichung lautet

$$F(\mathbf{a}, \mathbf{x}_k) = a_1 x_k^2 + a_2 x_k y_k + a_3 y_k^2 + a_4 x_k + a_5 y_k + a_6 = 0. \quad (21.104)$$

Da $f_0(\mathbf{x}) = 0$ ist, müssen wir eine lineare Restriktion einführen, z. B. $a_1 + a_3 = 1$. Der Einfachheit halber seien nur 6 Bildpunkte (x_i, y_i) , $i = 1, \dots, 6$ gegeben. Wir führen noch eine Restriktion zur Beschränkung des zulässigen Bereiches ein, so dass wir 8 Restriktionen und 25 Variable haben. In der folgenden Matrixdarstellung ist der erste Zeile der Vektor \mathbf{c} , dann die Restriktionsmatrix \mathbf{A} , und dann der Vektor der rechten Seite \mathbf{b} :

$$\begin{array}{cccccccccccccccccccccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 10^6 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 1 \\ x_1^2 & -x_1^2 & x_1y_1 & -x_1y_1 & y_1^2 & -y_1^2 & x_1 & -x_1 & y_1 & -y_1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_2^2 & -x_2^2 & x_2y_2 & -x_2y_2 & y_2^2 & -y_2^2 & x_2 & -x_2 & y_2 & -y_2 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3^2 & -x_3^2 & x_3y_3 & -x_3y_3 & y_3^2 & -y_3^2 & x_3 & -x_3 & y_3 & -y_3 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_4^2 & -x_4^2 & x_4y_4 & -x_4y_4 & y_4^2 & -y_4^2 & x_4 & -x_4 & y_4 & -y_4 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_5^2 & -x_5^2 & x_5y_5 & -x_5y_5 & y_5^2 & -y_5^2 & x_5 & -x_5 & y_5 & -y_5 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_6^2 & -x_6^2 & x_6y_6 & -x_6y_6 & y_6^2 & -y_6^2 & x_6 & -x_6 & y_6 & -y_6 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \end{array} \quad (21.105)$$

Die Schranke M wurde willkürlich auf 10^6 gesetzt. Wählt man diese zu klein, wird womöglich das Optimum verfälscht. Man sollte deshalb die Punkte auf ihren Schwerpunkt transformieren und womöglich noch skalieren, damit die Zahlen in der Matrix nicht so groß werden.

Natürlich können wir mit der L_1 -Norm auch adaptiv Fitten und eine gewichtete Ausgleichsrechnung durchführen, die Gewichte gehen dann in die Zielfunktion ein:

$$z = \sum_{k=1}^K g_k (z_k^+ + z_k^-) \rightarrow \text{Minimum} \quad (21.106)$$

$$\text{bei } F(\mathbf{a}, \mathbf{x}_k) + z_k^+ - z_k^- = 0, z_k^+, z_k^- \geq 0, k = 1, \dots, K. \quad (21.107)$$

In direktem Zusammenhang zur Ausgleichsrechnung stehen die Begriffe der Singulärwertzerlegung und der Pseudoinversen, deshalb gehen wir auf diese Begriffe im Folgenden näher ein. Weiterhin sind gewisse Begriffe der Stochastik, wie z. B. die Normalverteilung, von besonderem Interesse für die angegebenen Schätzer sowie für Verfahren des maschinellen Lernens.

22.1 Pseudoinverse

Da pseudoinverse Matrizen eine große Rolle für die Ausgleichsrechnung spielen, wollen wir diesen Begriff genau klären. Gegeben sei eine Matrix \mathbf{A} mit m Zeilen und n Spalten. Nun betrachten wir die Abbildung:

$$\mathbf{y} = \mathbf{Ax} : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m. \quad (22.1)$$

Wir betrachten den Bildraum:

$$R(\mathbf{A}) = \{\mathbf{y} \mid \mathbf{y} = \mathbf{Ax}, \mathbf{x} \in \mathbb{R}^n\}. \quad (22.2)$$

Dieser wird offensichtlich von den Spaltenvektoren von \mathbf{A} aufgespannt. Es sei weiterhin $N(\mathbf{A})$ der Nullraum von \mathbf{A} und $S(\mathbf{A})$ das orthogonale Komplement zum Nullraum von \mathbf{A} . Dies sind lineare Räume und beide spannen zusammen den \mathbb{R}^n auf. Genauso können wir die Räume der transponierten Matrix \mathbf{A}^T betrachten. Wir betrachten demnach den Nullraum $N(\mathbf{A}^T)$ und sein orthogonales Komplement $S(\mathbf{A}^T)$, beide spannen analog den \mathbb{R}^m auf. Betrachten wir nun den Nullraum der transponierten Matrix etwas genauer:

$$N(\mathbf{A}^T) = \{\mathbf{y} \mid \mathbf{A}^T \mathbf{y} = 0, \mathbf{y} \in \mathbb{R}^m\}. \quad (22.3)$$

Offensichtlich sind die Zeilenvektoren von \mathbf{A}^T orthogonal zum Nullraum $N(\mathbf{A}^T)$ oder was dasselbe ist, die Spaltenvektoren von \mathbf{A} sind orthogonal zu $N(\mathbf{A}^T)$. Damit spannen aber die Spaltenvektoren von \mathbf{A} das orthogonale Komplement $S(\mathbf{A}^T)$ auf. Demzufolge können wir schlussfolgern:

$$R(\mathbf{A}) = S(\mathbf{A}^T), \quad R(\mathbf{A}^T) = S(\mathbf{A}). \quad (22.4)$$

Wir behaupten nun: die Abbildung

$$\mathbf{x} \leftrightarrow \mathbf{y} : \mathbf{y} = \mathbf{Ax}, \quad \mathbf{x} \in S(\mathbf{A}), \mathbf{y} \in R(\mathbf{A}) = S(\mathbf{A}^T) \quad (22.5)$$

ist eine bijektive Abbildung, d. h. sie ist eindeutig umkehrbar. Dies ist sehr einfach zu zeigen. Dazu betrachten wir die Gleichungen $\mathbf{Ax} = \mathbf{z}$, $\mathbf{Ay} = \mathbf{z}$, mit $\mathbf{x} \neq \mathbf{y}$ und $\mathbf{x}, \mathbf{y} \in S(\mathbf{A})$. Daraus folgt $\mathbf{A}(\mathbf{x} - \mathbf{y}) = \mathbf{0}$ und damit $\mathbf{x} - \mathbf{y} \in N(\mathbf{A})$. Da aber $\mathbf{x} \in S(\mathbf{A})$ und $\mathbf{y} \in S(\mathbf{A})$ folgt auch $\mathbf{x} - \mathbf{y} \in S(\mathbf{A})$, dies ist aber nur für das Nullelement möglich $\mathbf{x} - \mathbf{y} = \mathbf{0}$, damit muss $\mathbf{x} = \mathbf{y}$ sein, was im Widerspruch zur Annahme steht. Wenn wir im Folgenden die kleinen Indizes o und n verwenden, meinen wir ein Element des orthogonalen Komplements bzw. des Nullraumes. Also wir betrachten jetzt die bijektive Abbildung

$$\mathbf{y}^o = \mathbf{Ax}^o, \quad \mathbf{x}^o \in S(\mathbf{A}), \mathbf{y}^o \in R(\mathbf{A}) = S(\mathbf{A}^T) \quad (22.6)$$

die wir in der inversen Form schreiben können

$$\mathbf{x}^o = \mathbf{A}^+ \mathbf{y}^o \quad (22.7)$$

und nennen die Matrix \mathbf{A}^+ die Pseudoinverse von \mathbf{A} . Da $\mathbf{y}^o \in S(\mathbf{A}^T)$, können wir setzen $S(\mathbf{A}^T) = S(\mathbf{A}^+)$ und damit auch $N(\mathbf{A}^+) = N(\mathbf{A}^T)$, womit wir jetzt den ganzen \mathbb{R}^m als Definitionsbereich für \mathbf{A}^+ eingeführt haben. Nun wollen wir ein klassisches lineares Ausgleichsproblem betrachten $\mathbf{Ax} = \mathbf{b}$, wobei „in der Regel“ $m > n$, $R(\mathbf{A}) = S(\mathbf{A}^T) \subset \mathbb{R}^m$ und $\mathbf{b} \in \mathbb{R}(\mathbf{A})$, $\mathbf{b} \in N(\mathbf{A}^T)$ gilt. Da wir $\mathbf{b} = \mathbf{b}^o + \mathbf{b}^n$ als Summe dieser Projektionen schreiben können, folgt $\mathbf{x}^o = \mathbf{A}^+ \mathbf{b} = \mathbf{A}^+(\mathbf{b}^o + \mathbf{b}^n) = \mathbf{A}^+ \mathbf{b}^o$ mit $\mathbf{Ax}^o = \mathbf{b}^o$, und somit ist $\mathbf{x}^o = \mathbf{A}^+ \mathbf{b}$ Lösung von

$$\|\mathbf{Ax} - \mathbf{b}\|^2 \rightarrow \text{Minimum}, \quad (22.8)$$

weil \mathbf{b}^o die orthogonale Projektion von \mathbf{b} in $R(\mathbf{A}) = S(\mathbf{A}^T)$ darstellt. Damit ist \mathbf{x}^o eine Lösung des Ausgleichsproblems. Falls nun das Ausgleichsproblem nicht eindeutig lösbar ist, welche von den möglichen Lösungen stellt dann aber \mathbf{x}^o dar? Wenn \mathbf{x}^o Lösung ist, dann ist auch $\mathbf{x}^b = \mathbf{x}^o + \mathbf{x}^n$ Lösung. Wir schätzen nun die Länge von \mathbf{x}^b ab:

$$\|\mathbf{x}^b\|^2 = (\mathbf{x}^o + \mathbf{x}^n)^T (\mathbf{x}^o + \mathbf{x}^n) = \|\mathbf{x}^o\|^2 + \|\mathbf{x}^n\|^2 + 2\mathbf{x}^o T \mathbf{x}^n \geq \|\mathbf{x}^o\|^2. \quad (22.9)$$

Folglich ist $\mathbf{x}^o = \mathbf{A}^+ \mathbf{b}$ die **normkleinste** Lösung, also Lösung von

$$\min(\|\mathbf{x}\|^2) \quad \text{bei } \|\mathbf{Ax} - \mathbf{b}\|^2 \rightarrow \text{Minimum} \quad (22.10)$$

und wird oft als **Pseudonormallösung** von $\mathbf{Ax} = \mathbf{b}$ bezeichnet.

Als einfaches Beispiel der Pseudoinversen betrachten wir eine $m \times n$ Diagonalmatrix \mathbf{D} der Form:

$$\mathbf{D} = \begin{pmatrix} a_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & a_{rr} & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & 0 \end{pmatrix}, \quad (22.11)$$

wobei $m > r$ sein soll. Durch folgende Schreibweise erkennen wir sofort die entsprechenden Räume:

$$\mathbf{Dx} = \mathbf{D} \cdot \left(\begin{pmatrix} x_1 \\ \vdots \\ x_r \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ x_{r+1} \\ \vdots \\ x_n \end{pmatrix} \right) = \begin{pmatrix} b_1 \\ \vdots \\ b_r \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_{r+1} \\ \vdots \\ b_n \end{pmatrix}. \quad (22.12)$$

Durch alle Vektoren $(x_1, \dots, x_r, 0, \dots, 0)^T$ wird das orthogonale Komplement $S(\mathbf{D})$ aufgespannt und durch die Vektoren $(0, \dots, 0, x_{r+1}, \dots, x_n)^T$ der Nullraum $N(\mathbf{D})$. Wir sehen leicht leicht, dass diese Räume orthogonal zueinander sind. Demzufolge wird der Bildraum $R(\mathbf{D}) = S(\mathbf{D}^T)$ durch die Vektoren $(b_1, \dots, b_r, 0, \dots, 0)^T$ und der Raum $N(\mathbf{D}^T)$ durch die Vektoren $(0, \dots, 0, b_{r+1}, \dots, b_n)^T$ aufgespannt. Folglich können wir nun ganz leicht die Pseudoinverse angeben:

$$\mathbf{D}^+ = \begin{pmatrix} \frac{1}{a_{11}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{a_{22}} & 0 & 0 & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \frac{1}{a_{rr}} & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & 0 \end{pmatrix}. \quad (22.13)$$

Als eine erste Anwendung betrachten wir ein allgemeines lineares Ausgleichsproblem der Form:

$$\mathbf{Ax} = \mathbf{b} \leftrightarrow \|\mathbf{Ax} - \mathbf{b}\|^2 = \text{Minimum} \leftrightarrow \mathbf{Ax} = \mathbf{b}^o. \quad (22.14)$$

Wir könnten also einfach die letzte Gleichung $\mathbf{Ax} = \mathbf{b}^o$ lösen, das Problem ist aber, dass wir \mathbf{b}^o nicht kennen. Da $\mathbf{b}^o \in S(\mathbf{A}^T) = S(\mathbf{A}^+)$, multiplizieren wir jetzt linke und rechte Seite mit \mathbf{A}^T und ergänzen:

$$\mathbf{Ax} = \mathbf{b}^o \leftrightarrow \mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}^o = \mathbf{A}^T (\mathbf{b}^o + \mathbf{b}^n) = \mathbf{A}^T \mathbf{b}. \quad (22.15)$$

Folglich erhalten wir als lineares Gleichungssystem die **Gaußschen Normalengleichungen**

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}, \quad (22.16)$$

welche uns schon aus (21.32) bekannt sind. Jede Lösung dieser Gleichung ist eine Lösung von $\mathbf{Ax} = \mathbf{b}$ im Sinne der kleinsten Quadrate, die Lösung muss aber nicht eindeutig sein. Ist die Lösung eindeutig, also existiert die Inverse $(\mathbf{A}^T \mathbf{A})^{-1}$, dann können wir die Lösung formal aufschreiben und sie muss identisch mit der Lösung über die Pseudoinverse sein:

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b} \leftrightarrow \mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \rightarrow \mathbf{A}^+ = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T. \quad (22.17)$$

Wir haben also eine erste explizite Darstellung der Pseudoinversen gefunden und zwar nur für den Fall, wenn das Ausgleichsproblem eindeutig lösbar ist. Wie kann man nun eine allgemeine Darstellung finden? Dies geht recht einfach mittels der **Singulärwertzerlegung** der Matrix A , siehe (22.44).

Satz 22.1 (Pseudoinverse) *Es sei $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$ die Singulärwertzerlegung der Matrix \mathbf{A} mit den Singulärwerten $\sigma_1, \sigma_2, \dots, \sigma_r$. Dann berechnet sich die Pseudoinverse \mathbf{A}^+ von \mathbf{A} zu*

$$\mathbf{A}^+ = \mathbf{V}\Sigma^+\mathbf{U}^T. \quad (22.18)$$

Dabei ist Σ^+ die Pseudoinverse der Diagonalmatrix Σ , deren Diagonalelemente einfach zu $\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}$ berechnet werden.

Wir wollen nun noch folgende naheliegende Frage klären: Wenn von einer quadratischen Matrix \mathbf{A} die Inverse \mathbf{A}^{-1} existiert, dann gilt $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{E}$, wobei \mathbf{E} die Einheitsmatrix ist. Die Frage ist nun, was sind das für Matrizen \mathbf{AA}^+ und $\mathbf{A}^+\mathbf{A}$, wenn die Inverse \mathbf{A}^{-1} nicht existiert bzw. wenn \mathbf{A} keine quadratische Matrix ist? Auf jeden Fall sind erst einmal beide Matrizen \mathbf{AA}^+ und $\mathbf{A}^+\mathbf{A}$ quadratische Matrizen, wobei die Dimensionen verschieden sein können.

Wir betrachten demzufolge nun quadratische Matrizen \mathbf{P} mit ihren Räumen $N(\mathbf{P})$ und $S(\mathbf{P})$, dann gilt

$$\mathbf{Px} = \mathbf{P}(\mathbf{x}^o + \mathbf{x}^n) = \mathbf{Px}^o. \quad (22.19)$$

Nun fordern wir eine zusätzliche, sogenannte **Projektionseigenschaft**

$$\mathbf{P}\mathbf{x}^o = \mathbf{x}^o. \quad (22.20)$$

Damit können wir rechnen:

$$\mathbf{P}(\mathbf{Px}) = \mathbf{P}(\mathbf{x}^o) = \mathbf{P}(\mathbf{x}^o + \mathbf{x}^n) = \mathbf{Px}. \quad (22.21)$$

Daraus folgt $\mathbf{P}^k = \mathbf{P} \forall k$. Weiterhin gilt:

$$\mathbf{x}^T \mathbf{P}^T \mathbf{y} = (\mathbf{Px})^T \mathbf{y} = \mathbf{x}^{oT} (\mathbf{y}^o + \mathbf{y}^n) = \mathbf{x}^{oT} \mathbf{y}^o = (\mathbf{x}^o + \mathbf{x}^n)^T \mathbf{y}^o = \mathbf{x}^T \mathbf{Py}. \quad (22.22)$$

Daraus folgt die Symmetrie $\mathbf{P}^T = \mathbf{P}$.

Wir schlussfolgern: eine quadratische Matrix \mathbf{P} heißt **Projektionsmatrix**, wenn gilt:

- $\mathbf{P}^k = \mathbf{P} \forall k$,
- $\mathbf{P}^T = \mathbf{P}$.

Wir bilden nun

$$\mathbf{A}^+ \mathbf{y}^o = \mathbf{A}^+ \mathbf{Ax}^o = \mathbf{x}^o. \quad (22.23)$$

Damit ist $\mathbf{P} = \mathbf{A}^+ \mathbf{A}$ eine Projektionsmatrix im \mathbb{R}^n . Weiterhin ist auch

$$\mathbf{Ax}^o = \mathbf{AA}^+ \mathbf{y}^o = \mathbf{y}^o. \quad (22.24)$$

Damit ist $\mathbf{Q} = \mathbf{AA}^+$ Projektionsmatrix im \mathbb{R}^m . Wir wissen also jetzt, was diese quadratischen Matrizen für Eigenschaften besitzen, sie sind Projektionsmatrizen. Daher können wir nun auch schreiben:

$$R(\mathbf{A}) = S(\mathbf{A}^+) = S(\mathbf{A}^T) \subset \mathbb{R}^m \rightarrow (\mathbf{AA}^+) \mathbf{A} = \mathbf{A}, \quad (22.25)$$

$$R(\mathbf{A}^+) = R(\mathbf{A}^T) = S(\mathbf{A}) \subset \mathbb{R}^n \rightarrow (\mathbf{A}^+ \mathbf{A}) \mathbf{A}^+ = \mathbf{A}^+. \quad (22.26)$$

Wenn wir in den beiden Bedingungen einfach $\mathbf{G} = \mathbf{A}^+$ setzen und diese beiden Gleichungen nochmals aufschreiben und berücksichtigen, dass die Projektionsmatrizen \mathbf{GA} und \mathbf{AG} symmetrisch sein müssen, dann erhalten wir die sogenannten **Moore-Penrose-Bedingungen** zur Definition der Pseudoinversen. Wenn also eine Matrix \mathbf{G} die vier Bedingungen

$$1. (\mathbf{GA})^T = \mathbf{GA} \quad 2. (\mathbf{AG})^T = \mathbf{AG} \quad 3. \mathbf{AGA} = \mathbf{A} \quad 4. \mathbf{GAG} = \mathbf{G} \quad (22.27)$$

erfüllt, dann nennen wir \mathbf{G} die Pseudoinverse von \mathbf{A} .

22.2 Regularisierung nach Tichonov und Arsenin

Die Instabilität eines Systems kann beträchtliche Folgen haben, wie wir dies beim Invers-Filter gezeigt haben. Daher bildet man oft ein Ersatzproblem, welches stabil ist, aber leider nicht die gewünschte Lösung hat. Man geht dabei oft einen Kompromiss zwischen Stabilität und dem „Abtriften“ der Lösung ein, dies nennt man **Regularisierung** des Problems. Die Pseudonormallösung eines linearen Gleichungssystems bezüglich der Pseudoinversen löst das Optimierungsproblem:

$$\min(\|\mathbf{x}\|^2) \quad \text{bei } \|\mathbf{Ax} - \mathbf{b}\|^2 \rightarrow \text{Minimum.} \quad (22.28)$$

Die Lösung war $\mathbf{x} = \mathbf{A}^+ \mathbf{b}$. Nun schwächen wir einmal die Forderung nach dem Minimum ab:

$$\min(\|\mathbf{x}\|^2) \quad \text{bei } \|\mathbf{Ax} - \mathbf{b}\|^2 \leq \delta^2. \quad (22.29)$$

Dabei fordern wir lediglich, dass der Fehler eine Schranke nicht überschreiten soll, dies ist also ein schwächeres Ersatzproblem. Dies können wir auch äquivalent in der Form

$$\min(\|\mathbf{x}\|^2) \quad \text{bei } \|\mathbf{Ax} - \mathbf{b}\| \leq \delta \quad (22.30)$$

schreiben. Das Minimum wird auf dem Rand angenommen. Dazu sei

$$0 < \|\mathbf{x}_{\min}\|^2, \quad \delta_{\min} = \|\mathbf{Ax}_{\min} - \mathbf{b}\| < \delta. \quad (22.31)$$

Wir setzen

$$\kappa = \min\left(1, \frac{\delta - \delta_{\min}}{\|\mathbf{A}\| \cdot \|\mathbf{x}_{\min}\|}\right), \quad \mathbf{x} = (1 - \kappa)\mathbf{x}_{\min} \quad (22.32)$$

und es gilt

$$\|\mathbf{Ax} - \mathbf{b}\| \leq \|\mathbf{Ax}_{\min} - \mathbf{b}\| + \kappa \|\mathbf{A}\| \cdot \|\mathbf{x}_{\min}\| \leq \delta, \quad (22.33)$$

womit \mathbf{x} keine optimale, aber eine zulässige Lösung ist. Aber es ist auch

$$\|\mathbf{x}\|^2 = (1 - \kappa)^2 \|\mathbf{x}_{\min}\|^2 < \|\mathbf{x}_{\min}\|^2, \quad (22.34)$$

d. h. \mathbf{x} ist eine bessere Lösung als \mathbf{x}_{\min} . Daher kann das Minimum nur auf dem Rand angenommen werden. Demzufolge brauchen wir nur noch die Minimum-Aufgabe

$$\min(\|\mathbf{x}\|^2) \quad \text{bei } \|\mathbf{Ax} - \mathbf{b}\|^2 = \delta^2 \quad (22.35)$$

zu betrachten. Nun formulieren wir das Minimumproblem mit der Lagrange-Funktion:

$$L = \|\mathbf{x}\|^2 + \alpha(\|\mathbf{Ax} - \mathbf{b}\|^2 - \delta^2) \rightarrow \text{Minimum.} \quad (22.36)$$

Wir bilden die Ableitung und setzen gleich Null:

$$\nabla_{\mathbf{x}} L = 2\mathbf{x} + 2\alpha(\mathbf{A}^T \mathbf{Ax} - \mathbf{A}^T \mathbf{b}) = \mathbf{0}. \quad (22.37)$$

Wir teilen durch α und setzen $\lambda = \frac{1}{\alpha}$:

$$\lambda \mathbf{Ix} + (\mathbf{A}^T \mathbf{Ax} - \mathbf{A}^T \mathbf{b}) = \mathbf{0}. \quad (22.38)$$

Nun fassen wir zusammen:

$$(\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I}) \mathbf{x} = \mathbf{A}^T \mathbf{b}. \quad (22.39)$$

Auf der linken Seite steht die regularisierte Matrix durch Störung der Hauptdiagonalen. Der Regularisierungsparameter λ beschreibt die „Stärke“ des Ersatzproblems. Dieses wird stabiler, je größer λ wird.

22.3 Singulärwertzerlegung (SVD)

22.3.1 Grundlagen

Die Singulärwertzerlegung (SVD – Singular Value Decomposition) wird in der Bildverarbeitung an sehr vielen Stellen benötigt, deshalb seien hierzu die Grundlagen kurz aufgeschrieben. Gegeben sei eine reelle, rechteckige $m \times n$ Matrix \mathbf{B} , die wir in zwei orthogonale Matrizen \mathbf{U} und \mathbf{V} sowie in eine beliebige Diagonalmatrix \mathbf{A} nach der Vorschrift

$$\mathbf{B} = \mathbf{U} \mathbf{A} \mathbf{V}^T \quad (22.40)$$

zerlegen wollen. Dazu bilden wir die Produkte:

$$\begin{aligned} \mathbf{B} \mathbf{B}^T &= \mathbf{U} \mathbf{A} \mathbf{V}^T \mathbf{V} \mathbf{A}^T \mathbf{U}^T = \mathbf{U} \mathbf{A} \mathbf{A}^T \mathbf{U}^T \leftrightarrow \mathbf{B} \mathbf{B}^T \mathbf{U} = \mathbf{U} \mathbf{A} \mathbf{A}^T \\ \mathbf{B}^T \mathbf{B} &= \mathbf{V} \mathbf{A}^T \mathbf{U}^T \mathbf{U} \mathbf{A} \mathbf{V}^T = \mathbf{V} \mathbf{A}^T \mathbf{A} \mathbf{V}^T \leftrightarrow \mathbf{B}^T \mathbf{B} \mathbf{V} = \mathbf{V} \mathbf{A}^T \mathbf{A}. \end{aligned} \quad (22.41)$$

An diesen beiden Darstellungen können wir sofort etwas ablesen. Dazu betrachten wir als Wiederholung ein klassisches Eigenwertproblem, siehe auch (21.20): $\mathbf{Cx} = \lambda \mathbf{x}$. Dabei sei \mathbf{C} eine quadratische, reelle und symmetrische Matrix, damit sind die Eigenwerte

λ stets reell. Wenn \mathbf{C} zusätzlich noch positiv semidefinit ist, dann sind alle Eigenwerte nichtnegativ. Lässt sich \mathbf{C} darstellen als Matrixprodukt $\mathbf{C} = \mathbf{B}^T \mathbf{B}$ oder $\mathbf{C} = \mathbf{B} \mathbf{B}^T$, dann ist \mathbf{C} trivialerweise symmetrisch und positiv semidefinit. Zu den reellen Eigenwerten gehört dann ein orthonormiertes System von Eigenvektoren. Diese Eigenvektoren schreiben wir als Spaltenvektoren in eine Matrix \mathbf{X} , die dann orthogonal ist. Schreiben wir dann noch alle Eigenwerte in eine Diagonalmatrix Λ , dann lässt sich das Eigenwertproblem in Matrixschreibweise formulieren:

$$\mathbf{C}\mathbf{X} = \mathbf{X}\Lambda \Leftrightarrow \mathbf{C} = \mathbf{X}\Lambda\mathbf{X}^T. \quad (22.42)$$

Wenn wir dies mit obigen Gleichungen bez. \mathbf{U} und \mathbf{V} vergleichen, lesen wir sofort ab:

In \mathbf{U} stehen die orthonormierten Eigenvektoren von $\mathbf{B}\mathbf{B}^T$ mit den Eigenwerten in $\Lambda_1 = \mathbf{A}\mathbf{A}^T$, in \mathbf{V} stehen die orthonormierten Eigenvektoren von $\mathbf{B}^T\mathbf{B}$ mit den Eigenwerten $\Lambda_2 = \mathbf{A}^T\mathbf{A}$. Damit sind nun auch Λ_1 und Λ_2 Diagonalmatrizen. Man kann nun noch zeigen, dass $\Lambda_1 = \Lambda_2 = \Lambda$ für quadratische Matrizen \mathbf{B} ist. Für nichtquadratische Matrizen sind alle Eigenwerte ungleich Null auch gleich, der Rest ist eben eine unterschiedliche Anzahl von Eigenwerten, die aber alle Null sind. Damit ergibt sich als Singulärwertzerlegung für allgemeine, reelle und rechteckige Matrizen \mathbf{B} :

$$\mathbf{B} = \mathbf{U}\Lambda\mathbf{V}^T. \quad (22.43)$$

Wir fassen nochmals zusammen: in \mathbf{U} stehen die orthonormierten Eigenvektoren von $\mathbf{B}\mathbf{B}^T$, in \mathbf{V} die orthonormierten Eigenvektoren von $\mathbf{B}^T\mathbf{B}$, in Λ stehen die sogenannten singulären Werte, die nichts anderes als die Wurzeln aus den nichtnegativen Eigenwerten von $\mathbf{B}\mathbf{B}^T$ bzw. $\mathbf{B}^T\mathbf{B}$ sind. (Die positiven Eigenwerte von $\mathbf{B}\mathbf{B}^T$ und $\mathbf{B}^T\mathbf{B}$ sind die gleichen, der Rest ist sowieso gleich Null)

Im Folgenden seien die Spaltenvektoren von \mathbf{U} mit \mathbf{u}_k und die Spaltenvektoren von \mathbf{V} mit \mathbf{v}_l bezeichnet. Weiterhin bezeichnen wir die Diagonalmatrix der Singulärwerte nicht mehr mit Λ , sondern mit Σ , die Singulärwerte folglich mit $\sigma_1, \dots, \sigma_r$. Wir schreiben auch im Folgenden immer die Singulärwerte der Größe nach absteigend geordnet in die Diagonale von Σ , d. h. $\sigma_1 \geq \sigma_2 \geq \sigma_3 \dots$. Dies können wir tun, wenn wir auch entsprechend die Spaltenvektoren von \mathbf{U} und \mathbf{V} umordnen. Wir können die Singulärwertzerlegung auch als Summe schreiben:

$$\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^T \Leftrightarrow \mathbf{B} = \sum_{k=1}^n \sigma_k \mathbf{u}_k \mathbf{v}_k^T, \quad \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n. \quad (22.44)$$

Die Matrix \mathbf{B} lässt sich also als gewichtete Summe (Gewichte sind die Singulärwerte) von Matrizen schreiben, deren Rang jeweils eins ist. Diese Matrizen $\mathbf{u}_k \mathbf{v}_k^T$ vom Range eins ergeben sich als dyadisches Produkt der beiden Spaltenvektoren \mathbf{u}_k und \mathbf{v}_k oder als Tensorprodukt der beiden Vektoren. Wichtig für Anwendungen in der Bildverarbeitung ist der folgende

Approximationssatz Es sei $\mathbf{B} \in M_{m,n}(\mathbb{R})$ mit $\text{rang}(\mathbf{B}) = r$ und $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^T$. Mit $k \in \{0, 1, \dots, r-1\}$ bezeichne $M_{m,n}^k(R)$ die Menge aller Matrizen vom Range $\leq k$. Dann gilt:

$$\min_{\mathbf{X} \in M_{m,n}^k(R)} \|\mathbf{B} - \mathbf{X}\|_S = \|\mathbf{B} - \hat{\mathbf{X}}\|_S = \sigma_{k+1}, \quad (22.45)$$

wobei

$$\hat{\mathbf{X}} = \mathbf{U}\hat{\Sigma}\mathbf{V}^T, \quad \hat{\Sigma} = \begin{pmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \quad (22.46)$$

ist und $\|\cdot\|_S$ die Spektralnorm bedeutet.

Dieser Satz wird auch als *Eckart-Young-Theorem* bezeichnet und die Approximation ist auch richtig im Sinne der kleinsten Quadrate, d. h. wenn wir die Frobenius-Norm verwenden, allerdings ist dann der Fehler gleich:

$$\min_{\mathbf{X} \in M_{m,n}^k(R)} \|\mathbf{B} - \mathbf{X}\|_F = \|\mathbf{B} - \hat{\mathbf{X}}\|_F = \sqrt{\sum_{l \geq k+1} \sigma_l^2}. \quad (22.47)$$

Die beste Approximation ist dann:

$$\hat{\mathbf{X}} = \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^T. \quad (22.48)$$

Der kleinste singuläre Wert ist demnach ein Maß für die Nähe der regulären Matrix \mathbf{B} zu einer singulären Matrix $\hat{\mathbf{X}}$. Es sei erinnert: Wenn $\|\cdot\|_2$ die Euklidische Norm eines Vektors ist, dann ist die zugehörige Matrixnorm die *Spektralnorm*:

$$\|\mathbf{B}\|_S = \rho(\mathbf{B}) = \max\{|\lambda|, \lambda \text{ Eigenwert von } \mathbf{B}\}. \quad (22.49)$$

Weiterhin gilt die interessante Beziehung $\|\mathbf{B}\|_S = \sqrt{\rho(\mathbf{B}^T \mathbf{B})} = \sigma_1$.

22.3.2 Anwendungen

Später werden wir sehen, wie die SVD zur Faltungsberechnung eingesetzt werden kann. Aber zunächst einige grundsätzlich andere Anwendungen der SVD.

- Die Zerlegung $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^T$ lässt sich auch geometrisch interpretieren. Die quadratische, reguläre Matrix \mathbf{B} beschreibe eine affine Transformation (ohne Translation und Spiegelungen). Dann lässt sich diese affine Transformation stets als die Hintereinanderausführung einer Rotation (\mathbf{V}), einer anisotropen Skalierung (Σ) und einer zweiten Rotation (\mathbf{U}) interpretieren.

- Das Ausgleichsproblem $\|\mathbf{Bx} - \mathbf{b}\|_2 \rightarrow \text{Minimum}$, welches die Lösung $\mathbf{x}^+ = \mathbf{B}^+\mathbf{b}$ besitzt (siehe Abschn. 22.1), wird durch die Gaußschen Normalengleichungen $\mathbf{B}^T\mathbf{Bx} = \mathbf{B}^T\mathbf{b}$ gelöst. Wir geben einmal ein Beispiel für \mathbf{B} und $\mathbf{B}^T\mathbf{B}$ an:

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix} \rightarrow \mathbf{B}^T\mathbf{B} = \begin{pmatrix} 1 + \varepsilon^2 & 1 & 1 \\ 1 & 1 + \varepsilon^2 & 1 \\ 1 & 1 & 1 + \varepsilon^2 \end{pmatrix}. \quad (22.50)$$

Es sei ε eine „kleine“ Gleitkommazahl, also darstellbar auf einem Computer, dann kann es aber passieren, dass $1 + \varepsilon^2$ nicht mehr darstellbar ist und demzufolge $1 + \varepsilon^2 = 1$ ist. Damit wird aber die Matrix $\mathbf{B}^T\mathbf{B}$ singulär, obwohl wir eine darstellbare Messmatrix \mathbf{B} zur Verfügung hatten. Numerische Probleme mit den Gaußschen Normalengleichungen gibt es folglich auch, wenn die Messgleichungen stark korreliert sind. Daher ist es numerisch günstiger, direkte Lösungsmethoden zu verwenden anstatt die Gaußschen Normalengleichungen zu lösen. Eine Möglichkeit war die Nutzung der QR-Zerlegung, siehe (21.34). Eine weitere Möglichkeit bietet nun die SVD. Da die Lösung der Gaußschen Normalengleichungen durch $\mathbf{x}^+ = \mathbf{B}^+\mathbf{b}$ darstellbar ist, brauchen wir nur die Pseudoinverse \mathbf{B}^+ mit der SVD zu berechnen. Nach (22.18) ist $\mathbf{x}^+ = \mathbf{B}^+\mathbf{b} = \mathbf{V}\Sigma^+\mathbf{U}^T\mathbf{b}$. Die Lösung mit der SVD gilt als die numerisch stabilste Lösung, ist allerdings auch das aufwendigste Lösungsverfahren.

- Die Grauwerte eines Bildes oder eines Bildblocks lassen sich als Matrix auffassen. Damit kann man Bildblöcke oder ganze Bilder einer SVD unterziehen, aber wozu? Wir nutzen den Approximationssatz. Approximationssätze aus der Mathematik können immer für Datenkompressionen herangezogen werden. Dazu nehmen wir an B sei ein Bild oder Bildblock, dann können wir die Zerlegung (22.48) $\mathbf{B} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ benutzen. Die Tensorprodukte (oder dyadische Produkte) $\mathbf{u}_i \mathbf{v}_i^T$ sind wieder Matrizen, allerdings mit dem Rang eins. Diese können als Basismatrizen oder Basisbilder interpretiert werden. Wir entwickeln also \mathbf{B} in eine Summe nach diesen Basismatrizen. Wenn wir die Summation eher abbrechen, dann erhalten wir eine Approximation von B und der Approximationssatz gibt uns über die singulären Werte auch noch Auskunft über die Güte der Approximation. Wenn die Matrix \mathbf{B} die Dimension (n, n) hat, also n^2 Elemente enthält, dann sollte unsere Approximation weniger als n^2 Elemente enthalten. Ein Summand in der Approximation wird dann aus $2n$ Elementen und einem Singulärwert gebildet, also $2n + 1$ Werten. Für den Abbruchindex k in der Approximation sollte dann $k(2n + 1) < n^2$ gelten, als grobe Regel kann man $k < n/2$ ansetzen. In Abb. 22.1 ist die Approximationsgüte der SVD demonstriert.
- Schnelle Implementierung von LSI-Filters, siehe auch Abschn. 8.1.2
Grundlage dafür ist wieder der Approximationssatz und (22.44), wobei \mathbf{B} die Rolle der Filtermaske übernimmt. Wenn die Summe nur einen Summanden hat, dann ist die Maske separierbar. Liegt mehrfache Separierbarkeit vor (Summe hat mehr als einen Summanden), dann erhöht sich entsprechend der Aufwand oder man approximiert die



Abb. 22.1 SVD, 256×256 Bild Lenna, **a** 10 Basisbilder, komprimiert auf 8 %, **b** 30 Basisbilder, komprimiert auf 22 %, **c** 70 Basisbilder, komprimiert auf 55 %

Summe, indem man Summanden mit „kleinen“ Singulärwerten entsprechend (22.48) weglässt. Nachfolgend sollen die Singulärwerte von Filtermasken angegeben werden, die nicht separierbar sind:

- 3×3 -Laplacefilter

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 4,449 & 0 & 0 \\ 0 & 0,449 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (22.51)$$

- 7×7 -Mexican-Hat

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & -1 & -1 & -1 & 0 & 0 \\ 0 & -1 & -3 & -3 & -3 & -1 & 0 \\ -1 & -3 & 0 & 7 & 0 & -3 & -1 \\ -1 & -3 & 7 & 24 & 7 & -3 & -1 \\ -1 & -3 & 0 & 7 & 0 & -3 & -1 \\ 0 & -1 & -3 & -3 & -3 & -1 & 0 \\ 0 & 0 & -1 & -1 & -1 & 0 & 0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 28,9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7,7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1,14 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (22.52)$$

22.4 Blockmatrizen

In manchen Anwendungsfällen (z. B. Abschn. 18.5.5) lässt sich eine spezielle Blockstruktur einer Matrix für die Invertierung ausnutzen.

Lemma 22.2 (Invertierung einer Blockmatrix) *Sei eine Matrix \mathbf{M} mit folgender Blockstruktur gegeben:*

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}. \quad (22.53)$$

Die Inverse der Matrix \mathbf{M} kann dann auf die Inverse der Teilmatrix \mathbf{A} und des sogenannten Schur-Komplements $\mathbf{S} = -\mathbf{C}\mathbf{A}^{-1}\mathbf{B} + \mathbf{D}$ zurückgeführt werden:

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{S}^{-1} \end{bmatrix}. \quad (22.54)$$

Beweis Wir betrachten zunächst die folgenden algebraischen Manipulationen:

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \cdot \mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \quad (22.55)$$

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix}. \quad (22.56)$$

Die Matrizen, welche für diese einzelnen Manipulationen verwendet haben, entsprechen genau den Operationen beim Gauß'schen Eliminationsverfahren und führen zu folgender Gleichung:

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \cdot \mathbf{M} \cdot \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix}. \quad (22.57)$$

Die Inversen der Transformation und die diagonale Blockmatrix auf der rechten Seite lassen sich durch ihre speziellen Strukturen einfach berechnen und wir erhalten das Resultat des Lemmas:

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^{-1} \quad (22.58)$$

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \quad (22.59)$$

$$= \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1} \\ \mathbf{0} & \mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \quad (22.60)$$

$$= \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{S}^{-1} \end{bmatrix}. \quad (22.61)$$

Für eine symmetrische Matrix \mathbf{M} mit $\mathbf{C} = \mathbf{B}^T$, können wir das obige Resultat noch vereinfachen und wir erhalten:

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1}\mathbf{B}^T\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1} \\ (-\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1})^T & \mathbf{S}^{-1} \end{bmatrix}. \quad (22.62)$$

□

Lemma 22.3 (Satz von Woodbury) *Für reguläre Matrizen \mathbf{A} und beliebige Matrizen \mathbf{B}, \mathbf{C} und \mathbf{D} passender Größe gilt:*

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}. \quad (22.63)$$

Beweis Um die Aussage des Lemmas zu beweisen, verwenden wir die Grundidee des vorherigen Beweises mit einer anderen Gauß'schen Elimination:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \mathbf{M} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{D}^{-1}\mathbf{C} & \mathbf{0} \end{bmatrix} \mathbf{I} = \begin{bmatrix} \tilde{\mathbf{S}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}. \quad (22.64)$$

Mit der gleichen Schlußfolgerung wie schon bei der Invertierung von Blockmatrizen erhalten wir eine andere Variante des Lemmas mit einem modifizierten Schurkomplement $\tilde{\mathbf{S}} = -\mathbf{B}\mathbf{D}^{-1}\mathbf{C} + \mathbf{A}$ in der oberen linken Ecke der Matrix \mathbf{M}^{-1} . Ein Vergleich mit der obigen Variante der Invertierung von Blockmatrizen liefert direkt die Aussage des Satzes von Woodbury. \square

22.5 Normalverteilungen

Wir werden uns im Folgenden einige Eigenschaften multivariater Normalverteilungen ansehen und bedingte Verteilungen sowie Randverteilungen herleiten. Zunächst betrachten wir aber den einfachen Fall der Summe von zwei normalverteilten Größen:

Lemma 22.4 (Summe von zwei normalverteilten unabhängigen Größen) *Seien zwei unabhängige multivariate normalverteilte Zufallsvariablen $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{A}_1)$ und $\mathbf{x}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \mathbf{A}_2)$ gegeben. Die Summe dieser zwei Variablen ist dann ebenfalls normalverteilt:*

$$\mathbf{x}_1 + \mathbf{x}_2 \sim \mathcal{N}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \mathbf{A}_1 + \mathbf{A}_2). \quad (22.65)$$

Beweis Die Verteilung der Summe von zwei unabhängigen Zufallsvariablen lässt sich mittels der Faltung bestimmen. In Abschn. 2.1.1 und dort besonders im Beispiel 1 haben wir diesen Zusammenhang bereits schon gesehen und auch die entsprechende Herleitung für die Normalverteilung vorgenommen (siehe (2.12)). \square

Die bedingte Verteilung spielt bei der Herleitung der Gaußprozess-Regression eine ganz entscheidende Rolle (Abschn. 18.5.5):

Lemma 22.5 (Bedingte Normalverteilungen) *Sei $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_d)^T$ eine multivariate Normalverteilung, d. h. $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{M})$, mit dem folgenden Mittelwertvektor und Kovarianzmatrix:*

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_d \end{bmatrix} \mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{D} \end{bmatrix}. \quad (22.66)$$

Die bedingte Verteilung $p(\mathbf{x}_a | \mathbf{x}_d)$ ist ebenfalls eine Normalverteilung mit Mittelwertvektor $\boldsymbol{\mu}'$ und Kovarianzmatrix \mathbf{M}' :

$$\boldsymbol{\mu}' = \boldsymbol{\mu}_a + \mathbf{B}^T \mathbf{D}^{-1} (\mathbf{x}_d - \boldsymbol{\mu}_d) \mathbf{M}' = \mathbf{A} - \mathbf{B}^T \mathbf{D}^{-1} \mathbf{B}. \quad (22.67)$$

Beweis Ein Beweis findet sich in [3, Abschn. 2.3.2, S. 89]. \square

Lemma 22.6 (Randverteilung einer Normalverteilung) Seien zwei normalverteilte Zufallsvariablen \mathbf{x}_a und \mathbf{x}_d gegeben und folgendermaßen verteilt:

$$\mathbf{x}_a \sim \mathcal{N}(\boldsymbol{\mu}_a, \mathbf{A}) \quad (22.68)$$

$$\mathbf{x}_d | \mathbf{x}_a \sim \mathcal{N}(\mathbf{F}\mathbf{x}_a + \mathbf{b}, \mathbf{G}). \quad (22.69)$$

Die Randverteilung \mathbf{x}_d kann dann wie folgt angegeben werden:

$$\mathbf{x}_d \sim \mathcal{N}(\mathbf{F}\boldsymbol{\mu}_a + \mathbf{b}, \mathbf{G} + \mathbf{F}\mathbf{A}\mathbf{F}^T). \quad (22.70)$$

Beweis Ein Beweis findet sich in [3, Abschn. 2.3.3, S. 90–93]. \square

Literatur

1. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. *IEEE Trans. Comput.* **23**(1), 90–93 (1974)
2. Beyerer, J., Leon, F.P., Frese, C.: *Automatische Sichtprüfung*. Springer (2012)
3. Bishop, C.M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1. Aufl. Springer (2006)
4. Blake, A., Kohli, P., Rother, C.: *Markov Random Fields for Vision and Image Processing*. MIT Press (2011)
5. Bogert, B.P., Healy, M.J.R., Tukey, J.W.: The frequency analysis of time series for echoes: Cepstrum, pseudo-austovariance, cross-cepstrum and saphe cracking. In: *Proc. Sym. of Time Series Analysis*, S. 209–243 (1963)
6. Boukerroui, D., Noble, J.A., Brady, M.: On the choice of band-pass quadrature filters. *Journal of Mathematical Imaging and Vision* **21**(1), 53–80 (2004)
7. Boyd, S., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press, New York, NY, USA (2004)
8. Breiman, L.: Bagging predictors. *Machine Learning* **24**(2), 123–140 (1996)
9. Breiman, L.: Random forests. *Machine Learning* **45**(1), 5–32 (2001)
10. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: *Classification and Regression Trees*, 1. Aufl. Chapman and Hall/CRC (1984)
11. Brodatz, P.: *A Photographic Album for Artists and Designers*. Dover Publishing (1966)
12. Butz, T.: *Fouriertransformation für Fußgänger*. Teubner (2003)
13. Cathey, W.T., Dowski, E.R.: New paradigm for imaging systems. *Appl. Optics* **41**, 6080–6092 (2002)
14. Chaudhuri, S., Rajagopalan, A.N.: Depth from defocus – a real aperture imaging approach. Springer (1999)
15. Coates, A., Ng, A.Y.: The importance of encoding versus training with sparse coding and vector quantization. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, S. 921–928 (2011)
16. Costa, L., Cesar, J.: *Shape Analysis and Classification*. CRC Press (2001)
17. Nister, D.: An efficient solution to the five-point relative pose problem. *PAMI* **26**(6), 756–770 (2004)

18. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), S. 886–893 (2005)
19. Denzler, J.: Probabilistische Zustandsschätzung und Aktionsauswahl im Rechnersehen. Logos Verlag Berlin (2003)
20. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **6**(99), 1030–1037 (2011)
21. Dudgeon, D.E., Mersereau, R.M.: Multidimensional Digital Signal Processing. Prentice-Hall (1984)
22. Faugeras, O.: Three-Dimensional Computer Vision, 1. Aufl. MIT Press (1993)
23. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(9), 1627–1645 (2010)
24. Fenimore, E., Cannon, T.M.: Coded aperture imaging with uniformly redundant rays. *Appl. Optics* **17**, 337 (1978)
25. Frischholz, R.W., Spinnler, K.P.: Class of algorithms for realtime subpixel registration. In: Proceedings of the Europto-Conference, S. 1–10 (1993)
26. Fröhlich, B., Rodner, E., Denzler, J.: A fast approach for pixelwise labeling of facade images. In: Proceedings of the International Conference on Pattern Recognition (ICPR 2010), S. 3029–3032 (2010)
27. Fröhlich, B., Rodner, E., Denzler, J.: Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In: Asian Conference on Computer Vision (ACCV), S. 218–231 (2012)
28. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. *Maching Learning* **63**(1), 3–42 (2006)
29. Gonzales, R.C., Woods, R.E.: Digital Image Processing. Pearson (2008)
30. Haralick, R.M., Shapiro, L.G.: Computer and robot vision, volume 1. Addison-Wesley (1992)
31. Ho, T.K.: The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(8), 832–844 (1998)
32. -K. Hu, M.: Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on* **8**, 179–187 (1962)
33. Kadir, T., Brady, M.: Scale, saliency and image description. *Int. Journal of Computer Vision* **45**, 83–105 (2001)
34. Kang, H.R.: COLOR Technology for Electronic Imaging Devices. SPIE-Press (1997)
35. Klette, R., Rosenfeld, A.: Digital Geometry. Elsevier (2004)
36. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances of Neural Information Processing Systems S. 1097–1105 (2012)
37. Kuglin, C.D., Hines, D.C.: The phase correlation image alignment method. *Proceedings of the IEEE on Cybernetics and Society*, S. 163–165 (1975)
38. Kuncheva, L.I., Rodríguez, J.J.: An experimental study on rotation forest ensembles. In: Multiple Classifier Systems, S. 459–468 (2007)
39. Lagendijk, R.L., Biemond, J.: Iterative Indentification and Restoration of Images. Kluwer Academic Publishers (1991)

40. Lezoray, O., Grady, L.: *Image Processing and Analysis with Graphs*, 1. Aufl. CRC Press (2012)
41. Lindeberg, T.: *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers (1994)
42. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* **60**(2), 91–110 (2004)
43. Lucy, L.B.: An iterative technique for the rectification of observed distributions. *The Astronomical Journal* **74**, 745–754 (1974)
44. Lutzke, P., Schaffer, M., Kühmstedt, P., Kowarschik, R., Notni, G.: Experimental comparison of phase-shifting fringe projection and statistical pattern projection for active triangulation systems. In: *Proceedings of the Conference: Optical Measurement Systems for Industrial Inspection*, SPIE 8788 (2013)
45. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: *Proceedings of the British Machine Vision Conference (BMVC 2002)*, S. 384–393 (2002)
46. Mercer, J.: Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London* **209**, 415–446 (1909)
47. Minka, T.P.: A family of algorithms for approximate bayesian inference. PhD thesis, Massachusetts Institute of Technology (2001). AAI0803033
48. Mukundan, R., Ramakrishnan, K.R.: Fast computation of Legendre and Zernike moments. *Pattern Recognition* (1995)
49. Nayar, S.K., Nakagawa, Y.: Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**, 824–831 (1994)
50. Nickisch, H., E. Rasmussen, C.: Approximations for binary gaussian process classification. *Journal of Machine Learning Research* **9**, 2035–2078 (2008)
51. Nischwitz, A., Haberäcker, P.: *Masterkurs Computergrafik und Bildverarbeitung*. Vieweg (2004)
52. Ojala, T., Pietikainen, M., Harw, D.: Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In: *Proceedings of the International Conference on Pattern Recognition*, S. 582–585 (1994)
53. Oppenheim, A.V., Schafer, R.W., Buck, J.W.: *Zeitdiskrete Signalverarbeitung*, 2. Aufl. Pearson (2004)
54. Park, S.C., Park, M.K., Kang, M.G.: Super-resolution image reconstruction: A technical overview. *IEEE Signal Processing Magazine* **20**, 21–36 (2003)
55. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: *Advances in Large Margin Classifiers*, S. 61–74. MIT Press (1999)
56. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, P.B.: *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3. Aufl. Cambridge University Press, New York, NY, USA (2007)
57. Rasmussen, C.E., Williams, C.K.I.: *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press (2005)
58. Richardson, W.H.: Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America* . **62**, 55–59 (1972)
59. Rodner, E., Süße, H., Ortmann, W., Denzler, J.: Difference of boxes filters revisited: Shadow suppression and efficient character segmentation. In: *Proceedings of the Eighth IAPR Workshop on Document Analysis Systems*, S. 263–269 (2008)

60. Rosenfeld, A., Thurston, M.: Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers* **20**, 562–569 (1971)
61. Rosin, P.L.: Measuring shape: ellipticity, rectangularity, and triangularity. *Machine Vision and Applications* **14**, 172–184 (2003)
62. Rothe, I., Suesse, H., Voss, K.: The method of normalization to determine invariants, Bd. 18, S. 2067–2068 (1996)
63. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
64. Sedgewick, R.: Algorithmen, 2. Aufl. Pearson (2002)
65. Seeger, M.: Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations. PhD thesis, University of Edinburgh (2003)
66. Shepp, L.A., Logan, B.F.: The fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science* **21**(3), 21–43 (1974)
67. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: Computer vision and pattern recognition (CVPR), S. 1–8 (2008)
68. Sniedovich, M.: Dijkstra's algorithm revisited: The dynamic programming connexion. *Control and Cybernetics* **35**(3), 599–620 (2006)
69. Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. *Journal of Machine Learning Research* **7**, 1531–1565 (2006)
70. Stegmann, M.B., Ersbøll, B.K., Larsen, R.: FAME - a flexible appearance modelling environment. *IEEE Transactions on Medical Imaging* **22**(10), 1319–1331 (2003)
71. Suesse, H., Ditrich, F.: Robust determination of rotation-angles for closed regions using moments. In: Proceedings of the International Conference on Image Processing (ICIP 2005), S. 337–340 (2005)
72. Suesse, H., Ortmann, W.: A new efficient algorithm for fitting of rectangles and squares. In: Proceedings of the International Conference on Image Processing (ICIP 2001), S. 809–812 (2001)
73. Suesse, H., Ortmann, W.: Robust matching of affinely transformed objects. In: Proceedings of the International Conference on Image Processing (ICIP 2003), S. 383–386 (2003)
74. Suesse, H., Ortmann, W., Lautenschläger, J., Lautenschläger, C., Körner, M., Grosskreutz, J., Denzler, J.: Quantitative analysis of pathological mitochondrial morphology in neuronal cells in confocal laser scanning microscopy images. *Proceedings IWBBIO* (2014)
75. Takeda, M., Mutch, K.: Fourier transform profilometry for the automatic measurement of 3-d object shape. *Appl. optics* **22**(24), 3977–3982 (1983)
76. Trummer, M., Suesse, H., Denzler, J.: Coarse registration of 3d surface triangulations based on moment invariants with applications to object alignment and identification. In: Proceedings of IEEE 12th International Conference on Computer Vision (ICCV 2009), S. 1273–1279 (2009)
77. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: Computer Vision and Pattern Recognition (CVPR), S. 586–591 (1991)
78. Vanderwalle, P., Süsstrunk, S., Vetterli, M.: A frequency domain approach to registration of aliased images with application to super-resolution. In: EURASIP Journal on Applied Signal Processing, S. 1–14 (2006)
79. Vapnik, V.N.: The nature of statistical learning theory. Springer (2000)

80. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **34**(3), 480–492 (2012)
81. Viola, P., Jones, M.: Robust real-time object detection. *International Journal of Computer Vision* **57**(2), 137–154 (2004)
82. Vondrick, C., Khosla, A., Malisiewicz, T., Torralba, A.: HOGgles: Visualizing Object Detection Features. *Proceedings of IEEE 16th International Conference on Computer Vision (ICCV 2013)*
83. Voss, K., Suesse, H.: Invariant fitting of planar objects by primitives, Bd. 19, S. 80–84 (1997)
84. Voss, K., Suesse, H.: A new one-parametric fitting method for planar objects, Bd. 21, S. 2067–2068 (1999)
85. Voss, K., Suesse, H.: Affine point pattern matching. In: *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, S. 155–162. Springer, London, UK (2001)
86. Voss, K., Süsse, H.: Adaptive Modelle und Invarianten für zweidimensionale Bilder, 1. Aufl. Shaker (1995)
87. Wahl, F.M.: Coded light approach for 3-dimensional (3d) vision. Technical Report RZ 1452, IBM (1984)
88. Weickert, J.: Anisotropic Diffusion in Image Processing. B.G. Teubner (1998)
89. Winkler, G.: Image Analysis, Random Fields and Dynamic Monte Carlo Methods. Springer (1991)
90. Wolpert, D.H.: The supervised learning no-free-lunch theorems. In: *Soft Computing and Industry*, S. 25–42. Springer (2002)

Sachverzeichnis

- 3D-Bildverarbeitung
E-Matrix, 328
Epipolargeometrie, 322
Fundamentalmatrix, 331
Kalibrierung, 339
Kruppa Matrix, 350
Projektionsmatrix, 324, 356
- A**
Absolute Conic, 311
Abstände, 542
Abtastfunktion, 36, 90
Abtasttheorem, 3, 90
 - Aliasing, 5Affine Kamera, 296
Affine Transformation, 547
Albedo, 401
algebraische Distanz, 624
Approximation
 - Kontur, 99
 - trigonometrische, 98Autofaltung, 33
Autokorrelation, 33, 47, 48, 80
Autoregressiver Prozess, 256
average recognition rate, 506
- B**
back propagation, 596
Bagging, 460, 468
Bag-of-Features, 577, 591
Bag-of-Words, 577, 591
baseline, 322
Basis
 - affine, 102, 262, 313
- projektive, 263, 266
Standardbasis, 267
Basis eines Raumes, 54
Basisklassifikatoren, 466
Bayes-Ansätze, 458
Bayes-Schätzwert, 459
Beleuchtung, 158, 398
Bewegung einer Kamera, 310
bias-variance, 454
Bildhauptpunkt, 307
Bildklassifikation, 589, 591
Bildmodell, 17
Bildregistrierung, 542, 547
Bildrestaurierung
 - Bewegungsunschärfe, 203
 - Inversfilter, 196
 - Pseudoinversfilter, 199
 - Richardson-Lucy, 202
 - unter Zwang, 196
 - Wienerfilter, 199Bildverarbeitung
 - Punktinvarianten, 312Biofilme, 181, 221
bipartiter Graph, 564
Blickebene, 309
Blockmatching, 96, 549, 558
Blockzirkularmatrix, 40
Bogenlänge, 109
Bündelausgleichung, 344
Boosting, 470, 602
Bootstrapping, 600
bounding box, 538
- C**
CAMSIZER, 403

- canonical frame, 522
 Cauchy-Schwarzsche Ungleichung, 544
 CCD-Sensor, 5
 CCIR-Norm, 5
 Chi-Quadrat-Kern, 483
 Cholesky-Verfahren, 465
 Clique, 9
 Closing, 178
 CLSM, 8
 CMOS-Sensor, 5
 coded aperture, 35, 209
 Codierter Lichtansatz, 384
 complex spectrogram, 148
 computational photography, 209
 Computertomographie, 240
 conjugate prior, 459
 Cooccurrence-Matrix, 190
 cross validation, 504
- D**
 Datenkompression, 583
 DC-Komponente, 72
 decision tree, 466
 deep learning, 596
 Deinterlacing, 5
 Delaunay-Graph, 218, 231
 Delaunay-Triangulation, 14, 218, 231
 Deskriptor, 571
 Detektor, 572
 Diffusionsprozess, 30
 Digitalisierung, 3
 Dijkstra-Algorithmus, 226
 Dilatation, 176
 Dirac-Funktion, 34, 88
 Dirac-Kamm, 89
 Dirichletscher Integralkern, 92, 155
 Diskriminative Klassifikatoren, 452
 Disparität, 322
 Doppelkegel, 238
 Duale quadratische Form, 269
 Duale Quaternionen, 289
 Duale zahlen, 288
 Duales Optimierungsproblem, 486
 Dynamische Programmierung, 223, 570
- E**
 Earth Mover's Distance (EMD), 251, 545, 546, 595
 Eckendetektion, 173, 174
 eigenfaces, 585
 Eigenspace-Methode, 585
 Eigenwertproblem, 171, 619
 Einheitsimpuls, 34, 79, 88
 Ellipse, 630
 - Elliptizität, 100
 - Fitting, 100, 524
 - Streuungsellipse, 517
 - Trägheitsellipse, 517
 EM-Algorithmus, 511
 Energiefilter, 153
 Energiemodell, 118
 Ensemble, 460
 Ensemble-Klassifikatoren, 460
 Entfaltung, 49
 Entropie
 - differentielle, 186
 - diskrete, 183, 466
 Entscheidungsbäume, 466
 Epipolarlinien, 322
 Erkennungsrate, 505
 Erosion, 176
 Essential Matrix, 331, 356
 Euler Formel, 53
expectation propagation, 459, 499
- F**
 Faktorenanalyse, 585
 Falschpositivrate, 507
 Faltung, 44, 48, 366
 - analoge, 27, 36
 - Diffusionsgleichung, 30
 - diskrete, 27, 606
 - effektive Implementierung, 29, 51
 Einheitsimpuls, 35
 gemischte, 27, 36
 inverse, 35
 Iterationen, 83
 Konditionszahl, 42
 Normalengleichungen, 43
 Polynome, 30
 Stabilität, 42
 - Summe von ZV, 29
 Farbe

- Auge, 243
CIELAB-Raum, 246
CIELUV-Raum, 246
CMY-Modell, 248
Color Naming System, 244
Farbmodelle, 246
Farbtafel, 244
Farbtensor, 445
HSI-Modell, 249
Kanten, 250
RGB-Modell, 246
Spektrum, 242
YIQ-System, 249
YUV-System, 249
- Filter
Ableitungen, 165
Bandpass, 159, 161
Diffusion, 174
DoB, 158, 174
DoG, 37
Gauss, 158
Gradient, 171
homomorphe, 158
Hückel-Operator, 174
Koeffizienten, 165
Krümmung, 171, 173
linear, 45
LoG, 37, 158, 165, 573
LSI, 45, 165
Maximum, 179, 180, 221
Median, 180
Mexican hat, 37
Minimum, 179, 180, 221
Mittelwertfilter, 169
morphologische, 176
nichtlinear, 46, 174
Pruning, 158
RangordnungsfILTER, 180
Relaxationsfilter, 176
scharfstellen, 161
Tiefpass, 74, 123, 155
top hat, 179, 181
Unscharfe Maskierung, 156
- Filtermaske, 51
FIR-Filter, 49
- Fitting
adaptiv, 634, 635
Bookstein-Fit, 632
Dreiecke, 530
- Ebenen, 628
Ellipsen, 99, 624, 632
Geraden, 623, 628
invariant, kovariant, 525, 630
Kovarianz, 632
Kreis, 624
momentenbasiert, 530
Rechtecke, 530, 533
Superellipsen, 530
Trägheitsellipse, 524
Flächenberechnung, 535
Fluchtpunkt, 324
focus stacking, 164
Fokussierung, 161
Formfaktor, 535, 538
Fourier transform profilometry, 393
Fourier-Mellin-Transformation, 66
Fourier-Projection-Slice-Theorem, 363
Fourierreihe, 390
Fourier-Slice-Theorem, 240
Fouriertransformation, 53
Affine Transformation, 116
DFT, 60
Diskrete Kosinus Transformation (DCT), 112
Faltungstheorem, 78
Fourier transform profilometry, 393
Fourierdeskriptoren, 104
Fourierreihe, 57
gefilterte Rückprojektion, 365
Gibbs, 72
Hartley, 84
Integraltransformation, 62
Kohärenz, 80
Leistungsspektrum, 80
Modifizierte DFT, 110
Parsevalsche Gleichung, 77
reelle, 59, 84
Ringing, 73
Slice-Theorem, 365
unendliche DFT, 65
Unschärferelation, 115
Freeman-Kode, 214
Frobenius-Norm, 332, 647
Fundamentalmatrix, 325, 334, 335
- G**
Gaborfilter, 150, 152, 153

Gauß-Funktion, 30, 31, 92
 Gauß-Prozess, 490
 Definition, 491
 Klassifikation, 498
 Regression, 495
 Gaußsche Normalengleichungen, 40, 621, 641, 642, 648
 Generalisierung, 454
 Generative Klassifikatoren, 452
 Geometrische Transformation, 21
 Affine Transformation, 22, 262
 Euklidische Transformation, 21, 262
 Homogene Koordinaten, 261
 Ähnlichkeitstransformation, 22, 262
 Projektive Transformation, 23, 262
 Rotation, 21, 261
 Scherung, 21
 Skalierung, 22
 Translation, 21, 261
 Geradendetektion, 231
 Gesichtsdetektion, 600
 Gibbs-Verteilung, 194
 Gimbal Lock, 286
 Gleicheilteil, 72
 GMM, 510
 GP, 490
 Gradient, 171
 Grauwerttransformation, 24
 Gray-Kode, 385
 Greenscher Integralsatz, 539
 Gruppierung, 509
 überwacht, 594
 zufällig, 594

H

Haarmerkmale, 600, 610
 Hauptpunkttransformation, 312
 Hilbert-Transformation, 67
 Histogrammausgleichung, 19
 Histogrammschnittkern, 484
 HOG, 605
 Homogene Koordinaten, 263
 Homographie, 263, 304, 311
 Hough Transformation
 generalized, 238
 Hough-Transformation
 klassisch, 233
 Hyperparameter, 483, 490, 491, 503, 595

I

ICF, 611
 ICP-Algorithmus, 568
 IIR-Filter, 49
 image retrieval, 547
 ImageNet, 596
 Impulsantwort, 45
 Instanzerkennung, 597
 Integralbilder, 579
 Interlace-Modus, 5
 Interpolation, 36
 bilineare, 23
 Kontur, 97
 trigonometrische, 97
 Invarianten
 affine, 313
 affine shape, 314
 Algebraische Distanz, 321
 Determinantenmethode, 317
 Doppelverhältnis, 315, 327
 Ellipsenpaare, 322
 Flächeninvarianten, 318
 Fourierdeskriptoren, 106
 geometric hashing, 314
 Hu-Invarianten, 527
 Kamerainvarianten, 307, 310
 komplexe, 101
 Momentinvarianten, 527
 Normalisierung, 101
 Nullraum, 314
 Paar von Ellipsen, 322
 Punktmengen, 101
 Teilverhältnis, 315
 Inverse Faltung, 48
 Ising Modell, 194
 Isolinien, 173
 Iteration, 48

J

Jakobimatrix, 171

K

Kalibrierung, 339, 352
 Kamerakalibrierung, 339
 Kameramodelle, 293
 Kameraparameter, 299, 307, 324
 Kanade-Tomasi-Detektor, 174, 445

- Kanatani-Transformtion, 312
Karhunen-Loeve-Transformation, 580
Kerneldichte, 484
Kernelfunktion, 478, 482
 χ^2 , 484, 493
Gaußkern, 483, 493
homogen, 483
Minimumkern, 484, 493
positiv definit, 481
RBF, 483
stationär, 483
Kernel-Trick, 480
Kernmatrix, 481
Klassifikation, 449
Klassifikationsproblemen, 450
Klassifikator
 diskriminativ, 452
 Entscheidungsbaum, 468
 generativ, 452
 GP, 490
 Kaskade, 602
 k -NN, 462
 nichtlinear, 478
 Nächster Nachbar, 462
 Normalverteilung, 464
 Parzen, 484
 SVM, 473
Kleinste Quadrate Methode, 286, 642
 k -Means, 509, 594
Konditionszahl, 43, 83
Kontextmerkmale, 611
Konturfolgeverfahren, 214
Konturglättung, 14, 98, 178
Konvexe Hülle, 12
 limitierte, 12, 218
Konvexität, 11, 219
Korrelation, 31, 80
Korrelationskoeffizient, 517, 544
Kovarianz, 524
Kovarianzmatrix, 39
Kreuzkorrelation, 31
Kreuzleistungsspektrum, 80
Kreuzvalidierung, 504
Kruppa-Gleichungen, 345, 351
Kruppa-Matrix, 311
- L
label regression, 498
- Lacunarity, 255
lacunarity, 255
Lagrange-Funktion, 171
Lambert-Oberfläche, 399
Laplace-Approximation, 459, 499
Leck-Effekt, 147
Leibnizsche Sektorformel, 535
Lernen
 überwacht, 461
 unüberwacht, 508
Levenberg-Marquardt-Methode, 629
Lichtschnittverfahren, 383
lineare Klassifikatoren, 478
Lineare Optimierung, 547, 636
Lineares Filter, 32
Local binary pattern (LBP), 576
Log-Polar-Darstellung, 24, 67
LSI-Operator, 43, 50
LTI-System, 43
- M
MAP-Schätzung, *siehe*
 Maximum-A-Posteriori-Schätzung
Marginalisierung, 458
Markov chain Monte Carlo, 459
Matching
 Abstände, 542
 Kontur, 557
 PCA, 585
 Transformationen, 547
Maximum-A-Posteriori-Schätzung, 457
Maximum-Likelihood-Schätzung, 456
MCMC, *siehe* Markov chain Monte Carlo
Median, 180, 636
Mehrklassen-Klassifikationsproblem, 450
Mercer-Bedingung, 478
Metrik, 542
Minimal aufspannender Baum (MST), 229
Minimumkern, 484
Mischverteilung, 510
ML-Schätzung, *siehe*
 Maximum-Likelihood-Schätzung
Modellkombination, 460
Modellkomplexität, 454
Moire-Technik, 387
Momente
 Affine Invarianten, 529
 Axiale Momente, 518

-
- effektive Berechnung, 539
 Flächenmomente, 516
 Hu-Invarianten, 527
 Komplexe Momente, 518
 Liniennmomente, 517
 Momentinvarianten, 527, 564
 Punktmomente, 518
 Schwerpunkt, 516, 522
 Transformation, 520
 Trägheitsellipse, 524
 Zentrale Momente, 522
- Moore-Penrose Bedingungen, 643
 Motion deblurring, 203
 MRT, 8
 MSER-Methode, 526
 multiple kernel learning, 595
 mutual information, 186, 466
- N**
- Nachbarschaftsrelation
 Achter, 9
 Sechser, 9
 Sechsundzwanziger, 9
 Vierer, 9
- Nodalpunkt, 311
 non-maximum suppression, 605
 Norm, 41
 Normalisierte Koordinaten, 303
 Normalisierung, 522
 Normalverteilung, 464, 490, 651
 Nyquist-Frequenz, 4
- O**
- Objekte
 blobs, 181
 Orientierung, 517
 spots, 181
 Trägheitsellipse, 517
- Objektlokalisation, 589, 598
 OCR, 24, 159
 Überanpassung, 454
 one-vs-all, 477, 502
 one-vs-X, 477, 478
 Opening, 178
 Orientierungen, 172
 Orthogonale Momente, 518
 Orthografische Projektion, 294
- Ortho-Theorem, 324
 overall recognition rate, 506
- P**
- Panoramabilder, 311
 Parallaxe, 322
 Paraperspektive, 296
 Parsevalsche Gleichung, 54
 Parzen, 484
 PCA, 580
 Perkolation, 16
 Phasenkongruenz, 119
 Phasenkorrelation, 550
 Phasenshift-Verfahren, 390
 phase-only transformation, 95
 phase-unwrapping, 389, 394
 Photometrisches Stereo, 402
 Picksche Formel, 538
 pill box, 47
 pillbox, 396
 Plugin-Prinzip, 457
 Point pattern matching, 563, 567
 Point Spread Function (PSF), 46
 Poissonsche Formel, 89
 Poissonverteilung, 202
 Polarkoordinaten, 20
 Pooling, 592
 positiv definit, 481
 Potts Modell, 194
 precision, 508
probit, 498
 Progressive Scan, 5
 Projektionsmatrix, 302, 356, 642
 Prototypen, 509
 Pseudoinverse, 639
 Punktschätzung, 456
 Pyramide
 Gauß, 145, 599
 Laplace, 145
- Q**
- QR-Zerlegung, 356, 622
 Quadratische Form, 269
 Quadratur-Filter, 153
 Quantisierung, 509, 591
 Quaternionen, 273, 280
 query expansion, 598

- R**
- Radon Transformation, 363
 - Radon-Transformation, 239
 - Random Forest, 468
 - RANSAC-Methode, 234
 - Rauschen
 - Datenkompression, 193
 - Dämpfung, 192
 - Energiefunktionen, 195
 - Kanalrauschen, 191
 - Modelle, 190
 - Salz und Pfeffer, 192
 - signal to noise ratio(SNR)*, 191
 - weißes, 190
 - Rauschmodell, 494
 - RDF, 468, 594, 609
 - recall, 508
 - receiver operator characteristics, 507
 - Rechteckfunktion, 155
 - Reflexion, 158, 398
 - Registrierung, 547, 558
 - Regression, 449
 - Regressionsprobleme, 450
 - Regularisierung, 644
 - Regularisierungsparameter, 487
 - Reklassifikation, 504
 - Reklassifikationsfehler, 504
 - Rekonstruktion
 - Affine, 369
 - Aktives Sehen, 383
 - Polyedrische Objekte, 367
 - Projektive, 370
 - Punkte, 376
 - Triangulation, 376
 - Relativer Nachbarschaftsgraph, 230
 - Richtigpositivrate, 507
 - Richtungen, 172
 - Richtungsableitung, 171
 - ROC-Kurve, 507
 - Rückprojektion
 - gefilterte, 240, 365
 - Rotationen
 - 3D, 269
 - Drehachse, 271
 - Drehmatrix, 269
 - Eulersche Winkel, 269
 - Exponentielle Form, 279
 - Kanonische Form, 283
 - Quaternionen, 273
- S**
- Registrierung, 555
 - Rodrigues, 278
 - Rundheit, 535
 - Schätzer
 - Bayes, 494
 - Bayes-Ansatz, 458
 - LAD, 617
 - LAD-Methode, 636
 - LSE-Methode, 620
 - MAP-Methode, 456
 - ML-Methode, 456
 - MMSE-Methode, 459
 - MSE, 617
 - stochastische, 456
 - Schur-Komplements, 650
 - Schwache Perspektive, 295, 352
 - Schwerpunkt, 516
 - Segmentierung
 - Otsu-thresholding, 213
 - Regionen, 608
 - seeded region growing, 220
 - semantisch, 607
 - Sehstrahl, 309
 - Semantische Segmentierung, 589
 - shading, 181
 - shape from focus/defocus, 395
 - Shi-Tomasi-Detektor, 174
 - Siebanalyse, 403
 - SIFT, 591, 597, 607
 - SIFT-Deskriptor, 572
 - Signaturen, 546
 - Silhouette einer Punktmenge, 12, 16, 218
 - Simplexmethode, 547, 637
 - Sinusoid, 60
 - Skalarprodukte, 53
 - Skalierungs-Kovarianz, 573
 - Skelett, 179
 - sliding window, 599
 - Sliding-Window Klassifikation, 599
 - Sobelfilter, 45
 - space variant filtering, 46
 - spatial pooling, 592
 - Spektralnorm, 647
 - Spiegel, 399
 - Spiegelung, 79
 - Spiegelungsoperator, 32

square wave, 92
stationäres Feld, 39, 201
Stereo-Anordnung, 304
Stereologie, 236
Stützgerade, 235
Strukturtensor, 173, 174, 443, 445
Subpixelgenauigkeit, 555
Superellipsen, 532
Superresolution, 134
support vector machines, 473
SVD, 52, 642, 645
SVM, 473, 486, 500, 605

T

Tensoren, 415
 Dualer Vektorraum, 416
 Farbtensor, 445
 m -Vektoren, 432
 Rang, 433
 Strukturtensor, 443
 Tensoralgebra, 423
 Tensorfelder, 438
 Trifokaler Tensor, 440
 Verjüngung, 425
Textur, 255
 LBP, 577
Thickening, 178
Thinning, 178
Tichonov-Arsenin Regularisierung, 644
Tieffpass
 idealer, 98, 129, 391
 Kontur, 98
time warping, 569
Toeplitz-Matrix, 39
Transferfunktion, 72
Transinformation, 186

Triangulation, 376, 390

U

Umfang, 535
Ungarische Methode, 567
Unsharp Masking, 156
unsupervised, 508

V

Vandermonde-Matrix, 125, 142
Vermessungsaufgaben, 380
Verschiebungsinvarianz, 44, 47
Verschiebungsoperator, 33, 47
Verschiebungstheorem, 87
Vertauschungsmatrix, 506
Verzeichnungen, 357
Viola und Jones, 600
voting, 231

W

wavefront coding, 209
whitening, 35, 94
Whitening Transformation, 584
Wiener-Chintchin-Theorem, 80
Wiener-Lee-Beziehung, 48
Wigner Ville Transformation, 151
Wrap-around effect, 37

Z

Zero padding, 37, 555
Zirkularmatrix, 38
Zylinderhut-Transformation, 179