Signalanalyse-Verfahren zur Segmentierung von Multimediadaten

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Diplom-Wirtschaftsinformatiker Thomas Haenselmann
aus Erlangen

Mannheim, 2004

Dekan:	**Professor Dr. Jürgen Potthoff, Universität Mannheim**
Referent:	**Professor Dr. Wolfgang Effelsberg, Universität Mannheim**
Korreferent:	**Privatdozent Dr. Jürgen Hesser, Universität Mannheim**

Tag der mündlichen Prüfung: 02. März 2004

INHALTSVERZEICHNIS

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Dank an Mitwirkende
Einteilung der Arbeit
Begleitende Software
Formales

Kapitel 1 - Transformationen in der Multimediaanalyse

1.1 Einleitung 13
1.2 Beitrag zur Forschung 15
1.3 Reale Bilder als Punktecluster 16
1.4 Bildtransformation mit einer zufällig gewählten Basis 19
1.5 Segmentierung durch statistische Analyse 20
1.6 Die Peak Signal-to-Noise-Ratio 21

Kapitel 2 - Faktoren- und Hauptkomponentenanalyse

2.1 Motivation der Faktor- und Hauptkomponentenanalyse 25
2.2 Anwendung der Hauptkomponentenanalyse 40

Kapitel 3 - Diskrete Cosinus-Transformation

3.1 Eigenschaften der DCT 61
3.2 Blockweise Darstellung 62
3.3 Bezug zur Fourieranalyse 62
3.4 Das Markov Modell 64
3.5 Eignung der DCT für Analyse und Kompression 66
3.6 Die Vorwärtstransformation 71
3.7 Die inverse Transformation 72
3.8 Quantisierung 73

Kapitel 4 - Wavelet-Transformation und Anwendungen

4.1 Motivation 81
4.2 Wavelet-basierte Signalkodierung 96
4.3 Exkurs: Grundlagen des Volumenraytracing 106

Kapitel 5 - Waveletbasierte Segmentierung

5.1 Herausforderung des Auffindens von Objektgrenzen 111
5.2 Der Verfahren *SemiSeg* 116
5.3 SemiSeg im Probandentest 127
5.4 Problem unscharfer Objektgrenzen 137

Kapitel 6 - Verfeinerung des Suchalgorithmus

6.1 Stärken und Schwächen von SemiSeg 141
6.2 Das Live-Wire Verfahren 141
6.3 Richtungssuche von Live-Wire und Mustererkennung von SemiSeg 154
6.4 Vergleich von Live-Wire und SemiSeg 157

Kapitel 7 - Segmentierung von Volumendaten

7.1 Segmentierung im Dreidimensionalen 159
7.2 Vorarbeiten und Stand der Technik 160
7.3 Problem der 3D-Segmentierung mit Live-Wire 163
7.4 Erzeugung der Oberfläche 164

7.5 Das generalisierte baryzentrische Koordinatensystem 170
7.6 Kostenmaß für 3D-LiveWire 173
7.7 Waveletanalyse im Raum 174

Kapitel 8 - Vergleich verschiedener Transformationen

8.1 Aufbau des Vergleichsverfahrens 177
8.2 Der Testalgorithmus 178
8.3 Versuch einer repräsentativen Auswahl 179
8.4 Untersuchte Transformationen 180
8.5 Evaluation und Interpretation 183

Kapitel 9 - Fazit und Ausblick

9.1Fazit 187
9.2 Ausblick auf Nahziele 189
9.3 Der größere Bogen 190

Anhang - Testmethode zum Vergleich von Segmentierungsverfahren

A.1 Der Faktor *Benutzer* in der Evaluation 193
A.2 Die Referenzsegmentierung 195
A.3 Problem der optimalen Verteilung der Stützstellen 195
A.4 Dynamische Programmierung zur Suche des globalen Optimums 198
A.5 Erzeugen der manuellen Segmentierung 199
A.6 Abstand zweier Kurven 200
A.7 Suche der optimalen Knotenmenge 208

ABBILDUNGSVERZEICHNIS

Abbildung 1.1: Frühe Grafikdarstellung aus dem Artikel *The Representation and Matching of Pictorial Structures*, IEEE Transactions on Computers, 1973 14

Abbildung 1.2: Originales Photo im Ortsraum (links). Transformiertes, quantisiertes und rücktransformiertes Photo (rechts) 20

Abbildung 1.3: Original Abbildung (links oben), künstlich generiertes Rauschen mit PSNR von 8,4 (rechts oben), verrauschtes Original mit PSNR von 8,4 (links unten), diffusionsgefilterte Version des linken unteren Bildes (rechts unten) 23

Abbildung 1.4: Das Bild wurde so erstellt, dass zum Original aus 1.3 (links oben) eine minimale PSNR von ca. 2,8 entsteht. 23

Abbildung 2.1: Ein Merkmalsvektor 2 wird auf einen anderen Merkmalsvektor 1 projiziert. Ist der Winkel dabei null, so zeigen beide in die gleiche Richtung bzw. erklären den gleichen Sachverhalt. Daher ist der entsprechende Wert in der Korrelationsmatrix cos(0°) = 1. 34

Abbildung 2.2: Die Beziehungen der fünf Merkmale aus dem Beispiel wurden hier in der Ebene verdeutlicht. Zeigen zwei Vektoren in die gleiche Richtung, so geben sie auch die gleiche Information wieder. 36

Abbildung 2.3: Eine Wolke von Punkten wurde in der Ebene abgetragen. Die beiden Geraden zeigen die Hauptkomponenten der Verteilung an. Offensichtlich zeigt die erste Komponente den wichtigeren Aspekt der ovalen Verteilung an. 37

Abbildung 2.4: Zuerst wird die erste Hauptachse bestimmt (oben). In den folgenden Schritten sollen weitere Achsen ermittelt werden. Diese können nur noch in dem Unterraum liegen, der zur ersten Achse orthogonal ist (unten). In diesem Unterraum findet die weitere Suche nach den verbleibenden Vorzugsrichtungen der Messpunkte statt. 40

Abbildung 2.5: Resynthese nach [HAE02] von Text (link) und Haarstruktur (rechts). Die keinen schwarzen Blöcke werden zur Analyse verwendet. Daneben wurde auf Grundlage dieser Analyse neue Textur synthetisiert. 43

Abbildung 2.6: Die Bereiche zwischen den weißen Häkchen wurden resynthetisiert. Die auf der Hauptkomponentenanalyse basierende Resynthese funktioniert am besten auf ungeordneten Strukturen (links), wogegen in der stark ausgerichteten Struktur des Strohs Blockartefakte deutlicher hervortreten (rechts). 45

Abbildung 2.7: 256 Eigenwerte wurden ihrer Größe nach sortiert abgetragen. Typischerweise sind nur etwa 20%-25% von Null verschieden. 46

Abbildung 2.8: Alle Bilder zeigen das originale Photo. Nur der Bereich zwischen den kleinen weißen Haken wurde mit einer unterschiedlich großen Anzahl von Koeffizienten (bzw. Zufallszahlen) resynthetisiert. 47

Abbildung 2.9: Holztextur, die die in Abbildung 2.10 gezeigte Verteilung der Koeffizienten verursacht. 49

Abbildung 2.10a: zeigt die grafische Darstellung der Verteilung eines jeden Koeffizienten von (0,0) bis (7,7) für die Texturanalyse aus Abbildung 2.9. 52

Abbildung 2.10b: Barcode als Texturblock interpretiert. 53

Abbildung 2.11: Verteilung der Koeffizienten des Tabellenbildes aus Abbildung 2.5. Offensichtlich passen sich die Koeffizienten der Gaussverteilung hier nicht gut an. 55

Abbildung 2.12: Das linke Bild zeigt das originale Photo. Der zu resynthetisierende Teil wurde mit Kästchen der Größe 16x16 markiert. Im rechten Teil wurde der gleiche Bereich durch Zufallszahlen resynthetisiert. 56

Abbildung 2.13: Das fett gedruckte Raster soll synthetisierte Texturblöcke veranschaulichen. Die dünnen, überlappenden Blöcke werden zur Interpolation verwendet, um Blockartefakte zu unterdrücken. 57

Abbildung 3.1: Jean Baptiste Joseph Fourier (1768-1830) 64

Abbildung 3.2: Für jeden Punkt des (hier dreidimensionalen) Raumes ist in einem Markov Modell eine eigene Wahrscheinlichkeitsverteilung definiert. 65

Abbildung 3.3: Die vier ersten tieffrequenten Basisvektoren für die Korrelationsmatrix 3.3 für p=0.2 bei einer Auflösung von 16 Samplepunkten. 67

Abbildung 3.4: Vollständige Basis der DCT für die Transformation von 8x8 Blöcken. Jedes der 64 Kästchen entspricht einem Basisvektor. Helle Bereiche entsprechen positiven Werten, das mittlere Grau entspricht 0 und dunkle Schattierungen repräsentieren Werte kleiner Null. 68

Abbildung 3.5: Die Basis der Sinustransformation. 70

Abbildung 3.6: Abgebildet wurden die ersten vier Basisvektoren der DCT mit 16 Samplepunkten. 71

Abbildung 3.7: Die fünf bezeichneten AC-Koeffizienten werden in der AC-Prädiktion vorhergesagt. 75

Abbildung 3.8: Zielblock mit benachbarten Bildblöcken. Die Korrektur des quantisierten Zielblockes erfolgt so, dass ein möglichst glatter Übergang zu den Nachbarn gewährleistet ist. 76

Abbildung 3.9: Die Gruppe von neun Blöcken wird durch ein Polynom approximiert (zur Verdeutlichung nach oben verschoben). Dieses Polynom soll möglichst stetig mit der Bildinformation an den Grenzen zwischen dem mittleren Block und seinen Nachbarn abschließen. 78

Abbildung 3.10: Quantisierung durch Abrunden. 80

Abbildung 4.1: Signal im Ortsraum (oben) und die Hartley-Transformierte im Frequenzraum unten. 85

Abbildung 4.2: Im oberen Teil (Ortsraum) wurde eine ganze Schwingung erzeugt. Im unten dargestellten Frequenzraum wird deutlich, dass es sich tatsächlich nur um eine einzelne Schwingung handelt. 86

Abbildung 4.3: Die Schwingung aus Abbildung 4.2 wurde innerhalb des Fensters um eine halbe Phase verkürzt, so dass zwischen Anfang und Ende ein Sprung entsteht. Durch die Verlängerung ist ein vollkommen anderes Frequenzspektrum entstanden. 87

Abbildung 4.4: Bei der WT wird ein Signal auf unterschiedlichen Skalen mit Fenstern unterschiedlicher Größe abgetastet. Das Fenster für die Abtastung passt sich der Skala in der Weise an, dass die Schwingung jeweils genau einmal hineinpasst. (Hier wurde der Sinus als Schwingung nur exemplarisch verwendet, um die vollständige Phase deutlich zu machen.) 88

Abbildung 4.5: Im Ortsraum (oberer Teil) sieht man das *Mexican Hat* Wavelet. Im unteren Teil sind die beteiligten Frequenzen zu sehen. Der Mexican Hat enthält dabei wenige große tieffrequente Anteile, die hochfrequenten sind nur marginal ausgeprägt. Damit ist klar, dass dieser Filter ein schmales, tieffrequentes Band herausfiltert, im Frequenzraum also schön lokalisiert ist. 89

Abbildung 4.6: Das Haar Wavelet im Ortsraum (oben) und die Frequenzzerlegung (unten). 92

Abbildung 4.7: Bei der kontinuierlichen WT (links) wird das kontinuierlich skalierte Wavelet wiederum selbst kontinuierlich über das Signal verschoben. In der dyadischen Version (rechts) wird dagegen jeweils um eine ganze Fensterbreite verschoben und um den Faktor 2^{-n} skaliert. 95

Abbildung 4.8: Das Mexican Hat Wavelet wurde mit unterschiedlichen Skalierungen (Z-Achse) über ein Rechtecksignal verschoben (X-Achse). Der Koeffizient wurde in die Höhe (Y-Achse) abgetragen. 96

Abbildung 4.9: DCT komprimiertes Bild bei einer Datenrate von 0.15 Bits/Pixel (links oben), JPEG 2000 kodiertes Bild bei 0.15 Bits/Pixel (rechts oben), DCT kodiertes Bild bei gleicher PSNR wie waveletkodiertes Bild (links unten), originales Bild (rechts unten) 98

Abbildung 4.12: Standard Beispiel der Bildverarbeitung *Lena* (links). Wavelet-zerlegte Version (rechts).Die Skalen sind von der feinsten zur gröbsten Struktur durchnummeriert. In jeder Stufe, von der Gröbsten zur Feinsten, kann das Bild durch die zusätzlichen Koeffizienten verbessert werden. Ein analoges Vorgehen ist auch für die Bewegungskompensation möglich. 101

Abbildung 4.13: Beispiel einer kurzen Videosequenz, die im Raum als Würfel veranschaulicht wird. Auf der Oberfläche des Würfels ist das erste Bild zu sehen, von allen anderen Bildern lediglich die jeweils erste Zeile am unteren Rand und die letzte Spalte am rechten Rand. Die Zeitachse zeigt somit nach unten. Die verwischte Struktur an den senkrechten Würfelflächen läßt vermuten, dass in der Sequenz ein Kameraschwenk erfolgt. 103

Abbildung 4.14: Die kurze Videosequenz aus Abb. 4.13 wurde Wavelet-Transformiert und als transparentes Volumenmodell dargestellt. Im Gegensatz zur Darstellung 4.13, kann man so ins Innere des Würfels sehen. Am offensichtlichsten ist, dass ein Großteil der Information in den dunklen Koeffizienten links unten konzentriert ist. 104

Abbildung 4.15: Ein Strahl wird von einem virtuellen Betrachter in die Szene emittiert. Auf seinem Weg durch den Würfel werden Dichtewerte aufsummiert. 106

Abbildung 4.16: Darstellung des interaktiven Modus von *VolRay* zur Visualisierung von Schnitten durch Volumendaten. 108

Abbildung 4.17: Gleiches Volumen wie in 4.14, jedoch wurden alle Koeffizienten unterhalb eines festen Schwellwertes gelöscht. Der Raum enthält nur noch 1% der betragsmäßig größten Koeffizienten. 109

Abbildung 4.18: Die oberen vier Bilder zeigen eine Sequenz aus dem originalen Video, die unteren vier entstanden aus dem ersten Prozent, der absteigend nach ihrer Wichtigkeit sortierten Koeffizienten. Trotz des Verlustes von 99% der Koeffizienten ist der Inhalt noch gut erkennbar. 110

Abbildung 4.19: Fehlerbild zwischen der oberen und unteren Reihe aus Abbildung 4.19, also die Information, die durch Verwerfen von 99% der betragsmäßig kleinsten Koeffizienten verloren ging. 110

Abbildung 5.1: Originales Photo (links). Die mittlere und rechte Abbildung machen deutlich, dass den Grauwerten alleine nicht zu entnehmen ist, was das zu segmentierende Objekt sein könnte. 113

Abbildung 5.2: Typisches Beispiel für eine unscharf definierte Objektgrenze. Der Analysebereich ist so gewählt, dass dessen Mitte das Bild in Objekt- und Hintergrundbereich aufteilt. 117

Abbildung 5.3: Das vom Benutzer festgelegte Rechteck definiert die Objektgrenze und dient als Initialisierung des SemiSeg Verfahrens. 119

Abbildung 5.4: Am Ende des benutzerdefinierten Rechtecks schließt sich ein Suchrechteck an. Dieses hängt am vorherigen Rechteck fest, hat somit also nur noch die Drehung als verbleibenden Freiheitsgrad. Für unterschiedliche Winkel entsteht so jeweils ein Bildausschnitt. 120

Abbildung 5.5: Der durch das benutzerdefinierte Rechteck gewählte Bildausschnitt muss ebenso wie alle anderen Ausschnitte durch Drehung in ein gemeinsames Koordinatensystem normiert werden. 122

Abbildung 5.6: Das benutzerdefinierte Rechteck wird Spalte für Spalte analysiert. 123

Abbildung 5.7: Beispiel *Africa*. Originales Bild (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts). 130

Abbildung 5.8: Bild *Fashion* (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts) 132

Abbildung 5.9: Bild *Fashion* mit Rauschen (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts) 133

Abbildung 5.10a: Bild *Sea* (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts) 134

Abbildung 5.10b: Bild *Clouds* (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts) 135

Abbildung 5.10c: Bild *Veget* (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts) 135

Abbildung 5.11: Pixel P liegt im halbtransparenten Bereich zwischen Objekt und Hintergrund. In der Umgebung von P (das Rechteck) können typische Objekt- und Hintergrundfarben, jenseits der beiden Segmentierungsgrenzen, ermittelt werden. 138

Abbildung 5.12: Objekt- und Hintergrundfarben bilden im RGB-Raum je eine Punktewolke, die durch ein Ellipsoid angenähert wird. 140

Abbildung 6.1: Das weiße Kreuz markiert den Seed-Point S, das schwarze Kreuz die interaktive Cursorposition. Während der Benutzer die Maus im Bild bewegt, richtet sich die weiße Segmentierungslinie (der Live-Wire) an der stärksten Kante aus. 142

Abbildung 6.2: Dynamische Programmierung. In jeder der N Stufen kann für den Übergang zur nächsten eine Entscheidung getroffen werden. Gesucht sind die n-1 Entscheidungen, die zu einem Pfad minimaler Kosten führen. 144

Abbildung 6.3: Dynamische Programmierung im Ablauf. Die fett gedruckten Pfeile markieren den jew. optimalen Weg eines Knoten n nach (n-1). 146

Abbildung 6.4: Nachbarschaftsbeziehungen von Pixeln. Pixel 1 ist der Seed-Point. Unter allen Kandidaten erzeugt der Übergang zu Pixel 2 die geringsten Kosten. 148

Abbildung 6.5: Live-Wire Algorithmus während der Berechnung des Baumes. Der Seed-Point ist mit einem weissen Kreuz markiert. Die bereits bearbeitete Pixelmenge ist heller dargestellt. Im Ablauf (links oben nach rechts unten) wird deutlich, dass texturreiche Bereiche früher eingenommen werden, als glatte. 149

Abbildung 6.6: Der Seed-Point in der Mitte ist mit einem Sternchen markiert. Diesen erreicht man durch Verfolgen der Pfeile von jedem anderen Punkt des Bildes. Dabei versucht jeder mögliche Weg sich dicht entlang der lokalen Kanten zu bewegen. 151

Abbildung 6.7: Skalenbasierte Mustererkennung von SemiSeg (links) in Verbindung mit der Graphensuche von Live-Wire. Bei der Entscheidung für eine der acht Richtungen (rechts) wird nicht mehr nach der stärksten Kante, sondern der besten Übereinstimmung mit dem Muster (links) gesucht. 156

Abbildung 6.8: In einigen Fällen nimmt Live-Wire aufgrund der globalen Optimierung ungewollte Abkürzungen (oben) wogegen SemiSeg auch einen längeren Weg in Kauf nimmt (unten) 157

Abbildung 7.1: Eine benutzerdefinierte Startlinie setzt sich in Richtung der Seed-Linie fort. Dabei neigt sie dazu in der Mitte ihre Kosten zu minimieren. 161

Abbildung 7.2: Initialer Pfad einer Oberfläche durch dunkle Voxel gekennzeichnet (links). Exemplarisch wurden zwei mögliche angrenzende Pfade durch helle Voxel dargestellt (rechts). 162

Abbildung 7.3: Live-Wire Graph im Raum. In jedem Voxel ist eine Referenz gespeichert, die auf einen von 26 möglichen Nachbarn zeigt. Folgt man den Pfeilen konsequent, so erreicht man von jedem Voxel aus den Seed-Point. Dabei entsteht auf dem Weg durch das Volumen keine Fläche, sondern nur ein Pfad. 164

Abbildung 7.4: Darstellung eines Volumenmodells mit segmentierter Nase. Die benutzerdefinierten Seed-Punkte sind durch weiße Kreuze markiert. Die helle Oberfläche ist das Ergebnis der Segmentierung. 166

Abbildung 7.5: Der graue benutzerdefinierte Pfad wird sukzessiv durch weitere Ketten von Voxeln erweitert, bis die Oberfläche geschlossen ist. 167

Abbildung 7.6: Die durch einzelne Live-Wire verbundenen Seed-Punkte S grenzen die Oberfläche (den Patch) ein. In der Mitte werden gerade die Kosten für Voxel P berechnet. Diese Kosten setzen sich aus den Teilkosten zusammen, die alle Pfade c verursachen. 168

Abbildung 7.7: Eine Oberfläche wird durch fünf Kurven begrenzt (links). Rechts befindet sich der Parameterraum zur Fläche. Gesucht wird nach einer Abbildung, die einem Punkt im gleichseitigen Parameterpolygon einen Punkt auf der Oberfläche zuweist. 170

Abbildung 7.8: Ein Punkt p spannt mit jeder Seite des Polygons eine Fläche α_i auf. 171

Abbildung 7.9: Im zweidimensionalen Fall ist die Richtung der Wavelet-Transformation mit der Objektgrenze eindeutig festgelegt. Im Dreidimensionalen kann das lokale Koordinatensystem dagegen frei um den Gradienten a gedreht werden. Die Oberfläche selbst legt diesen Freiheitsgrad nicht fest. 175

Abbildung 8.1a: Testbilder - Seemann (1), Ente (2), Waschbär (3), Kaffeetasse (4), Photograph (5), Pärchen (6) , Katze (7), Bär (8), Gründerzeit (9), Halloween(10) 181

Abbildung 8.1b: Testbilder Fortsetzung - BBQ (11), Souris (12), Tisch (13), Gelage (14), Fußgänger (15), Golden Gate Bridge (16) , Mama (17), Sessel (18), Matrose (19), Mittag (20) 182

Abbildung 8.2: Kantenbild Kaffeetasse (links) und Waschbär (rechts) 184

Abbildung 9.1: Offensichtliche Trajektorie (oben). Die ebenfalls korrelierte Struktur unten wird erst nach einer Vertauschung jedes zweiten Pixelpaares als ebenso kontinuierlich wahrgenommen. Dabei ist die Vertauschung die Transformation, die die Struktur noch deutlicher hervorhebt. 188

Abbildung A.1: Die weiße Objektgrenze (links) wurde vollständig durch den Benutzer definiert. Die schwarzen Punkte werden durch den Live-Wire Algorithmus verbunden. Das Ergebnis der Segmentierung zwischen den Punkten wird als überlagerte schwarze Linie (rechts) dargestellt. 194

Abbildung A.2: Der unkorrigierte Live-Wire sucht sich entlang des Beines die stärkste Kante als Pfad, nämlich den Übergang von Licht zu Schatten. Tatsächlich liegt die echte Grenze des Objektes aber bereits selbst im Schatten. 196

Abbildung A.3: Im interaktiven Modus von SemiSegDP muss der Benutzer zuerst eine aus menschlicher Sicht optimale Referenzsegmentierung erzeugen. An dieser lassen sich im nachfolgenden Schritt unterschiedliche Verfahren messen. Das Fadenkreuz in der Vergrößerung hilft bei der pixelgenauen Positionierung der Objektgrenze. 199

Abbildung A.4: Beide Bilder entsprechen sich bis auf den kleinen Punkt. Daher kann der Hausdorff Abstand beider Bilder nicht geringer als der Abstand dieser beiden Punkte werden. 200

Abbildung A.5: Der mittlere Abstand der Linie S zu L ist kleiner als der Abstand zwischen L und S. 201

Abbildung A.6: Die stark umrandeten Felder kennzeichnen die Pixel des Objektsilhouette. Jeder Pixel versteht sich nicht als Grauwert, sondern als Cache mit dem Eintrag des kürzesten Abstands zum nächsten Pixel des Umrisses. 202

Abbildung A.7: Bild 1 (links oben) bis Bild 4 (rechts unten) zeigt unterschiedliche Stadien des Segmentierungsverlaufes 203

Abbildung A.8: Links wurden lediglich zwölf Stützstellen um das Objekt herum festgelegt, um Tabelle A.1 übersichtlich zu halten. Im rechten Bild wurde der Live-Wire von jedem Knoten zu jedem anderen eingezeichnet. 208

Abbildung A.9: Suche nach einer optimalen Auswahl von k Knoten aus einer Menge von 12 mit Hilfe der dynamischen Programmierung. 209

Abbildung A.10: Fügt man dem Optimierungsproblem (links) die Nebenbedingung hinzu, dass die Segmentierung einen bereits gefundenen Pfad nicht zurücklaufen darf, so reduziert sich das Problem wesentlich (rechts). 210

Abbildung A.11: Schließt man aus, dass ein Seed-Point nicht zweimal gewählt werden darf, so können die untersten Knoten zu Anfang nicht gewählt werden. Als weitere Nebenbedingung wurde festgelegt, dass der letzte Seed-Point in jedem Fall erreicht werden muss (rechts). Der Graph dünnt sich jeweils weiter aus. 211

TABELLENVERZEICHNIS

Tabelle 2.1: Zur Verdeutlichung der Methodik wurde ein kleines reales Beispiel mit einer überschaubaren Anzahl von Messwerten gewählt. 29

Tabelle 2.2: Die Daten der Befragung wurden bzgl. der Personen aggregiert. 30

Tabelle 2.3: Korrelationsmatrix mit normierten Koeffizienten. Die zwei schwarz umrandeten Felder lassen bereits vermuten, dass die jeweils darin enthaltenen Werte durch nur je einen Faktor bestimmt werden. 32

Tabelle 2.4: Statt den Korrelationskoeffizienten wurden hier in der oberen Dreiecksmatrix die Winkel zwischen den Merkmalen eingetragen. Dies entspricht jeweils dem Arcuscosinus des Koeffizienten, der sich gespiegelt noch einmal in der unteren Dreiecksmatrix findet. 34

Tabelle 5.1: Bewertung der unterschiedlichen Segmentierungsverfahren (1-4) für unterschiedliche Bilder durch sieben Personen (P1-P7). Die Bewertungsskala reicht von 1=schlecht bis 10=sehr gut. 131

Tabelle 5.2: Aggregiertes Ergebnis der Auswertung. Dargestellt ist die Qualität (Spalte *Summe Noten*), der Aufwand (Spalte *Klicks*) und der Quotient aus beiden Werten. 136

Tabelle 8.1: Die 20 Bilder aus Abbildung 8.1 wurden mit fünf verschiedenen Transformationen segmentiert. Die Einträge entsprechen der Anzahl von Seed-Punkten (bzw. benötigten Maus-Klicks) für Segmentierungen vergleichbarer Qualität. 183

Tabelle A.1: Jedes Element der Matrix M stellt die Kosten dar, die der Live-Wire von Knoten i zu Knoten j erzeugt. 207

KAPITEL 1

TRANSFORMATIONEN IN DER MULTIMEDIAANALYSE

1.1 Einleitung

Ein Multimediasystem ist charakterisiert durch die integrierte Erzeugung, Verarbeitung, Speicherung, Darstellung und Übertragung verschiedener Medienströme. Diese dürfen zeitkontinuierlich, aber auch zeitdiskret sein [EFF98, WAT91].

Obwohl die digitale Verarbeitung von Bildern, Videos und Audioströmen schon seit den 70er Jahren wissenschaftlich erforscht wurde, kann man frühe Ansätze [ROS71, FIS73, CHI74] eher als theoretische Trockenübungen ansehen. Dabei wurden Bilder wegen mangelnder Rechenzeit, z. B. für optische Flussberechnungen, nur zeilenweise analysiert und die Ergebnisse erster Segmentierungsversuche auf reinen Textdruckern ausgedruckt, die helle und dunkle Pixel durch unterschiedliche Zeichen annähern mussten (siehe Abbildung 1.1).

Wirklich zeitkontinuierliche Darstellungen, im Verständnis der obigen Definition von Multimedia, waren erst in den 80er Jahren auf sogenannten Grafikworkstations möglich, wie sie unter anderem von der Firma Silicon Graphics produziert wurden. Nur durch den Einsatz massiver Parallelisierung konnten Videos im Format Quarter CIF- (180x144 Pixel) oder CIF (360x288 Pixel) aufgenommen und wiedergegeben werden.

Anfang des dritten Jahrtausends hat sich der Fokus von der reinen Aufnahme und Wiedergabe, nicht nur in der Forschung, sondern auch in der Praxis, in Richtung der Analyse von Multimediadaten verlagert. Denn auch private Urlaubsphotos und Videos werden in absehbarer Zeit nur noch digital aufgenommen werden. Damit der Anwender mit der Flut der Daten zurechtkommt, werden automatische Analysetechniken interessanter. Auch die automatische Überwachung von öffentlichen Räumen - mit allen damit verbundenen ethischen Problemen - sowie der Bereich der Produktionsüberwachung in der Industrie treffen zunehmend auf kommerzielles Interesse.

Getrieben durch die praktischen Anwendungen, konvergieren früher disjunkte Forschungsbereiche wie Bilderverarbeitung, Mustererkennung und Multimediatechnik immer mehr zueinander.

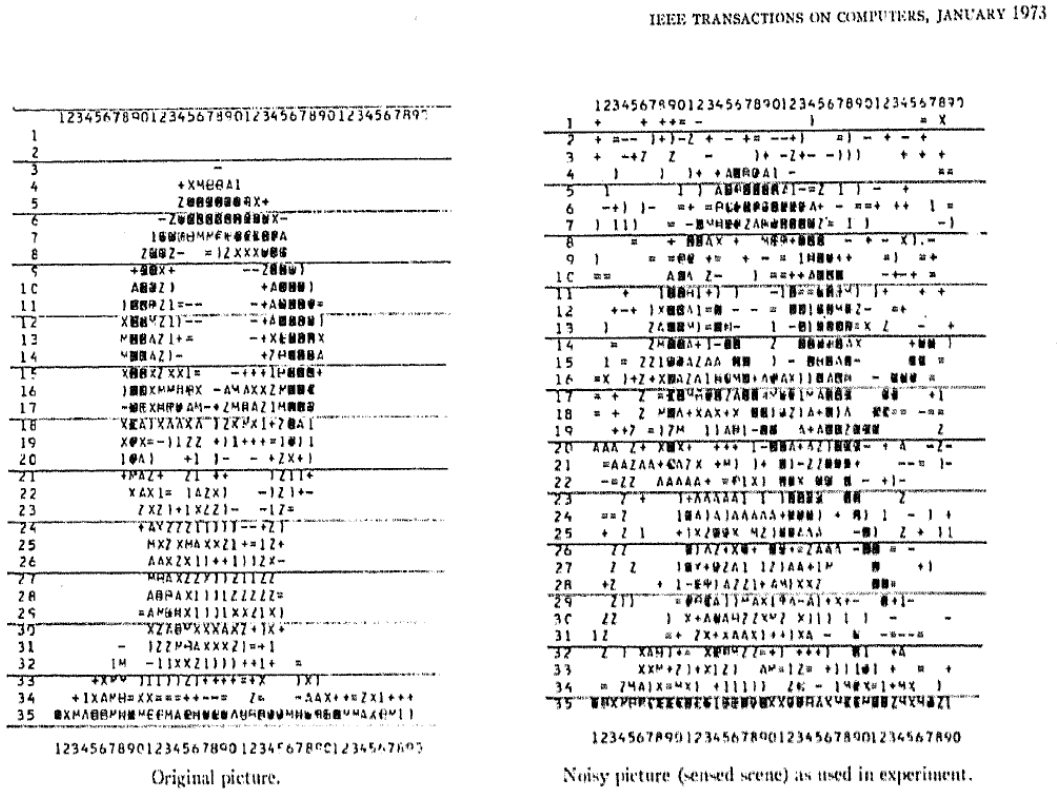

Abbildung 1.1: Frühe Grafikdarstellung aus dem Artikel *The Representation and Matching of Pictorial Structures*, IEEE Transactions on Computers, 1973

Die wachsende Anzahl von Publikationen, Konferenzen und Workshops, die den Begriff *Multimedia Signal Analysis* im Titel enthalten, unterstreicht dies. Ebenso wie die

reine Bildverarbeitung Erkenntnisse aus der Mathematik, Statistik und Physik aufgreift, diffundieren klassische Themen der Bildverarbeitung zunehmend in die Multimediaforschung.

Trotz dieser Vermischung klafft zwischen der multimedialen Signalanalyse und der dazu benötigten mathematischen Statistik immer noch eine Lücke. Diese Erkenntnis war auch der allgemeine Tenor des *21. Heidelberger Bildverarbeitungsforums*, das im März 2003 in Mannheim abgehalten wurde. Eine Erklärung der Podiumsdiskusion war, dass trotz verwandter Inhalte in der Signalanalyse andere Begriffe als in der mathematischen Statisktik verwendet werden. Berechtigterweise finde man in der Signalanalyse statistische Größen wie Mittelwert, Varianz oder Gaussverteilung. Dann folgen aber erst nach einer großen Lücke jüngere Analyseverfahren, wie z. B. die Waveletanalyse.

Die nächsten drei Kapitel konzentrieren sich darauf, die Lücke dazwischen zu schliessen und die erörterte Statistik im Rahmen multimedialer Anwendungen zu interpretieren. Dazu werden aus den Daten die für die Anwendung interessanten Aspekte extrahiert. Letztendlich soll ein unübersichtliches Konglomerat von gesampelten Informationen so gedreht werden, dass die interessanten Aspekte besonders offensichtlich werden. Und in der Tat handelt es sich dabei mathematisch um Drehungen. Dazu wird nur nach einer günstigeren Basis gesucht, die eigentliche Information aber nie verändert. Jedes Kapitel enthält mindestens eine Anwendung der diskutierten Verfahren. Die letzten vier Kapitel konzentrieren sich spezieller auf die Verbesserung der bis heute bekannten halbautomatischen Segmentierungsverfahren.

1.2 Beitrag zur Forschung

Die Arbeit enthält drei wesentliche Ideen, die zum heutigen Zeitpunkt über den bekannten Stand der Forschung hinausgehen.

Zum einen handelt es sich dabei um die in Kapitel 2.2 beschriebene Idee, die Bildkompression dadurch weiter zu verbessern, dass Bilder nicht mehr vollständig explizit gespeichert werden, sondern dass ausgewählte texturierte Bereiche, während der Dekodie-

rung eines Bildes, einfach neu erzeugt werden. Dabei muss darauf geachtet werden, dass der Charaker des Bildes nicht verfremdet wird, es ist aber nicht zwingend nötig, eine pixelweise Übereinstimmung zwischen Original und Rekonstruktion zu erreichen. Vor allem bei Strukturen wie Fell, Sand, Pflanzen oder gleichmäßigen Oberflächenstrukturen stellt man fest, dass der Betrachter dem Detail, also z. B. einem bestimmten Haar, weniger Interesse entgegenbringt als dem Charakter der Struktur an sich, also z. B. einer Ausrichtung oder Körnung.

Der größte Teil der Arbeit beschäftigt sich ab Kapitel 5 mit der Frage, wie Computer Objekte, unter möglichst sparsamer Zuhilfenahme des Benutzers, segmentieren können. Der Kern des wissenschaftlichen Beitrags steckt in der Idee, zum Zweck der Dekorrelation von Bildinformation eine Wavelet-Transformation zu verwenden. Auf den ersten Blick scheint dies ein von der Texturresynthese unabhängiges Thema zu sein. Tatsächlich sind beide aber eng verwandt. Denn bei der Texturresynthese findet eine Bildanalyse, gefolgt von einer Synthese statt. Im Rahmen der Segmentierung werden Bildbereiche nur analysiert, jedoch mit genau dem gleichen Ziel, nämlich der Dekorrelation der Zufallsvariablen, also der Pixel.

Der dritte große Beitrag ist die Ausdehnung der benutzergesteuerten Segmentierung auf Volumendaten in Kapitel 7. Bisher sind in der Literatur hierzu kaum Vorschläge zu finden. Segmentierung findet bisher meist in zweidimensionalen Bildern statt und muss für jedes Folgebild mehr oder weniger wiederholt werden.

1.3 Reale Bilder als Punktecluster

In diesem und den folgenden Kapiteln werden Bilder, die in n Zeilen und m Spalten aufgelöst sind, als einzelne Punkte in einem Vektorraum der Dimension $n \times m$ betrachtet. Um keine übertrieben hohen Dimensionen zu bekommen und die Berechnungen handhabbar zu halten, beschränkt man sich an einigen Stellen praktischerweise darauf, das Bild in kleinere Blöcke aufzuteilen und diese getrennt zu behandeln.

Geht man davon aus, dass Grauwerte im Intervall [0, 255] liegen können, so lässt sich der Raum aller darstellbaren Bilder als achsenparalleler Würfel beschreiben, in dem jeder mögliche Punkt für ein ganzes Bild steht. Aufgrund der Tatsache, dass Grauwerte im Rechner meist byteweise abgelegt sind, sind auch die Seiten des Würfels diskret in 256 Einheiten aufgeteilt (für jedes Pixel ist eine Achse zuständig). Es ergeben sich folglich $256^{n \times m}$ darstellbare monochrome Bilder. Bei einer Auflösung von 512x512 Pixeln erhält man etwa $10^{630.860}$ Möglichkeiten. Interessanterweise sind in der Menge dieser Bilder alle vergangenen und zukünftigen historischen Ereignisse enthalten, ebenso wie alle interessanten Formeln, die die Menschheit nie entdecken wird, und neben den schönsten einseitigen Gedichten die besten Limericks.

Die weitaus größte Gruppe von Bildern beinhaltet allerdings vollkommen zufälliges Rauschen von untereinander unkorrelierten Grauwerten. Ohne dies genauer zu quantifizieren, kann man doch sagen, dass die Gruppe der sinnvollen Bilder nur einen marginalen Bruchteil aller Möglichkeiten ausmacht. Reale Bilder zeichnen sich dadurch aus, dass Grauwerte vor allem lokal stark korreliert sind. Bis auf Kanten und Texturen sind die meisten Pixel also von ähnlichen Grautönen umgeben. Im Raum der darstellbaren Bilder kann man sich die Menge der sinnvollen Bilder oder typischen Photos als eine Punktewolke mit einer bestimmten Form vorstellen. Unterliegen die Pixel z. B. einer Gaussverteilung, so entstehen ovale Formen in der Ebene, eiförmige Gebilde im dreidimensionalen Raum und ähnliche, weniger anschauliche Körper in Räumen höherer Dimension (wie im Fall von Bildern). Es ist die Aufgabe der Datenkompression, nach Kodierungsverfahren zu suchen, die wahrscheinlichen Bildern möglichst kurze Bitmuster zuweisen. Dennoch sollen alle möglichen Bilder, also auch die untypischen, kodierbar sein. Diese können jedoch durch längere Bitfolgen dargestellt werden.

Interessanter als die Kompression ist im Rahmen dieser Arbeit die Analyse der Daten. Gesucht wird dabei nicht nach kurzen Bitmustern, sondern man möchte die Daten (Vektoren) so umformen, dass die für die Anwendung interessante Information in möglichst wenigen Komponenten des Vektors erkennbar wird oder die Komponenten bestimmte Eigenschaften intuitiv erkennen lassen. Im Idealfall trennt eine Transformation die in den Daten enthaltenen Aspekte so genau, dass diese absteigend sortiert nach der Rei-

henfolge ihrer Wichtigkeit in den ersten Komponenten eines Vektor stehen und die letzten Komponenten nur noch unspezifisches Rauschen enthalten. Genau diese Art der Umformung wird in Kapitel zwei beschrieben. Sucht man in Bildern oder Videos nach einer tieferen Semantik, so kann man dies natürlich unmittelbar auf Grundlage der Grauwerte selbst tun [MES89]. Dabei steht man jedoch dem Problem gegenüber, große Datenmengen unmittelbar analysieren zu müssen, obwohl dem einzelnen Pixel nur eine geringe Bedeutung zukommt. Es verteilt sich die Information redundant über eine größere Anzahl von Pixeln, so dass man viele Informationen auch noch aus Bildern der halben Auflösung entnehmen könnte. Offensichtlich wäre es günstiger die interessanten Informationen bereits in wenigen, stark aggregierten Koeffizienten beobachten zu können, wozu man eine sorgfältig gewählte Transformation benötigt. So reichen z. B. für die Gesichtserkennung bereits 20-30 Koeffizienten aus, um ein Gesicht zweifelsfrei zu identifizieren - eine Eigenschaft die den Grauwerten nicht eigen ist. In Kapitel fünf wird die Transformationskodierung dazu verwendet Objektsilhouetten zu erkennen, die man auf Grundlage der Grauwerte nur schwer finden würde und bereits im nächsten Kapitel werden durch Analyse und Erzeugung von Koeffizienten Texturregionen, in beliebiger Größe, generiert.

Für eine gegebene Menge von Bildern wird in beiden Fällen, der Kompression und der Analyse, eine Transformation gesucht, die Abhängigkeiten zwischen Pixeln erkennt und "gewinnbringend" ausnutzt. Unter allen möglichen linearen Transformationen erfüllt die sogenannte *Karhunen-Loève-Transformation* diese Aufgabe am besten. In der Literatur ist sie auch als *Hauptachsenzerlegung* (engl. *Principal Component Analysis*, kurz PCA) bekannt.

Der Begriff der Transformation bedeutet in diesem Zusammenhang, dass eine Drehung im mehrdimensionalen Raum (des Bildes) gesucht wird, die einem Merkmalsvektor bestimmte Eigenschaften verleiht. Auch die Hauptachsenzerlegung aus Kapitel zwei meint nichts anderes: Aus ihrer Sicht kann man sich vorstellen, dass ein Koordinatensystem in einer bestimmten Weise in die bereits erwähnte ovale Punktwolke gelegt wird. Beides, Drehung und Aufbau der Koordinatenachsen, meint den gleichen Sachverhalt.

1.4 Bildtransformation mit einer zufällig gewählten Basis

Das Programm *AnyTrans* demonstriert die Drehung kleiner Blöcke. Dabei wird ein Bild in Quadrate der Größe 14x14 aufgeteilt und jeder als Vektor mit 14x14=196 Komponenten aufgefasst. Durch das Programm wird ein Block im 196-dimensionalen Raum gedreht, das Urbild gelöscht und die gedrehten Koeffizienten zurücktransformiert. In den folgenden Kapiteln wird die zufällige Drehung aus dem Beispiel durch sinnvolle Drehungen ersetzt, die im Rahmen der Analyse besonders wünschenswerte Eigenschaften haben.

Wie zu erwarten, kommt das identische Bild heraus. Um zu demonstrieren, dass die Drehung tatsächlich stattgefunden hat, wurden in Abbildung 1.2 die Koeffizienten durch starke Quantisierung gestört. Dadurch treten Blockartefakte auf. Die eigentliche Drehung wurde völlig zufällig erzeugt. Aus diesem Grund wird durch die Quantisierung auch ein gleichmäßiges Rauschen hinzugefügt, wobei anders als bei der DCT (dazu später mehr) kein Frequenzband bevorzugt manipuliert wird. Die Drehung wurde durch 196 Vektoren, mit je 196 zufällig generierten Komponenten, erzeugt. Denn ein Raum der Dimension n wird durch genau n Vektoren aufgespannt. Man spricht daher auch von Spannvektoren. Auch ist der Drehung eigen, dass alle Spannvektoren senkrecht aufeinander stehen. Bei zufällig erzeugten Vektoren ist das nicht zu erwarten. Was fehlt, ist also noch das "Zurechtbiegen" der Vektoren, damit auch wirklich eine Drehung entsteht, die Längen und Winkel bewahrt. Dazu wurde das *Schmidt'sche Orthogonalisierungsverfahren* verwendet, welches eine saubere, orthonormale Basis zu einem Raum liefert, der durch ein Bündel von Vektoren aufgespannten wird. Die Erzeugung dieser zufälligen Drehung soll an dieser Stelle nicht weiter vertieft werden. Es geht nur darum zu zeigen, dass man Bilder bzw. Bildblöcke verlustfrei transformieren kann.

Abbildung 1.2: Originales Photo im Ortsraum (links). Transformiertes, quantisiertes und rücktransformiertes Photo (rechts)

1.5 Segmentierung durch statistische Analyse

Mit dem Bezug der Segmentierung zur Hauptkomponentenanalyse und zu den anderen in dieser Arbeit verwendeten Transformationen beschäftigen sich die letzten Kapitel dieser Arbeit ausführlich. Zur Motivation der im nächsten Kapitel folgenden mathematischen Statistik soll trotzdem kurz auf die Anwendung vorgegriffen werden: Mit den Merkmalsvektoren aus Kapitel zwei sind in der Anwendung Photos gemeint, meist nicht die gesamten Photos, sondern die Grenzregionen von Objektsilhouetten, die ein Objekt vom Hintergrund trennen. Diese Teile der Photos sollen einer Analyse unterzogen werden, die herausfindet, was das Objekt vom Hintergrund trennt. Dabei kann man davon ausgehen, dass auch in diesen Bildbereichen nicht alle enthaltenen Informationen die Segmentierung wirklich unterstützen. Tatsächlich sind gerade die irreführenden dafür verantwortlich, dass klassische Ansätze der Segmentierung in vielen Fällen schlecht funktionieren. Daher wird im Rahmen dieser Arbeit versucht, durch geschickte Transformation der Datenvektoren diese so zu drehen, dass die für den Zweck der Segmentierung nützlichen Aspekte besonders hervortreten und die irreführenden so weit wie möglich ausgeblendet werden.

1.6 Die Peak Signal-to-Noise-Ratio

In der Multimediaanalyse wird zur Beurteilung der Abweichung eines Signales oft die Peak Signal-to-Noise-Ratio (kurz PSNR) genutzt, die hier noch kurz eingeführt werden soll [BHA97, S. 9]. Hinter der Verwendung von Abstandsmaßen in der Multimediatechnik steckt der Wunsch, die menschliche Wahrnehmung durch eine möglichst kompakte Metrik abzubilden [KUH01]. Dass dies ohne weiteres nicht funktionieren kann, zeigt schon die Ungültigkeit der Dreiecksungleichung. Sie ist eine notwendige Bedingung für jede Metrik und beschreibt den Sachverhalt, dass die Distanz zwischen zwei Punkten (A,C) kleiner oder gleich dem Umweg über B, also (A,B)+(B,C) sein muss. Mit anderen Worten kann kein Umweg kürzer sein, als die Gerade zwischen zwei Punkten. Für die menschliche Wahrnehmung können aber Beispiele gefunden werden, in denen eine sanfte Veränderung eines Bildes insgesamt als angenehmer empfunden wird als der unmittelbare Sprung.

Die PSNR ist also ein eher technisches Maß, ohne den Anspruch zu haben, die Wahrnehmung zu modellieren. Für einen Vergleich verschiedener Signale untereinander, ohne unmittelbaren Bezug zum Menschen, eignet sie sich jedoch durchaus. Statistisch betrachtet, gibt sie eine normierte Form des mittleren quadratischen Fehlers e_{MSE} an und wird wie folgt definiert:

$$e_{MSE} = \frac{1}{XY}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left[\hat{p}(x,y) - p(x,y)\right]^2$$
$$PSNR = 10\log_{10}\left(\frac{S^2}{e_{MSE}}\right) \qquad \text{(Ausdruck 1.1)}$$

X, Y : Auflösung des Bildes
$p(x, y)$: Bild an der Stelle (x, y)
$\hat{p}(x, y)$: Approximation an der Stelle (x, y)

Die Konstante S steht für den maximalen Wert, den das Signal annehmen kann. Je kleiner der mittlere quadratische Fehler wird, desto größer wird der Bruch in der Klammer, da dieser bei Null eine Polstelle hat. Damit für gute Signale, also kleine Fehler e_{MSE}, die entsprechende Maßzahl nicht zu unhandlich wird, wird auf das Ergebnis noch einmal

der Logarithmus angewendet. Offensichtlich darf der eingesetzte Fehler im Nenner nicht null werden.

Je weiter der Fehler eines Signales gegen null konvergiert, desto größer wird die PSNR. PSNR-Werte für schlechter werdende Signale sind jedoch weniger intuitiv zu interpretieren - ein Umstand, der in der Literatur kaum erwähnt wird. Ist S der größte Wert, den ein Signal annehmen kann, so wird der mittlere Fehler in Bezug auf ein vollkommen zufällig generiertes Signal S/2 sein.

$$PSNR = 10\log_{10}\left(\frac{S^2}{(S/2)^2}\right) = 10\log_{10}\left(\frac{S^2}{(S^2/4)}\right) = 10\log_{10}(4) \approx 6{,}02 \qquad \text{(Ausdruck 1.2)}$$

Ein zufälliges Signal hat also immer noch eine PSNR von ca. 6,0 im Mittel. Abbildung 1.3 zeigt ein Beispiel zweier Bilder (rechts oben und links unten), die die gleiche PSNR haben.

Nach einer Bearbeitung des linken unteren Bildes durch den Diffusionsfilter *Enhance,* der ebenfalls im Rahmen dieser Arbeit entstanden ist, wird das Original zumindest wieder schemenhaft sichtbar. Wie kann es sein, dass beide Bilder, das Zufällige und das Verrauschte, auf Grundlage der PSNR gleich sind?

Die Erklärung liegt darin, dass lediglich ein Vergleich auf Basis isolierter Pixel durchgeführt wird. Trotzdem können benachbarte Pixel immer noch korreliert sein. Dies wird im mittleren quadratischen Fehler nicht gemessen. Interessanterweise werden sich Bild und Approximation sogar visuell wieder ähnlicher, wenn man die PSNR unter 6,02 absenkt. Dazu muss man das approximierte Signal wie folgt konstruieren:

Liegt das originale Signal in der oberen Hälfte des Wertebereichs, also z. B. im Intervall [128, 255], so muss das approximierende Signal seinen Minimalwert annehmen.

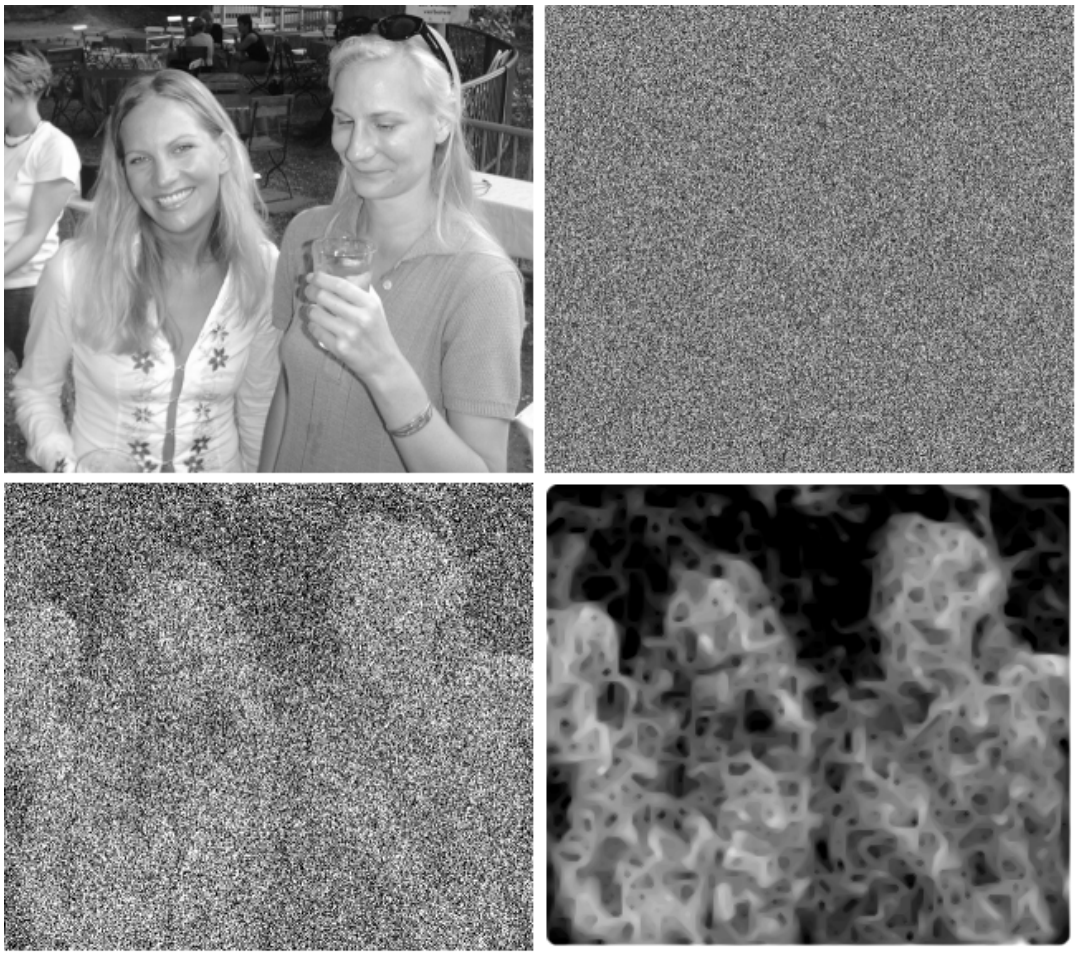

Abbildung 1.3: Original Abbildung (links oben), künstlich generiertes Rauschen mit PSNR von 8,4 (rechts oben), verrauschtes Original mit PSNR von 8,4 (links unten), diffusionsgefilterte Version des linken unteren Bildes (rechts unten)

Im Fall kleiner Signalwerte nimmt die Approximation ihr Maximum an, um immer den größtmöglichen Abstand zum Original zu erzeugen. Abbildung 1.4 zeigt ein Beispiel. Tatsächlich wird in diesem Fall eine PSNR von ca. 2,8 erreicht.

Abbildung 1.4: Das Bild wurde so erstellt, dass zum Original aus 1.3 (links oben) eine minimale PSNR von ca. 2,8 entsteht.

KAPITEL 2

FAKTOREN- UND HAUPTKOMPONENTENANALYSE

2.1 Motivation der Faktor- und Hauptkomponentenanalyse

Traditionell finden sowohl die Hauptkomponenten- als auch die Faktoranalyse in vielen Wissenschaftsgebieten, vor allem in der Biologie, der Volkswirtschaftslehre und den Sozialwissenschaften Anwendung. Die Anwendung im Bereich der digitalen Signalanalyse ist noch recht jung.

In allen Fällen werden Merkmalsvektoren (hier Bilder) mit vielen Merkmalen (die Pixel) betrachtet. Unabhängig von der Natur der Anwendung stellt man häufig fest, dass ein großer Teil der Merkmale untereinander korreliert ist. Dies bedeutet, dass hinter einer großen Anzahl oft nur wenige, unabhängige Sachverhalte stehen, die die vielen Merkmale bestimmen. Die Hauptkomponentenanalyse hat zum Ziel, diese wenigen unabhängigen Sachverhalte (im folgenden Hauptkomponenten oder Hauptachsen) aus einer großen Menge von Merkmalen herauszuziehen und gleich absteigend nach ihrer Wichtigkeit zu sortieren. Dieser, auch als Hauptachsenzerlegung bezeichnete Vorgang, ist noch ein rein mathematisches Verfahren (nämlich eine Drehung), das zu einem objektiven und nicht bestreitbaren Ergebnis führt.

Im Gegensatz dazu geht es in der Faktoranalyse darum, diese bisher noch abstrakten Hauptkomponenten im Kontext des realen Phänomens, welches analysiert werden soll, zu interpretieren, ihnen sozusagen "Namen zu geben". Dabei hat der Statistiker meist schon eine Vermutung über Art und Anzahl der Faktoren. Hier setzt auch gleich die Kritik an der Faktoranalyse an, denn die unterstellte Anzahl der vermuteten Faktoren kann willkürlich sein [BAC96, Abschnitt 5.2.7]. Denn meist ist die Zahl der Hauptkomponenten deutlich größer als die der Faktoren, die hinter einem Phänomen vermutet werden. Der Statistiker behilft sich damit, die verbleibenden Komponenten als Rauschen zu deklarieren, welches nicht interpretiert werden kann. Kurz gesprochen, versucht die Faktoranalyse den Zusammenhang unterschiedlicher Merkmale eines Merkmalsvektors aufzudecken und semantisch zu interpretieren [RIN00].

Zur Motivation sei hier noch ein konkretes Beispiel aus der Multimediaanalyse erwähnt: Bei der Analyse von Grauwert- oder Farbkorrelationen von Passbildern könnte ein enger Zusammenhang zwischen zwei bestimmten Pixelgruppen gefunden werden. Vielleicht stammen diese Pixel in den meisten Fotos jeweils vom rechten und linken Auge der abgebildeten Person [LAN95]. Die Erklärung für den Sachverhalt wäre, dass sich die Farben der beiden Augen bis auf seltene Fälle gleichen. Diesen Zusammenhang zwischen Pixeln könnte man durch die Hauptkomponentenzerlegung aufzeigen. Die meisten Korrelationen zwischen Grauwerten haben aber weniger intuitiv verständliche Hintergründe, so dass die Interpretation der Zusammenhänge zwischen den erhobenen Variablen kaum von Interesse ist (also z. B. der Zusammenhang zwischen rechtem und linkem Auge). Wichtig ist nur, die Zusammenhänge in der Korrelationsmatrix festzuhalten, um sie im Sinne der gesuchten Analyse später zu nutzen.

Die Hauptkomponentenanalyse begnügt sich mit dem Auffinden von beeinflussenden Sachverhalten. Mathematischer gesprochen, versucht sie vornehmlich, die Varianz der ursprünglichen Merkmale durch möglichst wenige Komponenten zu reproduzieren. Der Zusammenhang der Komponenten untereinander interessiert nicht, zumindest nicht im Sinne einer Interpretation.

Dazu ein einfaches Beispiel: Gegeben seien fünf Merkmalsvektoren mit fünf Komponenten.

$$\begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix},\begin{pmatrix}5\\6\\7\\8\\9\end{pmatrix},\begin{pmatrix}3\\4\\5\\6\\7\end{pmatrix},\begin{pmatrix}2\\3\\4\\5\\6\end{pmatrix},\begin{pmatrix}4\\5\\6\\7\\8\end{pmatrix} \qquad \text{(Ausdruck 2.1)}$$

Beim Betrachten der Vektoren wird deutlich, dass alle Komponenten einer Spalte voneinander abhängen, d. h. vollständig miteinander korreliert sind. Die folgende Abbildungsvorschrift ist offensichtlich.

$$\vec{m}_x = \begin{pmatrix} x \\ x+1 \\ x+2 \\ x+3 \\ x+4 \end{pmatrix} \qquad \text{(Ausdruck 2.2)}$$

Der Nutzen einer solchen Abstraktion für die Kompression ist offensichtlich, denn statt fünf Merkmale zu speichern, kann man den Aufwand auf nur eine Komponente reduzieren, die Daten sind dennoch vollständig wiederherstellbar. Auch bei der Analyse konzentriert man sich besser auf eine Komponente, insbesondere dann, wenn die restlichen ohnehin vollständig abhängig sind und keinerlei eigene Varianz enthalten.

2.1.1 Kommunalität im Kontext der Signalanalyse

In den meisten Fällen sind die Verbindungen der Komponenten untereinander weniger offensichtlich. Für Merkmale aus tatsächlichen Erhebungen ist es auch nicht wahrscheinlich, dass ein funktionaler Zusammenhang besteht, sondern eben nur eine Korrelation. Dies ist im folgenden Beispiel der Fall, in dem jeder Wert um eine zufällige Störung erweitert wurde.

$$\begin{pmatrix} 1+0.07 \\ 2-0.01 \\ 3+0.09 \\ 4+0.08 \\ 5-0.03 \end{pmatrix}, \begin{pmatrix} 5+0.02 \\ 6+0.04 \\ 7-0.05 \\ 8-0.00 \\ 9+0.08 \end{pmatrix}, \begin{pmatrix} 3-0.00 \\ 4+0.00 \\ 5+0.09 \\ 6-0.01 \\ 7-0.04 \end{pmatrix}, \begin{pmatrix} 2+0.02 \\ 3+0.07 \\ 4+0.03 \\ 5-0.08 \\ 6+0.05 \end{pmatrix}, \begin{pmatrix} 4-0.05 \\ 5-0.04 \\ 6-0.01 \\ 7+0.02 \\ 8-0.08 \end{pmatrix} \quad \text{(Ausdruck 2.3)}$$

Auch in diesem Beispiel kann der Zusammenhang zwischen den Komponenten erklärt werden, allerdings nicht vollständig. Mit dem Vorwissen darüber, dass die Vektoren im Hundertstelbereich gestört sind, würde man auch gar nicht den Versuch machen, sie perfekt zu reproduzieren oder sie vollständig zu analysieren.

> Bemerkung: Eine solche Störung schleicht sich bei der Analog-Digitalwandlung immer ein. So liefert eine Kamera niemals identische Standbilder, sondern fügt dem Bild ein in den elektronischen Schaltkreisen vorhandenes Rauschen hinzu. Gute Videocodecs können solche Szenen detektieren und sich entsprechend darauf einstellen, kein (informationsfreies) Rauschen zu übertragen. Auch wurde der Vorschlag gemacht, Bilder nicht im Original zu kodieren, sondern zuvor leicht zu glätten, um für die durch die Kamera induzierte Fehlinformation keine Bandbreite zu verbrauchen. Damit wird klar, dass auch die vollständige Analyse von Multimediadaten nicht nötig ist, weil ein kleiner Anteil der Information keine Bedeutung hat, egal ob man sich auf Bilder, Videos oder Audiodaten bezieht. Nicht immer ist die Elektronik schuld. Bei CT- (Computertomographie) oder MR- (Magnetresonanz) Aufnahmen, die im siebten Kapitel analysiert werden, entsteht aufgrund des Aufnahmeverfahrens selbst ein erhebliches Rauschen und eine so genannte Drift. Je besser die Analyse ein Rauschen in nachgelagerten Komponenten isolieren kann, die gar nicht mehr interpretiert werden, desto besser ist die Chance, dass sich z. B. ein Segmentierungsverfahren nicht fehlleiten lässt.

Den Grad, bis zu dem eine Variable durch Faktoren oder Komponenten reproduziert werden kann, bezeichnet man in der Statistik auch als Kommunalität. Aufgrund des Signalrauschens ist z. B. eine Kommunalität von eins, also die vollständige Reproduktion, nicht sinnvoll. Gleichwohl ist eine verlustfreie Reproduktion durchaus möglich, denn spätestens mit dem Einsatz von fünf Faktoren lassen sich auch fünf Komponenten vollständig wiederherstellen. Nun wird auch deutlicher, worin die oben erwähnte Kritik an der Faktoranalyse begründet ist: Möglicherweise erklären die letzten Faktoren nur

noch das zufällige Rauschen im Signal, oder es fehlen Faktoren, die einen Restsachverhalt analysieren könnten, der andernfalls als Rauschen ignoriert wird.

2.1.2 Verfahren der Faktoranalyse am Beispiel

Die Vorgehensweise bei der Hauptkomponenten- und der Faktoranalyse wird am besten an einem alltäglichen Beispiel deutlich. Insbesondere kann hier der später wichtige Zusammenhang zwischen den Korrelationen von Zufallsvariablen und den Hauptkomponenten noch grafisch verdeutlicht werden. Im Fall der hohen Dimensionen in der multimedialen Signalanalyse sind diese Zusammenhänge nur schwer anschaulich zu zeigen.

Die folgenden, noch überschaubaren Daten haben eine Umfrage zur Grundlage, in der insgesamt 30 Hausfrauen gebeten wurden, sechs Emulsionsfette (Butter und Margarine) subjektiv nach ihrer persönlichen Einschätzung zu bewerten [BAC96].

Merkmale	**Produkte**
Anteil ungesättigter Fettsäuren	Rama
Kaloriengehalt	Sanella
Vitamingehalt	Becel
Haltbarkeit	Du darfst
Preis	Holländische Butter
	Weihnachtsbutter

Tabelle 2.1: Zur Verdeutlichung der Methodik wurde ein kleines reales Beispiel mit einer überschaubaren Anzahl von Messwerten gewählt (aus [BAC96]).

Für alle Merkmale konnten ganze Zahlen aus dem Intervall [1, 7], von 1 = negativ bis 7 = positiv, gewählt werden. Da alle 30 Probanden sechs Produkte nach fünf Kriterien bewerten sollten, ergibt sich ein dreidimensionales Feld mit Bewertungsergebnissen. Die Anzahl der Probanden entsprich der Anzahl der Samples, die später einem multimedialen Signal entnommen werden. Dabei dient eine große Anzahl dazu, Zusammenhänge zwischen den Merkmalen so zu bestimmen, dass die gewonnenen Erkenntnisse (nicht aber die Merkmale selbst) mit einer möglichst geringen Varianz verbunden sind. Die Merkmale (z. B. der Vitamingehalt oder ein Pixel) haben ihre eigene gegebene Varianz. Die Varianz, die bei der Analyse minimiert werden soll, bezieht sich hier z. B. auf

das Wissen, wie stark die Haltbarkeit vom Preis abhängt oder wie stark ein Pixel seinen Nachbarn bestimmt. Denn auch diese Erkenntnis selbst hat natürlich eine Varianz, die einer Irrtumswahrscheinlichkeit entspricht. Aufgrund weniger, untypischer Stichproben könnte ein falsches Bild entstehen. Würde man theoretisch alle möglichen Signale einbeziehen (oder hier alle möglichen Hausfrauen befragen), so hätte man sichere Informationen mit einer Varianz von null über alle Zusammenhänge zwischen Merkmalen.

Zur statistischen Auswertung wurden nun die Mittel der Merkmalswerte über die Probanden berechnet, d. h. die Daten wurden bzgl. der befragten Personen anonymisiert. Die folgende Tabelle enthält das Ergebnis der Aggregation:

	Rama	**Sanella**	**Becel**	**Du darfst**	**Holl. B.**	**Weih. B.**
Fettsäuren	1	2	4	5	2	3
Kalorien	1	6	5	6	3	4
Vitamine	2	3	4	6	3	4
Haltbarkeit	1	3	4	2	5	6
Preis	2	4	5	3	7	7

Tabelle 2.2: Die Daten der Befragung wurden bzgl. der Personen aggregiert.

Bereits beim genaueren Betrachten der unmittelbaren Befragungsergebnisse fällt auf, dass bestimmte Eigenschaften zusammenhängen. So korreliert z. B. ein hoher Preis mit einer langen Haltbarkeit. Beide Merkmale unterscheiden sich bei fast allen Produkten lediglich um eine Skaleneinheit.

2.1.3 Abstraktion durch Ermitteln der Korrelation

Andere Merkmale bedingen sich dabei fast gar nicht. So lassen sich z. B. zwischen Preis und Vitaminen sowohl ähnliche als auch ganz unterschiedliche Wertepaare finden. Durchgängig unterschiedliche Werte würden eine negative Korrelation bedeuten, wogegen vorwiegend gleiche Werte eine positive Korrelation wiederspiegeln. Lediglich ein ausgewogenes Verhältnis beider lässt auf zwei miteinander unkorrelierte Größen schließen.

Um festzustellen, welche Variablen andere ganz oder teilweise bedingen, berechnet man die Korrelation zwischen je zwei Variablen. Dabei werden alle Werte der Befragung bzgl. ihres Mittelwertes normiert, so dass diese alle um den Ursprung "schwingen".

$$\bar{x}_m = 1/6 \sum_{p=1}^{6} x_{m,p}$$
$$x_{m,p}^{norm} = x_{m,p} - \bar{x}_m \qquad \text{(Ausdruck 2.4)}$$

$\bar{x}_m$ = Mittelwert des Merkmales m
m = Index Merkmal
p = Index Produkt

Nun unterscheiden sich die Befragungsergebnisse in $x_{m,p}^{norm}$ zwar nicht mehr durch deren Mittelwert, jedoch durch die Standardabweichungen, d. h. sie schwingen noch unterschiedlich stark um den Ursprung. Auch diese unterschiedlich starke Schwingung soll herausnormiert werden. Hierzu berechnet man für jede Variable die Standardabweichung $\sigma(x_m)$. Jeder einzelne Befragungswert wird zuletzt noch durch $\sigma(x_m)$ geteilt.

$$\sigma(x_m) = \sqrt{\frac{1}{6} \sum_{p=1}^{6} (x_{m,p} - \bar{x}_m)^2} \qquad \text{(Ausdruck 2.5)}$$
$\sigma(x_m)$ = Standardabweichung des Merkmals m

$$z_{m,p} = \bar{x}_{m,p} / \sigma(x_m) \qquad \text{(Ausdruck 2.6)}$$
$z_{m,p}$ = Standardisiertes Befragungsergebnis des Merkmals m bei Prokukt p

Nach der Vorarbeit zum Zweck der Normierung folgt nun die Berechnung der Korrelation $r_{m1,m2}$ zwischen je zwei Merkmalen m_1 und m_2.

$$r_{m1,m2} = \frac{1}{6}\sum_{p=1}^{6} z_{m1,p} * z_{m2,p}$$

(Ausdruck 2.7)

m_1 = Merkmal 1
m_2 = Merkmal 2
$r_{m1,m2}$ = Korrelation zwischen Merkmal 1 und Merkmal 2

So wie ursprünglich die Befragung bezüglich der Personen anonymisiert wurde, so versteht sich der Korrelationskoeffizient $r_{m1,m2}$ zusätzlich bezüglich der Produkte als normiert.

Ausdruck 2.8 fasst die Berechnung in eine einzige Gleichung zusammen.

$$r_{m1,m2} = \frac{\frac{1}{6}\sum_{p=1}^{6}\left[(x_{m1,p} - \bar{x}_{m1})(x_{m2,p} - \bar{x}_{m2})\right]}{\sqrt{\frac{1}{6}\sum_{p=1}^{6}(x_{m1,p} - \bar{x}_{m1})^2}\sqrt{\frac{1}{6}\sum_{p=1}^{6}(x_{m2,p} - \bar{x}_{m2})^2}}$$

(Ausdruck 2.8)

Tabelle 2.3 lässt bereits vermuten, dass hinter der Beurteilung von Emulsionsfetten nur zwei Konzepte stehen. Dies wird in der Tabelle durch die beiden fettgedruckten Rechtecke gekennzeichnet. Scheinbar sind die Größen Fettsäuren, Kalorien und Vitamine miteinander korreliert, da sich hier Werte > 0,7 finden.

$r_{m1,m2}$	**Fettsäuren**	**Kalorien**	**Vitamine**	**Haltbarkeit**	**Preis**
Fettsäuren	1,00	0,71	0,96	0,11	0,04
Kalorien	0,71	1,00	0,70	0,14	0,07
Vitamine	0,96	0,70	1,00	0,08	0,02
Haltbarkeit	0,11	0,14	0,08	1,00	0,98
Preis	0,04	0,07	0,02	0,98	1,00

Tabelle 2.3: Korrelationsmatrix mit normierten Koeffizienten. Die zwei schwarz umrandeten Felder lassen bereits vermuten, dass die jeweils darin enthaltenen Werte durch nur je einen Faktor bestimmt werden.

Dagegen lassen sich zwischen der Gruppe Fettsäuren, Kalorien und Vitamine und der Gruppe Preis und Haltbarkeit (also den nicht markierten Bereichen) keine oder zumindest keine signifikanten Korrelationen finden (alle < 0,14). Zwischen Preis und Haltbarkeit findet sich dagegen eine sehr hohe Korrelation von 0,98. Dass sich jeder Wert

selbst bedingt, ist trivial und spiegelt sich in dem Wert von 1,00 auf der Diagonalen wieder.

> Hinweis: Ebenso wie der Durchschnitt eine Schätzung des wahren, aber unbekannten Mittelwertes ist, kann auch der Korrelationskoeffizient eine schlechte Schätzung der tatsächlichen Verbindung zweier Zufallsvariablen sein. In der Statistik wird diesem Umstand insbesondere dann Beachtung geschenkt, wenn nur wenige Stichprobenwerte erhoben wurden. Ein einfacher Test für die "Glaubwürdigkeit" der Korrelationen besteht in der Berechnung der Inversen Matrix [BAC96]. Dabei sollte sich eine mehr oder weniger deutliche Diagonalmatrix ergeben, mit tendenziell großen Werte um die Diagonale und eher kleinen in der rechten oberen und linken unteren Ecke. Eine genauere und allgemeingültige Quantifizierung der zu erwartenden Werte ist in der Statistik nicht bekannt. Genauere Beurteilungen der Güte der Korrelationskoeffizienten sind mit dem Bartlett-Test und der Image-Analyse von Guttmann [GUT50] möglich.

Für den Zweck der multimedialen Signalanalyse kann die Korrelationsmatrix im Allgemeinen als glaubwürdig angesehen werden, da die Anzahl der Grauwerte meist groß gewählt werden kann. Auch ist im Gegensatz zu den Sozialwissenschaften eine Erklärung der Korrelationen nicht von Interesse.

2.1.4 Winkel und Korrelationskoeffizienten

Die Korrelation zwischen zwei Variablen kann man sich auch als Winkel zwischen zwei Vektoren im Raum vorstellen. Sind zwei Größen weitgehend miteinander korreliert, so zeigen deren entsprechende Vektoren etwa in die gleiche Richtung. Sind sie dagegen unkorreliert, d. h. ihr Tabelleneintrag ist null, so stehen sie senkrecht aufeinander. Ein Traversieren entlang der einen Achse hat dann keinerlei Auswirkung auf die Richtung der anderen Achse. In Tabelle 2.4 wurden in der oberen Dreiecksmatrix die Winkel zu den entsprechenden Cosinuswerten eingetragen. Man erkennt, dass Fettsäuren, Kalorien und Vitamine paarweise annähernd senkrecht auf den Merkmalen Haltbarkeit und Preis stehen.

$r_{m1,m2}$	**Fettsäuren**	**Kalorien**	**Vitamine**	**Haltbarkeit**	**Preis**
Fettsäuren	0°	44,77°	16,26°	83,68°	87,71°
Kalorien	0,71	0°	45,57°	81,95°	85,99°
Vitamine	0,96	0,70	0°	85,41°	88,85°
Haltbarkeit	0,11	0,14	0,08	0°	11,48°
Preis	0,04	0,07	0,02	0,98	0°

Tabelle 2.4: Statt den Korrelationskoeffizienten wurden hier in der oberen Dreiecksmatrix die Winkel zwischen den Merkmalen eingetragen. Dies entspricht jeweils dem Arcuscosinus des Koeffizienten, der sich gespiegelt noch einmal in der unteren Dreiecksmatrix findet.

Bemerkung: Der Winkel zwischen zwei Merkmalen errechnet sich einfach aus dem Arcuscosinus des Korrelationskoeffizienten. Die anschauliche Bedeutung des Cosinus ist einfach die Projektion eines Vektors der Länge eins auf die Abszisse (meist als X-Achse bezeichnet). Im Kontext der Statistik denkt man sich die Abszisse als einen der beiden Merkmalsvektoren, der zu projizierende Vektor entspricht dem verbleibenden Merkmal. Liegt ein Vektor z. B. unmittelbar auf der Achse, so hat die Projektion gerade die Länge eins. Beide Vektoren sind also vollständig korreliert und ihr Winkel damit null Grad. In Abbildung 2.1 wird ein Merkmalsvektor auf einen anderen projiziert.

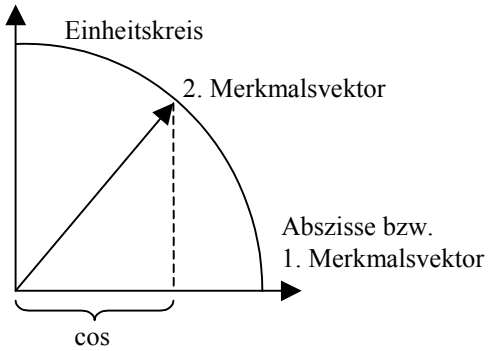

Abbildung 2.1: Ein Merkmalsvektor 2 wird auf einen anderen Merkmalsvektor 1 projiziert. Ist der Winkel dabei null, so zeigen beide in die gleiche Richtung bzw. erklären den gleichen Sachverhalt. Daher ist der entsprechende Wert in der Korrelationsmatrix cos(0°) = 1.

Da die fünf mehr oder minder unabhängigen Merkmale einen fünfdimensionalen Raum aufspannen und alle Winkel zwischen den Merkmalen aus der Tabelle 2.4 bekannt sind, lässt sich das schiefwinklige Bündel von fünf Vektoren aufbauen. Wie gesagt, muss der Aufbau so geschehen, dass je zwei Vektoren zueinander die Winkel aus Tabelle 2.4 annehmen. So entnimmt man z. B. der Tabelle, dass der Vektor zwischen *Fettsäure* und *Kalorien* 44,77° betragen muss. Konsistenterweise hat der Vektor Fettsäure übrigens

einen Winkel von 0° zu sich selbst. Im Allgemeinen lassen sich die in der Tabelle beschriebenen Winkel zwischen fünf Vektoren in der zweidimensionalen Ebene nicht veranschaulichen. Woran liegt das? Die Antwort ist, dass in einem Raum mit fünf Dimensionen die Winkel zwischen allen möglichen Paaren von Vektoren, frei gewählt werden können. In der Ebene reichen die Freiheitsgrade dafür bei weitem nicht aus. Vielmehr läßt die Ebene lediglich zu, dass der Winkel zwischen benachbarten Vektoren festgelegt wird. Die Winkel zwischen allen nicht-benachbarten Vektoren sind damit implizit definiert und können nicht mehr gewählt werden. In Abbildung 2.2 wurde jedoch eine beispielhafte Konstellation erzeugt, der jedoch keine echte Korrelationsmatrix zugrunde liegt.

2.1.5 Konstruktion der ersten Hauptkomponente

Es fällt auf, dass alle Vektoren zwar in unterschiedliche Richtungen weisen, in Summe jedoch eine gemeinsame Stossrichtung besitzen. Diese als gestrichelter Vektor gezeichnete gemeinsame Richtung ist bereits die erste und wichtigste Hauptkomponente oder auch die erste Resultante. Sie ist sowohl graphisch also auch rechnerisch durch Addition der beteiligten Vektoren ermittelbar und ergibt den wichtigsten Faktor, der in diesem Beispiel Einfluss auf alle Variablen hat. Mit anderen Worten: Keine der fünf Variablen entzieht sich in diesem Beispiel ganz der Richtung der Resultanten. Dieser wichtigste Faktor beeinflusst somit alle Variablen, abhängig von deren Winkel mehr oder weniger stark. Lediglich ein zur Resultanten senkrechtes Merkmal wäre mit ihr unkorreliert.

Das Ausmaß, in dem die erste Resultante die einzelnen Ausgangsvariablen x1, ..., x5 beschreibt, wird in der Statistik auch als die Faktorladung bezeichnet. Ein kleiner Winkel zwischen Resultante und einem Ausgangsvektor drückt sich in einer hohen Faktorladung (max. eins) aus.

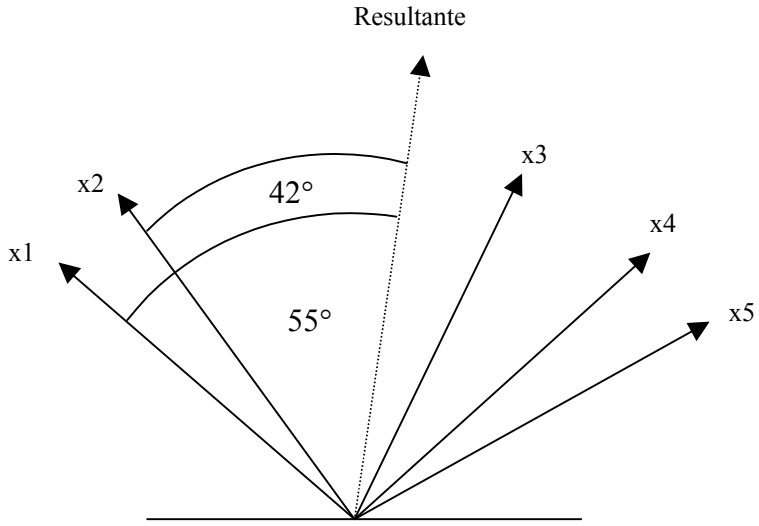

Abbildung 2.2: Die Beziehungen der fünf Merkmale aus dem Beispiel wurden hier in der Ebene verdeutlicht. Zeigen zwei Vektoren in die gleiche Richtung, so geben sie auch die gleiche Information wieder.

Diese errechnet sich gerade aus dem Cosinus des Winkels (siehe Bemerkung weiter oben). Durch Berechnung der Winkel zwischen der Resultanten und den Ausgangsvektoren kann also der erste Faktor unmittelbar bestimmt werden. Ausdruck 2.9 zeigt den ersten Faktor auf Grundlage der (geometrisch) abgemessenen Winkel mit den einzelnen Faktorladungen als Komponenten des Vektors.

$$w_1 \approx \begin{pmatrix} \cos(55°) \approx 0{,}574 \\ \cos(42°) \approx 0{,}743 \\ \cos(17°) \approx 0{,}956 \\ \cos(37°) \approx 0{,}799 \\ \cos(49°) \approx 0{,}656 \end{pmatrix} \qquad \text{(Ausdruck 2.9)}$$

Wie weiter oben bereits erwähnt, ist die Resultante derjenige Vektor, der seinen Winkel zu allen anderen Vektoren minimiert. Dies motiviert deren Konstruktion eher geometrisch. Rein rechnerisch kann die Suche nach der erste Hauptkomponente auch als Ergebnis einer Optimierungsaufgabe gesehen werden:

$$w_1 = \arg\max_{\|w_1\|=1} \mathrm{E}\left\{\left(w_1^T x\right)^2\right\} \qquad \text{(Ausdruck 2.10)}$$

Der Vektor w_1 bezeichnet die erste Hauptkomponente. Wegen $\|w_1\|$ wird über alle denkbaren Vektoren w_1 der Länge Eins optimiert. Der arg-Operator bedeutet lediglich, dass nicht das Ergebnis der Optimierung selbst, sondern das Argument (also das beste w_1) zurückgeliefert wird, mit welchem das Optimum erreicht wurde.

Das Ziel der Optimierung ist, den Erwartungswert des Produktes aus w_1 und der Zufallsvariablen x zu maximieren. Dabei kann x z. B. ein Befragungsergebnis aus dem oben bemühten Beispiel sein, später werden im Zusammenhang mit der Mustererkennung aber Grauwertbilder oder Ausschnitte davon eingesetzt.

Hierzu wieder ein konkretes Beispiel aus der Multimediatechnik [KIR90, TUR91]: Alle x seien Passbilder einer Personaldatenbank. Mit dem Vektor w_1 ist ein Durchschnittsgesicht gesucht, welches so konstruiert sein soll, dass es allen Gesichtern möglichst nahe kommt. Vielleicht ist ein solches Gesicht in der Datenbank selbst enthalten, wahrscheinlicher ist aber ein "Mischbild", welches nicht zwingend überall sehr scharf definiert sein muss. Trotz dessen: Hätte man nur einen Parameter zur Verfügung bzw. könnte oder wollte man Gesichter mit nur einer Variablen anzeigen (statt mit zehntausenden von Pixeln), so würde man den kleinstmöglichen Fehler machen, dieses Gesicht w_1 durch einen entsprechenden Parameter einzublenden.

Allgemein kann man sich alle x als Punktewolke in einem mehrdimensionalen Raum vorstellen. Abbildung 2.3 zeigt ein Beispiel im R^2.

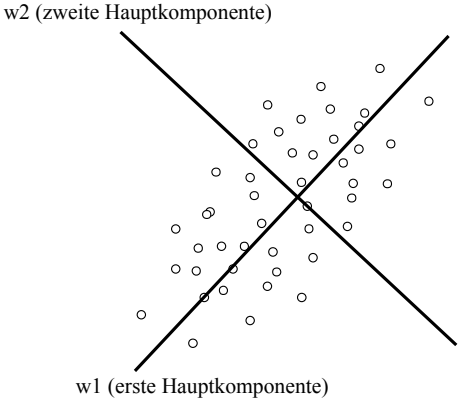

Abbildung 2.3: Eine Wolke von Punkten wurde in der Ebene abgetragen. Die beiden Geraden zeigen die Hauptkomponenten der Verteilung an. Offensichtlich zeigt die erste Komponente den wichtigeren Aspekt der ovalen Verteilung an.

Jede Ellipse im R^n hat n eindeutige Achsen. Zumindest gilt dies dann, wenn keine Projektion der Ellipse einen Kreis ergibt. Dabei gibt die längste Achse anschaulich die längste Ausdehnung der Ellipse im Raum an.

In erster Näherung kann man nun die w_1-Achse traversieren, um möglichst nahe an einen bestimmten Punkt zu gelangen. Oder mit anderen Worten: Hätte man nur einen Skalar, um einen Punkt der Wolke zu beschreiben, so würde man allen Punkten im Mittel entlang der w_1-Achse am nächsten kommen. Keine andere im Raum gedrehte Achse ist allen Punkten gleichzeitig so nahe. Genau dieser Sachverhalt wird in Ausdruck 2.10 optimiert. Im zweidimensionalen Raum ergibt sich die zweite Komponente eindeutig als Lot der ersten, so dass sich die weitere Optimierung erübrigt. In Räumen höherer Dimension ist dagegen mehr Arbeit nötig.

Die Achse w_1 ist nichts anderes als die Resultante, die in Abbildung 2.2 ermittelt wurde. Um sie zeichnerisch oder rechnerisch zu ermitteln, braucht man allerdings zuerst das ebenfalls in 2.2 dargestellte schiefwinklige System der Merkmalsvektoren. Zur Erinnerung: Die Winkel zwischen den Merkmalen können unmittelbar aus der Korrelationsmatrix abgelesen werden. Im Allgemeinen ist die Konstruktion des Systems schiefwinkliger Vektoren im mehrdimensionalen Raum weder trivial noch sinnvoll.

Ausdruck 2.10 bedient sich einer bequemen Schreibweise, die zwar definiert welcher Eigenschaft ein bestimmter Vektor genügen muss, die jedoch keine Aussage darüber macht, wie dieser Vektor zu berechnen ist. Im folgenden Abschnitt wird daher der Weg (im Sinne einer Prozedur) zum Ermitteln der Vektoren gesucht.

Neben der rein rechnerischen Definition werden die Hauptkomponenten statistisch auch als künstliche Merkmale bezeichnet, welche genau so erzeugt werden, dass mit allen anderen Merkmalen eine möglichst große Korrelation entsteht.

Durch die gerade ermittelte Hauptkomponente ist die Varianz aller Variablen (also z. B. den Pixeln der Passbilder) zu einem gewissen Teil reproduzierbar. Der Grad der Reproduktion ist bei nur einem extrahierten Faktor gerade dessen Faktorladung selbst. So ist

aus Ausdruck 2.9 unmittelbar abzulesen, dass Variable 1 zu etwa 57%, Variable 2 zu 74% etc. durch den Faktor w_1 erzeugt wird. Den Grad der Erklärbarkeit bezeichnet man auch als Kommunalität der Variablen. Obwohl die Anzahl der Faktoren, insbesondere im Sinn der Datenreduktion und Analyse, signifikant kleiner sein soll als die Dimension der erhobenen Daten, ist eine Kommunalität von 57% (durch die Verwendung nur eines Faktors), noch nicht ausreichend. Daher müssen weitere Faktoren berechnet werden.

2.1.6 Weitere Hauptkomponenten

Bei mehr als einem Faktor ergibt sich die Kommunalität einer Variablen aus der Summe aller beteiligter Faktorladungen. Wie bereits früher erwähnt, ist die Reproduktion der Variablen nur bis zu dem Maße sinnvoll, bis zu dem "echte" Information enthalten ist. Ebenso wie die Befragung von Probanden, erzeugt auch die Messung von Grauwerten ein gewisses Maß an Rauschen, das im Kontext der Anwendung nicht sinnvoll erklärt werden kann. Somit ist eine Kommunalität von 100% selten nötig. Daher wird man die Suche nach weiteren Faktoren beim Erreichen eines ausreichenden Maßes an Reproduzierbarkeit abbrechen. Der Grad an Informationsverlust, den man dabei in Kauf nehmen kann, hängt von der Stärke des Rauschens ab. Im oben gewählten Beispiel ist dieses anhand der einen Befragung sicher nicht ermittelbar, in der Messtechnik ist der Rauschabstand i. d. R. bekannt.

Mit dem nächsten Faktor soll die Varianz beschrieben werden, die der erste Faktor nicht erfassen konnte. Bezogen auf das Beispiel der Passbilder soll das Durchschnittsgesicht nun um weitere Details wie z. B. die spezielle Ausprägung von Augen, Nase, Mund etc. erweitert werden. Zu diesem Zweck projiziert man alle Datenpunkte entlang des ersten Faktors in den Untervektorraum, der ohne den ersten Faktor verbleibt. In Abbildung 2.4 ist der Vorgang grafisch verdeutlicht. Nach der Projektion der Punktewolke bleibt in diesem Beispiel ein Punktecluster auf einer Ebene übrig (unterer Teil der Abb.). In dieser Ebene wird nach einer neuen Hauptkomponente gesucht usw.

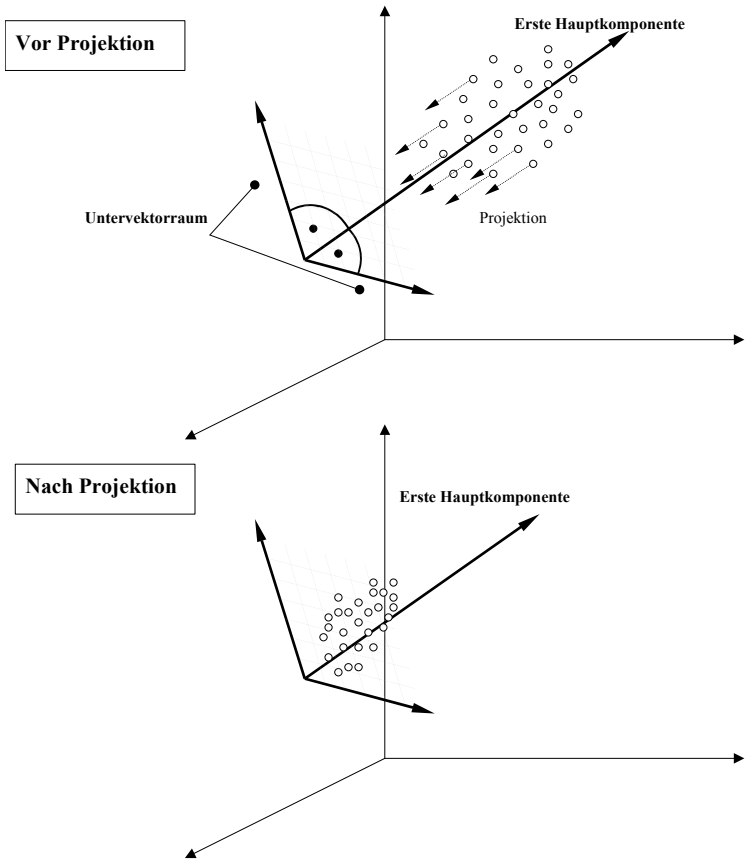

Abbildung 2.4: Zuerst wird die erste Hauptachse bestimmt (oben). In den folgenden Schritten sollen weitere Achsen ermittelt werden. Diese können nur noch in dem Unterraum liegen, der zur ersten Achse orthogonal ist (unten). In diesem Unterraum findet die weitere Suche nach den verbleibenden Vorzugsrichtungen der Messpunkte statt.

Hinweis: Der letzte Vektor ist gerade so zu setzen, dass er auf allen vorherigen orthogonal steht. Durch den Vorgang des "Wegprojizierens" der jeweils ermittelten Faktoren ist die Bedingung der Orthogonalität der vorherigen Faktoren bereits gegeben. Zur Veranschaulichung: Beispielsweise steht jeder beliebige von Null verschiedene Vektor in der XY-Ebene senkrecht zur Z-Achse.

2.2 Anwendung der Hauptkomponentenanalyse

Eine unmittelbare Anwendung der Hauptkomponentenanalyse in der Multimediatechnik wurde in [HAE02] vorgeschlagen. Dabei soll eine möglichst gleichmäßig texturierte Region analysiert und danach eine ähnliche Textur über einer beliebig großen Fläche resynthetisiert werden. Gebraucht werden solche Texturregionen z. B. bei Computer-

spielen, bei denen Texturen heute noch explizit gespeichert werden. Einfacher wäre die Generierung nach dem Programmstart oder evtl. in Echtzeit während des Spiels.

2.2.1 Bildkompression durch Texturresynthese

Eine weitere Anwendung ist die fortgeschrittene Bildkompression. Wie schon weiter oben gezeigt, bietet die gezielte Konstruktion einer Transformation die Möglichkeit, ein Signal maximal zu dekorrelieren. Leider ist damit auch die theoretische Obergrenze der Kompression durch lineare Abbildungen festgelegt, zumindest wenn kein Informationsverlust in Kauf genommen wird:

> Exkurs: Diese Obergrenze besteht nur unter der Voraussetzung, dass die Abbildung des Bildes durch eine lineare Funktion wie z. B. die diskrete Cosinus-Transformation (DCT) oder die diskrete Wavelet-Transformation (DWT) geschieht. Die nicht-linearen Funktionen bergen dagegen - zumindest theoretisch - Potenzial für noch stärkere verlustfreie Kompression. Zu den wenigen Ansätzen, die bereits in der Praxis verwendet werden, gehört die fraktale Bildkompression, die vom britischen Mathematiker Michael F. Barnsley, in Zusammenarbeit mit Lyman P. Hurd, vorgeschlagen wurde [BAR88, BAR93, BAR96]. Anders als bei der konventionellen Bildkompression wird dabei nicht das Bild selbst, sondern eine Funktion gespeichert. Deren besondere Eigenschaft besteht darin, für jeden beliebigen nicht-leeren Startwert gegen ein Zielbild zu konvergieren, das implizit in der Funktion enthalten ist. Es handelt sich also um eine spezielle Art der Fixpunktiteration. Die Dekodierung erfolgt tatsächlich durch Einsetzen eines beliebigen Bildes. Das Ergebnis der Abbildung durch die Funktion wird wieder erneut eingesetzt usw. Natürlich konvergiert nicht jedes Bild gleich schnell gegen den Zielwert. Man kann jedoch zeigen, dass es für jedes schwarz-weiße Bild mindestens eine Funktion gibt, die das Zielbild als Fixpunkt enthält. Meist gibt es sogar eine unendlich große Anzahl von Abbildungen mit dem gleichen Fixpunkt, so dass man z. B. nach derjenigen suchen könnte, die am schnellsten konvergiert. Jede dieser (Kompressions-) Funktionen besteht selbst aus einer mehr oder weniger großen Anzahl von affinen Funktionen der Form Ax+b. Diese bezeichnet Barnsley als ein iteriertes Funktionensystem (kurz IFS). Ein Pixel x eines Startbildes wird also durch den linearen Teil A der Abbildung z. B. gedreht, gestaucht etc. und ggfs. durch das b verschoben. Das geschieht innerhalb der Funktion evtl. sehr oft, je nachdem, aus wievielen affinen

Abbildungen das IFS besteht. Daher ist die Dekodierung einfach auszuführen. Viel problematischer und bisher nicht richtig gelöst ist das Auffinden eines minimalen IFS. Hierzu ist bisher kein Verfahren, schon gar kein automatisches bekannt. In den praktischen Implementierungen der fraktalen Bildkomression beschränkt man sich daher auf die Betrachtung sehr kleiner Blöcke, für die überhaupt nur entsprechend wenige Funktionen in Frage kommen.

Obwohl eine Obergrenze für die verlustfreie Kompression durch nicht-lineare Abbildungen noch nicht abgeschätzt werden kann, kann man sich dennoch vorstellen, dass für ausreichend chaotische Bildinhalte nicht ohne weiteres eine Funktion zu finden ist, die selbst wesentlich kleiner als der Inhalt des Bildes ist. Die Idee der Texturanalyse bzw. Resynthese verfolgt daher das Ziel, ein Bild gar nicht Pixel für Pixel abzuspeichern, sondern beim Betrachter nur einen visuellen Eindruck zu reproduzieren, so wie er auch vom Originalbild erzeugt wurde. Der Ansatz funktioniert vor allem in stark texturierten Bereichen. Texturen bestehen aus einer großen Anzahl winziger Details, so z. B. Gras, Haare oder Fell, eine Rauhfasertapete etc. Die einzelnen Details sind in diesem Fall für den menschlichen Betrachter von untergeordnetem Interesse. Lediglich die grobe Charakteristik der Textur sollte erhalten bleiben, also z. B. die Körnung der Tapete oder die vorherrschende Richtung der Haare eines Felles. Tatsächlich können diese Charakteristika mit Hilfe des nachfolgend beschriebenen Verfahrens analysiert und im Anschluss ohne Zugriff auf die ursprünglich gespeicherten Pixel wiederhergestellt werden.

2.2.2 Vorangegangene Arbeiten

Frühe Arbeiten im Bereich der Texturanalyse arbeiten im Ortsraum. Ein Beispiel hierfür ist der autoregressive moving average (ARMA) process [JAI81]. Dieses Verfahren besteht aus zwei Komponenten (siehe Ausdruck 2.11): Der Grauwert eines betrachteten Pixels wird hauptsächlich durch einen Filter $a_i f()$ berechnet. Eine zweite Komponente $b_i \omega()$ generiert eine zusätzliche Zufallszahl mit dem Mittelwert Null. Auch diese Zufallszahl ist ein gefiltertes Mittel eines Zufallsfelds $\omega()$, welches ebenso groß wie das Bild selbst ist.

$$f(x) = \sum_i a_i f(x + \Delta x_i) + \sum_i b_i \omega(x + \Delta x_i)$$
$$\Delta x_i \neq (0,0) \qquad \text{(Ausdruck 2.11)}$$

Die Wahl der Filter und die Verteilung der Zufallswerte bestimmt vollständig die Charakteristik der generierten Textur. Der erste Teil der Summe aus Ausdruck 2.11 repräsentiert gewissermaßen den deterministischen Anteil der Textur, der bei diesem Vorschlag von der Nachbarschaft abhängig ist, die zweite Summe bringt den chaotischen Anteil in die Textur ein. In anderen Ansätzen werden Markov-Modelle berechnet, die die Korrelation von Grauwerten in einer definierten Nachbarschaft analysieren und entsprechende neue Zufallswerte generieren [MES89]. All diese Techniken haben gemein, dass sie im Ortsraum arbeiten. Natürlich muss auch jede wichtige Charakteristik in irgendeiner Weise in der Verteilung der Grauwerte im Ortsraum erkennbar werden. Trotz dessen kann eine geschickt gewählte Transformation die Aufgabe erleichtern, wichtige statistische Merkmale aus dem Bild herauszuziehen. Einige Autoren suchen diese Merkmale nach einer Wavelet-Transformation im Frequenzraum [UNS96]. Dahinter steckt die Hoffnung, dass sich Eigenschaften in bestimmten Frequenzbändern besonders konzentrieren bzw. durch die Verdichtung der Information in einzelnen Koeffizienten wiederfinden lassen. Das Ausmaß der Verdichtung hängt von der verwendeten Filterbank und vor allem der zugrunde liegenden Textur ab [CHA93, KUN92].

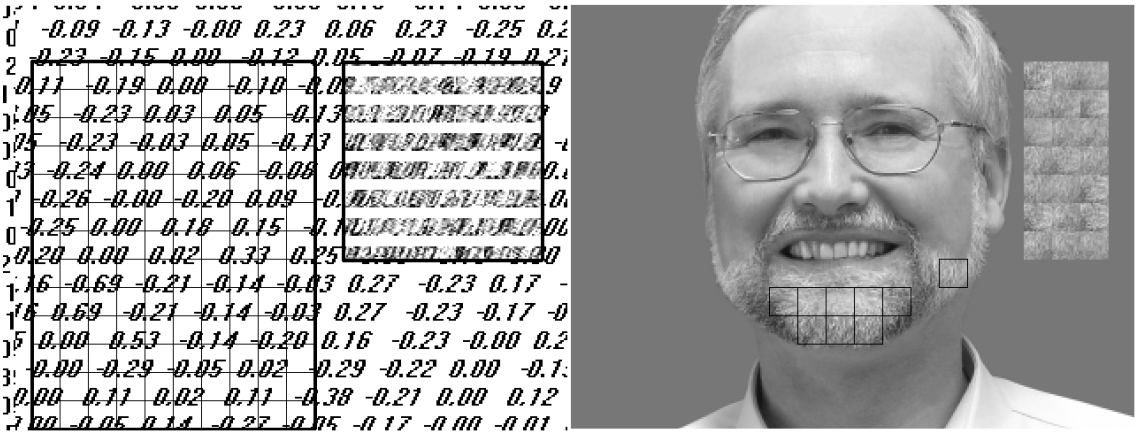

Abbildung 2.5: Resynthese nach [HAE02] von Text (links) und Haarstruktur (rechts). Die keinen schwarzen Blöcke werden zur Analyse verwendet. Daneben wurde auf Grundlage dieser Analyse neue Textur synthetisiert.

Der hier vorgestellte Ansatz basiert auf der Hauptkomponentenanalyse, in der die Abhängigkeiten der Grauwerte untereinander in einem begrenzten Bereich analysiert werden. Das Verfahren arbeitet also nicht auf dem gesamten Bild, sondern ist blockbasiert (siehe Abbildung 2.5). Die noch bessere Verdichtung der Information, im Vergleich zur Wavelet-Transformation, ist eine wichtige Eigenschaft für die spätere Resynthese.

Im folgenden Abschnitt wird das eigentlich Verfahren der Texturresynthese beschrieben. An dessen Ende folgt eine Zusammenfassung als Pseudocode. In Abschnitt 2.2.4 wird die Unterstellung untersucht, dass die Koeffizienten Gaussverteilt sind. Abschnitt 2.2.5 zeigt die praktische Anwendung des Verfahrens und in 2.2.6 wird eine Idee zur Unterdrückung von Blockartefakten erläutert.

2.2.3 Ermitteln der optimalen Transformation

Gegeben sei ein Bildblock der Größe $n \times n$ als ein Vektor von Grauwerten $x = (x_1,...,x_{n*n})$, so wie dies z. B. von der JPEG Kompression (siehe Kapitel 3) her bekannt ist.

> Hinweis: Das "Plattklopfen" der 2D-Pixelwerte kann willkürlich erfolgen, d. h. die Reihenfolge der Pixel ist beliebig. Zwar wird durch die Reihenfolge die spätere Korrelationsmatrix und die Transformation beeinflusst, die Rücktransformation, also die Resynthese in den Ortsraum erfolgt aber analog, so dass es keine Rolle spielt, ob die Pixel z. B. in einer Zick-Zack Reihenfolge oder zeilenweise betrachtet werden. Mit anderen Worten: Die Reihenfolge der Analyse wird die gleiche wie die der Synthese sein. Im Ergebnis spielt sie keine Rolle.

Der gewählte Block sollte aus einem Teil des Bildes stammen, welcher typische Textur enthält. Auch sollten sich mehrere Bereiche der Größe $n \times n$ mit der gleichen Textur finden lassen. Je größer die Anzahl der Beispielblöcke ist, desto besser kann die Analyse erfolgen, weil die Korrelationsmatrix immer zuverlässiger wird. Auf Grundlage nur eines Blockes könnte man überhaupt keine Texturresynthese betreiben. Denn aufgrund nur eines (vektorwertigen) Wertes kann man keinen sinnvollen Mittelwert und keine Varianz berechnen.

In der aktuellen Implementierung des Programmes *TextureResynthesis* werden die Blöcke manuell ausgewählt. Danach wird eine Korrelationsmatrix C auf Grundlage der gewählten Blöcke berechnet. Offensichtlich ist C eine symmetrische Matrix, da jeder Grauwert x_i in gleicher Weise mit x_j korreliert ist wie umgekehrt. Nach der in diesem Kapitel erläuterten Methode werden nun alle $n \times n$ Eigenvektoren E_i aus C berechnet und entsprechend der Größe ihrer zugehörigen Eigenwerte e_i in absteigender Reihenfolge sortiert. Ob diese Sortierung bereits bei der Berechnung entsteht, hängt vom verwendeten Verfahren ab. Im Rahmen dieser Arbeit soll die Berechnung der Eigenwerte nicht weiter vertieft werden. Es stehen hierzu eine Reihe von kommerziellen und Open-Source-Softwarebibliotheken zur Verfügung [PRE02]. Grundsätzlich ist der Vorgang aber der gleiche wie der Vorgang der Berechnung der Korrelationen zwischen den Merkmalen der Butter-Produkte, im oben bemühten Beispiel. Die sich ergebende Matrix $T = (E_1^T, ..., E_{n*n}^T)$ ist bereits diejenige Transformation, die die Informationen eines der typischen $n \times n$-Texturblöcke in der bestmöglichen Weise dekorreliert.

Innerhalb der praktischen Auswertung der Texturresynthese zeigt sich, dass nur ein kleiner Teil der Eigenvektoren wirklich relevant ist. Dies belegt auch Abbildung 2.7, in der die Eigenwerte als Graph abgetragen wurden. Von 256 Werten sind etwa 200 fast null.

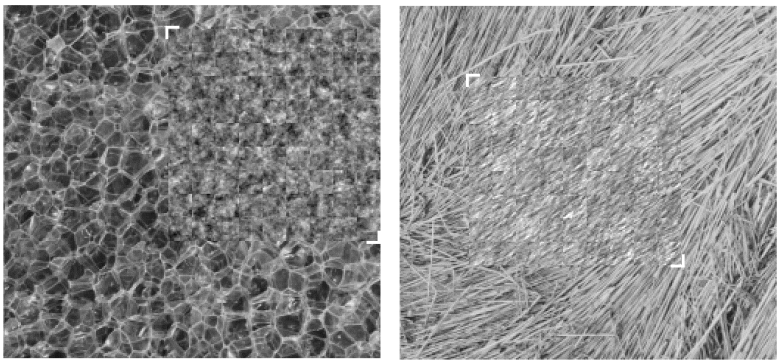

Abbildung 2.6: Die Bereiche zwischen den weißen Häkchen wurden in beiden Bildern resynthetisiert. Die auf der Hauptkomponentenanalyse basierende Resynthese funktioniert am besten auf ungeordneten Strukturen (wie der Blasenstruktur in der linken Abbildung), wogegen in der stark ausgerichteten Struktur des Strohs (rechts) Blockartefakte deutlicher hervortreten.

Das bedeutet für die Generierung neuer Textur, dass bei Verwendung von Pixel-Blöcken der Größe 8x8 lediglich die ersten 50 Eigenvektoren linear kombiniert werden müssen, um neue Textur zu generieren. Die restlichen - quasi Nullvektoren - liefern fast

keinen Beitrag. Tatsächlich kann man sogar mit deutlich weniger Vektoren auskommen, um später neue Textur zu generieren. Eine Kommunalität (siehe Abschnitt 2.3.1) von 0,8 ist meist vollkommen ausreichend.

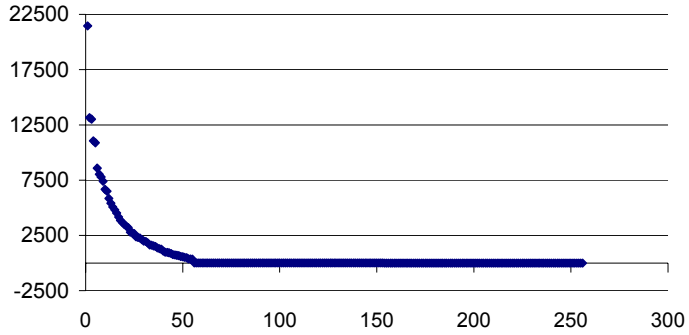

Abbildung 2.7: 256 Eigenwerte wurden ihrer Größe nach sortiert abgetragen. Typischerweise sind nur etwa 20%-25% von Null verschieden.

Da die optimale Transformation T nun gefunden ist, muss jeder Texturblock transformiert werden und die gewonnenen Koeffizienten müssen analysiert werden. Mathematischer ausgedrückt wird für jeden Block eine Basistransformation ausgeführt, nämlich vom Ortsraum (dem Raum der Grauwerte) in das Koordinatensystem, welches auf der Basis der Korrelationsmatrix berechnet wurde. Nach dem Basiswechsel wird für jeden Koeffizienten der Mittelwert und die Varianz ermittelt. Neben den wichtigsten Eigenvektoren der Transformation T sind Mittelwert und Varianz für die ersten 20% der Koeffizienten die einzigen Werte, die gespeichert werden müssen. Allein in diesen Werten ist die Charakteristik der Textur enthalten. Denn eine Analyse der Verteilung der Koeffizienten zeigt, dass sie grob einer Gaussverteilung entsprechen - eine Eigenschaft, die für Grauwerte nicht typisch ist. Eine genaue statistische Analyse der Verteilung der Koeffizienten findet im nächsten Abschnitt statt.

Die eigentliche Resynthese neuer Texturblöcke erfolgt nun durch Generierung künstlicher gaussverteilter Koeffizienten mit den ermittelten Mittelwerten und Varianzen. Die Rücktransformation in den Ortsraum erfolgt durch die zu T inverse Transformation T^{-1}. Da die Eigenwerte schnell null werden, müssen für die meisten Koeffizienten keine Zufallszahlen erzeugt werden.

Abbildung 2.6 zeigt im linken Bild eine grobe Blasenstruktur, im rechten sieht man Stroh mit einer deutlichen Ausrichtung nach rechts oben. In beiden Bildern wurde zwi-

schen den kleinen weißen Haken, ein rechteckiger Bereich synthetisierter Textur, eingebettet. Außerhalb der durch die Haken gekennzeichneten Rechtecke ist das originale Photo zu sehen, auf dessen Grundlage die Textur analysiert wurde. Die Blasen im linken Beispiel sind dem Original recht ähnlich. Das Stroh im rechten Beispiel hat eine relativ regelmäßige Struktur mit einer deutlichen Vorzugsrichtung. Zufällig bzw. chaotisch ist die Struktur also eher senkrecht zu ihrer Vorzugsrichtung. Innerhalb der Blöcke kann die stark gerichtete Struktur durchaus nachgebildet werden, jedoch haben die Blöcke untereinander keine Verbindung, so dass Artefakte sichtbar werden. Trotzdem bleibt die Richtung und die Granularität des Strohs grob erhalten. Grundsätzlich kann man sagen, dass die Resynthese besonders gut funktioniert, wenn der Grad der Zufälligkeit der Textur groß ist. Abbildung 2.8 zeigt ein Beispiel einer Sandstruktur, in dem die Synthese fast perfekt funktioniert.

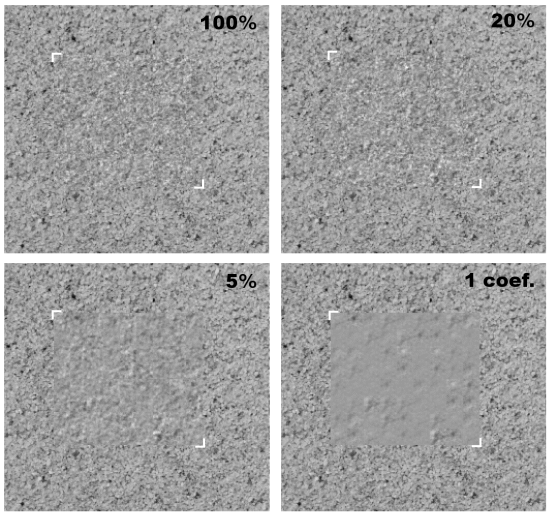

Abbildung 2.8: Alle Bilder zeigen das originale Photo. Nur der Bereich zwischen den kleinen weißen Haken wurde mit einer unterschiedlich großen Anzahl von Koeffizienten (bzw. Zufallszahlen) resynthetisiert.

Um den Einfluss der Eigenwerte auf die Bilderzeugung deutlich zu machen, wurde in Abbildung 2.8 die Textur bei der Resynthese mit einer unterschiedlichen Anzahl von Koeffizienten erzeugt. Rechts unten wurde nur ein Koeffizient verwendet. Die fehlende Anzahl von Freiheitsgraden zeigt sich ganz offensichtlich. Denn abgesehen vom konstanten Grauwert, der jedem Block a priori zugewiesen wird, kann der eine Koeffizient nur eine konstante Körnung mehr oder weniger hinzufügen. Daher ist auch in jedem Block ein ähnliches Muster erkennbar. Dieses grobe Muster ist jedoch eine gute Veranschaulichung des ersten Eigenvektors, der durch den ersten Koeffizienten hinzuge-

mischt wird. Verwendet man 5% der Koeffizienten zur Steuerung der wichtigsten Eigenvektoren, so erzeugen diese bereits ein deutlich überzeugenderes Bild (links unten). Zwischen 20% und 100% der Koeffizienten im oberen Teil der Abbildung ist kaum noch ein Unterschied zu erkennen. Die vorangegangene Analyse der Eigenwerte ließ dies bereits vermuten. Da lediglich etwa 20% der Eigenvektoren benötigt werden, kann auch die Matrix T^{-1} deutlich vereinfacht gespeichert werden, es reichen also die ersten 50 (= 20%) Spaltenvektoren aus, weniger ist auch möglich. In Ausdruck 2.12 können also die $a_{*,k}$ für k > 50 vernachlässigt werden.

$$e = \begin{pmatrix} e_1 & \dots & e_i & \dots & e_{n*n} \end{pmatrix}$$
$$T^{-1} = \begin{pmatrix} a_{1,1} & \dots & a_{1,i} & \dots & a_{1,n*n} \\ a_{2,1} & \dots & a_{2,i} & \dots & a_{2,n*n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n*n,1} & \dots. & a_{n*n,i} & \dots & a_{n*n,n*n} \end{pmatrix} \qquad \text{(Ausdruck 2.12)}$$

Pseudocode der Texturresynthese:

1. Führe eine Hauptachsentransformation durch

1.1 Ermittle die Korrelationsmatrix C, die die Korrelation der Pixel untereinander wiederspiegelt. Verwende dazu alle vom Benutzer definierten Texturblöcke.

1.2 Berechnet die Eigenvektoren von C. Jeder Eigenvektor von C ist nachfolgend ein Spaltenvektor der Matrix T. Die Eigenvektoren sollen, entsprechend ihrer Eigenwerte, in absteigender Reihenfolge geordnet sein. Die Matrix T hat die Eigenschaft einen Texturblock so zu transformieren, dass auch die durch die Transformation entstehenden Koeffizienten nach ihrer Wichtigkeit geordnet sind.

2. Führe die Texturanalyse durch

2.1 Transformiere nun alle benutzerdefinierten Texturblöcke durch die Matrix T.

2.2 Berechne für jeden Koeffizienten dessen Mittelwert und Varianz

3. Führe die Textursynthese für beliebig viele Blöcke durch

3.1 Erzeuge für jeden Koeffizienten eine Zufallszahl. Jedoch sollen dessen Erwartungswert dem zuvor berechneten Mittelwert entsprechen. Auch die Varianz soll erhalten bleiben.

3.2 Transformiere die zufällig erzeugten Koeffizienten mittels der Inversen Matrix T^{-1} in den Ortsraum und stelle die dadurch entstehenden Grauwerte dar.

2.2.4 Statistische Eigenschaften der Koeffizienten

Wie bereits zu Anfang erwähnt, kann die Texturresynthese dazu verwendet werden, Teile eines Bildes in visuell ähnlicher Weise neu zu erzeugen, statt sie in einer Datei zu speichern oder über ein Netzwerk zu übertragen. Um dabei gegenüber einer konventionellen Kompression effektiv Daten zu sparen, ist es wichtig, nur ein notwendiges Minimum an Informationen zu einer gegebenen Textur zu speichern. Neben den oben erwähnten wichtigsten Eigenvektoren müsste eigentlich auch die Verteilungsfunktion jedes Koeffizienten gespeichert werden. Erst auf Grundlage dieser Verteilung ist es möglich, zufällige Koeffizienten zu erzeugen, die nach der Rücktransformation in den Ortsraum die gewünschte Textur erzeugen.

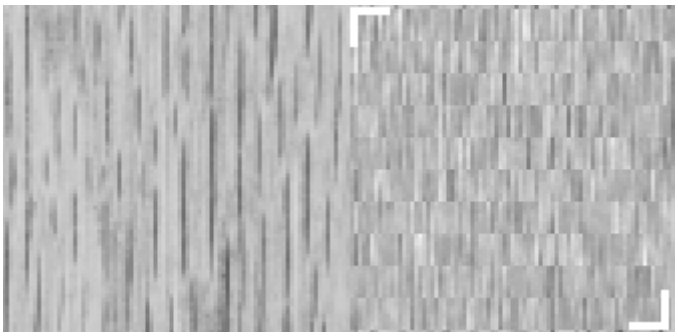

Abbildung 2.9: Holztextur, die die in Abbildung 2.10 gezeigte Verteilung der Koeffizienten verursacht.

Die exakte Speicherung der Verteilungsfunktion würde dazu beitragen, das gegenüber der herkömmlichen Kompression gesparte Speichervolumen wieder aufzuzehren. Daher ist es günstiger, wenn die Verteilungsfunktion a priori bekannt ist und als implizite Information fest in den Dekoder integriert werden kann. Tatsächlich geht die hier vorgestellte Texturresynthese von der Annahme aus, dass die Koeffizienten näherungsweise gaussverteilt sind, was im folgenden zu belegen ist.

Im Gegensatz zur Gaussfunktion kann die wahre Verteilungsfunktion durch endlich viele Stichproben, hier durch endlich viele Texturblöcke, nicht ermittelt werden. Statt dessen wird der Wertebereich der Koeffizienten in eine Anzahl I gleich großer Intervalle geteilt. Fällt der Wert eines Koeffizienten in ein bestimmtes Intervall, so wird dessen Intervallzähler N_i um eins inkrementiert. Zum Schluss wird jeder Zähler durch die Anzahl N der Ziehungen dividiert, um die relative Häufigkeit des Intervalls zu erhalten. Jede Ziehung im statistischen Sinn bedeutet die Analyse eines weiteren Texturblocks.

Da die Gaussverteilung ihrerseits stetig ist, muss auch sie in gleichgroße Intervalle unterteilt werden, um die Flächenstücke F_i unter der Kurve zu integrieren. Im Programm *TextureResynthesis* wird diese Analyse im Rahmen der Resynthese gleich durchgeführt. Dazu musste die Gaussfunktion numerisch integriert werden, da die Stammfunktion der Gaussverteilung, also deren Massefunktion, nicht analytisch geschlossen dargestellt werden kann. Die Anzahl der Unterteilungen ist zwar willkürlich gewählt, hat jedoch auf den weiter unten verwendeten χ^2 (Chi-Quadrat)-Test keinen Einfluss, da sich dieser an die Anzahl der Freiheitsgrade, also die Anzahl der Intervalle, anpasst.

Die in Abbildung 2.9 gezeigte Holzstruktur wurde in Blöcke der Größe 8x8 aufgeteilt. Nach der Abbildung der Pixel durch die Karhunen-Loeve Transformation (also die maximal dekorrelierende Transformation im Sinne dieses Kapitels) wurde die Verteilung eines jeden der 64 Koeffizienten analysiert und die Verteilung der relativen Häufigkeiten in Abbildung 2.10 dargestellt. Die breiten Balken repräsentieren die relativen Häufigkeiten der tatsächlichen Verteilung der Koeffizienten. Die schmalen Balken stehen für die integrierte Gaussverteilung im jeweils gleichen Intervall. Die unterschiedlichen Balkenbreiten dienen lediglich zur optischen Unterscheidung.

Wie ist die Darstellung zu interpretieren? Wenn die Balken der Gaussverteilung (also F_i) und der gemessenen Verteilung eines Koeffizienten (also N_i / N) exakt übereinstimmen, kann zumindest die These nicht verworfen werden, dass der entsprechende Koeffizient tatsächlich gaussverteilt sein könnte. Streng genommen ist damit aber auch nichts bewiesen. Denn die gemessene Verteilung ist zum einen nicht stetig, zum anderen könnte das "gute Ergebnis" Zufall sein. Je weiter die Höhen schmaler und breiter Balken voneinander abweichen, desto unwahrscheinlicher wird jedoch die Annahme der Gaussverteilung. Genau diese Differenz wird entsprechend Ausdruck 2.13 durch den χ^2-Anpassungstest ermittelt - ein in der Literatur häufig beschriebener Standardtest [HEI79].

$$\chi^2 = \sum_{i=1}^{I} \frac{\left(N_i / N - F_i\right)^2}{F_i} \qquad \text{(Ausdruck 2.13)}$$

Wie bereits angedeutet, könnte eine scheinbar passende Gaussverteilung zufällig bei einer Variablen auftreten, die tatsächlich dieser Verteilung gar nicht entspricht. Diesen Irrtum bezeichnet man in der Statistik als Fehler 2. Art. Ebenso könnte die Annahme der Gaussverteilung aufgrund einer schlechten Stichprobe fälschlicherweise abgelehnt werden, obwohl sie eigentlich zutrifft - was ein Fehler 1. Art wäre. Im Fall eines χ^2-Anpassungstests wird daher eine Irrtumswahrscheinlichkeit (für den Fehler 1. Art) festgelegt. Im Rahmen dieses Tests wurde hierfür 5% gewählt. Trifft die Annahme der Gaussverteilung tatsächlich zu, so wird diese These dennoch im Mittel in 5% der Fälle verworfen. Im Fall der Holztextur konnte die These für insgesamt 56 Koeffizienten nicht wiederlegt werden und musste für acht Koeffizienten verworfen werden. Es liegt nahe, die acht verworfenen Verteilungen mit den 5% in Beziehung zu setzen, die zu erwarten waren. Im Mittel sollten dies bei 64 Koeffizienten daher nur 3,2 sein. Es muss jedoch nicht zwingend zutreffen, dass alle Koeffizienten die gleichen Eigenschaften haben. So könnte es sein, dass manche der Gaussverteilung besonders gut entsprechen, dies für andere aber überhaupt nicht zutrifft. Bei insgesamt zwölf Intervallen (Balken) und einer Irrtumswahrscheinlichkeit von 5%, darf die durch χ^2 gezeichnete Abweichung 25,19 nicht übersteigen. In der Literatur sind Tabellen [WET67] zu Ablehnungsbereichen weit verbreitet und wurden daher nicht noch einmal in diese Arbeit aufgenommen. Beim Prüfen des Ablehnungsbereiches ist jedoch zu beachten, dass die Ablehnungsgrenze nicht in der Zeile zwölf (wegen der zwölf Intervalle), sondern in Zeile 10 abgelesen wird. Dies liegt darin begründet, dass bereits zwei Freiheitsgrade zur Ermittlung des Mittelwertes und der Varianz verbraucht wurden. Denn der wahre Erwartungswert und die tatsächliche Varianz konnten nur anhand der Stichprobenwerte geschätzt werden.

Die acht verworfenen Verteilungen wurden durch unterbrochene Ovale markiert. Manchmal sehen sich Verteilungen mit sehr unterschiedlichen Fehlern optisch ähnlich. Dies ist z. B. bei Koeffizient (1, 6) und (1, 7) der Fall. Es ist aber zu beachten, dass die Differenzen am äußeren Rand stärker in die Gewichtung eingehen als Differenzen in der Mitte, da die relativen Häufigkeit der Gaussverteilung in Ausdruck 2.13 im Nenner steht. An den Rändern der Verteilung bedeutet dies, dass durch besonders kleine Werte

geteilt wird, was zu einer starken Gewichtung führen kann. Die relativ gute Anpassung der Koeffizienten an die Gaussverteilung ist jedoch nicht bei jeder Art von Struktur garantiert. In Abbildung 2.11 wurde die Verteilung der Koeffizienten gemessen, die für die Resynthese des Bildes der Zahlentabelle aus Abbildung 2.5 verantwortlich ist. Hier konnten lediglich 5% der Koeffizienten nicht verworfen werden. Grundsätzlich passen sich die Koeffizienten der Gaussverteilung optisch gar nicht so schlecht an. Die Monotonie der Verläufe ist gleich und auch das Maximum befindet sich bei beiden Verteilungen meist an der gleichen Stelle. Die dennoch große Abweichung χ^2 kommt durch den starken Ausschlag bei den beiden mittleren Intervallen zustande. Da der Fehler bei der Berechnung in der Summe zusätzlich quadriert wird, entstehen so sehr hohe Gesamtfehler.

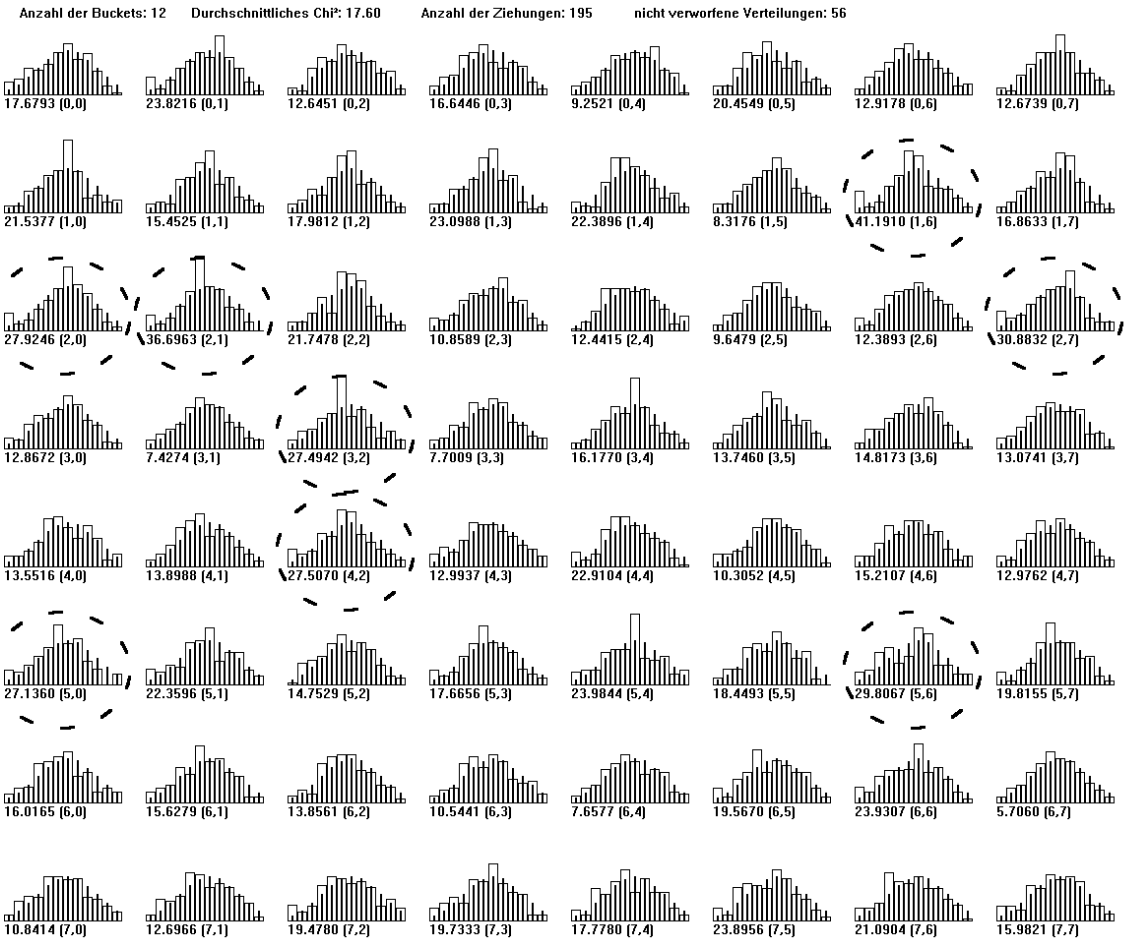

Abbildung 2.10a zeigt die grafische Darstellung der Verteilung eines jeden Koeffizienten von (0, 0) bis (7, 7) für die Texturanalyse aus Abbildung 2.9.

Eine Abgrenzung zwischen Textur und für sich bedeutungsvoller Bildinformation kann keine Statistik leisten. Ein gutes Beispiel hierfür ist Abbildung 2.5 (links). Die Zahlen-

kolonnen wurden hier als Textur aufgefasst und eine ähnliche Struktur resynthetisiert. Ein noch deutlicheres Beispiel zeigt Abbildung 2.10b. Zu sehen sind vier Blöcke des *Data Matrix* Barcodes, einer Kodierungstechnik der Firma RVSI Acuity CiMatrix. *Data Matrix* wurde mittlerweile als Public Domain Spezifikation veröffentlicht und verdrängt in Industrie und Handel auf vielen Gebieten den herkömmlichen eindimensionalen *EAN-13* Barcode des Uniform Code Councils. Ohne weiteres Wissen interpretiert man die Blöcke als zufälliges Rauschen einfarbiger Pixel, die von einem festen Rahmen umgeben sind.

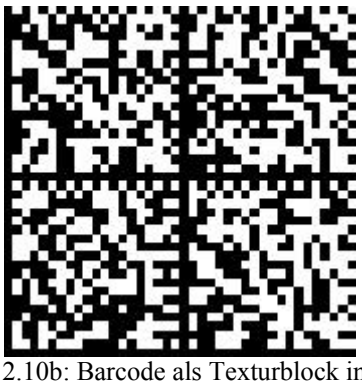

Abbildung 2.10b: Barcode als Texturblock interpretiert.

Liest man den Barcode jedoch nach der Spezifikation, so erschließen sich einem die Pixel als ASCII kodierte Text nach der im Standard festgelegten Reed-Solomon Kodierung. Der Rahmen dient der Synchronisation mit dem Bitmuster. Die Interpretation des Bilde als Rauschen (Textur) oder als ASCII Kodierung (an sich bedeutungsvolle Bildinformation), hängt im gewählten Beispiel ganz vom Betrachter ab.

In diesem Kapitel wurde unterstellt, dass die erzeugten Koeffizienten gaussverteilt sind. Ob diese Unterstellung zulässig ist hängt von der Textur ab, die resynthetisiert werden soll. In wieweit eine Abweichung der echten Verteilung von der Gaussverteilung zulässig ist, hängt dagegen ganz vom Betrachter und der Trägheit der menschlichen Wahrnehmung ab.

Sind die Koeffizienten wirklich gaussverteilt?

Im Fall der Synthese der Haare aus Abbildung 2.5 (rechts) betrug das mittlere χ^2 bei nur zwei verworfenen Verteilungen lediglich 12,84, d. h. einem Wert, der deutlich unter der Grenzen des Ablehungsbereiches von 25,19 liegt. Die Annahme scheint hier also richtig zu sein. Auch der visuelle Eindruck der resynthetisierten Blöcke ist zufriedenstellend. Sie könnten ohne weiteres dem echten Bild entnommen worden sein.

Reynthetisiert man das zufällige Rauschen aus dem obigen Beispiel, so entsteht ein ebenso zufälliges Rauschen. Jedoch wären die Koeffizienten alle vollkommen gleichverteilt. Paradoxerweise steht in diesem Fall einem visuell zufriedenstellenden Ergebnis eine vollkommen falsche Annahme gegenüber.

Welche Bedeutung kommt der Verteilungsfunktion überhaupt noch zu, wenn visuell überzeugende Ergebnisse wie in den beiden zuletzt genannten Beispielen mit und ohne Gaussverteilung zustande kommen? Die Erklärung könnte darin liegen, dass der Mensch ein gewisses Maß an Redundanz benötigt, um Bildinhalte zu erkennen. Bilder mit vollkommenem Rauschen scheinen für den Betrachter alle gleich auszusehen, obwohl ein Signal mit maximaler Entropie aus Sicht der Kodierungstheorie eigentlich ein Höchstmaß an Informationen enthält. Offensichtlich braucht der Mensch ein gewisses Maß an Redundanz, um in Bildern sinnvolle Inhalte wahrzunehmen. Man könnte daher vermuten, dass die Koeffizienten nur insofern einer spezifischen Verteilungsfunktion gehorchen müssen, wie die Koeffizienten auch Informationen enthalten, die für den Menschen bedeutungsvoll sind. Hat ein Koeffizient eine Auswirkung auf das Bild, welche ein Betrachter ohnehin nicht erkennt, so spielt auch die Verteilung des Koeffizienten keine Rolle. Das würde erklären, warum auch Bilder mit Koeffizienten, die der Gaussverteilung nicht gut entsprechen immer noch visuell überzeugende Ergebnisse erzielen können.

Als Fazit der Untersuchung kann man festhalten, dass die Karhunen-Loeve Transformation dazu führt, dass die i. d. R. nicht gaussverteilten Grauwerte in Koeffizienten transformiert werden, die zumindest bezüglich ihrer Monotonie und Extrema der Gaussverteilung recht nahe kommen bzw. ihr in vielen Beispielen auch entsprechen. Eine Re-

synthese ist auf dieser Grundlage einfacher durchzuführen, als dies im Ortsraum möglich wäre.

Offen bleibt, was eigentlich die Eigenschaften eines Bildes sind, die der Mensch in erster Linie wahrnimmt. Die Verteilungsfunktion scheint eine Rolle zu spielen, es muss aber noch andere Kriterien geben die bis zum heutigen Tag nicht gut verstanden sind.

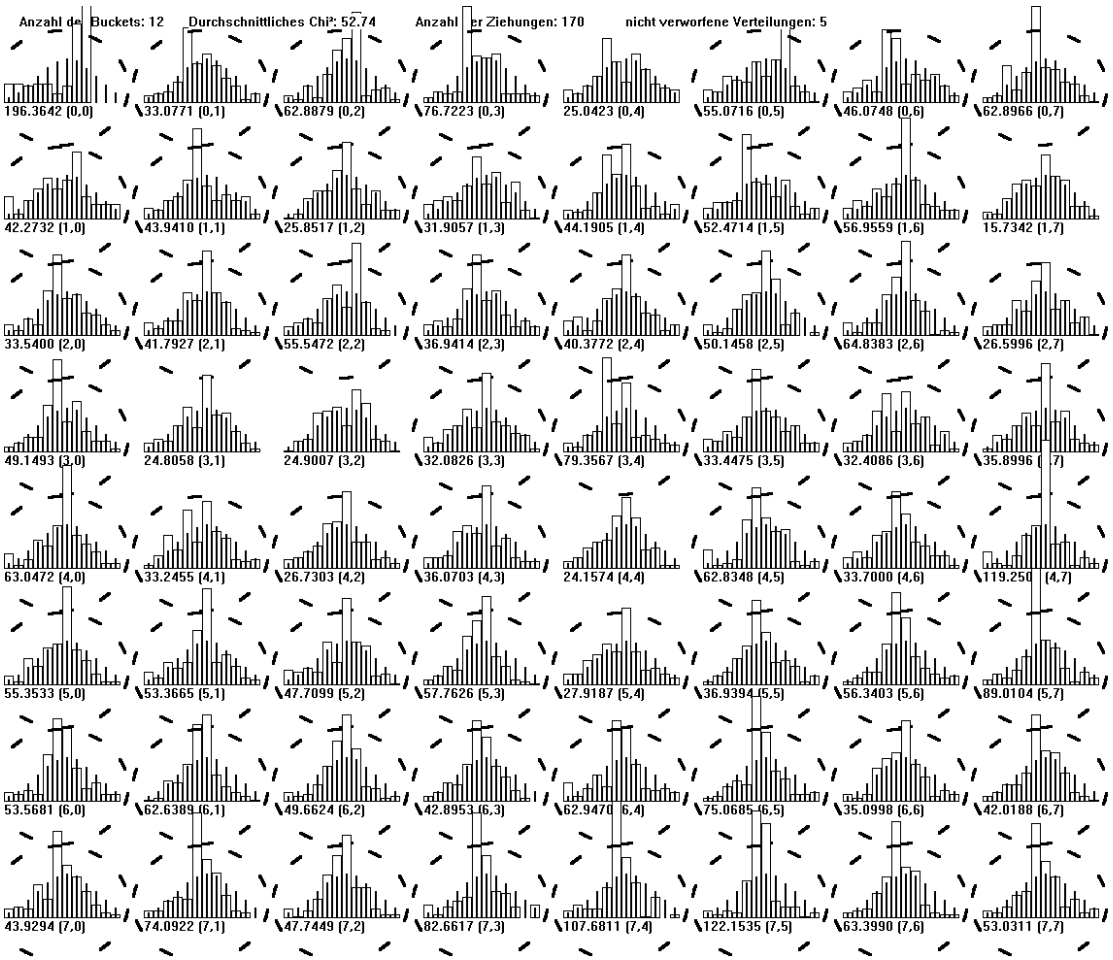

Abbildung 2.11: Verteilung der Koeffizienten des Tabellenbildes aus Abbildung 2.5. Offensichtlich passen sich die Koeffizienten der Gaussverteilung hier nicht gut an.

2.2.5 Anwendungsbeispiel *TextureResynthesis*

Eine konkrete Anwendung des Verfahrens durch das Beispielprogramm *TextureResynthesis* ist in Abbildung 2.12 gezeigt. Die Katze wurde explizit gespeichert, der Hintergrund synthetisiert. Da sich der Teppichboden in zwei unterschiedlich stark beleuchtete Bereiche aufteilt, wurde der helle und der dunkle Bereich getrennt analysiert. Aber

auch der dunkle Bereich enthält unterschiedlich beleuchtete Felder. Dies spiegelt sich in der Resynthese wieder, in der auch Blöcke mit leicht unterschiedlicher Helligkeit erzeugt werden. Für die praktische Anwendung des Verfahrens zur Kompression von Bildern ist es also empfehlenswert, den mittleren Grauwert für jeden Block explizit zu ermitteln, abzuspeichern und vom Bild zu subtrahieren. Erst auf dem so normierten Bild sollte die Analyse stattfinden. Die unterschiedliche Helligkeit der Blöcke sollte man nicht dem Zufall überlassen, sondern besser exakt wiederherstellen.

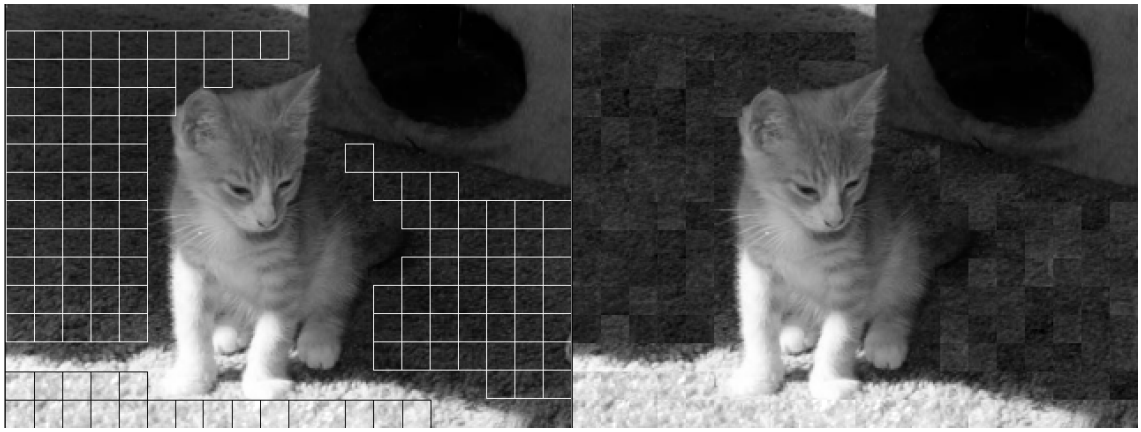

Abbildung 2.12: Das linke Bild zeigt das originale Photo. Der zu resynthetisierende Teil wurde mit Kästchen der Größe 16x16 markiert. Im rechten Teil wurde der gleiche Bereich durch Zufallszahlen resynthetisiert.

Es kostet nicht viel Speicher, die grobe Approximation des Bildes zu erhalten, die mittleren Grauwerte können sogar differenziell mit minimalem Aufwand gespeichert werden. Dass die restliche Texturinformation künstlich ist, stört das Auge deutlich weniger als kleine Fehler in den tiefen Frequenzanteilen des Bildes.

2.2.6 Unterdrückung von Blockartefakten

Die dem blockbasierten Verfahren immanenten Artefakte können nach der Generierung reduziert werden. Wie in Abbildung 2.12 gezeigt, werden hierzu zusätzliche Blöcke über den Grenzen von je vier Bildblöcken generiert. Nähert man sich im Bild einer Grenze an, so muss zwischen dem originalen und dem zusätzlichen Block in der Art einer Überblendung interpoliert werden, so dass genau auf der Grenze nur noch der Zusatzblock für die Bildinformation verantwortlich ist. Mit fortschreitender Konvergenz gegen

die Mitte des originalen Blockes muss der Einfluss wieder abnehmen. Zwar werden die Blockartefakte auf diese Weise vollständig eliminiert, die Textur bekommt aber ein etwas geglättetes Aussehen. Daher sollte zumindest die Geschwindigkeit der Überblendung der Textur empirisch angepasst werden. Je nach Größe der Regionen, in denen Textur analysiert bzw. wiederhergestellt werden soll, kann man auch die Blöcke vergrößern, so dass automatisch weniger Grenzen entstehen. Abgesehen von der Tatsache, dass der Rechenaufwand polynomial anwächst und, wie bereits erwähnt, eine gewisse Anzahl von Blöcken für die Statistik benötigt werden, ist der Blockgröße keine Grenze gesetzt.

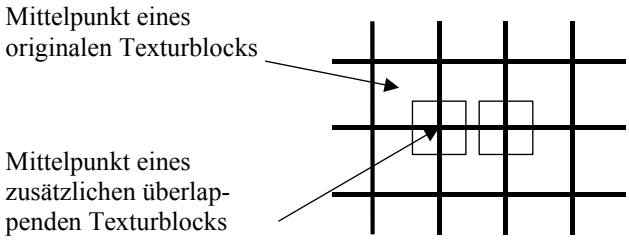

Abbildung 2.13: Das fett gedruckte Raster soll synthetisierte Texturblöcke veranschaulichen. Die dünnen, überlappenden Blöcke werden zur Interpolation verwendet, um Blockartefakte zu unterdrücken.

2.2.7 Potenzial für die Kompression

Für die reine Erzeugung von Textur müsste man sich um das Speichervolumen für die statistischen Daten keine Gedanken machen. Daher könnte man eine genauere Verteilungsfunktion für jeden Koeffizienten speichern, anstatt die Gaussverteilung zu unterstellen.

Soll die Texturresynthese jedoch zum Zweck der Kompression genutzt werden, so muss analysiert werden, in welchen Fällen ein Vorteil zu erwarten ist. Betrachtet werden im folgenden eine Anzahl b von Bildblöcken der Größe NxN mit gleichartiger Textur. Der Speicherplatz für die Texturinformation muss also kleiner als bN^2 sein.

Die Texturinformation erzeugt folgenden Aufwand: Die Eigenvektoren wurden wie oben erläutert aus der Korrelationsmatrix berechnet. Da ein einzelner Bildblock N^2 Pixel enthält, existieren N^2 Eigenvektoren der Größe N^2. Wie bereits erläutert, sind meist nur

etwa 20% der Eigenwerte signifikant größer Null. Zu jedem Eigenvektor muss aber noch ein Mittelwert und eine Standardabweichung gespeichert werden, um die näherungsweise gaussverteilten Koeffizienten erzeugen zu können. In Ausdruck 2.14 beschreibt die linke Seite den Speicherplatz für die originalen Texturblöcke, die rechte den Platz für die abstrakte Texturinformation.

$$bN^2 \geq \frac{N^2\left(N^2+2\right)}{5} \quad \text{für } N^2 \neq 0 \text{ folgt}$$
$$b \geq \frac{\left(N^2+2\right)}{5} \qquad \text{(Ausdruck 2.14)}$$

Die 2 in der Klammer steht für Mittelwert und Varianz, die für jeden Eigenwert gespeichert werden müssen. Die Teilung durch 5 resultiert aus den etwa 20% Eigenwerten > 0. Hat man sich also auf eine Blockgröße NxN festgelegt, so kann man bereits ermitteln, wie viele Texturblöcke ein Bild mindestens enthalten muss, damit sich die Anwendung der Texturresynthese lohnt.

Zur weiteren Steigerung der Effizienz kann man möglicherweise die Kommunalität geringfügig absenken, ohne dabei starke visuelle Einbußen hinnehmen zu müssen und so den Nenner in Ausdruck 2.14 noch etwas vergrößern. Wie bereits zu Anfang des Kapitels erwähnt, beschreibt die Kommunalität, in welchem Maß die Varianz der Zufallsvariablen, oder in diesem Kontext in welchem Maß die Lebendigkeit der Textur, wiederhergestellt werden soll. Eine Kommunalität von eins bedeutet, dass man bezogen auf Abb. 2.7 den gesamten Flächeninhalt unter der Kurve erhalten will. Lockert man diese Forderung jedoch leicht, so könnte man durch Weglassen der kleinsten Eigenwerte nahe der Abszisse, die schon beinahe Null sind, auch die entsprechenden Eigenvektoren einsparen. Geht man dabei von den kleinsten Werten ganz rechts behutsam nach links vor, so verzichtet man, bezogen auf das Integral unter der Kurve, nur auf wenig Flächeninhalt, kann aber evtl. einige Werte einsparen. Ein Bestehen auf einer Kommunalität von eins ist also nur bedingt sinnvoll.

Die linke Seite der Gleichung könnte insofern kritisch beurteilt werden, als die Bildinformation bN^2 auch bei einer konventionellen Kompression kaum im Verhältnis 1:1 ge-

speichert werden muss. Selbst bei verlustfreier Kompression sind Raten von etwa 2:1 zu erwarten. Auf der anderen Seite gilt vermutlich auch für die Eigenvektoren, dass eine gewisse Redundanz enthalten ist. Wie stark ein Bild durch die DCT Komprimierbar ist hängt ebenso vom Bild ab, wie die zu erwartende Redundanz in den Eigenvektoren.

Zuletzt ein Beispiel: Bei einer Blockgröße von 8x8 Pixeln gilt die Ungleichung für $b > (64+2)/5=13,2$. Ab 14 Texturblöcken würde sich die Texturresynthese bei einer relativ strengen Kommunalität von 1 bereits lohnen. Digitalkameras der aktuellen Generation erzeugen 3-5 Mio. Pixel pro Bild, so dass bereits 0,018% - 0,3% gleichartige Texturregion ausreichen würde, um die Kompression zu verbessern. Welche Kompressionsraten dabei letztendlich zu erzielen sind, hängt davon ab, welche unterschiedlichen Texturen in welchen Mengen im Bild vorhanden sind. Ist überhaupt keine Texturinformation im Bild enthalten, so kann auch nur konventionell komprimiert werden. Das Bild einer Rasenfläche könnte man dagegen bis auf einige niederfrequente Wellen, die man wie früher erwähnt immer explizit speichern sollte, evtl. vollständig mit einem Speicheraufwand erzeugen, der dem von 14 Bildblöcken entspricht. Eine vergleichende Bewertung mit anderen Kompressionsverfahren auf Grundlage der Peak Signal-to-Noise-Ratio (kurz PSNR - einer Form des mittleren quadratischen Fehlers - vergleiche Kapitel 1.6) wäre kaum sinnvoll, da ja nicht die Intention verfolgt wird das tatsächliche Bild zu rekonstruieren, sondern nur ein visuell ähnliches. Eine Metrik zwischen zwei Bildern, die eher der menschlichen Wahrnehmung entspricht, wurde von C. Kuhmünch [KUH01] vorgeschlagen.

KAPITEL 3

DISKRETE COSINUS-TRANSFORMATION

3.1 Eigenschaften der DCT

Mit den in Kapitel 2 beschriebenen Überlegungen zu Transformationen im Allgemeinen, ist die Diskrete Cosinus-Transformation (DCT) nun eine von vielen möglichen Drehungen eines Bildes in dessen Raum. Bei der DCT werden die Basisvektoren durch Cosinusschwingungen unterschiedlicher Frequenzen erzeugt. Der erste Koeffizient heißt DCT-Koeffizient (engl. Direct Current oder deutsch Gleichstromkomponente) - eine sprachliche Anleihe aus der Elektrotechnik. Er nimmt eine Sonderrolle ein, da er einer Schwingung von null Hertz entspricht. Dieser Komponente kommt besondere Bedeutung bei der Kodierung konstanter Signalanteile zu. Eine genauere Bewertung der Basisvektoren, besonders unter Berücksichtigung der menschlichen Wahrnehmung, folgt weiter unten. Alle anderen Koeffizienten werden als AC-Koeffizienten (engl. Alternating Current oder deutsch Wechselstromkomponente) bezeichnet und kodieren im Gegensatz zum konstanten Anteil ausschließlich die variablen Anteile eines Signales, denn das Integral aller AC-Koeffizienten ist null. Schon deswegen ist eine DC-Komponente für die Kodierung unverzichtbar. Im Rahmen der Bildanalyse zur Segmentierung wird die DCT dazu verwendet, Bildausschnitte vollständig in eine Frequenzdarstellung zu zerlegen, in der (im Gegensatz zur Wavelet-Transformation) keine Aussagen mehr über den Ortsraum gemacht werden können.

Grundsätzlich lassen sich unendlich viele orthonormale Basen finden, die eine Zerlegung eines Bildes mit DCT-ähnlichen Eigenschaften erzeugen.

3.2 Blockweise Darstellung

Die standardisierten Implementierungen der DCT transformieren ein Bild nicht als Ganzes, sondern arbeiten lediglich auf kleineren Blöcken. Der JPEG-Standard [ISO 10918] verwendet 8x8-Blöcke, ebenso wie die standardisierte Videokompression MPEG2. Andere Kompressionsformate auf Basis der DCT erlauben auch größere oder kleinere Blöcke, auch mit unterschiedlichen Seitenlängen, so z. B. der ITU (International Telecommunication Union) Standard H.26L. Grundsätzlich werden Bilder aber niemals als Ganzes, sondern immer blockweise transformiert. Der Grund hierfür liegt zumindest bei der Standbildkompression darin, dass die Anwendung der DCT auf das gesamte Bild zu aufwändig wäre (die Transformation eines Vektors mit n Komponenten erfordert eine NxN Matrix). Bei der Videokodierung nutzt man die Blockstruktur zusätzlich zur Kompensation von Bewegung aus, d. h. Blöcke werden wenn möglich nur verschoben, statt neu kodiert. Im Rahmen dieser Arbeit werden die Bildausschnitte vollständig DC-transformiert, zum einen weil die Transformation der Analyse und nicht der Kompression dient, zum anderen da die Anwendung weniger zeitkritisch ist als z. B. die Videokodierung, bei der viele Vollbilder in der Sekunde verarbeitet werden müssen.

3.3 Bezug zur Fourieranalyse

Wie bereits oben erwähnt, ist die Wahl der diskreten Zerlegung eines Multimediasignales in Cosinusschwingungen, wie sie 1974 von Ahmed, Natarajan und Rao [AHM74] vorgeschlagen wurde, nicht aus sich heraus selbstverständlich. Zum einen lässt sie sich jedoch statistisch herleiten (näheres hierzu in Kapitel 3.5), zum anderen liegt eine enge Verwandtschaft zur Fouriertransformation (FT) vor. Bei der Fouriertransformation werden Signale in überlagerte Sinus- und Cosinusschwingungen zerlegt. Dabei entstehen komplexwertige Koeffizienten, die sowohl die Phase als auch die Amplitude jeder Frequenz enthalten.

In Zusammenhang mit der Multimediaanalyse bringt das Aufblähen eines reellwertigen Signales in komplexwertige Koeffizienten keinen Vorteil. Die Cosinus-Transformation unterscheidet sich von der FT nur insofern, als keine Sinusschwingungen verwendet werden und daher der komplexe Anteil fehlt. Ein weiterer Unterschied besteht darin, dass die Phase einer Schwingung nicht Ergebnis der Analyse ist, sondern implizit mit jedem Koeffizienten (fest) definiert ist.

Die Basis der DCT lässt sich aber auch aus der typischen Grauwertverteilung von Photos direkt herleiten. Mit dieser statistisch motivierten Herleitung [JAI89] beschäftigt sich der nächste Abschnitt.

Historischer Exkurs: Die Fourieranalyse hat bis heute großen Einfluss auf die Mathematik und insbesondere auf die Ingenieurwissenschaften. Fourier selbst könnte man guten Gewissens als Vollblutmathematiker bezeichnen. Aus den späteren Aufzeichnungen seines Landsmannes V. Cousin *Notes biographiques pour faire suite à l'éloge de M. Fourier*, die 1831 veröffentlicht wurden, geht hervor, dass Fourier Kind einer zwölfköpfigen Familie war. Nachdem im Alter von zehn Jahren bereits beide seiner Eltern gestorben waren, konnte Jean Baptiste Joseph Fourier dennoch die königlich militärische Akademie von Auxerre besuchen. Cousin berichtet in seiner Veröffentlichung vom ungewöhnlich starken Interesse des dreizehnjährigen Fourier an Mathematik. Dabei soll er am Tag Kerzenstümpfe gesammelt haben. Nachts schlich er heimlich in sein Klassenzimmer, um bei Kerzenschein seinen mathematischen Studien nachzugehen.

Neben seinem Interesse an Mathematik schrieb Fourier anlässlich eines längeren Aufenthaltes in Ägypten ein Buch über das Land. Einigen heutigen Ägyptologen ist Fourier daher für dieses Buch bekannt, ohne dass diese von seinen mathematischen Entdeckungen wüssten.

Abbildung 3.1: Jean Baptiste Joseph Fourier (1768-1830)

Während der Wirren der Französischen Revolution verstrickte sich Fourier in einige gefährliche Situationen. Nur die Tatsache, dass die Guillotine den großen Revolutionär Robespierre zuerst ereilte, ersparte sie Fourier. In den nachfolgenden, ruhigeren Jahren der Republik, in denen Fourier endlich auch zu akademischen Ehren gekommen war, hatte er 1822 Gelegenheit seine bahnbrechende Entdeckung der Fourieranalyse unter dem Titel *Théorie analytique de la chaleur* zu veröffentlichen.

3.4 Das Markov-Modell

Das im Folgenden kurz vorgestellte Markov-Modell wird im Abschnitt 3.5 gebraucht, um ein Modell der typischen Grauwertverteilung aufzustellen, auf dessen Grundlage die DCT fußt.

Eine Sequenz von Zufallsvariablen wird als Markov-Modell p-ter Ordnung bezeichnet, wenn die Wahrscheinlichkeitsverteilung einer Variablen u(n) an der Stelle n der Sequenz vollständig durch die bereits realisierten vorher gehenden p Zufallsvariablen gegeben ist. Mit anderen Worten dürfen die vor (n-p) liegenden Werte keine Rolle mehr spielen. Formal ausgedrückt gilt folgender Sachverhalt:

$$\begin{aligned}&\text{Prob}[u(n) \mid u(n-1),(n-2),...,u(1)] = \\ &\text{Prob}[u(n) \mid u(n-1),(n-2),...,u(n-p)]\end{aligned} \qquad \text{(Ausdruck 3.1)}$$

Anschaulich kann man sich ein Markov-Modell wie folgt vorstellen. Gegeben sei ein p-dimensionaler Würfel der Kantenlänge eins. Jede Kante des Würfels entspricht einer Zufallsvariablen. Haben sich diese alle realisiert, so wird durch dieses Zufallsereignis genau ein Punkt im Würfel "adressiert". Mit jedem Punkt des Würfels ist dann eine Wahrscheinlichkeitsverteilung assoziiert.

Die Anforderung, diese Verteilungsfunktionen für große Werte von p für jeden Punkt zu kennen, ist in der Praxis meist unrealistisch, da zu komplex. Daher werden bevorzugt Modelle mit p=1 betrachtet.

Ist zusätzlich bekannt, dass die einzelnen Zufallsvariablen einer Gaussverteilung folgen, so ist auch die folgende schwächere Bedingung aus Ausdruck 3.2 ausreichend, bei der lediglich die Erwartungswerte gleich sein müssen.

$$E(\text{Prob}[u(n) \mid u(n-1),(n-2),...,u(1)]) = E(\text{Prob}[u(n) \mid u(n-1),(n-2),...,u(n-p)]) \qquad \text{(Ausdruck 3.2)}$$

In der Bildverarbeitung hat dieser Fall allerdings nur untergeordnete Bedeutung, da Grauwerthistogramme i. d. R. nicht gaussverteilt sind.

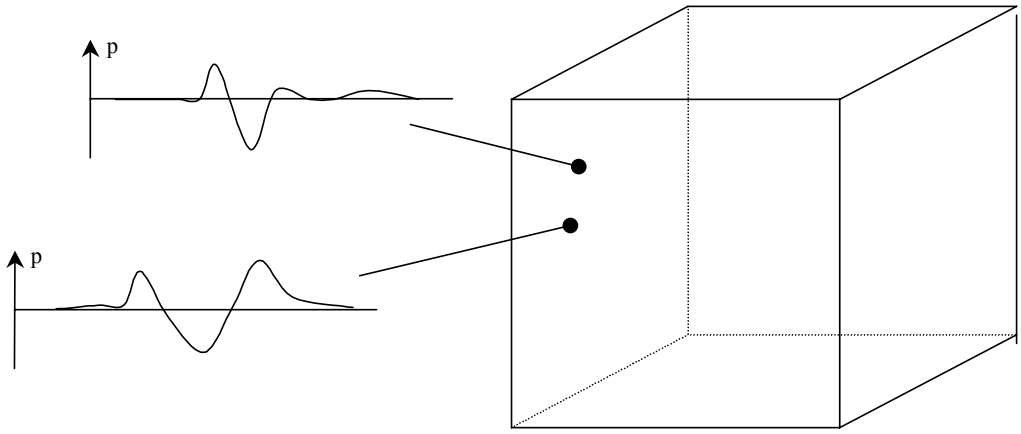

Abbildung 3.2: Für jeden Punkt des (hier dreidimensionalen) Raumes ist in einem Markov-Modell eine eigene Wahrscheinlichkeitsverteilung definiert.

Grundsätzlich macht die Annahme eines Markov-Modelles für Bilder Sinn. Denn meist gehören benachbarte Pixel zum gleichen Objekt, so dass sich ein Grauwert mit einer hohen Wahrscheinlichkeit in seiner Umgebung fortsetzt.

3.5 Eignung der DCT für Analyse und Kompression

In der Bildverarbeitung wird gern die folgende Korrelation für einen linearen Vektor von Grauwerten unterstellt:

$$\begin{pmatrix} 1 & p & p^2 & \dots & p^{N-1} \\ p & 1 & p & \dots & p^{N-2} \\ \dots & & & & \dots \\ \dots & & & 1 & p \\ p^{N-1} & \dots & & p & 1 \end{pmatrix} \qquad \text{(Ausdruck 3.3)}$$

Je größer der Exponent über p ($p \in [0,1]$) wird, desto unwahrscheinlicher wird der Fall, dass ein Bildpunkt seinem Nachbarn ähnelt. Dies bedeutet, dass schon für kleine Abstände ein Pixel von seinen Nachbarn kaum noch beeinflusst wird. Für unterschiedliche Bildtypen kann dieser Einfluss jedoch verschieden schnell abnehmen. In texturreichen Bildbereichen ändern sich Grauwerte oft über eine Entfernung von 2-3 Pixel völlig, in gleichförmigen Bildbereichen, wie z. B. bei blauem Himmel, können vollständige Zeilen des Bildes von der gleichen Farbe dominiert sein. Ein passendes p kann also nur im Sinn eines optimalen Durchschnitts über viele Bilder gefunden werden. Genau der gleiche Vorgang wurde in Kapitel 2 exemplarisch anhand der Probandenbefragung durchgeführt. Dort wurde ermittelt, ob und in welchem Maß bestimmte Merkmale eines Produktes in der Meinung der Probanden miteinander verkoppelt sind. Ausdruck 3.3 beschreibt in grundsätzlich gleicher Weise, in welchem Verhältnis ein Pixel zu seinen Nachbarn steht.

Der Korrelationsmatrix in Ausdruck 3.3 kann man entnehmen, dass für ein gegebenes p im Intervall [0, 1] ein exponentiell schnelles Abfallen der Abhängigkeiten, mit zunehmender Entfernung, erwartet wird. Mit dem Programm *CosineCorrelation* lässt sich für ein solches prototypisches Bild und ein fest gewähltes p die optimale Basis mittels

Hauptkomponentenanalyse berechnen, so wie dies in Kapitel 2 vorbereitet wurde. In Abbildung 3.3 wurden die ersten vier Basis- bzw. Eigenvektoren abgetragen. Welche Aussage macht die Abbildung? Nimmt man an, dass sich in einem Bild benachbarte Grauwerte (durchschnittlich) so verhalten, wie in Ausdruck 3.3 wiedergegeben, so ist zu erwarten, dass Schwingung 1 aus Abb. 3.3 am besten geeignet ist, einen Großteil der Bildinformation zu kodieren. Oder mit anderen Worten, es ist zu erwarten, dass Schwingung 1 im höchsten Maß vorkommt.

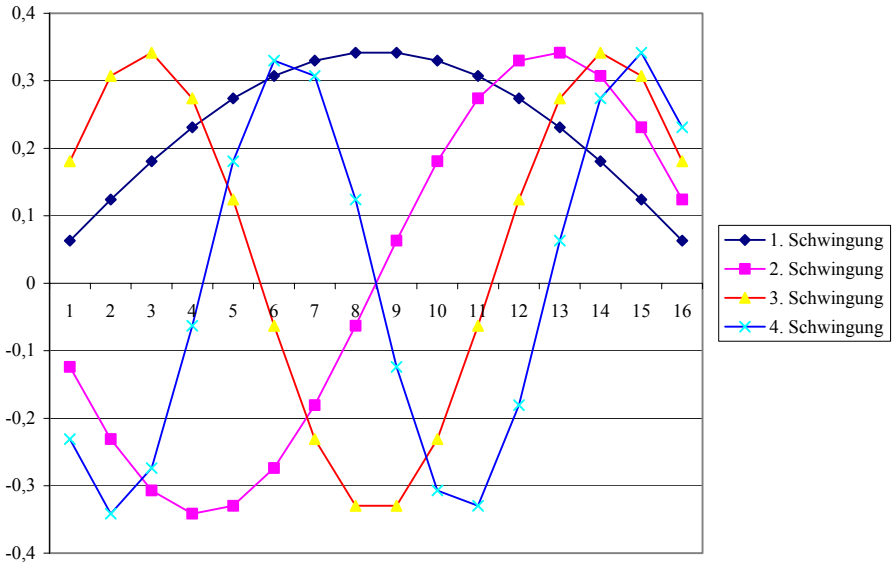

Abbildung 3.3: Die vier ersten tieffrequenten Basisvektoren für die Korrelationsmatrix 3.3 für p=0.2 bei einer Auflösung von 16 Samplepunkten.

Hat man das Bild um diesen Schwingungsanteil bereinigt, so ist zu erwarten, dass Schwingung 2 den größten Teil der verbleibenden Information aufnimmt usw. Auf diese Weise baut sich die optimale Basis Vektor für Vektor auf.

Die Ähnlichkeit mit der Basis der DCT (siehe Abb. 3.6) ist unverkennbar, jedoch mit dem Unterschied, dass in Abb. 3.3 keine konstante DC-Komponente vorkommt. Auch sind Phasen und Frequenzen leicht unterschiedlich. Grundsätzlich lässt sich aber bereits vermuten, dass auch die DCT-Basis, aufgrund der engen Verwandtschaft mit der Basis aus Abb. 3.3, den Eigenvektoren der Korrelationsmatrix 3.3 sehr ähnlich sein sollte. Zwar wurde die Basis explizit für die angegebene Korrelation 3.3 optimiert. Die Basis ist jedoch nur auf Bilder abgestimmt, deren benachbarte Pixel einen durchschnittlichen Korrelationskoeffizienten p haben. Für einen gegebenen Satz von Testbildern lässt sich

dieses p eindeutig bestimmen. Letztlich entsteht p aber nur aus einem gewichteten Mittel von sich wiederholenden bzw. von unterschiedlichen Pixeln. In realen Bildern kommen aber neben weitgehend unkorrelierten Bereichen auch solche vor, in denen Grauwerte über längere Strecken konstant sind. Unkorrelierte Grauwerte lassen sich nicht sinnvoll durch spezielle Basisvektoren berücksichtigen. In ihrem Fall ist sozusagen jede Basis geeignet, da sich die Energie des Bildes nicht konzentrieren lässt.

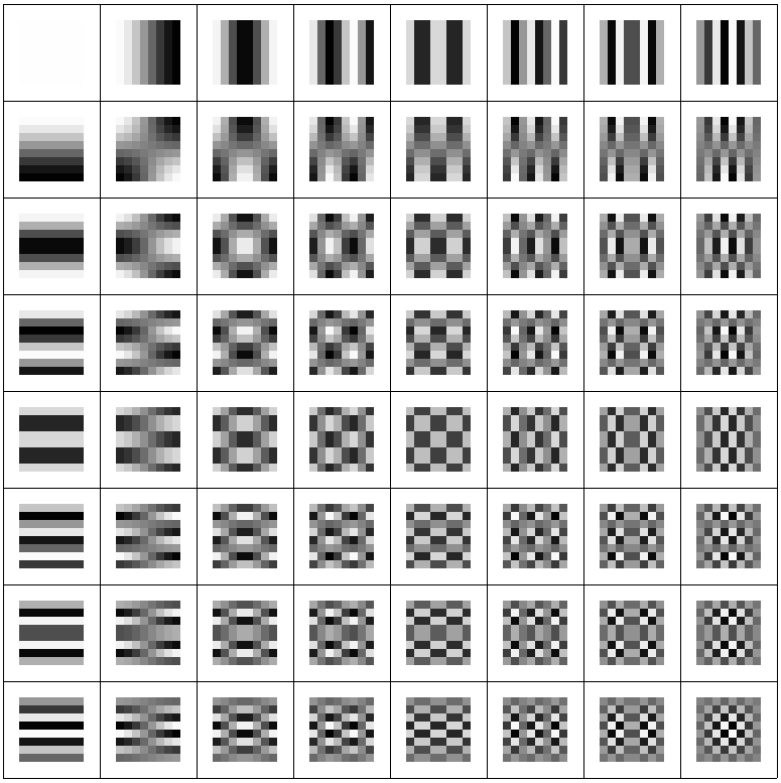

Abbildung 3.4: Vollständige Basis der DCT für die Transformation von 8x8 Blöcken. Jedes der 64 Kästchen entspricht einem Basisvektor. Helle Bereiche entsprechen positiven Werten, das mittlere Grau entspricht 0 und dunkle Schattierungen repräsentieren Werte kleiner Null.

Die in der Praxis häufig vorkommenden Bereiche konstanter Grauwerte können dagegen besonders gut durch eine DC-Komponente erfasst werden. Dies ist einer der wenigen Fälle anschaulich verständlicher Basisvektoren, denn die DC-Komponente repräsentiert ja gerade selbst den konstanten Grauwertanteil eines Bildes.

Mit der Basis aus Abbildung 3.3 können natürlich auch konstante Signalbereiche kodiert werden (neben den vier abgetragenen sind hierzu aber alle Basisvektoren nötig). Allerdings setzt sich ein konstantes Signal dann aus einer verhältnismäßig komplexen

Linearkombination zusammen. Sind konstante Bereiche wahrscheinlich, so sollten diese im Sinne der Kodierungseffizienz möglichst einfach repräsentiert werden. Daher eignet sich die DC-Basis für reale Bilder besser.
Neben der Kodierungseffizienz spielt auch die menschliche Wahrnehmung eine Rolle. Würde man bei der Basis aus Abbildung 3.3 einzelne Koeffizienten quantisieren, so würde aus einem konstanten Signal ein leicht wellenförmiges, da die Konstante eine eher fragile Linearkombination vieler Schwingungen ist. Etwas bekannter ist dieses Phänomen im Fall der Fouriertransformation einer Rechteckfunktion. Diese besteht aus einer unendlichen Anzahl von Frequenzen. Stört man diese z. B. durch Quantisieren ab einem bestimmten Frequenzbereich, so werden die Überschwinger an den Kanten der Rechtecke schnell deutlich. Ein vergleichbarer Effekt würde bei der oben gezeigten Basis in konstanten Bereichen eines Signales entstehen.

Die DCT ist dagegen absolut unempfindlich. Zwar kann auch der DC-Koeffizient mehr oder weniger gestört werden, dies wirkt sich aber nicht durch Überschwinger aus, da die Konstante lediglich etwas nach oben oder unten verschoben wird.

Sieht man vom Nutzen des DC-Koeffizienten ab, so könnte man auch die in Abbildung 3.5 dargestellte Sinustransformation verwenden. Die Tatsache, dass bei ihr die "Gleichstromkomponente" fehlt, zeigt sich darin, dass es kein gleichmäßig graues oder weißes Kästchen gibt, also keinen Basisvektor mit einheitlichen Werten.

Interessant ist nun die umgekehrte Frage, wie die Korrelationsmatrix zur gegebenen Basis der DCT aussieht. Ausdruck 3.4 zeigt das Ergebnis.

$$\begin{pmatrix} (1-\alpha) & -\alpha & 0 & \dots & 0 \\ -\alpha & 1 & -\alpha & \dots & 0 \\ 0 & -\alpha & & & \dots \\ \dots & & \dots & 1 & -\alpha \\ 0 & \dots & & -\alpha & (1-\alpha) \end{pmatrix}$$

(Ausdruck 3.4)

Im Gegensatz zur Korrelationsmatrix 3.3, wird in diesem Fall die Toeplitz-Bedingung nicht mehr eingehalten.

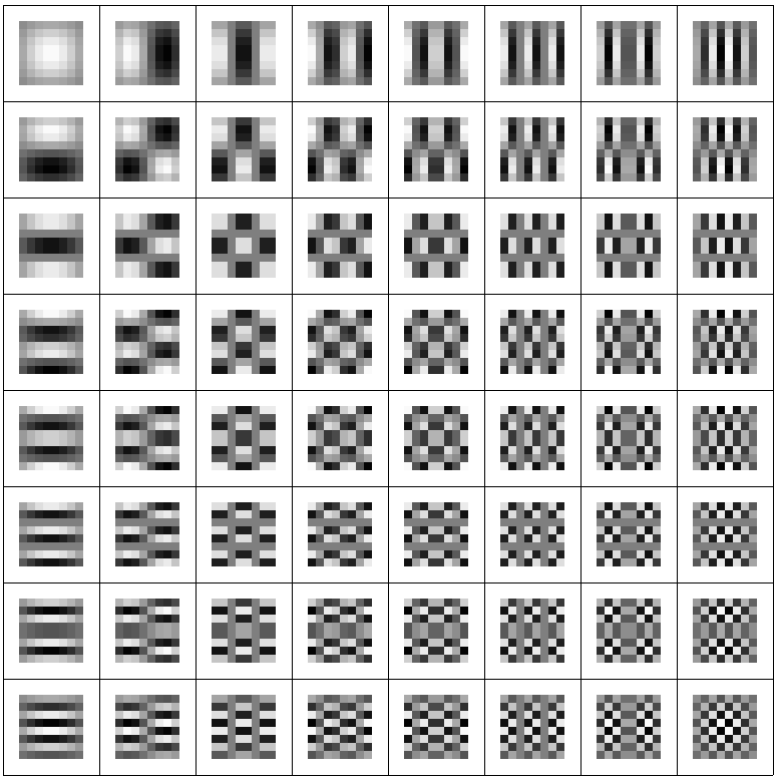

Abbildung 3.5: Die Basis der Sinustransformation.

Exkurs: Die Toeplitz-Bedingung fordert, dass die Diagonale mit einem einheitlichen Wert besetzt sein muss. Gleiches gilt auch für alle zur Diagonalen parallelen Linien in der Matrix. Weiterhin müssen die Linien über und unter der Diagonalen (im jeweils gleichen Abstand) ebenfalls den gleichen Wert besitzen. Damit kann eine n x n Matrix nur noch n unterschiedliche Werte enthalten.

Die Toeplitz-Bedingung hatte vor allem in den letzten Jahrzehnten bei der numerischen Invertierung großer Matrizen Bedeutung, wenn die gesamte Matrix sehr groß wurde. Unter Einhaltung dieser speziellen Bedingung muss die Matrix für die Invertierung nicht vollständig gespeichert werden, und die Zeitkomplexität für die Berechnung der Inversen sinkt von $O(n^3)$ auf $O(n^2)$.

Genau genommen handelt es sich nicht einmal um eine Korrelationsmatrix, da der erste und letzte Wert der Diagonalen nicht eins sind. Somit würde es, auf ein reales Problem übertragen, für alle $\alpha \neq 0$ zwei Variablen geben, die jeweils nicht vollständig mit sich selbst korreliert sind.

Zum Beweis, dass die Korrelationsmatrix 3.4 wirklich der DCT zugrunde liegt, wurde mit dem Programm *TextureResynthesis* wieder die Basis berechnet und in Abbildung 3.6 dargestellt. TextureResynthesis wurde insofern leicht verändert, als die Korrelationsmatrix fest hinein kodiert und nicht aus einem Bild ermittelt wurde. Wie zu erwarten entsteht tatsächlich die DC-Basis.

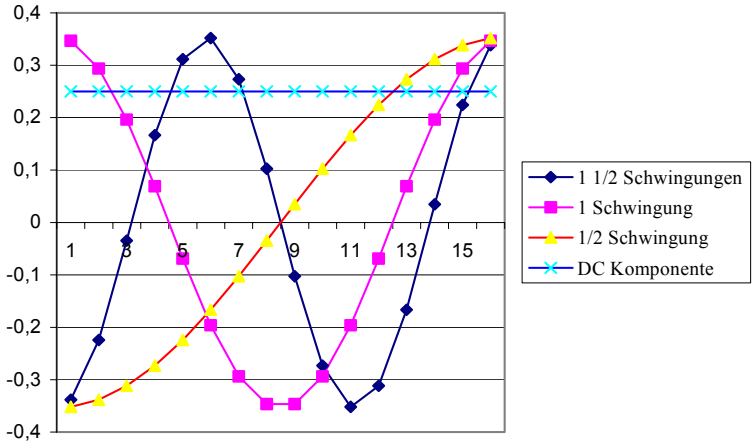

Abbildung 3.6: Abgebildet wurden die ersten vier Basisvektoren der DCT mit 16 Samplepunkten.

3.6 Die Vorwärtstransformation

In Ausdruck 3.5, der Transformation vom Orts- in den Frequenzraum, wird das Bild mit s(y, x), die DC-Koeffizienten mit S(v,u) bezeichnet. Für alle ganzzahligen $v \in [0,7]$ und $u \in [0,7]$, also für jeden der 64 Koeffizienten, muss das Bild einmal mit dem Basisvektor des Koeffizienten multipliziert werden. Ist also n die Anzahl der Pixel, so ist jedes einmal bei der Multiplikation mit jedem der n Basisvektoren beteiligt. Daher entsteht auch eine Komplexität von O(n²), von der anfänglich gesprochen wurde.

$$S(v,u) = \frac{C(v)}{2}\frac{C(u)}{2}\sum_{y=0}^{7}\sum_{x=0}^{7} s(y,x)\cos[(2x+1)u\pi/16]\cos[(2y+1)v\pi/16] \qquad \text{(Ausdruck 3.5)}$$

$$C(a) = 1/\sqrt{2} \text{ für } a = 0$$
$$C(a) = 1 \qquad \text{für } a > 0$$

Die in Ausdruck 3.5 verwendeten Vorfaktoren C(v) und C(u) dienen lediglich der Normierung der DC-Komponente. Die (orthonormale) Basis zeichnet sich ja gerade dadurch aus, dass jeder Vektor die Länge eins hat, d. h. mit sich selbst multipliziert eins ergibt. Ohne den Vorfaktor würde für den DC-Koeffizienten der eindimensionalen DCT gelten:

$$\text{Vektor der Länge 8} \left\{ \left\| \begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix} \right\|_2 = \sqrt{1^2 + \dots + 1^2} = \sqrt{8 * 1^2} \neq 1 \right. \qquad \text{(Ausdruck 3.6)}$$

Mit dem Vorfaktor $1/(2\sqrt{2})$ gilt:

$$\text{Vektor der Länge 8} \left\{ \left\| \begin{pmatrix} 1/2\sqrt{2} \\ \dots \\ 1/2\sqrt{2} \end{pmatrix} \right\|_2 = \sqrt{\left(1/2\sqrt{2}\right)^2 + \dots + \left(1/2\sqrt{2}\right)^2} = \sqrt{8\left(1/2\sqrt{2}\right)^2} = \sqrt{8\left(1/8\right)} = 1 \right.$$

3.7 Die inverse Transformation

Die inverse Transformation unterscheidet sich, wie zu erwarten, darin, dass die Informationen vom Frequenzraum zurück in den Ortsraum transformiert werden sollen. Daher stehen nun in der Summe statt den Grauwerten der Pixel die Koeffizienten. y und x werden festgehalten, während über alle Koeffizienten v, u iteriert wird.

$$s(y,x) = \sum_{v=0}^{7} \frac{C(v)}{2} \sum_{u=0}^{7} \frac{C(u)}{2} S(v,u) \cos[(2x+1)u\pi/16] \cos[(2y+1)v\pi/16] \qquad \text{(Ausdruck 3.7)}$$

3.8 Quantisierung

3.8.1 Problem der Blockartefakte

Die DCT hat gegenüber der Diskreten Wavelet-Transformation (DWT) einen wesentlichen Nachteil. Wie in Kapitel 2 ausgeführt wurde, kann man ohne Kenntnis des zu transformierenden Bildes nicht vorhersagen, ob die DCT oder die DWT die Grauwerte eines Bildes besser dekorreliert. Vielmehr liegt der Unterschied darin, dass die gängigen Implementierungen der DCT Bilder nie als Ganzes, sondern in Blöcke aufgeteilt sehen. Damit werden nur Redundanzen innerhalb der Blöcke ausgenutzt. Würde man die DCT auf das gesamte Bild anwenden, so wäre a priori nicht mehr klar, dass sie der DWT nicht ebenbürtig ist. Die Berechnung der DCT ist allerdings insofern komplexer, als alle Koeffizienten von allen Pixeln beeinflusst werden. Bei der DWT dagegen werden die meisten Koeffizienten nur von wenigen Pixeln bestimmt, oder die Abhängigkeit der Koeffizienten von den Grauwerten kann schrittweise von Skala zu Skala aufgehoben werden (hierzu mehr im Kapitel 4). Aus Sicht der Linearen Algebra könnte man sagen, dass die DWT, als Matrix betrachtet, einer Diagonalmatrix näher kommt. In der Matrix der DCT sind dagegen keine Nullen zu finden.

Exkurs: Dass in der DCT-Matrix keine Nullen zu finden sind, wird in Abbildung 3.4 anschaulich. Nur der mittlere Grauwert in den 64 Basisvektoren entspricht Nulleinträgen in der 64x64 Werte großen Transformationsmatrix. Die DCT ist aber genau so aufgebaut, dass der Cosinus an den abgetasteten Werten eben niemals null wird. In Ausdruck 3.5 kann man dies daran erkennen, dass die innere Klammer im Zähler jeweils um (+1) zu (2x+1), bzw. (2y+1) erweitert ist. Dadurch wird die Funktion nie null. Dies würde auch keinen Sinn machen, denn offensichtlich kann die Multiplikation eines Signales mit einer konstanten Null zumindest an der entsprechenden Stelle keine Information extrahieren. Würde man auf die Erweiterung des Zählers um (+1) verzichten, so wären 64 Koeffizienten nicht in der Lage, 64 Pixel zu kodieren. Als Beweis kann man die Determinante der so verfälschten Transformation berechnen und erhält eine Null. Der Würfel, dem die Matrix der DCT im 64-dimensionalen Raum entspricht, ist also in sich zusammengefallen (denn die Determinante beschreibt genau dessen Rauminhalt) und kann folglich den Raum nicht mehr aufspannen.

Die Aufteilung des Bildes in Blöcke bringt durch die Quantisierung unschöne Blockartefakte mit sich. Zwar treten Quantisierungsartefakte bei der DWT im gleichen Maß auf. Im Unterschied zur DCT können sie allerdings überall im Bild auftauchen und nicht nur an Blockgrenzen. So verteilen sich die Störungen gleichmäßig und entgehen der menschlichen Wahrnehmung eher.

Durch die Quantisierung werden Koeffizienten dem jeweils nächsten Quantisierungsintervall zugeordnet. Im Inneren eines Blockes wirkt dies weniger störend, da hierdurch keine Brüche in der Bildfunktion entstehen können.

> Beweis: Per Definition bestehen alle Basisvektoren der DCT aus Cosinusschwingungen. Da der Cosinus selbst stetig ist, kann auch durch die Überlagerung dieser Schwingungen keine Unstetigkeit auftreten. Dass die Summe stetiger Funktionen wieder eine stetige Funktion ist, wird z. B. in [FOR83] §10, Satz 1 bewiesen. Die Quantisierung der Koeffizienten ändert daran grundsätzlich nichts. Sie verändert lediglich die Gewichtung der überlagerten Schwingungen.
>
> Anders ist dies bei der Wavelet-Transformation, denn hier sind, je nach verwendetem Wavelet, die Basisvektoren selbst nicht stetig. Zwar lassen sich durch Linearkombination unstetiger Schwingungen durchaus stetige Funktionen erzeugen. Deren Stetigkeit ist aber insofern fragil, als sie bereits durch eine marginale Störung eines Koeffizienten zerstört werden kann.

Da unterschiedliche Blöcke keine "Kenntnis voneinander" haben, entstehen zwischen den Blockgrenzen Sprünge in der Bildfunktion. Es ist naheliegend, diese Sprünge nach der Rücktransformation des Bildes auszugleichen. Dies könnte durch Glätten des gesamten Bildes geschehen. Allerdings würden dadurch auch Kanten innerhalb der Blöcke geglättet.

3.8.2 AC-Prädiktion

Mit der so genannten AC-Prädiktion sollen nach einem Vorschlag von Niss [NIS88] die AC-Koeffizienten eines Blocks so angepasst werden, dass Sprünge an den Grenzen zu den Nachbarblöcken minimiert werden.

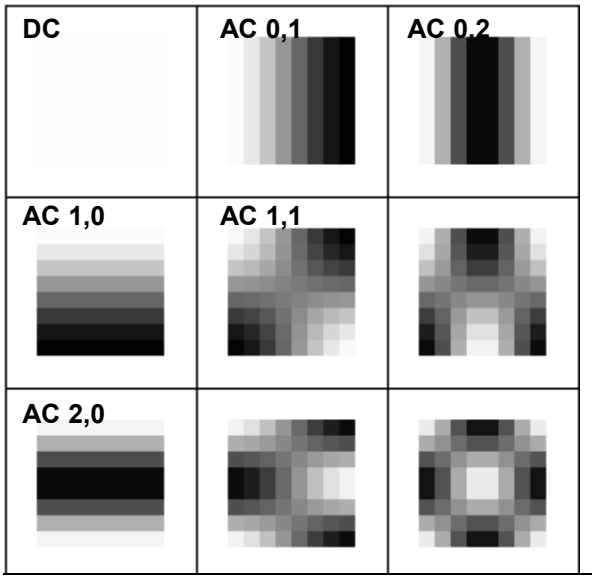

Abbildung 3.7: Die fünf bezeichneten AC-Koeffizienten werden in der AC-Prädiktion vorhergesagt.

Der zu optimierende Ziel-Block eines Bildes ist in Abbildung 3.8 in der Mitte dargestellt. Für diesen Block sollen die Koeffizienten zu den Basisvektoren (0, 1), (1, 0), (2, 0), (1, 1) und (0, 2) (siehe Abbildung 3.7) optimiert werden. Ziel der Optimierung ist dabei, die fünf Koeffizienten so anzupassen, dass die Grenzen zu den benachbarten acht Blöcken möglichst glatt werden.

Das Verfahren unterstellt einem Bild, dass es grundsätzlich glatt ist, d. h. dass die Bildfunktion keine Sprünge enthält. Wird ein solcher nach der Rücktransformation der Koeffizienten dennoch festgestellt, so muss ein Fehler vorliegen, der nur durch die Quantisierung herrühren kann.

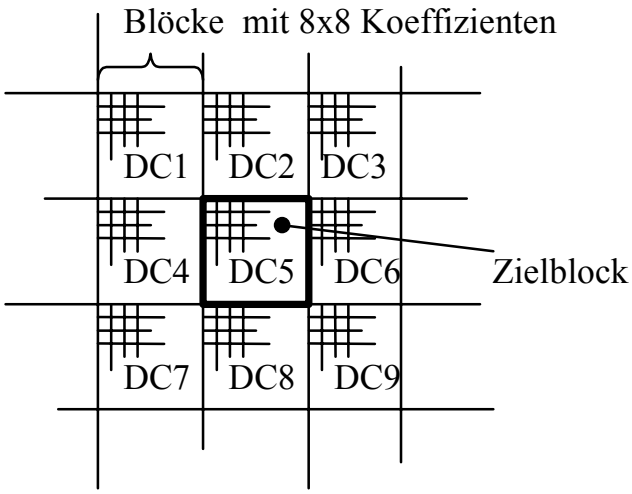

Abbildung 3.8: Zielblock mit benachbarten Bildblöcken. Die Korrektur des quantisierten Zielblockes erfolgt so, dass ein möglichst glatter Übergang zu den Nachbarn gewährleistet ist.

Um die nach Meinung der AC-Prädiktion besseren Koeffizienten zu ermitteln, geht man wie folgt vor: Zunächst wird ein Polynom über dem gesamten Bereich der neun Blöcke definiert.

$$P(x,y) = A_1x^2y^2 + A_2x^2y + A_3xy^2 + A_4x^2 + A_5xy + A_6y^2 + A_7x + A_8y + A_9 \qquad \text{(Ausdruck 3.8)}$$

Dabei werden dessen neun Koeffizienten $A_1,...,A_9$ eindeutig so optimiert, dass über jedem der neun Blöcke der mittlere Grauwert so gut wie möglich getroffen wird. In Abbildung 3.9 ist ein solches Polynom über einer Gruppe von neun Blöcken dargestellt. Der Umweg über die Berechnung eines Polynoms erfolgt lediglich aufgrund dessen leichter Optimierbarkeit z. B. durch das Gauß'sche Eliminationsverfahren.

Im Anschluss wird das eben optimierte Polynom DC-Transformiert, um die acht AC-Koeffizienten des Zielblocks zu erhalten. Die neu ermittelten Koeffizienten sind in Ausdruck 3.9 nur von den mittleren Grauwerten ihrer Nachbarn abhängig. Genau diesen Zusammenhang wollte man ja auch herstellen. Ob die so ermittelten AC-Koeffizienten besser als die gespeicherten Werte sind, wird weiter unten noch kritisch hinterfragt.

$$
\begin{aligned}
AC_{01} &= (1.13885/8)(DC_4 - DC_6) \\
AC_{10} &= (1.13885/8)(DC_2 - DC_8) \\
AC_{20} &= (0.27881/8)(DC_2 + DC_8 + 2DC_5) \\
AC_{11} &= (0.16213/8)((DC_1 - DC_3) - (DC_7 - DC_9)) \\
AC_{02} &= (0.27881/8)(DC_4 + DC_6 - 2DC_5)
\end{aligned}
\qquad \text{(Ausdruck 3.9)}
$$

Abbildung 3.7 veranschaulicht die Bildung der AC-Koeffizienten jedoch auch ohne Rechnung. So kann man sich gut vorstellen, dass der vertikale Grauwertverlauf des $AC_{1,0}$ Koeffizienten nur durch die Subtraktion der Grauwerte $DC_2 - DC_8$ entstehen kann. Denn $AC_{1,0}$ enthält keinerlei Änderung in horizontaler Richtung. Analoges gilt für die restlichen Koeffizienten.

Bei der Rekonstruktion des Bildes werden trotz der gerade berechneten Vorhersage in erster Linie die gespeicherten AC-Koeffizienten verwendet. Erst wenn der Vorhergesagte ins gleiche Quantisierungsintervall fällt wie der Gespeicherte, so wird die Vorhersage als die exaktere Wahl angesehen. Das Programm *DC-Transform* implementiert die Transformation, Quantisierung, Dequantisierung nebst Prädiktion und die Rücktransformation.

Die Idee der AC-Prädiktion ist durchaus kritisch zu sehen. Vor allem die in der Literatur gelegentlich bemühte Behauptung, man könne damit die Peak Signal-to-Noise Ratio (kurz PSNR - mehr dazu in Kapitel 1.6), also den Rauschabstand verbessern, konnte in der Implementierung nicht nachgewiesen werden. Vielmehr hielten sich zufällige Verbesserungen und Verschlechterungen in etwa die Waage.

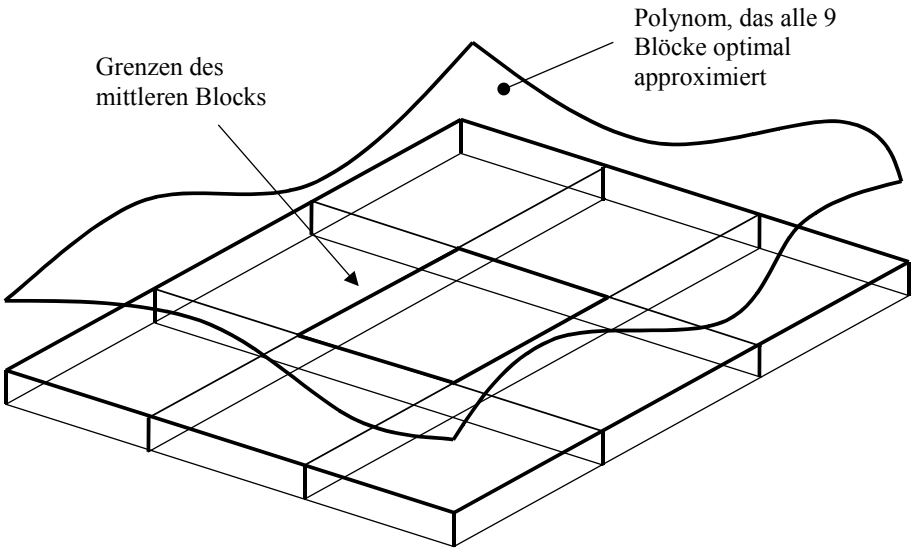

Abbildung 3.9: Die Gruppe von neun Blöcken wird durch ein Polynom approximiert (zur Verdeutlichung nach oben verschoben). Dieses Polynom soll möglichst stetig mit der Bildinformation an den Grenzen zwischen dem mittleren Block und seinen Nachbarn abschließen.

3.8.3 Kritik an der AC-Prädiktion

Kritik kann man in mehreren Punkten üben: Ziel war es, Blockartefakte möglichst gering zu halten. Zu diesem Zweck sollen die ersten fünf AC-Koeffizienten so optimiert werden, dass eine möglichst stetige Bildfunktion erzeugt wird - zumindest an den Blockgrenzen. Dabei steht die Annahme im Hintergrund, dass Bilder in der Regel stetige Funktionen sind, die nur durch die Quantisierung an den Blockgrenzen Sprünge machen.

Kritikpunkt 1: Machen hochfrequente Koeffizienten die Prädiktion kaputt?

Der in Abbildung 3.9 in der Mitte gezeigte Block wird so optimiert, als hätte er nur die fünf tieffrequentesten Koeffizienten. In der Tat wird hierdurch der Abstand zu den umgebenden mittleren Grauwerten minimiert. Nun wurden für den betrachteten Block aber in der Regel mehr Koeffizienten gespeichert, auch solche höherer Frequenzen. Fügt man dem "naiv optimierten" Block diese Koeffizienten später hinzu, so ist er an den Grenzen evtl. nicht mehr so bündig, wie dies mit der Prädiktion beabsichtigt wurde. Mit anderen Worten: Die Prädiktion nach Niss, so wie sie auch in den Empfehlungen des Standards ISO International Standard 10918 Part 1 steht (allerdings im informativen

Teil - im Gegensatz zum Normativen), geht bei der Optimierung nur von der Existenz der fünf wichtigsten AC- und dem einen DC- Koeffizienten aus und ignoriert die restlichen 58 Koeffizienten, die die Prädiktion im Nachhinein zerstören könnten.

Kritikpunkt 2: Warum werden die mittleren Grauwerte der Nachbarblöcke verwendet?

Die AC-Koeffizienten des Zielblocks werden so optimiert, dass die mittleren Grauwerte der acht Nachbarblöcke möglichst gut getroffen werden. Nun fragt sich aber, ob die Nachbarblöcke an den Grenzen überhaupt ihren mittleren Grauwert annehmen. Vielmehr könnten deren überlagerte Schwingungen dazu führen, dass sie an den Grenzen ganz andere Werte annehmen. Setzt man den DC-Koeffizient eines Nachbarn z. B. auf 0 und den ersten AC-Koeffizient auf sein Maximum, so hat man bereits den Fall, dass eine der Grenzen eine große Abweichung zum Mittelwert aufweist.

Kritikpunkt 3: Ist die Stetigkeit überhaupt erwünscht?

Starke Kanten innerhalb eines Blockes werden meist erhalten, da sie ausreichend große Koeffizienten erzeugen, die der Quantisierung nicht zum Opfer fallen. Nun könnten die gespeicherten AC-Koeffizienten aber genau so gewählt sein, dass an der Grenze eine erwünschte Kante entsteht. Eine solche Kante wird durch die AC-Prädiktion per Definition vernichtet. Berücksichtigt man, dass Blöcke mehrheitlich nur acht Pixel breit und hoch sind, so ist die Wahrscheinlichkeit für diesen Fall verhältnismäßig hoch.

Kritikpunkt 4: Warum ist die Prädiktion besser als der gespeicherte Wert?

In der Optimierung weiter oben wurden die fünf AC-Koeffizienten so angepasst, dass ein glattes Abschneiden mit den benachbarten Blöcken erreicht wird. Nun sind die fünf Koeffizienten aber bereits gegeben, z. B. in einer JPEG-Datei. Warum sollten die "künstlich" Errechneten besser als die Gespeicherten sein?

Diesem Kritikpunkt könnte man als einzigem entgegen halten, dass innerhalb des Quantisierungsintervalles eine Unsicherheit über den tatsächlichen Wert besteht und ein glattes Abschließen mit dem Nachbarblock wahrscheinlich ist. Daher sollte der Koeffizient das glatte Abschließen auch ermöglichen. In der Evaluation führte dies jedoch zu keiner signifikanten Verbesserung.

Bemerkung: Bei der Quantisierung von Koeffizienten werden die Werte durch einen bestimmten Faktor geteilt, oder die am wenigsten signifikanten Bits werden verworfen. Letzteres entspricht einer Teilung durch 2^n, wobei n die Anzahl der verworfenen Bits ist.

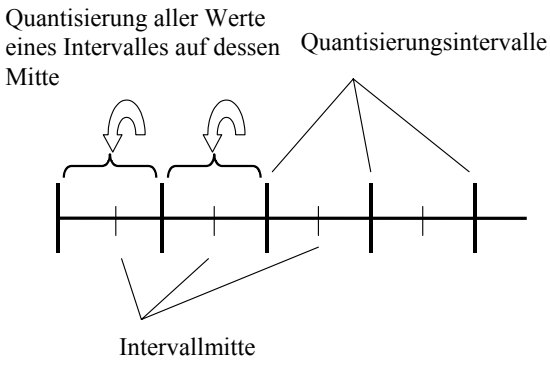

Abbildung 3.10: Quantisierung durch Runden auf die Mitte eines Intervalles

Durch die Teilung und die anschließende Dequantisierung werden alle Werte aus einem Intervall der unteren Grenze zugeordnet. Richtigerweise sollte man jedem Wert aber noch eine halbe Intervallbreite hinzurechnen, da so der mittlere Fehler minimiert wird. Vergisst man diesen Schritt bei der Implementierung des Dekoders, so bringt die AC-Prädiktion tatsächlich einen signifikanten Vorteil. Überhaupt zeigt sich, dass bereits kleine Nuancen in der Implementierung zu einer merklichen Veränderung der PSNR (dem Rauschabstand - mehr dazu in Kapitel 1.6) führen können.

KAPITEL 4

WAVELET-TRANSFORMATION UND ANWENDUNGEN

4.1 Motivation

Zum Verständnis der Wavelet-Transformation (kurz WT) ist es nützlich, noch einmal die Eigenschaften der Fouriertransformation (FT) zu betrachten. Bei der FT wird ein komplexwertiges Signal in eine Menge von überlagerten Sinus- und Cosinus-Schwingungen zerlegt. Jede Schwingung wird dabei durch einen komplexen Koeffizienten repräsentiert. Der Betrag des Koeffizienten gibt an, in welchem Maße die entsprechende Schwingung im Signal enthalten ist, also deren Amplitude [MAY97, S. 79]. Die Semantik des zerlegten Signales interessiert an dieser Stelle erst einmal nicht. Es könnte sich z. B. um Audiosignale, Messwerte oder Bilder handeln.

Ebenso wie die Cosinus-Transformation oder die WT, kann auch die Fouriertransformation kontinuierlich oder diskret durchgeführt werden. Kontinuierlich bedeutet, dass eine überabzählbar unendliche Anzahl von überlagerten Schwingungen verwendet wird, um ein Signal darzustellen.

Bemerkung: Überabzählbar unendlich sind z. B. die reellen Zahlen, da es keine Obergrenze gibt (Eigenschaft der Unendlichkeit) und da zwischen zwei ungleichen Zahlen fortwährend

weitere zu finden sind (Eigenschaft der Überabzählbarkeit). Unendlich sind zwar auch die natürlichen Zahlen, jedoch nicht überabzählbar. Man kann sie in eine Reihenfolge bringen, in der jede Zahl einen eindeutigen Platz hat. Interessanterweise ist auch die Menge der Brüche abzählbar, auch wenn dies nicht auf den ersten Blick offensichtlich ist. Es lassen sich aber Abfolgen aller Brüche konstruieren, in der jeder Mögliche an einer bestimmten Stelle in einer Folge vorkommt.

$$\begin{array}{ccc} 1/1 & 1/2 & 1/3 \\ 2/1 & 2/2 & \cdots \\ \cdots & \cdots & \cdots \end{array}$$

Die nach rechts und unten offene Matrix kann z. B. in einer zick-zack Reihenfolge abgezählt werden.

Im Falle der diskreter Transformationen können Signale meist nur durch unendlich viele Schwingungen ohne Rest erzeugt werden. Dabei werden allerdings nur abzählbar viele verwendet. Die Begriffe kontinuierliche oder diskrete Transformation haben also nichts mit der Natur des Signales zu tun. Dieses kann selbst nur an diskreten Stellen im Definitionsbereich (abgetastet) vorliegen oder überall definiert sein. Es gibt sozusagen alle vier Kombinationen.

In Ausdruck 4.1 (unten) liegt das periodische Signal s(n) innerhalb des Intervalls [0, N-1] an N Stellen abgetastet vor. Für jede ganzzahlige Frequenz v entsteht ein komplexer Koeffizient $\hat{s}_v$, der sowohl die Phase als auch die Amplitude der Schwingung mit der Frequenz v enthält.

$$\hat{s}_v = \frac{1}{N}\sum_{n=0}^{N-1} s(n)\left[\cos\left(\frac{2\pi v n}{N}\right) + i\sin\left(\frac{2\pi v n}{N}\right)\right] \qquad \text{(Ausdruck 4.1)}$$
$$0 \le v < N$$

Bemerkung: In der Literatur wird meist der Ausdruck Frequenz benutzt, auch dann, wenn sich das Signal gar nicht in der Zeit ausbreitet. Will man den Raum, in dem sich die Schwingung ausbreitet, nicht genauer spezifizieren, so kann man auch von einer *Wellenzahl* sprechen.

Im Fall der fouriertransformierten stetigen Rechteckfunktion z. B. reisst die Reihe der überlagerten Wellen niemals ab, da die Funktion innerhalb einer Periode zwei Sprünge macht. Bei der diskreten Transformation der Rechteckfunktion ist keine der beteiligten Schwingungen redundant. Läßt man trotzdem welche weg, so entstehen die bekannten Überschwinger, die vor allem an den Ecken des Rechtecks (den Flanken) deutlich werden. Abbildung 4.6 zeigt den Sachverhalt anhand einer einzelnen Rechteck-Schwingung, die vom Ortsraum (oben) in den Frequenzraum (unten) transformiert wurde.

Für kontinuierliche Transformationen diskreter Signale erhält man zumindest theoretisch sogar eine unendlich redundante Zerlegung des Signales [HUB98, S. 37]. Aus diesem Grund werden die kontinuierlichen Transformationen im Rahmen der Analyse verwendet. Aufgrund der Redundanz liegt das Signal gewissermaßen in "unterschiedlichen Interpretationen" vor. Die diskrete Transformation liefert dagegen eine eindeutige Zerlegung eines Signales und ist so auch für die Kompression geeignet.

Bemerkung: Für den Zweck der Bildverarbeitung eignet sich die FT weniger, da Bilder keine komplexen Grauwerte besitzen und daher die imaginäre Komponente Null ist. Statt dessen kann eine ähnliche Transformation, die sogenannte Hartley-Transformation, verwendet werden. Sie unterscheidet sich lediglich dadurch, dass das imaginäre i durch eine 1 ersetzt wurde, so dass ein Vektor reeller Werte in einen ebensolchen der gleichen Dimension transformiert wird. Für den diskreten Fall lautet die Gleichung

$$\hat{s}_v = \frac{1}{N}\sum_{n=0}^{N-1} s(n)\left[\cos\left(\frac{2\pi v n}{N}\right) - \sin\left(\frac{2\pi v n}{N}\right)\right] \qquad \text{(Ausdruck 4.2)}$$
$$0 \le v < N$$

Alternativ kann auch die DCT verwendet werden, in der kein Sinusterm vorhanden ist und die sich in der inneren Klammer leicht unterscheidet. Damit wird ein Signal zwar in die enthaltenen Cosinus-Wellen zerlegt, im Gegensatz zur Fourier- bzw. der Hartley-Transformation kann man aber bei der Einheit der Koeffizienten nicht von *Hertz* sprechen. Für die Analyse innerhalb einer Anwendungsdomäne, also ohne Bezug zu anderen Messungen oder Auswertungen, spielt dies keine Rolle.

Weil zur Zerlegung nur die periodischen Sinus- und Cosinus-Funktionen verwendet werden, können auch nur periodische Signale zerlegt werden. Die meisten Signale sind aber im Ortsraum lokal stark begrenzt und konvergieren für +/- Unendlich gegen null. Um solche Signale dennoch zerlegen zu können, behilft man sich in der Regel dadurch, das Signal theoretisch unendlich oft hintereinander zu hängen. Auf diese Weise kann man beliebig lange, nicht-periodische Signale, wie z. B. eine ganze Symphonie, in das Spektrum ihrer beteiligten Schwingungen zerlegen.

Einer der Nachteile der Fouriertransformation ist, dass keinerlei Ortsinformationen enthalten sind. Das bedeutet, dass ein Signal zwar vollständig in seine Frequenzanteile zerlegt werden kann, die Zerlegung aber keine Information mehr darüber enthält, an welcher Stelle eine bestimmte Frequenz auftritt.

Das gewählte Beispiel der transformierten Symphonie soll nicht dazu verleiten zu glauben, dass der Mensch ebenso hört. Vielmehr sollte man sich die Fourieranalyse als abstrakte Linearkombination vorstellen, die genau so konstruiert ist, dass ein Signal über seine gesamte Laufzeit aus Sinus- und Cosinus-Wellen zusammengesetzt wird. Vor allem bei langen Signalen ist dabei die Aussagekraft der einzelnen Frequenz schon aufgrund numerischer Ungenauigkeiten begrenzt. Auch spielt der Startpunkt der Analyse innerhalb des Signales eine erhebliche Rolle. Schon eine kleine Verschiebung kann sich deutlich auf die hohen Frequenzen auswirken, weil sich die Phasen aller Schwingungen verschieben.

Für praktische Anwendungen der multimedialen Signalanalyse ist es häufig von Interesse zu wissen, an welcher Stelle eine Schwingung auftritt. Streng genommen ist diese Frage schlecht gestellt, denn per Definition der Fourieranalyse ist es ja gerade das gesamte Signal, welches zerlegt wird. Dass die Frage dennoch gern gestellt wird, hängt auch mit der Art zusammen, wie Menschen Signale wahrnehmen bzw. erzeugen. Dabei wird eine Note immer zu einem bestimmten Zeitpunkt gehört oder vom Musiker gespielt. So kann es vorkommen, dass der Kammerton A an einer Stelle gespielt wird, die Frequenz von 440 Hz in der Zerlegung des gesamten Musikstückes jedoch kaum vertre-

ten ist. Denn das kurzzeitig gespielte A, das im übrigen Stück nicht mehr zu hören ist, ist in Wirklichkeit eine Linearkombination vieler anderer Schwingungen.

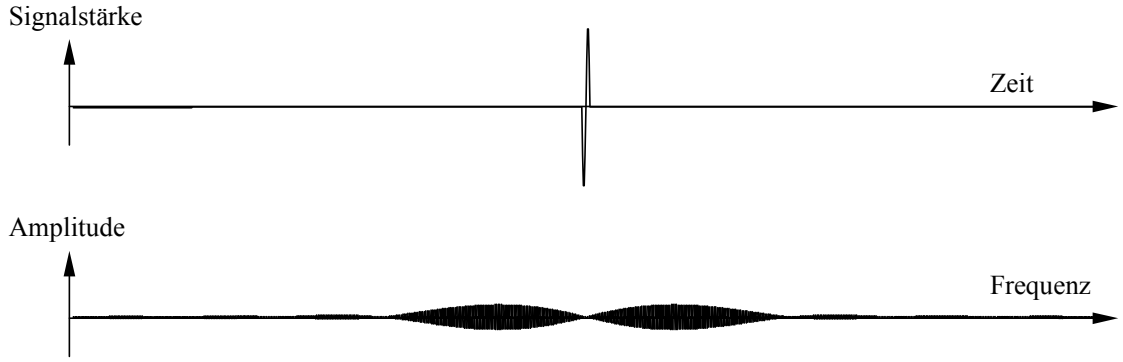

Abbildung 4.1: Signal im Ortsraum (oben) und die Hartley-Transformierte im Frequenzraum unten.

Eine einzelne Phase einer Schwingung wird in eine große Anzahl von Frequenzen zerlegt. Dabei ist es nicht mehr trivial, die eigentliche Frequenz des kurzen Signales in der Analyse zu erkennen.

Dieser Sachverhalt wird in Abbildung 4.1 dargestellt. Im oberen Teil der Darstellung, dem Ortsraum, ist ein Signal mit einer einzelnen Phase einer hochfrequenten Schwingung abgetragen. Im unteren Teil kann man die mittels Hartley-Transformation erzeugte Frequenzzerlegung sehen. Es fällt auf, dass die einzelne Schwingung in eine Vielzahl von Frequenzen zerlegt wurde.

Im Rahmen dieser Arbeit ist das Programm *Hartley* entstanden, mit dem die Beispiele in diesem Kapitel erzeugt wurden. Es stellt eine direkte Verbindung zwischen Ortsraum (im oberen Teil des Fensters) und Frequenzraum (unterer Teil) her. Nachdem man bei gedrückter linker Maustaste ein Signal im Ortsraum definiert hat, kann man mit der rechten Taste die Frequenzzerlegung im unteren Teil berechnen. Auf die gleiche Weise können auch die Koeffizienten im unteren Frequenzraum manipuliert und in den Ortsraum im oberen Teil des Fensters transformiert werden. Analytisch geschlossene Funktionen werden im Quelltext des Programmes eingegeben und zur Laufzeit erzeugt.

4.1.1 Bezug zur gefensterten Fourieranalyse

Um dennoch eine Vorstellung davon zu bekommen, wie sich das Frequenzspektrum über das gesamte Signal verändert, wird auf die so genannte *gefensterte Fourieranalyse* zurückgegriffen, die 1946 von D. Gabor [GAB46] vorgeschlagen wurde. Dabei wird ein Ausschnitt (auch Fenster) des Signales gewählt und dieser periodisch hintereinander gehängt. Nach der Analyse des gefensterten Bereichs ist gewährleistet, dass das resultierende Spektrum zwischen den Begrenzungen des Fensters auftreten muss. Je genauer die Ortsinformation dabei werden soll, desto enger müssen die Grenzen zueinander rücken. Gleichzeitig rückt dabei das zu beobachtende Frequenzspektrum immer höher, weil z. B. in einem Intervall von 200 ms keine tieferen Frequenzen als fünf Hertz auftreten können.

Das gewählte Fenster über einem Signal kann man sich wie eine gewöhnliche Rechteckfunktion vorstellen, die das Signal in einem bestimmten Intervall "passieren lässt" und außerhalb sofort auf null abfällt. Der so gewählte Ausschnitt wird, wie oben schon beschrieben, unendlich wiederholt, um künstlich ein periodisches Signal zu erzeugen.
An den Grenzen des Intervalls trifft das letzte Stück des Ausschnittes zwangsläufig wieder auf den Anfang. An dieser Stelle entsteht fast immer ein Sprung. Solche Sprünge erzeugen eine Vielzahl hochfrequenter Schwingungen, die dem Signal an sich nicht eigen sind.

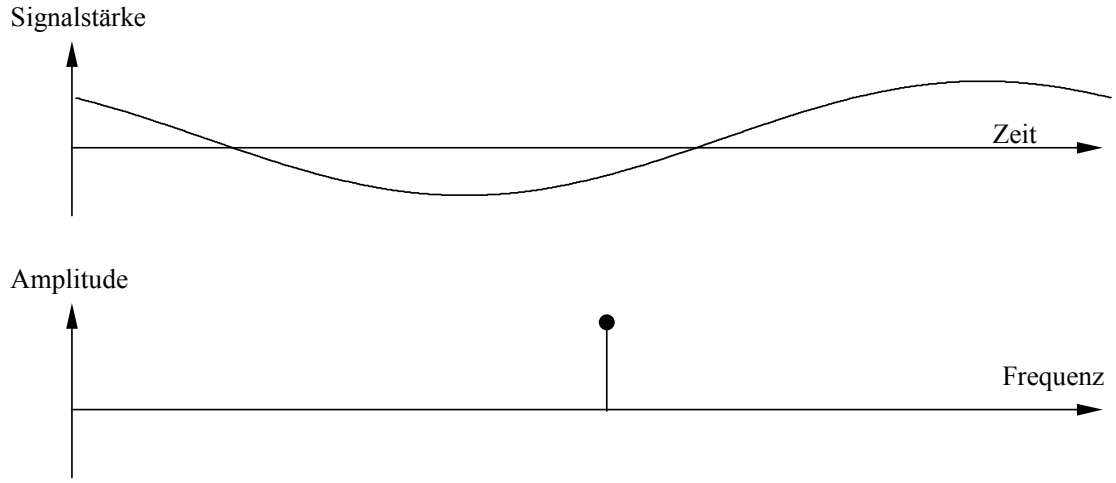

Abbildung 4.2: Im oberen Teil (Ortsraum) wurde eine ganze Schwingung erzeugt. Im unten dargestellten Frequenzraum wird deutlich, dass es sich tatsächlich nur um eine einzelne Schwingung handelt.

Daher wurde eine Reihe anderer Fensterformen vorgeschlagen, die den Übergang der Signalgrenzen besser glätten sollen [LOU98, S. 30ff].

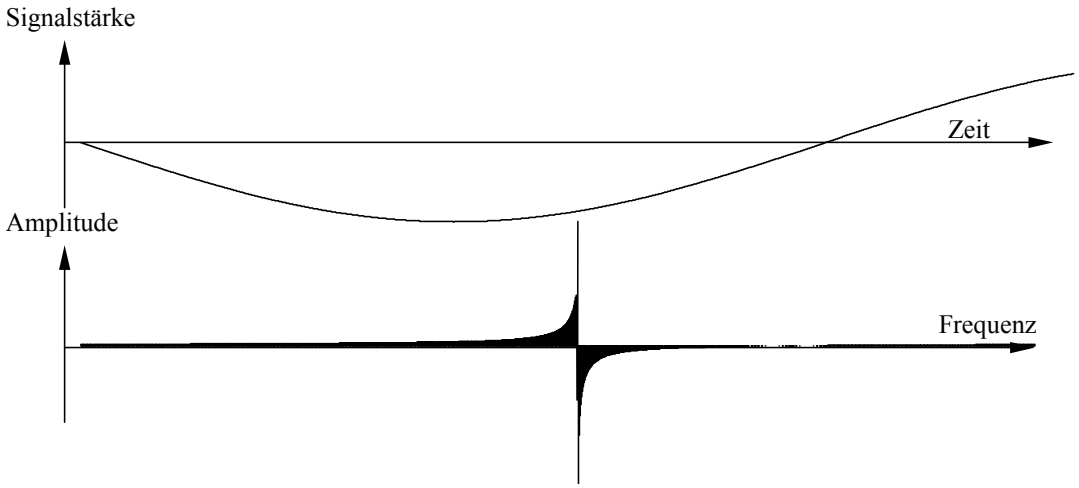

Abbildung 4.3: Die Schwingung aus Abbildung 4.2 wurde innerhalb des Fensters um eine halbe Phase verkürzt, so dass zwischen Anfang und Ende ein Sprung entsteht. Durch die Verlängerung ist ein vollkommen anderes Frequenzspektrum entstanden.

In Abbildung 4.2 wurde eine Schwingung genau so angepasst, dass sie vollständig in das gewählte Fenster passt. Entsprechend zeigt sich im Frequenzraum auch nur eine einzelne Spitze. In Abbildung 4.3 wurde die gleiche Schwingung lediglich um eine halbe Phase verkürzt. Dadurch passt der Anfang der Welle nicht mehr zu deren Ende, wenn man diese periodisch wiederholt. Durch den Sprung entsteht das im unteren Teil sichtbare Frequenzspektrum, das sich, im Gegensatz zu allen bisher Erzeugten, durch eine viel stärkere Gleichverteilung der beteiligten Schwingungen auszeichnet. Ein einfacher Rückschluss auf die ursprüngliche Schwingung ist so viel schwerer.

Das Beispiel zeigt, dass der Wahl des Fensters für die Analyse eine große Bedeutung zukommt, um das Spektrum nicht mit unnötig vielen Schwingungen zu versehen, durch die die eigentlich interessanten Informationen verdeckt werden. Das Problem der Fensterfunktion ist dem später noch beschriebenen Padding-Problem bei Wavelets eng verwandt.

4.1.2 Wavelet-Analyse als Weiterentwicklung der Fourieranalyse

Die Wavelet-Analyse ist gewissermaßen die konsequente Fortführung der gefensterten Fourieranalyse. Dabei wird ein Fenster nicht für die gesamte Transformation gewählt. Vielmehr richtet sich die Fensterbreite nach der Ausdehnung der "Analyseschwingung" (siehe Abbildung 4.4), also nach der Ausdehnung des jeweiligen Wavelets. Fensterbreiten werden also pro Skala genau so gewählt, dass eine volle Schwingung hinein passt. Genau genommen ist die Fensterfunktion bereits im Wavelet enthalten, da ein ausreichend schnelles Abfallen gegen null gefordert wird.

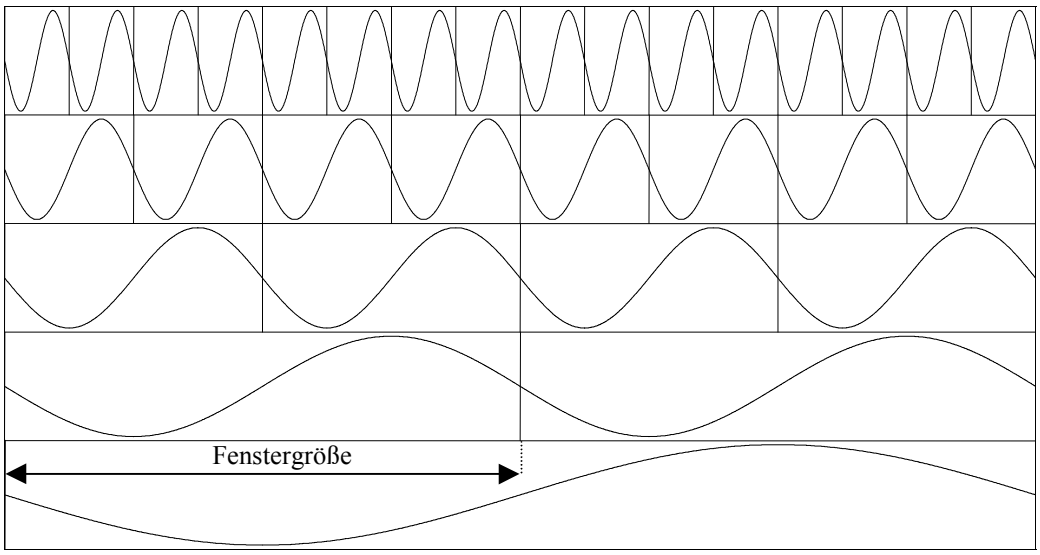

Abbildung 4.4: Bei der WT wird ein Signal auf unterschiedlichen Skalen mit Fenstern unterschiedlicher Größe abgetastet. Das Fenster für die Abtastung passt sich der Skala in der Weise an, dass die Schwingung jeweils genau einmal hineinpasst. (Hier wurde der Sinus als Schwingung nur exemplarisch verwendet, um die vollständige Phase deutlich zu machen.)

4.1.3 Die dyadische WT

Eine Schwingung von 1 Hz wird auf einem Intervall von 1000 ms beobachtet. Bei der so genannten *dyadischen Wavelet-Analyse* halbiert sich das betrachtete Intervall jeweils von einer Stufe zur nächsten. Daher würde als nächstes eine 2-Hz-Schwingung im Intervall von 500 ms analysiert werden, dann 4 Hz auf 250 ms usw., auf immer kleiner werdenden Fenstern. Im Unterschied zur Fourieranalyse erstreckt sich die Schwingung aber nicht auf das gesamte Signal, sondern immer nur auf die gerade gewählte Fenstergröße. Die 2-Hz-Schwingung kann somit innerhalb eines Signales von 1000 ms auch

zweimal analysiert werden. Daher entsteht bei der dyadischen Zerlegung in jeder Stufe (bzw. Skala) die doppelte Anzahl von Koeffizienten (siehe Abb. 4.4). Die Abgrenzung der Fenster voneinander kann abhängig vom verwendeten Wavelet unterschiedlich stark sein. Grundsätzlich kann man sagen, dass längere Filter zwangsläufig eher zur Überlappung neigen als kurze.

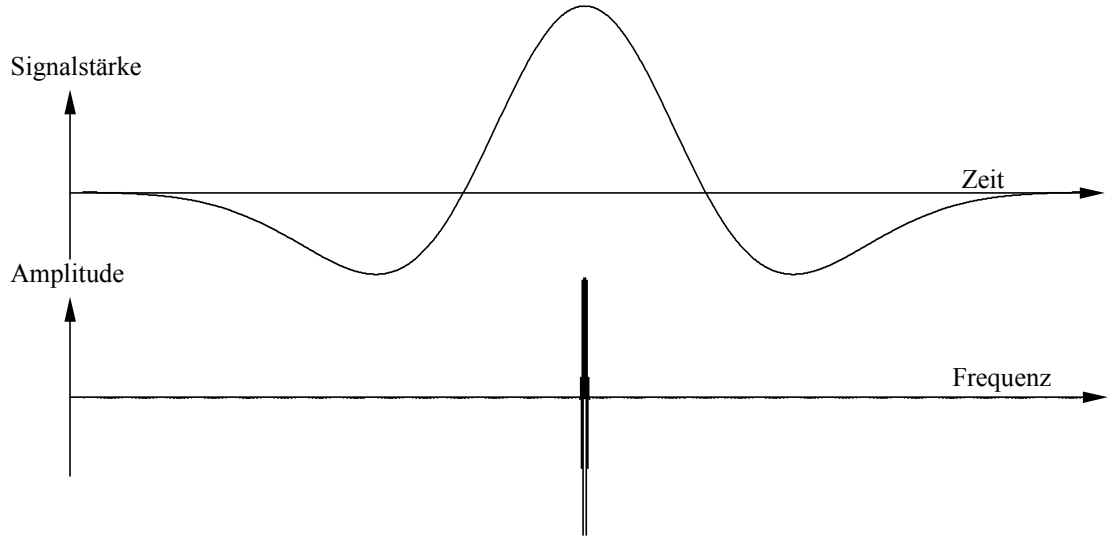

Abbildung 4.5: Im Ortsraum (oberer Teil) sieht man das *Mexican Hat* Wavelet. Im unteren Teil sind die beteiligten Frequenzen zu sehen. Der Mexican Hat enthält dabei wenige große tieffrequente Anteile, die hochfrequenten sind nur marginal ausgeprägt. Damit ist klar, dass dieser Filter ein schmales, tieffrequentes Band herausfiltert, im Frequenzraum also schön lokalisiert ist.

4.1.4 Notwendige Bedingungen an ein Wavelet

Ein weiterer Unterschied zur FT ist, dass die WT zur Analyse keine trigonometrische Funktionen verwendet. Vielmehr kann jede Funktion als Analyse-Welle dienen, die die im folgenden dargestellten Eigenschaften besitzt [LOU98, S. 17ff]:

$$0 < c_\psi := 2\pi \int_R \frac{|\hat{\psi}(\omega)|^2}{|\omega|} dw < \infty \qquad \text{(Ausdruck 4.3)}$$

Die Wavelet-Funktion selbst ist im obigen Ausdruck mit ψ bezeichnet. Das Mexican Hat Wavelet in Abbildung 4.5 ist ein Beispiel eines solches Wavelets. $\hat{\psi}$ steht für die Fourier-Transformierte des Wavelets, entspricht also der Zerlegung, die im unteren Teil der Abbildung zu sehen ist.

Die mit der Wavelet-Funktion verbundene sogenannte *Zulässigkeitskonstante* c_ψ kann man sich anschaulich als gewichtete Summe der quadrierten Koeffizienten aus Abb. 4.5 (unten) vorstellen. Zumindest würde man die Konstante numerisch so berechnen. Diese Konstante muss also endlich und größer null sein. Für ein verschwindendes c_ψ würde das Wavelet ohnehin zur Null-Funktion zusammenfallen. Man kann sich diesen Sachverhalt mit Anwendung *Hartley* durch Verkleinern der Koeffizienten veranschaulichen.

Weil die Zulässigkeitskonstante endlich sein muss folgt, dass auch der Bruch $|\hat{\psi}(\omega)|^2 / |\omega|$ für $\omega \to \infty$ gegen null konvergieren muss. Nach einem Satz von Riemann-Lebesque ist die Fourier-Transformierte $\hat{\psi}$ in R stetig. Aufgrund dieser Stetigkeit folgt:

$$0 = \hat{\psi}(0) = \int_R \psi(t)dt * e^{-2i\pi 0} = \int_R \psi(t)dt * 1 \qquad \text{(Ausdruck 4.4)}$$

Ausdruck 4.4 bedeutet anschaulich, dass der Mittelwert des Wavelets, bzw. sein Integral, null ist. Als Folge davon ist der Mittelwert des Signals durch die Wavelet-Analyse nicht zu ermitteln. Dies ist im Zusammenhang mit der weiter unten beschriebenen dyadischen Zerlegung wichtig, da der Mittelwert nach der Zerlegung des Signales immer übrig bleibt und gesondert gespeichert werden muss. Der Mittelwert ist - inhaltlich interpretiert - die grobst mögliche Approximation des Signales.

Die genaue Analyse von Wavelet-Funktionen sprengt den mathematischen Rahmen dieser Arbeit, da hier nur bekannte Filterbänke verwendet werden. Aber auch zum "Bau" eigener Analysewavelets ist es im Prinzip ausreichend zu wissen, dass das Wavelet auf einem kompakten Intervall definiert ist (auch als kompakter Support bezeichnet) und dass der Flächeninhalt über der Achse dem unter der Achse entspricht. Mathematisch reicht es sogar aus, wenn das Wavelet exponentiell schnell gegen null konvergiert, ohne die Null zu erreichen.

4.1.5 Transformation eines Signales

In Ausdruck 4.5 wird das Signal f unter Verwendung des Wavelets ψ für alle a, b des zulässigen Wertebereichs in seine Wavelet-Transformierte $L_\psi f(a,b)$ überführt.

$$L_\psi f_{a,b} = \frac{1}{\sqrt{c_\psi}} |a|^{-1/2} \int_R f(t) \psi\left(\frac{t-b}{a}\right) dt \qquad \text{(Ausdruck 4.5a)}$$
$$a \in R/\{0\}, b \in R$$

Der Parameter b verschiebt das Wavelet dabei entlang der reellen Achse, der Parameter a skaliert es in der Größe. Durch Variation von a werden also längere oder kürzere Schwingungen untersucht, durch die Veränderung von b geschieht dies an jedem Ort. Nun ist auch anschaulich, warum Wavelet-Koeffizienten immer sowohl Orts- als auch Frequenzinformationen über das analysierte Signal enthalten. Denn zu jedem Koeffizienten mit Index (a, b) gehört eine spezielle Ausprägung einer Wavelet-Funktion einer definierten Breite a und eines festgelegten Ortes b.

Damit der Flächeninhalt unter dem Wavelet gleich bleibt und somit alle Koeffizienten das gleiche Gewicht haben, wird außerhalb des Integrals mit $|a|^{-1/2}$ multipliziert. In Analogie zur Fourieranalyse wird in Ausdruck 4.5a das Produkt einer Analysefunktion mit dem Signal berechnet. Statt der Frequenz wird in diesem Fall die Größe des Wavelets skaliert. Kleine Werte für a resultieren in hohen schmalen Wavelets, große Werte in langgezogenen, niedrigen. Unterschiedliche Formen sind in Abbildung 4.7 dargestellt.

Ebenso wie die DCT oder die Fouriertransformation zerlegt die Wavelet-Transformation ein Signal in eine Art Frequenzspektrum. Streng genommen geben die Koeffizienten der WT aber keine Frequenzen im Sinn der Fourieranalyse - in Hertz - wieder.

Warum kann man den Einfluss eines Wavelet-Koeffizienten nicht als Frequenz durch eine Hertz-Einheit angeben? Transformiert man eine einzelne Sinusschwingung vom Orts- in den Fourierraum, so entsteht genau an der Stelle der Frequenz der Schwingung eine einzelne Spitze (siehe Abb. 4.2). Alle anderen Bereiche sind null, denn andere Frequenzen waren per Definition nicht vorhanden.

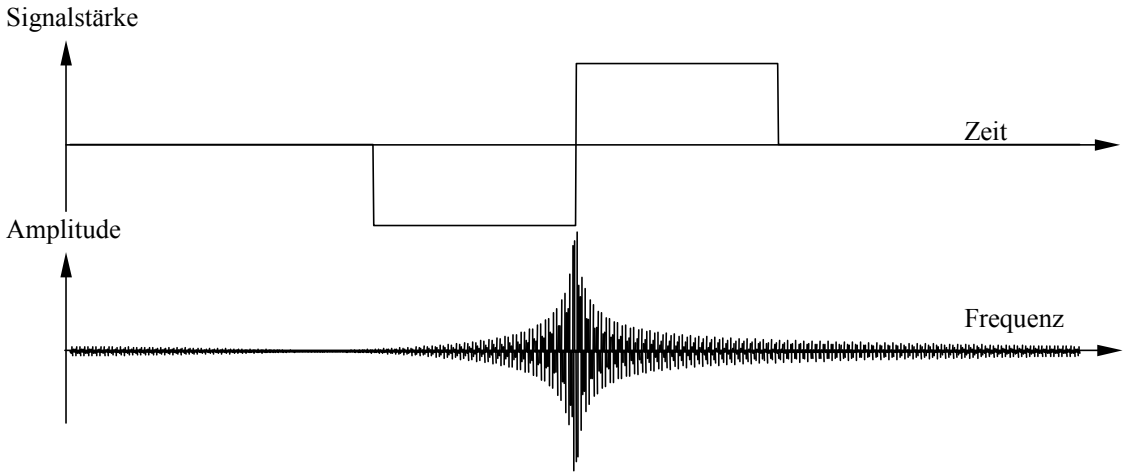

Abbildung 4.6: Das Haar Wavelet im Ortsraum (oben) und die Frequenzzerlegung (unten).

Transformiert man dagegen eine Wavelet-Funktion vom Orts- in den Fourierraum, so ergibt sich eine Zerlegung in unendlich viele überlagerte Schwingungen. Wird die Filterbank eines Wavelets z. B. in Schritten zu 200 ms verschoben (der Fenster passt also fünf mal in eine Sekunde), so kann zwar in der Analyse mit diesem Wavelet keine Schwingung enthalten sein, die tiefer als fünf Hertz liegt.

$$\psi_{Haar} = \begin{cases} -1 & t \in [0;0,5) \\ +1 & t \in (0,5;1] \\ 0 & sonst \end{cases} \qquad \text{(Ausdruck 4.5b)}$$

Die Abschätzung des Frequenzbandes, für die das Wavelet nach unten verantwortlich ist, ist also trivial. Für die enthaltenen höheren Frequenzen kann dagegen keine einfache Abschätzung stattfinden. Denn in der Regel werden Wavelet-Funktionen aus unendlich vielen Schwingungen von beliebig hohen Frequenzen zusammengesetzt, so z. B. im Fall des Haar Wavelets aus Ausdruck 4.5b (siehe Abb. 4.6). Das Haar-Wavelet nimmt insofern eine besondere Rolle ein, als aus ihm der einzige Filter mit nur zwei Werten abgeleitet werden kann, der sich für die dyadische Transformation eignet. Diese einfache Zerlegung eines Signales in Differenz- und Mittelwerte benachbarter Wertepaare geht eigentlich auf einen Vorschlag von Burt und Adelson [BUR84] aus dem Jahre 1984 zurück und wurde erst später im Rahmen der Wavelet-Theorie neu interpretiert.

In Abbildung 4.5 wurde exemplarisch das *Mexican Hat* Wavelet aus Ausdruck 4.5c zerlegt. Der Mexican Hat, welcher übrigens der zweiten Ableitung der Gaussfunktion ent-

spricht, ist besonders im Rahmen der Analyse beliebt, da er sowohl im Orts- wie auch im Frequenzraum gut lokalisiert ist. Im unteren Teil der Abbildung sieht man, dass das Mexican Hat Wavelet aus wenigen tiefen Frequenzen besteht.

Bei einer Wavelet-Analyse entstehen Koeffizienten, die angeben, in welchem Maße das betreffende Wavelet in dem Signal enthalten ist. Statt mit dem Wavelet könnte man das Signal aber ebenso einzeln mit allen Schwingungen analysieren, die im Wavelet enthalten sind (im unteren Teil der Abbildung dargestellt). Anschaulich gesprochen, fasst die Analyse eines Signales mit einem Wavelet also gleich die Analyse einer Reihe von Frequenzen der klassischen Fourieranalyse zusammen. Der direkte Vergleich zwischen Haar und Mexican Hat zeigt, dass unterschiedliche Wavelet-Funktionen im Frequenzraum mehr oder weniger stark konzentriert sind. Das Haar-Wavelet breitet sich dabei über ein breites Band von Frequenzen aus, wogegen der Mexican Hat in einem sehr schmalen Band definiert ist. Zum Zweck der Analyse ist eine möglichst scharfe Begrenzung des Bandes bzw. Spektrums wünschenswert, um die Existenz bestimmter Frequenzen nachzuweisen. Der Mexican Hat zeichnet sich durch eine relativ gute Lokalisierung sowohl im Orts- als auch im Frequenzraum aus.

$$\psi_{Mex} = \left(1 - t^2\right) e^{-t^2/2} \qquad \text{(Ausdruck 4.5c)}$$

Ähnliches gilt auch im Kontext der Bildkompression, bei der die langsam schwingenden Wellen im Bild von den schnellen getrennt werden sollen. Die Begründung wurde anhand der typischen Korrelation benachbarter Pixel bereits im Kapitel über die DCT diskutiert.

Je breiter die Filterbank bei der DWT gewählt wird, desto größer ist das Frequenzband, für das ein einzelner Koeffizient (ein einzelnes Wavelet) verantwortlich ist. Trotzdem kann man aber sagen, dass die Koeffizienten zu den hohen gedrängten Wavelets eher für die hohen Frequenzen verantwortlich sind. Je mehr sich das Wavelet in die Breite zieht, desto tiefer werden tendenziell auch die Frequenzen, die beeinflusst werden. Eine scharfe Eingrenzung des Bandes für einen Koeffizienten ist also nicht möglich. Im Fall der kontinuierlichen WT sind die Bänder sogar nach oben offen. Daraus folgt, dass sie

sich überlappen müssen. Ein Koeffizient einer bestimmten Skala beeinflusst also auch Frequenzbänder anderen Skalen.

Aufgrund der Orthogonalität der Wavelets ist aber gleichzeitig gewährleistet, dass jeder Koeffizient nur "einzigartige Aspekte" des Signales erfassen kann. Einfach einen Koeffizienten weglassen, käme der nicht-Beachtung eines der Fenster aus Abbildung 4.4 gleich. Offensichtlich würde man damit einen Aspekt des Signales vernachlässigen, der durch kein benachbartes Fenster ausgeglichen werden kann. Mit der DWT liegt also eine nicht-redundante Zerlegung des Signales vor.

Nach der letzten Stufe der Analyse bleibt ein Restsignal zurück, welches kürzer als die verwendete Filterbank ist und daher nicht weiter zerlegt werden kann. Diese grobst mögliche Approximation des Signales ist bei der Resynthese die Basis für die sukzessive Wiederherstellung. Ein Rest muss auf jeden Fall übrig bleiben, zumindest ein einzelner Wert, den man als Mittelwert des Signales interpretieren kann. Wie schon weiter oben erwähnt, kann das sich zu null integrierende Wavelet diesen Mittelwert nicht einfangen.

4.1.6 Die kontinuierliche WT

Neben der oben besprochenen dyadischen Variante der WT kann man diese auch kontinuierlich durchführen. Dabei wird die Fensterbreite nicht wie in Abb. 4.4 jeweils halbiert, sondern nimmt theoretisch kontinuierlich jede Breite an. Auch die Verschiebung des Wavelets auf einer bestimmten Skala findet nicht in Einheiten von Fensterbreiten statt, sondern ebenfalls kontinuierlich. Wegen der stufenlosen Variation gleich zweier Analyseparameter entsteht eine große Redundanz in der Zerlegung. Da die Parameter a und b aus Ausdruck 4.5 variiert werden, entsteht aus der Analyse einer eindimensionalen Funktion eine zweidimensionale Repräsentation. In Abbildung 4.8 ist die Analyse eines Rechtecksignales mit dem Mexican-Hat-Wavelet dargestellt.

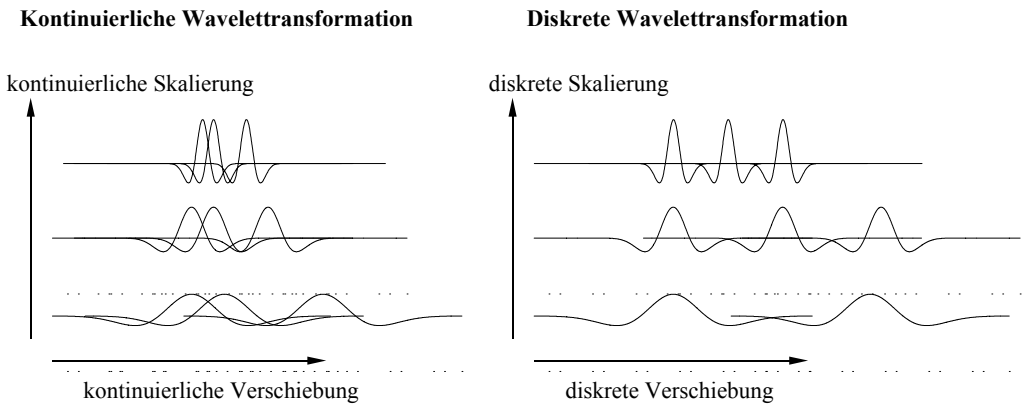

Abbildung 4.7: Bei der kontinuierlichen WT (links) wird das kontinuierlich skalierte Wavelet wiederum selbst kontinuierlich über das Signal verschoben. In der dyadischen Version (rechts) wird dagegen jeweils um eine ganze Fensterbreite verschoben und um den Faktor 2^{-n} skaliert.

Von links nach rechts wurde es im Ortsraum mit dem Parameter b verschoben, von vorne nach hinten wurde das Wavelet mit dem Parameter a skaliert. Jeder Punkt der Ebene entspricht also einem Parameter-Paar (Translation, Skalierung). Mit dieser Skalierung wurde dann an der (dem Parameter b) entsprechenden Stelle das Wavelet mit dem Signal gefaltet und das Ergebnis im Graphen als Höhe abgetragen.

4.1.7 Eigenschaften der WT

Die Stärke, vor allem der Analyse mit der kontinuierlichen WT, liegt gerade in der Redundanz der Zerlegung. Man hat gewissermaßen die Chance, einen Sachverhalt in einem ganzen Bereich von Koeffizienten zu finden.

Im Gegensatz dazu, beschreibt bei der dyadischen Transformation jeder Koeffizient einen einzigartigen Aspekt des Signales, der nachweislich in keinem anderen Koeffizienten zu finden sein kann. Besonders ungünstig ist bei dieser redundanzfreien Analyse auch, dass man nicht vorhersagen kann, ob ein kurzes Ereignis im Signal nur auf einen oder auf zwei Koeffizienten fällt. Dies hängt ganz davon ab, ob sich das Ereignis vollständig im Geltungsbereich eines Koeffizienten oder gerade auf der Grenze zwischen zwei benachbarten Wavelets manifestiert. Dies ist fast immer dem Zufall überlassen und für die Suche nach Mustern denkbar ungeeignet. Bei der kontinuierlichen Trans-

formation muss man sich darüber keine Gedanken machen, da die Geltungsbereiche der Wavelets kontinuierlich verschoben werden.

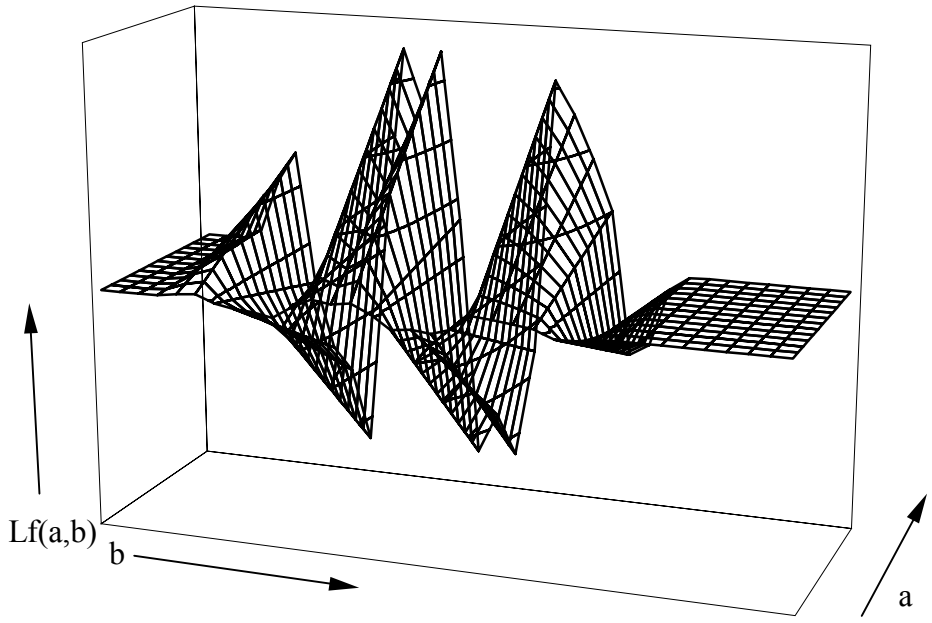

Abbildung 4.8: Das Mexican Hat Wavelet wurde mit unterschiedlichen Skalierungen (Z-Achse) über ein Rechtecksignal verschoben (X-Achse). Der Koeffizient wurde in die Höhe (Y-Achse) abgetragen.

Die Suche und Konstruktion spezieller Wavelets mit bestimmten Eigenschaften, sowohl für die Analyse als auch für die Kompression, ist eine eigenständige Disziplin der angewandten Mathematik geworden. Trotzdem ist nicht a priori klar, dass zum Zweck der Kompression, Analyse oder der Erkennung von Mustern die aus Wavelets gewonnenen Transformationen ein Signal besser dekorrelieren, als dies die DCT tut. Die Erfahrung zeigt allerdings, dass waveletbasierte Verfahren vor allem im Bereich der Kompression oft bessere Ergebnisse erzielen als solche, die auf der DCT beruhen.

4.2 Waveletbasierte Signalkodierung

Wie schon in Kapitel 3 beschrieben, bedingen typische Grauwertverteilungen digitaler Photos Transformationen, die der DCT bereits sehr ähnlich sind. Die besseren Kompressionsraten waveletbasierter Verfahren sind eher dadurch zu erklären, dass ein Bild als Ganzes und nicht in Blöcke aufgeteilt komprimiert wird. Dadurch hat die Kompression mit Wavelets den Vorteil, Redundanzen im Bild auszunutzen, die sich über größere

Regionen erstrecken. Die blockbasierte DCT kann dagegen nur Redundanzen innerhalb eines Blockes ausnutzen. Dabei wird jeder Block so kodiert, als wüsste man nichts vom Inhalt des letzten und nächsten Blockes. Vor allem tieffrequente Schwingungen müssen somit bei der DCT immer wieder erneut kodiert werden. Wavelets können dagegen eine Schwingung für ein ganzes Bild mit nur einem Koeffizienten erzeugen.

In der Praxis wird bei DCT-basierten Verfahren der DC-Koeffizient zum vorherigen Block differenziell fortgeschrieben. Dadurch wird in gewisser Weise doch eine Verbindung zwischen den Blöcken hergestellt, so dass auch tiefe Schwingungen nicht blockweise jeweils völlig neu gespeichert werden müssen. Die neuesten Verfahren, die z. B. in MPEG 4 spezifiziert sind, können sogar Koeffizienten für horizontale und vertikale Kanten von einem Block zum nächsten (bzw. zum Nachbarblock darunter) differenziell fortschreiben, da sich auch Kanten oft horizontal und vertikal durch das Bild fortsetzen.

Ist ein Bild stark durch hochfrequente Anteile bestimmt, so spielt die Kompression mit Wavelets ihre Stärken gegenüber der DCT weniger gut aus, da sich hochfrequente Schwingungen nur lokal auswirken. Blockgrenzen sind hier von geringer Bedeutung, denn auch innerhalb eines Blockes sind viele Informationen enthalten. Dabei spielen Redundanzen der Blöcke untereinander keine große Rolle.

In der Praxis zeigt sich auch tatsächlich, dass bei geringen Kompressionsraten und hoher Qualität zwischen DCT- und waveletbasierten Kompressionsverfahren kaum ein Unterschied besteht.

4.2.1 Art der Quantisierungsartefakte

In Kapitel 3 wurde in Zusammenhang mit der DCT auf Quantisierungsartefakte eingegangen. Der Vollständigkeit wegen sollen diese auch hier zum Vergleich kurz angesprochen werden.

Die Vermeidung von Blöcken bei der Wavelet-Kodierung hat neben der besseren Kodierungseffizienz auch Auswirkung auf Quantisierungsartefakte. Zwar treten durch die

Quantisierung von Koeffizienten auch hier Sprünge in der Bildfunktion auf, dies kann aber in jeder Zeile bzw. Spalte und nicht nur an Blockgrenzen stattfinden. Dadurch entsteht nicht das für stark komprimierte DCT-Bilder typische Kästchenschema, sondern Sprünge treten "zufällig" überall auf. Diese zufällige Verteilung der Fehler ist für den Betrachter weniger offensichtlich. In Abbildung 4.9 ist in den oberen beiden Bildern beispielhaft die Auswirkung der Quantisierungseffekte bei der DCT (links) und der Waveletkodierung (rechts) zu sehen. Links wird bei einer Datenrate von 0.15 Bit/Pixel eine Blockstruktur deutlich. Rechts wurde ebenfalls mit 0.15 Bit/Pixel waveletkodiert. Eine gleichmäßige Fehlerstruktur ist weniger zu erkennen. Dennoch treten Quantisierungsartfakte auch hier immer zeilen- und spaltenweise auf, da auch das Bild so gefiltert wurde. Mögliche Sprünge erscheinen aber überall mit gleicher Wahrscheinlichkeit, so dass keine regelmäßigen Muster entstehen können.

Abbildung 4.9: DCT-komprimiertes Bild bei einer Datenrate von 0.15 Bits/Pixel (links oben), JPEG 2000 kodiertes Bild bei 0.15 Bits/Pixel (rechts oben), DCT-kodiertes Bild bei gleicher PSNR wie waveletkodiertes Bild (links unten), originales Bild (rechts unten)

Der obige Versuch soll eigentlich nur die Quantisierungsartefakte isoliert bewerten. Dies ist insofern problematisch, als dabei keine strikte Trennung zwischen Quantisierungsfehlern und Kodierungseffizienz erfolgt. Mit anderen Worten: Das waveletkomprimierte Bild erzielt bei gleicher Datenrate eine deutlich bessere Peak Signal-to-Noise-Ratio (kurz PSNR, mehr dazu im folgenden Abschnitt), d. h. der Abstand zum Originalbild ist kleiner. Bewertet werden sollte aber nur der Eindruck, den die Quantisierung hinterläßt. In der Hoffnung auf eine bessere Vergleichbarkeit der Auffälligkeit von Quantisierungsfehlern wurde daher im Bild links unten bei der DCT-basierten Kompression dieses Mal die PSNR festgehalten, um eine annähernd gleiche Qualität mit dem waveletbasierten Bild rechts oben zu gewährleisten. Allerdings wurde die subjektive Bildqualität zugunsten der DCT dadurch nicht vergleichbar, sondern deutlich besser. In Anbetracht einer vier Mal höheren Datenrate ist dies auch nicht verwunderlich.

Da die PSNR lediglich eine Umformulierung des mittleren quadratischen Fehlers darstellt, ist die einzige Erklärung für den deutlichen visuellen Unterschied zwischen den Bildern, dass die DCT besonders starke Ausreißer haben muss. Die Waveletkodierung ist offensichtlich eher in der Lage, Fehler gleichmäßig über das Bild zu verteilen. Aus Sicht der DCT braucht diese also mehr Bits, um ihre starken maximalen Abweichungen verringern zu können (und so die PSNR abzusenken). Während sich die Datenrate erhöht, verteilen sich die Bits natürlich auch auf alle anderen Bereiche mit kleineren Fehlern, so dass ein besseres Bild entsteht. Als Fazit muss man feststellen, dass ein isolierter Vergleich der Quantisierungsartefakte beider Verfahren weder auf Grundlage der PSNR noch der Datenrate fair ist.

Zur Lösung dieses Problems wurden in der Literatur eine Reihe von Metriken vorgeschlagen, die die menschliche Wahrnehmung besser nachempfinden sollen. Unter anderem wurde von Kuhmünch für die Beurteilung der Qualität komprimierter Videosequenzen eine gewichtete Summe unterschiedlicher Merkmale von Differenzen der Kanten, von Farbabweichungen und von temporalen Artefakten entwickelt. Die Gewichte wurden dabei mittels empirischer Tests optimiert [KUH01]. Ob und inwiefern die Nachbildung der menschlichen Wahrnehmung zum heutigen Zeitpunkt möglich ist, ist

umstritten, da diese ein bisher wenig erforschtes Gebiet ist (siehe hierzu auch Kapitel 5.1.2).

4.2.2 Wavelets in der Videokodierung

Im Gegensatz zur weithin akzeptierten Kompression von Einzelbildern mit Wavelets, ist die Ausdehnung auf Videosequenzen noch Gegenstand intensiver Forschung. Zwar enthält die Videokodierung auch immer die Kompression von Einzelbildern. Da jedoch wegen der kontinuierlichen Filterung keine Blockstrukturen mehr vorhanden sind, gibt es keine offensichtlichen Strukturen mehr, die von den Bewegungsvektoren verschoben werden könnten. Zur Lösung dieses Problems werden zur Zeit zwei Varianten diskutiert, nämlich die explizite Kompensation der Bewegung, so wie bei MPEG und den H.26x Verfahren, oder die implizite Kodierung direkt im dreidimensionalen Raum des Videos.

Der erste Vorschlag versucht, zusammen mit der Waveletkodierung Bewegungsvektoren hierarchisch über die einzelnen Skalen zu verteilen [TSU94, YAN97]. Abbildung 4.12 zeigt das Beispiel einer Zerlegung in unterschiedliche Skalen. Sie sind von der Feinsten zur Gröbsten durchnummeriert. Bei einer realen Implementierung würde man die Zerlegung noch weiter fortführen, also über Stufe drei hinaus zu immer kleiner werdenden Approximationen. Eine Verteilung der Bewegungsvektoren auf unterschiedliche Skalen kann man sich nun wie folgt vorstellen: Die ersten Vektoren sind auf der gröbsten Approximation des Bildes definiert. Grundsätzlich muss auf dieser Approximation (Stufe 3 im gewählten Beispiel) für jeden Pixel ein Bewegungsvektor definiert werden, der beschreibt, in welche Richtung sich das Bild an der entsprechenden Stelle bewegt. Diese Verschiebungen werden auch analog auf die feineren Detailbilder übertragen, im Beispiel also Stufen 2 und 1. Mit jedem Pixel der gröbsten Approximation ist auch ein ganzer Block im originalen Bild assoziiert, denn die Koeffizienten haben je nach Skala einen größeren oder kleineren Geltungsbereich im vollständig rekonstruierten Bild.

Abbildung 4.12: Standard Beispiel der Bildverarbeitung *Lena* (links). Wavelet-zerlegte Version (rechts).Die Skalen sind von der feinsten zur gröbsten Struktur durchnummeriert. In jeder Stufe, von der Gröbsten zur Feinsten, kann das Bild durch die zusätzlichen Koeffizienten verbessert werden.

Die bisher sehr grobe Schätzung der Bewegung im Bild wird aber der tatsächlichen Bewegung des Bildes noch nicht gerecht, es sei denn die Kamera hat einen Schwenk vollzogen. Auf den zunehmend feineren Skalen, die mit immer größer werdenden Detailbildern verbunden sind (siehe Abb. 4.12), entstehen immer mehr Pixel, für die Bewegungsvektoren definiert werden müssen. Aufgrund der bereits kodierten groben Schätzung werden die Verfeinerungen auf den unteren Skalen betragsmäßig immer kleiner und verschwinden bis zur letzten bzw. vorletzten Stufe meist bis auf Objektkanten vollständig. Grundsätzlich nutzt man also zur Kodierung der Bewegung den gleichen Effekt aus, den man sich für die Kodierung des eigentlichen Bildes zunutze macht, nämlich dass die Informationen auf den feineren Skalen immer stärker gegen null konvergieren, und marginale Beträge überhaupt nicht mehr gespeichert werden müssen.

Im Gegensatz zur MPEG-Kodierung, bei der das zu verschiebende Raster festgelegt ist, kann die Erfassung der Bewegung hier gezielter erfolgen. Auch kann man sich entscheiden, ob man lieber die Bewegung gröber auflösen möchte und im Ausgleich dazu später stärkere Fehler im Bild kompensieren muss, oder ob man mehr Aufwand in die Präzision der Bewegung investiert, um zum Ausgleich nur noch marginale Differenzbilder erzeugen zu müssen. Trotzdem hat das oben beschriebene Verfahren mit MPEG die Gemeinsamkeit, dass Bild- und Bewegungsinformation als getrennte Aspekte betrachtet werden, die man mit unterschiedlichen Verfahren behandelt. Dies kommt ver-

mutlich auch der menschlichen Wahrnehmung am Nächsten, die auch zuerst Objekte identifiziert und sie daraufhin mit Bewegungen verbindet. Ob diese eher aus der Anschauung herrührende Trennung auch im Sinne der Kodierungseffizienz optimal ist, ist bis heute umstritten.

Die weiter oben erwähnte zweite Strömung in der Videokodierung versucht das Video oder Teile davon unter Einbeziehung der zeitlichen Dimension als Ganzes zu transformieren, also einen ganzen Quader von übereinander geschichteten Voxeln als dreidimensionales Signal aufzufassen und zu zerlegen [SER97]. Bisher ist in der Literatur jedoch noch keine Implementierung bekannt, die bei gleicher PSNR geringere Datenraten erzielt hätte als die etablierten DCT-basierten Varianten. Das große Potenzial besteht bei der impliziten Kodierung von Bild und Dynamik darin, zwischen beiden nicht mehr differenzieren zu müssen, da sie im Video eigentlich eine Einheit bilden. Redundanzen zwischen Bild und Bewegung können vielleicht besser ausgenutzt werden, wenn man sie eben nicht als getrennte Aspekte ansieht und sich damit bereits auf eine bestimmte Kodierung festlegt, die dem Signal evtl. gar nicht besonders gerecht wird. Tatsächlich kann man solche Beispiele auch leicht finden. Die kleinen Bewegungen eines Baumes mit sich bewegenden Blättern können kaum sinnvoll kompensiert werden. Auch gelegentlich abgefilmte Konfettiparaden produzieren in MPEG-Videos mitunter sehr starke Artefakte, weil Bewegungskompensation, vor allem auf Blockbasis, nicht möglich ist. Trotzdem verhalten sich die zahlreichen kleinen Objekte zwischen zwei Bildern durchaus vorhersagbar, die traditionelle Art der Bewegungskodierung eignet sich nur nicht dafür.

Zur besseren Veranschaulichung des 3D Orts- bzw. Frequenzraumes wurde daher im Rahmen dieser Arbeit eine Visualisierung, vor allem des Raumes der Waveletkoeffizienten, durch einen Volumenraytracer entwickelt [HAE01]. Damit ist die Hoffnung verbunden, durch ein intuitiveres Verständnis des Raumes der Koeffizienten bessere Kodierungsverfahren zu finden.

4.2.3 Interpretation des Frequenzraumes

Im Folgenden werden kurze Filmsequenzen als Datenwürfel interpretiert. Jede Scheibe des Würfels entspricht dabei einem Einzelbild. Die untereinander liegenden Bilder erzeugen das Volumen (siehe Abb. 4.13). In der Praxis kann und sollte man nicht ein vollständiges Video in einem Durchgang behandeln. Vielmehr ist es sinnvoll, z. B. Bereiche zwischen zwei Schnitten zusammen zu fassen, weil die Redundanzen aufeinander folgender Bilder innerhalb einer Kameraeinstellung am größten sind. Im Fall der DC-Transformation steht ein Punkt im dreidimensionalen Datenwürfel für eine Frequenz (genauer Wellenzahl), die sich in alle drei Richtungen ausbreitet.

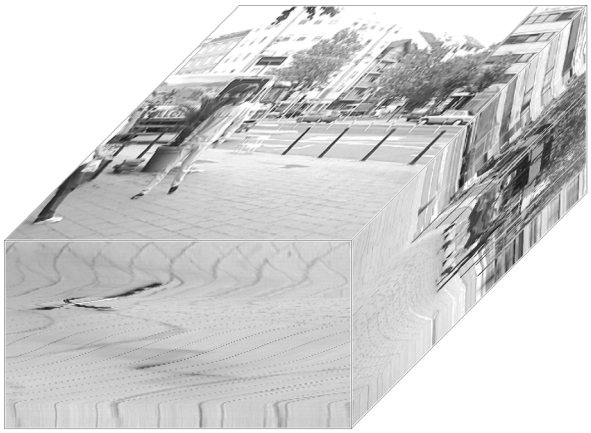

Abbildung 4.13: Beispiel einer kurzen Videosequenz, die im Raum als Würfel veranschaulicht wird. Auf der Oberfläche des Würfels ist das erste Bild zu sehen, von allen anderen Bildern lediglich die jeweils erste Zeile am unteren Rand und die letzte Spalte am rechten Rand. Die Zeitachse zeigt somit nach unten. Die verwischte Struktur an den senkrechten Würfelflächen läßt vermuten, dass in der Sequenz ein Kameraschwenk erfolgt.

Hat der Punkt eine große x-Komponente, jedoch eine kleine y- und z-Komponente, so oszilliert die Welle schnell in der x-Richtung und langsam entlang der y- und z-Achse. Der Betrag, den der Würfel an der entsprechenden Stelle annimmt, beschreibt die Größe der Amplitude. Null bedeutet, dass die entsprechende Frequenz nicht enthalten ist, ein großer Wert zeigt einen großen Einfluss der durch den Ort implizit definierten Welle an. Ein Video, das aus nur einem sich wiederholenden Bild besteht, hätte keine Schwingung in Richtung der Zeitachse. Alle Bereiche (x, y, z) im Frequenzwürfel mit $z>0$ wären damit null, da in der Zeit keine Veränderung stattfindet.

4.2.4 Interpretation des Raumes der WT-Koeffizienten

Im Fall einer Wavelet-Transformation lässt sich ein bestimmter Punkt im Würfel der Koeffizienten gleichzeitig, je nach Skala mehr oder weniger exakt, sowohl im Orts- wie auch im Frequenzraum lokalisieren. Abbildung 4.14 zeigt eine Volumendarstellung des Raumes der Waveletkoeffizienten, wie sie mit dem in dieser Arbeit entstandenen Programm *VolRay* berechnet werden kann.

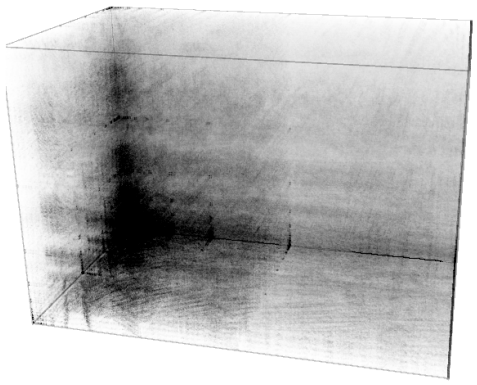

Abbildung 4.14: Die kurze Videosequenz aus Abb. 4.13 wurde Wavelet-transformiert und als transparentes Volumenmodell dargestellt. Im Gegensatz zur Darstellung 4.13, kann man so ins Innere des Würfels sehen. Am offensichtlichsten ist, dass ein Großteil der Information in den dunklen Koeffizienten links unten konzentriert ist. Kleine Koeffizienten werden dagegen durch helle Punkte repräsentiert.

Dabei wurde die Videosequenz aus Abb. 4.13 transformiert. Neu ist, dass die Verwendung eines Volumenraytracers *VolRay* einen Blick ins Innere des Raumes erlaubt. Ähnlich wie im zweidimensionalen Beispiel in Abb. 4.12 ist wieder eine Teilung in der Struktur der Daten zu erkennen, hier jedoch in Form von Oktanten im Raum, im Gegensatz zu den Quadranten in Abb. 4.12. Jeder Punkt im Würfel enthält einen Dichtewert, der entsprechend dem Betrag des Waveletkoeffizienten gesetzt wurde. Die besonders dunklen Bereiche im hinteren Teil des Würfels links unten zeigen besonders große Koeffizienten an. Dort liegt auch der Ursprung des Koordinatensystems. Die nach rechts oben deutlich durchsichtigeren Bereiche lassen auf überwiegend kleine Koeffizienten schliessen. Je größer ein Koeffizient betragsmäßig wird, desto weniger Licht läßt dieser passieren und desto dunkler wird der Bereich.

4.2.5 WT durch fortgesetzte Filterung

Die Transformation des "Video-Würfels" vom Ortsraum in den Raum der Wavelet-Koeffizienten erfolgt durch Filterung mit einem Hoch- bzw. Tiefpass, ganz analog zur Transformation von Einzelbildern in Beispiel 4.12. Zuerst wird das Bild z. B. entlang der x-Achse gefiltert und die entstehenden Koeffizienten in einem neuen Speicher abgelegt, der bis auf einige zusätzliche Werte an den Rändern der Größe des originalen Bildes entspricht. Dann wird das Bild der Koeffizienten entlang der y-Achse gefiltert und die Koeffizienten erneut gespeichert. Im Fall von Videos schließt sich nun noch eine Filterung entlang der Zeitachse an. Dabei werden die Veränderungen in der Zeit durch die Filterung zerlegt. Die Rücktransformation erfolgt in umgekehrter Reihenfolge, also in z-, y- und x-Richtung. Vor allem bei der Transformation in Richtung der Zeit zeigen sich die Probleme der Waveletkodierung. Zur Veranschaulichung soll das folgende Beispiel dienen:

Ein Zebra läuft durch das Bild, während die Kamera still steht. Ein bestimmtes Pixel fällt anfangs auf den Hintergrund der Szene. Während das Zebra den Pixel passiert, wird dieser je nach Färbung des Tiers schwarz oder weiß. Im Fall einer bewegungsbasierten MPEG-Kodierung verschieben sich einfach Blöcke des Bildes. Die Wavelet-Transformation entlang der Zeitachse kann man sich wie einen Strahl durch einen Pixel hindurch vorstellen. Entlang dieses Strahls wird ein Signal traversiert, das in seine Frequenzanteile zerlegt werden muss. Oszilliert das Signal zwischen schwarz und weiß, so nimmt es seinen Minimal- und Maximalwert an. Die Zerlegung eines solchen Signales ist insofern besonders ungünstig, da die fallenden und steigenden Flanken in eine große Zahl überlagerter Frequenzen bzw. Koeffizienten zerlegt werden müssen. Wo die Kodierung mit Bewegungsvektoren die Bewegung einfach kompensiert, leidet die naive Transformation eines Videos unter den sich ständig verändernden Grauwerten. Das Problem ist also die starre Transformation entlang der Zeitachse, die auf ihrem Weg immer wieder auf unterschiedliche Pixel hintereinander liegender Bilder stößt.
Würde es im Beispiel gelingen, die Transformationsrichtung schräg zu stellen, also der Geschwindigkeit des Zebras anzupassen, so würde die Filterung immer wieder einen schwarzen oder weißen Streifen treffen, was zu einem konstanteren, besser kodierbaren

Signal führen würde. Solche und ähnliche Erkenntnisse lassen sich anhand der oben gezeigten Visualisierung erkennen und besser verstehen.

Generell werden Volumenraytracer verwendet, um Objekte unterschiedlicher Dichte auch im Inneren zu visualisieren, weil die Oberfläche in diesem Fall wenig charakteristisch ist. Im Raum der Waveletkoeffizienten gibt es eigentlich überhaupt keine offensichtlichen Objekte oder Oberflächen, sondern nur abstrakte Dichteverteilungen zu beurteilen.

4.3 Exkurs: Grundlagen des Volumenraytracing

Ein Volumenraytracer funktioniert wie folgt: Ausgehend von einem virtuellen Beobachter im Raum werden Strahlen emittiert. Abbildung 4.15 zeigt eine Skizze.

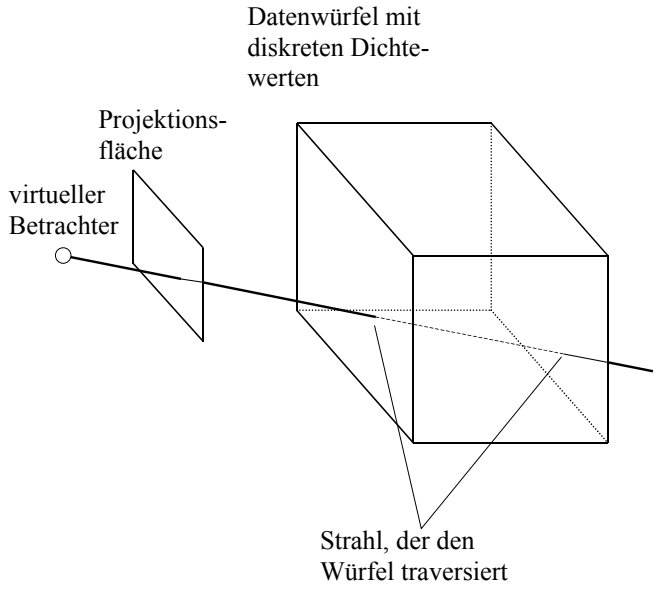

Abbildung 4.15: Ein Strahl wird von einem virtuellen Betrachter in die Szene emittiert. Auf seinem Weg durch den Würfel werden Dichtewerte aufsummiert.

Der Strahl schneidet eine Projektionsfläche und durchdringt den Raum der Dichtewerte. Die Dichtewerte, die der Strahl auf seinem Weg trifft, bestimmen zusammen die optische Durchlässigkeit für Licht aus dem Hintergrund. Der entsprechende Grauwert wird zum Schluss auf der Projektionsfläche an der Stelle des Schnitts abgetragen. Auf diese

Weise entsteht Pixel für Pixel ein perspektivisches Abbild aus Sicht des virtuellen Betrachters. Die Richtung der Strahlen ist auf den ersten Blick kontraintuitiv, weil in realen Szenen Strahlen von Objekten auf den Betrachter fallen und nicht umgekehrt. Beim Raytracing kehrt man die Richtung aus Gründen der Berechenbarkeit um. Denn die meisten Strahlen in einer realen Szene treffen den Betrachter nie. Definiert man ihn als Quelle, so müssen nur die Strahlen berechnet werden, die tatsächlich dazu beitragen, ein Bild im Auge des Betrachters zu erzeugen [HAE96].

4.3.1 Volumenraytracing des Koeffizientenraumes

Eine Anwendung des klassischen Volumenraytracing ist die Darstellung medizinischer Daten, wie sie bei der Computertomographie (CT) oder der Magnet-Resonanz-Tomographie (MR) entstehen [HES98, HES99, CHE01]. Wird entlang eines Strahls ein Wert größer null getroffen, so muss der Gradient (= die Richtung stärkster Grauwertänderung) an der entsprechenden Stelle berechnet werden. Er wird benötigt, um in der Umgebung eine Ebene zu ermitteln, die von der virtuellen Lichtquelle beleuchtet wird. Die Annahme einer Oberfläche ist insofern künstlich, als die Volumendarstellung ja gerade verwendet wird, um keine Oberflächen betrachten zu müssen, die der Natur der Daten gar nicht gerecht werden. Trotzdem braucht man etwas, das beleuchtet werden muss. Der Gradient gibt die Richtung der stärksten Grauwertänderung an. Senkrecht dazu ist also die geringste Grauwertänderung zu erwarten, die damit die Richtung definiert, die am besten einer Oberfläche entspricht. Den Gradienten kann man als Normale der Ebene interpretieren.

Bei medizinischen Daten ist die Annahme, auch in Volumendaten eine Oberfläche zu finden, nicht unrealistisch, denn tatsächlich bilden Knochen oder Gewebe mit ähnlichen Dichtewerten, wie z. B. das Gehirn, ein Objekt mit einer definierten Oberfläche. Aus diesem Grund werden medizinische Bilder sogar manchmal unter Berücksichtigung von Schlagschatten berechnet. Dazu wird am Punkt, an dem auch der Gradient berechnet wurde, ein weiterer Strahl in Richtung der Lichtquelle erzeugt und auf dessen Weg die Absorption des Lichtes als Maß für die Beschattung gewählt. Bei klar definierten Oberflächen kann die zusätzliche Schattenberechnung dazu beitragen, einem Bild mehr Tiefe

zu geben. Bei sehr komplexen Strukturen, die sich lokal oft verändern, kann ein Bild aufgrund zu vieler kleiner, unterschiedlich beschatteter Details allerdings auch unverständlich werden.

Bei den durch die Wavelet-Transformation erzeugten Volumendaten ist in der Regel nicht damit zu rechnen, dass die Koeffizienten zusammenhängende Oberflächen erzeugen. Zwar findet eine Konzentration in bestimmten Bereichen statt, die Verteilung in einer lokalen Umgebung ist aber eher chaotisch. Die Berechnung eines Gradienten führt daher zu keiner Oberfläche, die der Struktur der Dichtewerte gerecht werden könnte. Ein einfaches Aufsummieren der Dichtewerte hat in der Implementierung zu den verständlichsten Abbildungen geführt. Abbildung 4.14 zeigt ein Beispiel.
Zur genaueren Analyse ist es hilfreich, einzelne Schichten des Volumens isoliert betrachten zu können. Hierzu bietet die Anwendung *VolRay* einen interaktiven Modus, in dem ein achsenparalleler Schnitt als Graph hinter den Würfel projiziert werden kann. Der Benutzer steuert den Schnitt interaktiv mit der Maus. Abbildung 4.16 zeigt ein Bildschirmfoto.

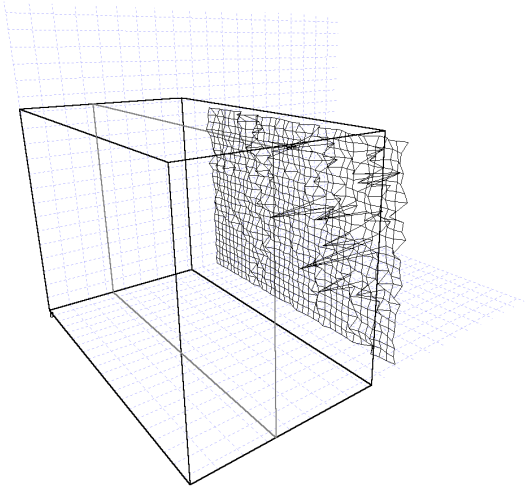

Abbildung 4.16: Darstellung des interaktiven Modus von *VolRay* zur Visualisierung von Schnitten durch Volumendaten.

4.3.2 Manipulation eines Videos im Orts- und Koeffizientenraum

Neben der reinen Transformation erlaubt das Programm *VolRay* auch die Manipulation der Daten im Frequenzraum und die Rücktransformation. Dies ist unter anderem auch

für Lehrzwecke hilfreich. In Abbildung 4.17 wurden alle Koeffizienten, die betragsmäßig unterhalb eines Schwellwertes liegen, auf null gesetzt. Der um etwa 99% seiner Koeffizienten bereinigte Raum erscheint dadurch deutlich heller. Entlang der Seiten des Würfels fallen kleine Cluster von Koeffizienten bei den Teilungen 1/2, 1/4, 1/8 auf. Diese Konzentrationen entstehen durch das in der Implementierung verwendete Zero-Padding am Ende des Bildes; wenn sich das Wavelet teilweise außerhalb des Bildes befindet, werden die dortigen Datenwerte als Null angenommen.

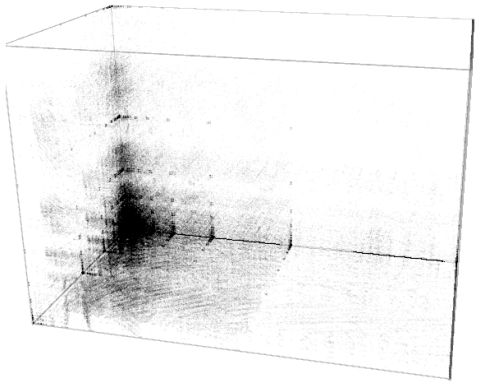

Abbildung 4.17: Gleiches Volumen wie in 4.14, jedoch wurden alle Koeffizienten unterhalb eines festen Schwellwertes gelöscht. Der Raum enthält nur noch 1% der Koeffizienten, nämlich die betragsmäßig größten.

Bei der Filterung werden den Daten an den Rändern Nullen hinzugefügt, um den überlappenden Wavelet-Filtern noch ausreichend viele Werte zu liefern. Dadurch entsteht im Signal ein Sprung, an dem eine Vielzahl von Frequenzen entsteht.

Interessant ist nun das Ergebnis der Rücktransformation in Abbildung 4.18. Im oberen Teil sind vier Bilder der originalen Sequenz zu sehen, im unteren Teil die vier entsprechenden Bilder, jedoch auf Grundlage der reduzierten Koeffizienten. In den unschärferen Bildern kann man zwei Arten von Artefakten erkennen, nämlich räumliche und zeitliche. Die räumlichen Fehler schlagen sich in einem gefilterten, leicht verschmierten Aussehen des Bildes nieder. Dies war zu erwarten, da 99% der kleinsten Koeffizienten gelöscht wurden und kleine Frequenzanteile meist für Kanten und Texturen stehen. Die zweite Sorte von Fehlern wirkt sich in der Zeit aus. Um die Beine des vorderen Fußgängers zeichnen sich schemenhaft Kanten um die schwarze Hose ab. Diese stammen von den im Video früher und später sichtbaren Beinen. Denn ein scharf begrenztes Objekt, das sich zwischen benachbarten Bildern bewegt, erzeugt eine Kante in der Zeit.

Wegen der 3D-Transformation werden Kanten in der Zeit ebenso geglättet wie solche im Ortsraum. Die schemenhafte Voraussicht von früheren und zukünftigen Beinen stammt von den betragsmäßig großen Koeffizienten. Da ein Daubechie-Filter mit vier Einträgen verwendet wurde [LOU98, S. 169ff], beeinflusst die entsprechende Welle durch ihre Ausdehnung auch die lokale Zeit um ein Ereignis herum, so dass kurzfristig frühere und zukünftige Ereignisse sichtbar werden.

Abbildung 4.18: Die oberen vier Bilder zeigen eine Sequenz aus dem originalen Video, die unteren vier entstanden aus dem ersten Prozent der absteigend nach ihrem Betrag sortierten Koeffizienten. Trotz des Verlustes von 99% der Koeffizienten ist der Inhalt noch gut erkennbar.

In Abbildung 4.19 wurde das Residuum des rücktransformierten Videoclips dargestellt. Man erkennt, dass 99% der Koeffizienten nur zu einem sehr schwachen Signal beigetragen haben. Trotzdem ist -die hier beschriebene, naive 3D-Transformation von Videos zum heutigen Zeitpunkt immer noch nicht besser als die fortgeschrittensten DCT-basierten Kodierungsverfahren.

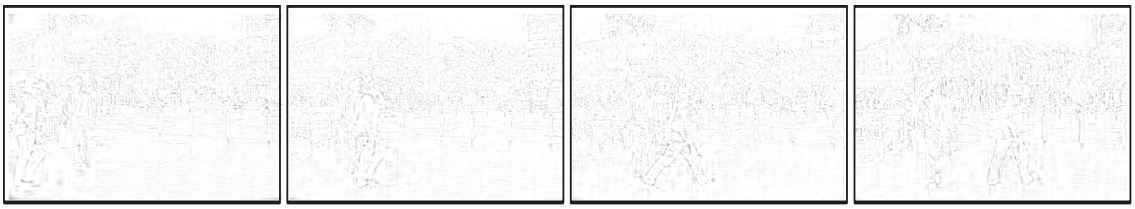

Abbildung 4.19: Fehlerbilder zwischen der oberen und unteren Reihe aus Abbildung 4.19, also die Information, die durch Verwerfen von 99% der betragsmäßig kleinsten Koeffizienten verloren ging.

KAPITEL 5

WAVELETBASIERTE SEGMENTIERUNG

5.1 Herausforderung des Auffindens von Objektgrenzen

Auf den ersten Blick scheint das Ausschneiden von Objekten aus Photos, also deren Segmentierung, eine einfache Aufgabe. Die dafür erforderliche Objekterkennung bzw. Abgrenzung vom Hintergrund ist eine Aufgabe, die selbst Kindern kaum Probleme macht. Innerhalb der Bildverarbeitung wird die Objektsegmentierung seit den frühen 70er Jahren erforscht [FIS73, CHI74]. Trotz langjähriger Bemühungen sind bisher allerdings keine allgemeingültigen Verfahren bekannt, die Objekte mit der gleichen Sicherheit abgrenzen könnten, wie es der Mensch vermag. Dies liegt zumindest zum Teil an der grundsätzlich schlecht definierten Aufgabenstellung, denn was ein zusammenhängendes Objekt ist, müsste eigentlich exakter formuliert werden. Lediglich der Mensch hat meist eine intuitive Vorstellung davon, was mit einem zusammenhängenden Objekt gemeint ist, wobei er in hohem Maße Erfahrungswissen einsetzt. Abbildung 5.1 zeigt die Problematik beispielhaft. Auf der Basis der Grauwertverteilungen ist eine Segmentierung i. d. R. nicht vollständig zu finden. Aus diesem Grund wurde in den letzten Jahren zunehmend gefordert, den Segmentierungsverfahren möglichst das Vorwissen mitzugeben, das auch dem Menschen zur Verfügung steht.

Den Schwerpunkt dieses Kapitels bildet ein im Rahmen dieser Arbeit entwickeltes Verfahren, welches auf einer Wavelet-Analyse des Bildes fußt. Dabei wird die Objektgrenze, also der Übergang zwischen Objekt und Hintergrund, auf unterschiedlichen Skalen analysiert. Die bei dieser Analyse entstehenden Erkenntnisse über die Charakteristik der Objektgrenze dient dann einer, verglichen mit herkömmlichen Verfahren, verlässlicheren Segmentierung. Eine genaue Analyse, warum und unter welchen Umständen eine solche skalenbasierte Zerlegung des Bildes vorteilhaft ist, widmet sich das Kapitel 8 vollständig. Dagegen ist auch schon in diesem Kapitel eine benutzergestützte Evaluation enthalten.

5.1.1 Mangelnde Signifikanz der Bildinformation

Dabei stellt sich die Frage, in welcher Weise Vorwissen einbezogen werden könnte. Probleme bereiten dabei zwei Sachverhalte: Zum einen hat man es meist mit deformierbaren oder beweglichen Objekten zu tun, zum anderen bieten unterschiedliche 3D-Ansichten eines Objektes je nach Perspektive eine Vielzahl unterschiedlicher Silhouetten, oft bei gleichzeitiger Deformierbarkeit. Der Umriss eines von der Seite betrachteten Menschen ist oft wenig charakteristisch, wenn man außer der Form keine weiteren Farb- oder Texturinformationen hinzuzieht; ein Auto von oben ist kaum als solches zu erkennen und Äpfel nicht von Orangen zu unterscheiden.

Offensichtlich kann man auf die Verwendung zusätzlicher Merkmale nicht immer verzichten. Farben und Texturen kann man vor einem komplexen Hintergrund aber oft erst dann richtig zuordnen, wenn man eine Idee von der Form des Objektes hat. Ganz zu schweigen von der Tatsache, dass Objekte teilweise verdeckt oder nur halb im Bild sichtbar sein können.

Bei genauerer Analyse ist die Abgrenzung von Formen ein komplexer Vorgang, der viel Vorwissen und Kombinationsgabe erfordert - und der im übrigen auch beim Menschen nicht immer funktioniert. Bei Unachtsamkeiten im Straßenverkehr spricht man manchmal davon, einen anderen Teilnehmer "nicht gesehen zu haben".

Abb. 5.1: Originales Photo (links). Die mittlere und rechte Abbildung machen deutlich, dass den Grauwerten alleine nicht zu entnehmen ist, was das zu segmentierende Objekt sein könnte.

Damit ist nicht immer gemeint, dass der andere gar nicht ins Blickfeld gekommen wäre, sondern eher, dass er nicht wahrgenommen wurde. Hier hat einfach die (rechtzeitige) Segmentierung nicht funktioniert.

5.1.2 Die menschliche Wahrnehmung

Wie das menschliche Gehirn Objekte und Bewegungen analysiert, ist in der Medizin und Psychologie kaum erforscht. Manche Tiere, wie z. B. Frösche, können Bewegungen bereits auf der Ebene der Retina (den Sehzellen des Augenhintergrundes) extrahieren. Beim Menschen weiß man, dass die Erkennung von Bewegung einer höheren kognitiven Ebene zugeordnet sein muss [ZEK91]. In der Medizin ist z. B. die sogenannte Bewegungsblindheit (med. Akinetopsie) bekannt [SCH00]. Die bekannten Fälle kommen von seltenen Beschädigungen des Gehirns oder treten in den letzten Jahren zunehmend bei Alzheimer-Patienten auf. Dabei kann der Patient normal sehen und in stehenden Bildern auch Objekte abgrenzen, nimmt jedoch keine Bewegungen wahr. Betroffene berichten, dass z. B. der Strahl eines eingegossenen Getränkes nicht mehr als fließend empfunden wird, sondern ähnlich einem Glasstab in der Tasse zu stehen scheint. Ein Überlaufen der Tasse wird nicht mehr vorhergesehen, weil die Veränderung der Bewe-

gung nicht fortgeschrieben werden kann. Erst wenn das nächste Standbild des überschwemmten Tisches wahrgenommen wird, realisiert ein Betroffener die Situation.

Die Fähigkeit, mit der Menschen Objekte vor einem Hintergrund abgrenzen, entwickelt sich schon beim Kleinkind. Aber erst mit etwa acht Jahren ist diese Entwicklung vollständig abgeschlossen. Zuerst fängt der Mensch an, zusammenhängende geometrische Formen, wie Linien und Flächen, in der Menge der vom Auge erfassten Farbwerte zu erkennen. Erst später werden Formen erkannt, denen Bedeutungen zugeordnet werden.
Wenn Patienten dank des medizinischen Fortschrittes in späten Jahren aufgrund von Operationen erstmals sehen können, kann dieser neue Sinneseindruck trotzdem ein Leben lang nicht mehr zur Wahrnehmung genutzt werden, da ein erwachsenes Gehirn die Informationen nicht mehr zu interpretieren lernt. Aus berührungsfreien Untersuchungen des Gehirns von Säuglingen und Kleinkindern weiß man, dass in den ersten Monaten die neuen Sinneseindrücke zu einer stark ansteigenden Zahl von synaptischen Verknüpfungen führen. Nachdem ein Maximum an Verknüpfungen erreicht wurde, werden die meisten in den folgenden Monaten wieder abgebaut. Offensichtlich findet in dieser Zeit eine zunehmende Selektion statt, in der den wahrgenommenen Sinneseindrücken tatsächlich stattfindende Ereignisse zugeordnet werden. In den ersten Monaten nach der Geburt scheint diese Verifikation noch nicht möglich zu sein. Die Forschung beschränkt sich momentan noch darauf festzustellen, in welchen Bereichen des Gehirns Aktivität herrscht. Was dort im Detail geschieht, lässt sich noch nicht erkennen. In nächster Zeit ist also auch nicht damit zu rechnen, dass die Medizin entschlüsselt, wie der Mensch die Segmentierung durchführt, ganz abgesehen von der Frage, ob eine solche Erkenntnis informatisch verwertbar wäre.

5.1.3 Die benutzergesteuerte Segmentierung

Zwischenzeitlich behelfen sich die Praktiker damit, die Anwendungsdomäne für die Segmentierung einzuschränken oder die Aufgabe unter (möglichst sparsamer) Zuhilfenahme des Menschen zu lösen.

Eine breite Anwendungsdomäne bietet sich in der industriellen Mustererkennung [JAE97, ROS82]. Frühe Verfahren basieren alleine auf Grauwerthistogrammen. Dabei wird im Histogramm ein Bereich eindeutig dem Hintergrund, ein anderer Teil eindeutig dem zu segmentierenden Objekt zugeordnet. Meist wird dabei der Hintergrund im Sinne einer optimalen Trennung der Grauwerte optimiert, also besonders hell oder dunkel gewählt.

Ähnlich arbeitet das aus dem Fernsehen bekannte Bluebox-Verfahren, bei dem ein Sprecher oder Schauspieler vor einem blauen Hintergrund agiert. Später wird das Blau durch ein Bild oder eine virtuelle 3D-Welt ersetzt, die sogar die Position und Bewegung der Kamera mit berücksichtigt [MIS93]. Fortgeschrittenere Verfahren stabilisieren die Segmentierungstechnik dadurch, dass auch die Umgebung eines Pixels analysiert wird. Wenn alle benachbarten Pixel einer Hintergrundfarbe entsprechen, so ist die Wahrscheinlichkeit auch klein, dass ein isoliertes Objektpixel gefunden wurde.

Sogenannte *Regiongrowing-Verfahren* fordern keinen monochromen Hintergrund, jedoch einen weitgehend texturfreien [BUR84]. Von einem Startpunkt, der sicher als Hintergrund identifizierbar ist, "fließt" eine Fläche in die Bildbereiche, in denen sich die Hintergrundfarbe nur langsam verändert oder sich die Grauwerte in einem bestimmten Intervall bewegen. Dadurch werden im Inneren eines Objektes auch keine Punkte fälschlicherweise als Hintergrund erkannt, wenn die Bereiche nicht zufällig eine Verbindung miteinander haben. All diesen Verfahren ist eigen, dass der Hintergrund mehr oder weniger künstlich festgelegt wird, um eine einfache Abgrenzung vom eigentlichen Objekt zu ermöglichen.

Für die nachträgliche Segmentierung, so wie sie z. B. täglich auf der Grundlage von Photos in den Medien stattfindet, eignen sich diese Verfahren nicht, weil der Hintergrund vorgegeben ist. Hier gibt es keine andere Möglichkeit, als die Grenze zwischen dem Objekt und dem Hintergrund aufzufinden. Unabhängig vom verwendeten Verfahren muss man eine Annahme darüber machen, was den Hintergrund oder das Objekt kennzeichnet, oder zumindest was typisch für den Übergang dazwischen ist [JAI98, MOR95].

5.2 Der Verfahren *SemiSeg*

Das im Rahmen dieser Dissertation neu entwickelte halbautomatische Segmentierungsverfahren SemiSeg [HAE00] gehört zu den Methoden, die versuchen, eine geschlossene Trajektorie (einen Pfad) zu finden, die ein Objekt begrenzt. Hierzu definiert der Benutzer beispielhaft eine Linie, die unmittelbar auf der Grenze des zu segmentierenden Objektes liegt. Dabei befindet sich das Objekt mehr oder weniger vollständig auf der einen Seite der Linie, der Hintergrund auf der anderen (siehe Abb. 5.2). Wie scharf dabei "mehr oder weniger" definiert ist, hängt von der Natur des Objektes ab. Viele Objekte der realen Welt unterscheiden sich eindeutig von ihrer Umgebung. Beispiele hierfür sind Menschen, Autos, Häuser etc. Da die Aufgabe der Segmentierung in der möglichst eindeutigen Abgrenzung liegt, erweisen sich alle Objekte mit unscharfen Rändern als problematisch. Zu solchen Objekten zählen viele Pflanzen, Haare, Wolken usw.

Der im Folgenden vorgestellte Algorithmus (siehe Software *SemiSeg* im gleichnamigen Verzeichnis) analysiert die benutzerdefinierte Linie und versucht den Pfad entlang des Objektes selbständig fortzusetzen. Dabei wird das aus der Analyse bekannte Wissen verwendet, um die richtige Richtung zu finden. Manchmal kann es vorkommen, dass das Objekt plötzlich seine Charakteristik verändert. Dann muss der Benutzer eingreifen, um den Algorithmus wieder auf den richtigen Pfad zu setzen.

Obwohl gerade das SemiSeg-Verfahren bei der Verfolgung schlecht definierter und unruhiger Grenzen seine Stärken ausspielt, wird in einigen Fällen auch eine dem Objekt optimal angepasste Trajektorie kein vollkommen zufriedenstellendes Ergebnis erzeugen. Dies ist besonders dann der Fall, wenn Teile des Objektes selbst Licht emittieren oder auch einen Teil des Hintergrundes durchscheinen lassen. Für wirklich perfekte Ergebnisse kann man dann die Einführung eines Alphakanals nicht vermeiden. Der Alphakanal gibt an, in welchem Maß eine Objektfarbe mit dem Hintergrund gemischt werden muss, wenn man das Objekt in einen anderen Kontext setzt.

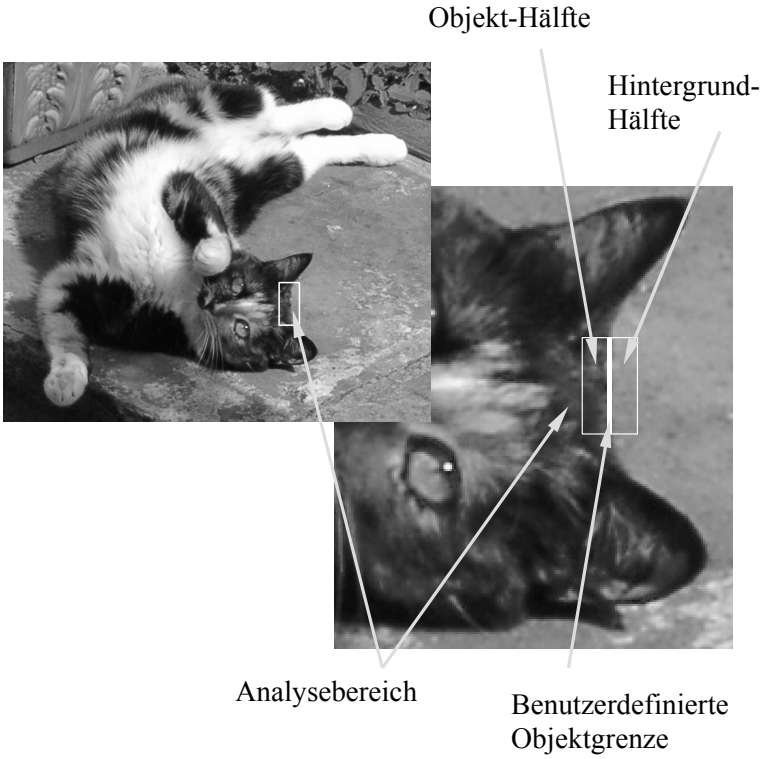

Abbildung 5.2: Typisches Beispiel für eine unscharf definierte Objektgrenze. Der Analysebereich ist so gewählt, dass dessen Mitte das Bild in Objekt- und Hintergrundbereich aufteilt.

Die Berechnung eines solchen Alphakanales für jedes Pixel ist in der Praxis nur unter besonderen Bedingungen möglich. Die Schwierigkeit liegt darin, dass jede Objektfarbe von der (teilweisen) Beeinflussung durch den durchscheinenden Hintergrund bereinigt werden muss. Hierzu muss aber die Hintergrundfarbe bekannt sein (siehe auch US Patent 6.134.345 und 6.134.346 von A. Berman [BER00] und [CHU01], [CHU02]).

In der Praxis der Segmentierung, in Film, Fernsehen, den Printmedien und mit Einschränkungen den medizinischen Anwendungen, sind Objekte ausreichend gut definiert, so dass die Problematik der Transparenz keine große Rolle spielt. Sie wird jedoch am Ende dieses Kapitels nochmals aufgegriffen.

5.2.1 Auffinden einer Kantencharakteristik

Damit eine Objektgrenze durch den Betrachter als solche wahrgenommen wird, muss sie irgendeine Form von Kontinuität aufweisen. Die Aufgabe der halbautomatischen Segmentierung besteht darin, eine solche Kontinuität möglichst verlässlich zu verfol-

gen. Leider kann die Art und Weise, in der sich eine Grenze entlang des Objektes stetig fortsetzt, sehr unterschiedlich definiert sein.

In den meisten Fällen wird das Objekt selbst eher kontinuierlich sein, während sich der Hintergrund unruhig verhalten kann. Hier sollte sich das Segmentierungsverfahren möglichst auf die Seite des Objektes verlassen, um nicht durch den Hintergrund fehlgeleitet zu werden. In anderen Fällen verdeckt ein sehr wechselhaftes Objekt einen eher uniformen Hintergrund. Dann sollte die umgekehrte Vorgehensweise gewählt werden.
Manchmal schlagen alle Annahmen fehl: Ein Gepard steht in einer unruhigen Graslandschaft. Die Kanten des Grases zeigen weitgehend zufällig in unterschiedliche Richtungen. Der Gepard selbst ist zwar ein geschlossenes Objekt, dessen begrenzende Kanten aber immer wieder durch schwarze Punkte unterbrochen werden. Fälle, in denen sich Objekte zumindest an bestimmten Stellen fast perfekt in ihren Hintergrund einpassen, lassen sich aber nicht nur im Tierreich finden. In Abbildung 5.1 hat der Regenmantel im Bereich der rechten Schulter wegen der Spiegelung exakt die Farbe des Hintergrundes. Ohne das Vorwissen des Benutzers über die typische Form von Menschen, in Verbindung mit der Art und Weise, in der unterschiedliche Stoffe am Körper "fallen" oder Falten schlagen, ist die Aufgabe der Segmentierung in diesem Teil des Bildes nicht zu leisten.

Abhängig von der Beschaffenheit des Objektes, können unterschiedliche Merkmale zum Verfolgen der Objektgrenze herangezogen werden. Der im nächsten Kapitel beschriebene *Live-Wire* Ansatz enthält eine ganze Reihe von Maßen, die mittels Gewichtungen mehr oder weniger in eine Bewertungsfunktion eingehen. Ob die Bewertungsfunktion tatsächlich die Charakteristik einer Objektbegrenzung beurteilt, hängt bei Live-Wire wesentlich von der Gewichtung der Maße ab. SemiSeg versucht, insbesondere in diesem Punkt einen anderen Weg zu gehen. Die vom Benutzer definierte Vorgabe wird auf verschiedenen Skalen untersucht. Meist findet sich dann die Kontinuität, die der menschliche Betrachter erkennt, auf einer oder mehreren Skalen, manchmal in wenigen oder allen Koeffizienten einer Skala wieder. Das genaue Vorgehen wird im Folgenden beschrieben.

5.2.2 Iterative Suche nach dem optimalen Pfad

Am Anfang initialisiert der Benutzer den Algorithmus durch Definition eines Rechtecks, das auf der Objektbegrenzung liegt (Abbildung 5.3). Das Rechteck ist sozusagen der Anfang des Pfades (auch Trajektorie) um das Objekt herum. SemiSeg analysiert den so definierten Bildausschnitt und führt den Pfad entlang des Objektes fort.

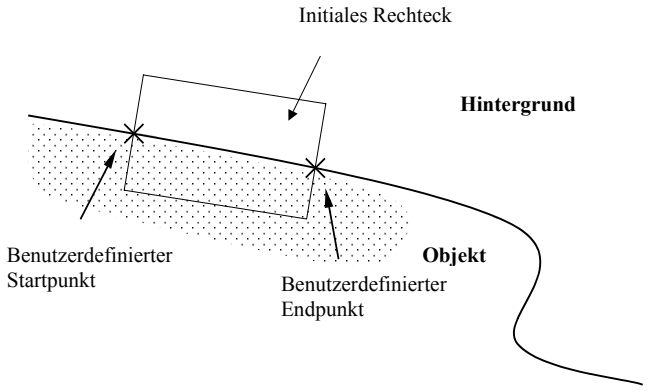

Abbildung 5.3: Das vom Benutzer festgelegte Rechteck definiert die Objektgrenze und dient als Initialisierung des SemiSeg Verfahrens.

Der Algorithmus muss nun entscheiden, in welche Richtung sich die Objektgrenze fortsetzt. Hierzu wird ein Bündel von sogenannten Suchrechtecken an das Ende des benutzerdefinierten Rechtecks angehängt. In Abbildung 5.4 ist zu sehen, dass deren Startpunkte mit dem Endpunkt des letzten Rechtecks zusammenfallen. Lediglich der Winkel der Suchrechtecke kann variiert werden.

Für einen bestimmten Winkel wird der Unterschied des Suchausschnittes zum benutzerdefinierten Ausschnitt minimal. Im Beispiel aus Abbildung 5.4 wäre das optimale Rechteck etwa das mit dem Winkel 0°, da sich die Objektgrenze zumindest kurzfristig geradeaus fortsetzt.

Das optimale Suchrechteck wird also durch Testen verschiedener Winkel im Suchbereich ermittelt. Die Öffnungswinkel des Suchbereichs wurden zuvor festgelegt. Je kleiner der Öffnungswinkel gewählt wird, desto schlechter folgt der Algorithmus schnellen Schwingungen in der Objektbegrenzung, desto geringer wird aber auch der Aufwand für die Suche.

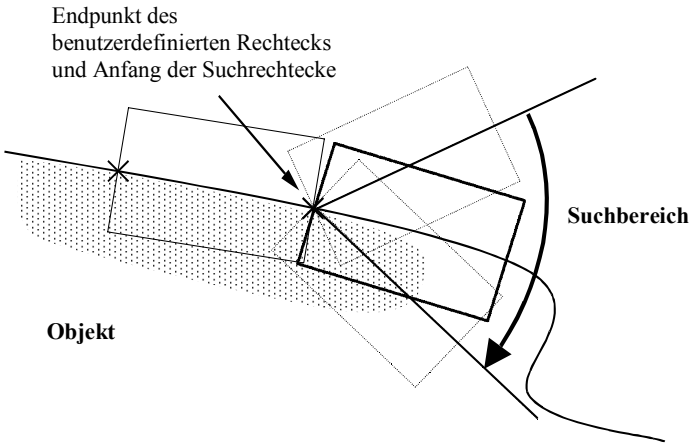

Abbildung 5.4: Am Ende des benutzerdefinierten Rechtecks schließt sich ein Suchrechteck an. Dieses hängt am vorherigen Rechteck fest, hat somit also nur noch die Drehung als verbleibenden Freiheitsgrad. Für unterschiedliche Winkel entsteht so jeweils ein Bildausschnitt.

Bemerkung: In der Praxis zeigt sich, dass das Laufzeitverhalten des Algorithmus im Vergleich zu anderen Algorithmen, wie z. B. dem Live-Wire Ansatz, wenig zeitkritisch ist. In der dieser Arbeit zugrundeliegenden Implementierung wurde mit unterschiedlichen Inkrementen für die Winkelauslösung experimentiert. So wurde das Suchrechteck z. B. in 5° anstatt in 1° Schritten gedreht. Zwischen den beiden besten Treffern wurde dann nochmals mit einer feineren Winkelauflösung abgetastet. Insgesamt konvergierte die gefundene Objektgrenze so schneller gegen die tatsächliche, wobei das Ergebnis nicht merklich unterschiedlich ausfiel.

Das weitere Verfahrens beschäftigt sich mit der Frage, welches der Suchrechtecke dem vorhergehenden möglichst ähnlich ist. Hierzu wird eine geeignete Metrik entwickelt.

5.2.3 Waveletbasierte Zerlegung der Bildinformation

Aus der Bildverarbeitung bieten sich eine Reihe bekannter Maße an, wie z. B. die PSNR (vergleiche Abschnitt 1.6). Auch könnte man je zwei Bildausschnitte zu zwei Vektoren "plattklopfen" und z. B. den euklidischen Abstand berechnen. Ein solches, eher absolutes Abstandsmaß wird aber nicht immer dem gerecht, was tatsächlich bewertet werden soll. Denn die Gleichmäßigkeit, die in der Objektgrenze enthalten ist, kann sich evtl. nur in bestimmten Eigenschaften zeigen. Vor allem dann, wenn sich Hintergrund oder Objekt entlang der Objektgrenze stark verändern, kann es leicht vorkommen, dass Bereiche

innerhalb oder außerhalb des Objektes besser zum vorherigen Rechteck passen als der weitere Verlauf der Objektgrenze. Daher soll nun ein Abstandsmaß entwickelt werden, welches sich anschaulich gesprochen an dem orientiert, was die Kontinuität in der Objektbegrenzung ausmacht und möglichst die Eigenschaften ausblendet, die mit der Begrenzung nichts zu tun haben.

Mehr noch als ein ähnlicher Bildausschnitt sollte also mit der Metrik eine möglichst kohärente Struktur bewertet werden, wie sie auch im jeweils vorausgegangenen Rechteck zu sehen war. Mit anderen Worten: Der Algorithmus bzw. die Metrik sollten möglichst die Regelmäßigkeit extrahieren, die der Benutzer durch Vorgabe des initialen Rechtecks definiert hat.

In diesem Zusammenhang bietet sich die Verwendung einer Wavelet-Zerlegung des Bildsignales an. Damit ist die Hoffnung verbunden, dass sich die Regelmäßigkeit in der kohärenten Struktur besonders auf bestimmten Skalen finden lässt. Skalen, die eine hohe Regelmäßigkeit aufweisen, sollen besonders in die Metrik einfließen, diejenigen Skalen ohne signifikante Regelmäßigkeit sollen dagegen möglichst ausgeblendet werden. Damit werden der Suche gewissermaßen "die Scheuklappen" für die irrelevanten Bildinformationen aufgesetzt.

> Bemerkung: Manchmal wird in Zusammenhang mit der Segmentierung auch dafür plädiert, das Bild zuvor z. B. durch Diffusionsfilter (wie in Kapitel 4 verwendet) zu bearbeiten, um kohärente Strukturen besser herauszuarbeiten. Weiter unten wird diese Möglichkeit genauer evaluiert. Nach den im Rahmen dieser Arbeit gewonnenen Erkenntnissen sollten bestimmte Eigenschaften wenn möglich eher in die Kosten- bzw. Bewertungsfunktion eingebaut werden. Denn durch die Filterung werden dem Bild niemals neue Informationen hinzugefügt, eher gehen diese verloren. Für den menschlichen Betrachter können durch Filtern eines Bildes Qualität und Erkennbarkeit bestimmter Strukturen deutlich verbessert werden. Für den Segmentierungsalgorithmus sollte die Vorverarbeitung aber keinen Unterschied machen. Vielmehr sollte man versuchen, die gewünschten Strukturen in die Bewertungsfunktion einzubauen, ohne dabei das eigentliche Bild zu verändern.

Eines der wichtigsten Argumente hierfür ist, dass sich eine Objektgrenze im Verlauf verändern kann. Dies bietet einem Segmentierungsverfahren die Möglichkeit, sich langsam vollziehenden Änderungen anzupassen. Eine globale Vorverarbeitung kann dagegen nur das gesamte Bild anhand eines Kriteriums verändern. Eine sukzessive Anpassung der Segmentierung an eine sich verändernde Objektgrenze ist so nicht möglich. Dagegen ist offensichtlich, dass im Gegensatz dazu jede globale Filterung des Bildes natürlich o. B. d. A. auch lokal in der Bewertungsfunktion stattfinden kann.

Um je zwei Bildausschnitte miteinander vergleichen zu können, müssen diese normiert werden. Zwar kann man per Definition die Rechtecke jeweils gleich groß halten, sie müssen jedoch für den Vergleich in das gleiche Koordinatensystem gedreht werden, oder mit anderen Worten: Die jeweils unterschiedliche Drehung des benutzerdefinierten Rechtecks und des Suchausschnittes muss angeglichen werden.

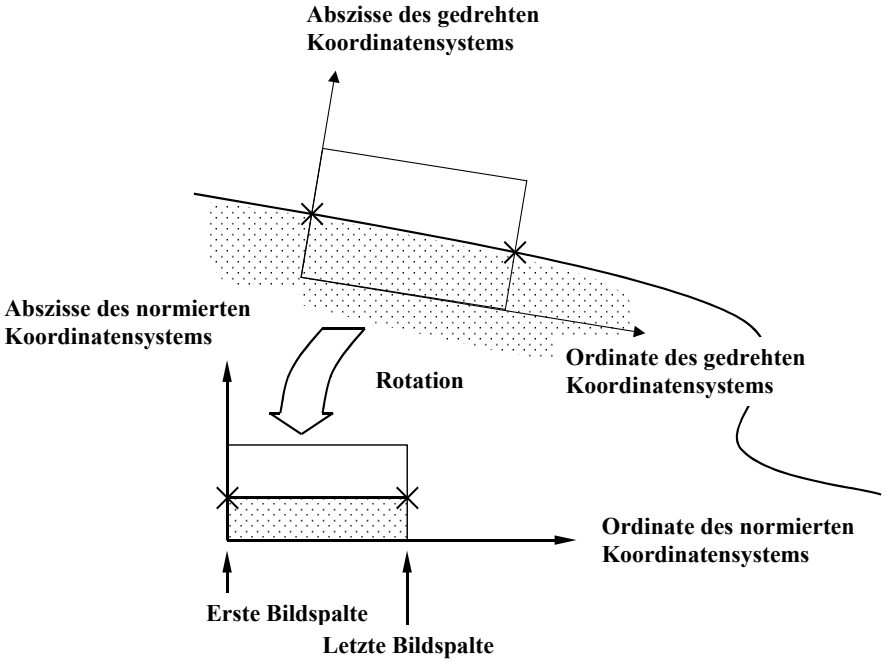

Abbildung 5.5: Der durch das benutzerdefinierte Rechteck gewählte Bildausschnitt muss ebenso wie alle anderen Ausschnitte durch Drehung in ein gemeinsames Koordinatensystem normiert werden.

Dabei soll die Seite des Rechtecks entlang der Objektgrenze nach der Drehung parallel zur Ordinate verlaufen (siehe Abbildung 5.5). Ein einfaches, erneutes Abtasten der Bildpunkte zum Zweck der Drehung würde zu starken Aliasing-Effekten führen. In der

späteren Analyse kann es daher vor allem auf den hochfrequenten Skalen Störungen geben, so dass diese für das Verfolgen der Objektbegrenzung nicht mehr brauchbar sind. Es ist daher zu empfehlen, das Bild bei der Drehung leicht zu filtern - z. B. mit einem schwachen Gaussfilter - um den verfremdenden Treppcheneffekt, vor allem bei kleinen Drehwinkeln, zu vermeiden.

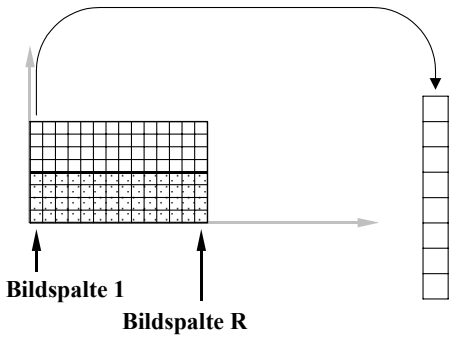

Abbildung 5.6: Das benutzerdefinierte Rechteck wird Spalte für Spalte analysiert.

Nach der Drehung eines Bildausschnittes kann die eigentliche Analyse beginnen. Ziel sollte sein, die vom Benutzer durch das Rechteck definierte Struktur zu erkennen und später wiederzufinden. Dazu wird der Bildausschnitt Spalte für Spalte mittels einer eindimensionalen kontinuierlichen Wavelet-Transformation auf verschiedenen Skalen analysiert (siehe Kapitel 4.1.6, insbesondere Abbildung 4.9). So entsteht auf unterschiedlichen Skalen eine redundante Zerlegung jeder einzelnen Spalte.

$$\bar{k}_{i,s} = \frac{1}{R}\sum_{r=1}^{R} k_{i,s}^{r} \qquad \text{(Ausdruck 5.1a)}$$

Die Koeffizienten der Zerlegung sind mit $k_{i,s}^{r}$ bezeichnet. Die Sub- und Superskripte haben folgende Bedeutung:

r: Die Spalte des rechteckigen Bildausschnittes
s: Skala, auf der der Koeffizient definiert ist
i: Index des Koeffizienten (auf Skala s)

R bezeichnet die Anzahl der Spalten, die nur von der Länge des benutzerdefinierten Rechtecks bzw. der Suchrechtecke abhängt und frei gewählt werden kann. Die Koeffi-

zienten werden dann über alle Spalten gemittelt. Die so entstandenen Durchschnittskoeffizienten werden mit $\bar{k}_{i,s}$ bezeichnet. Nun sind die Voraussetzungen geschaffen, die Varianz eines jeden Koeffizienten über alle Spalten zu berechnen (Ausdruck 5.1). Übrig bleibt eine Maßzahl $\mathrm{var}_{i,s}$ für jeden Koeffizienten i auf jeder Skala s, die angibt, wie stark jeder Koeffizient variiert. Der Spaltenindex r ist natürlich entfallen, denn auf Grundlage der Spalten wurde die Varianz ja gerade berechnet.

$$\mathrm{var}_{i,s} = \frac{1}{R}\sum_{r=1}^{R}\left(k_{i,s}^{r} - \bar{k}_{i,s}\right) \qquad \text{(Ausdruck 5.1b)}$$

Im Folgenden soll nochmals detailliert auf die Bedeutung der Skalen und der Varianz der einzelnen Koeffizienten eingegangen werden: Eine Skala s kann man sich in Einheiten von Pixeln vorstellen. Die Schwingung mit der größten denkbaren Frequenz findet für s=2 innerhalb jeweils benachbarter Pixel statt. Benachbarte Pixel können zwischen zwei Grauwerten oszillieren. Schnellere Schwingungen sind aufgrund des Theorems von Shannon nicht möglich. Weitere Schwingungen können nur noch auf tieferen Frequenzen stattfinden und werden durch die kontinuierliche Wavelet-Analyse für größere s berücksichtigt. In der Implementierung von SemiSeg wurde zugunsten einer einfachen Implementierung der Haar-Filter eingesetzt. Andere Filter sind aber ebenfalls denkbar und haben aus den in Kapitel 4 diskutierten Gründen evtl. bessere Eigenschaften für die Zerlegung eines Signales. Die Wahl fiel trotzdem auf den Haar-Filter, weil die zu analysierenden Pixel-Spalten (aus Abb. 5.6) meist sehr schmal sind. Die Größe der Spalten in Pixel variiert natürlich mit der Größe des Bildes und dem aufgenommenen Motiv. In den Tests im Rahmen dieser Arbeit wurden Bilder mit einer Auflösung von 500.000 bis 1 Mio. Pixel verwendet. Typische Breiten von Objekträndern bewegen sich in der Größenordnung von etwa 10 Pixeln. Deutlich breitere Suchrechtecke würden unnötig viel Information vom Hintergrund oder dem Objekt selbst erfassen. Betrachten will man aber nur den eigentlichen Übergang. Wenn das Signal selbst aber nur 10 Pixel "breit" ist, können keine Filter der Länge 6 oder mehr verwendet werden. Um überhaupt eine ausreichende Anzahl von Koeffizienten zu extrahieren, bietet sich also der Haar-Filter mit nur zwei Werten an. Zusammenfassend kann man festhalten, dass der Parameter s für die Skala einer mehr oder weniger starken Fokussierung auf das Signal entspricht. Klei-

ne Werte für s stehen für eine lokal begrenzte Analyse, große Werte berücksichtigen die tiefen Schwingungen.

Der Varianz eines Koeffizienten kommt im SemiSeg-Verfahren eine zentrale Bedeutung zu. Sie gibt an, in welchem Maß er entlang des benutzerdefinierten Rechtecks um seinen Mittelwert schwingt. Man kann den Grad dieser Schwingung nutzen, um die Signifikanz eines Koeffizienten für die Objektgrenze zu bewerten. Warum ist das so? Geht man durch das benutzerdefinierte Rechteck vom Start- zum Endpunkt, so lässt sich auf dieser Strecke ein bestimmter Koeffizient beobachten, ebenso wie man auch einzelne Pixel beobachten könnte. Wenn sich die Spalten des rechteckigen Bildausschnittes überhaupt nicht verändern, so werden auch die Koeffizienten konstant bleiben. Um die Objektgrenze weiter durch das Bild zu verfolgen, müsste man lediglich Pixelspalten finden, die der Vorgabe durch den Benutzer möglichst genau entsprechen. Dies ist gewissermaßen der triviale Fall. In der Regel ist mit einer solchen gleichförmigen Objektgrenze nicht zu rechnen. Das Objekt könnte z. B. gemustert sein und der Hintergrund texturiert. In diesem Fall sieht jede Spalte des Rechtecks evtl. anders aus als ihre Nachbarn. Trotzdem muss es eine bestimmte Form der Kontinuität geben, die für den menschlichen Betrachter offensichtlich ist. Z. B. könnte das Objekt vorwiegend dunkel und der Hintergrund vorwiegend hell sein - auch dann, wenn dies nicht unbedingt für jede einzelne Spalte zutreffen muss. Andere, weniger offensichtliche Arten von Kontinuität (man spricht in diesem Zusammenhang auch von Kohärenz) sind denkbar. Natürlich könnte man auch im Ortsraum nach Ähnlichkeiten suchen. Dabei könnte man z. B. feststellen, dass der mittlere Pixel entlang eines Objektes immer schwarz ist, sich alle anderen Pixel jedoch völlig zufällig verändern. Intuitiv würde man sich bei einer weiteren Suche darauf konzentrieren, eine dünne schwarze Linie zu finden. Alle anderen Pixel würde man nicht beachten, da sie zur Struktur der Grenze nicht beitragen. Der signifikante mittlere Pixel hat also eine geringe Varianz, da er meist schwarz ist. Alle anderen Pixel nehmen zufällige Werte an und haben zwangsläufig eine hohe Varianz. Ebenso wie sich also der Mensch bei der Suche nach ähnlichen Mustern auf die kontinuierlichen Aspekte konzentriert, soll dies auch die automatische Segmentierung tun.

Eine Suche im Ortsraum funktioniert in manchen Fällen tatsächlich schon ganz gut, ist aber suboptimal (hierzu mehr in Kapitel 8). Nun kann auf die ausführliche Vorarbeit der vorangegangenen Kapitel zurückgegriffen werden. Würde man den oben beschriebenen Sachverhalt im Kontext der Statistik beschreiben, so müsste man feststellen, dass die Pixel benachbarter Spalten in einer bestimmten Weise miteinander korreliert sind. Diese Korrelation kann ganz offensichtlich im Ortsraum erkennbar sein, also zur Feststellung führen, dass der mittlere Pixel immer schwarz ist. Es sind aber auch ganz andere Abhängigkeiten denkbar, die man sich nur schlecht vorstellen kann. Glücklicherweise ist diese Anschauung auch nicht nötig. Denn die Eigenvektoren einer Korrelationsmatrix beschreiben ja gerade diese Abhängigkeiten auch ohne Anschauung. Daher ist es sinnvoll, die Pixel einer Spalte nicht unmittelbar zu betrachten, sondern vorher zu transformieren. Die Transformation sollte dabei so gewählt sein, dass das Signal bereits möglichst gut dekorreliert ist. Ganz anschaulich gesprochen könnte man sagen: Was sich im Signal tut, sollte schon an möglichst wenigen Koeffizienten abzulesen sein, statt redundant über alle Pixel verteilt zu sein. Mit eben dieser Begründung wurde die kontinuierliche Wavelet-Analyse gewählt.

5.2.4 Eine Metrik für Kantenvergleiche

Nun fehlt noch eine Metrik, die zwei Bildausschnitte anhand der Koeffizienten auf ihre Ähnlichkeit überprüft. Dies geschieht in Ausdruck 5.2.

$$m = \sum_{r=1}^{R} \sum_{s=1}^{S} \sum_{i=1}^{I_s} \left| \frac{k_{i,s}^{r} - \bar{k}_{i,s}}{1 + var_{i,s}} \right| \qquad \text{(Ausdruck 5.2)}$$

Aus dem benutzerdefinierten Rechteck wurden, wie bereits beschrieben, über alle Spalten r die mittleren Koeffizienten $\bar{k}_{i,s}$ und die Varianz $\mathrm{var}_{i,s}$ berechnet. Ein Suchrechteck (siehe Abb. 5.4) wurde spaltenweise in seine Koeffizienten $k_{i,s}^{r}$ zerlegt. Im wesentlichen wird in Ausdruck 5.2 lediglich erfasst, wie stark die Koeffizienten eines zu prüfenden Suchrechtecks zu den mittleren Koeffizienten aus dem benutzerdefinierten passen. Koeffizienten, die besonders nahe an ihrem Mittelwert liegen, weisen auf eine hohe Ähnlichkeit der Bildausschnitte hin. Ist der Unterschied dagegen groß, so kann eine hohe

Varianz des Koeffizienten verhindern, dass dieser zu stark in die Summe eingeht. Hat der Koeffizient dagegen eine hohe Signifikanz (also geringe Varianz), so ist der Nenner klein und der Beitrag zur Summe berechtigterweise groß.
Die Maßzahl m wird für jedes Suchrechteck berechnet. Dasjenige mit dem kleinsten Wert passt am besten zur Vorgabe des Benutzers und gibt die Richtung an, in die sich die Objektgrenze fortsetzt. In unserer Implementierung SemiSeg wird die gefundene Silhouette des Objektes jedoch nur einige wenige Pixel fortgesetzt und nicht um die gesamte Länge des Suchrechtecks erweitert. Das Fortschreiben in entsprechend kleinen Schritten hat zur Folge, dass die Grenze exakter gefunden werden kann. Die Länge des Suchrechtecks gibt dem Algorithmus also eine gewisse Stabilität in der Suche, weil z. B. kurze Lücken in der Kontur nicht so ins Gewicht fallen und besser überbrückt werden können. Das eigentliche, langsame Fortschreiten führt aber zu einem exakteren Ergebnis.

5.3 SemiSeg im Probandentest

Schon im Vorfeld war klar, dass halbautomatische Segmentierungsverfahren nicht besser sein können als eine vollständig vom Benutzer gesteuerte Segmentierung. Ein Schwerpunkt des Vergleiches von SemiSeg mit anderen Verfahren war daher, den möglichen Nutzen für den Benutzer vor allem in Hinblick auf Zeitersparnis zu untersuchen. Mortensen und Barret machen einen der wenigen dokumentierten Versuche, ihren Algorithmus zu evaluieren [MOR95, MOR99, BAR97]. Dabei ist das einzige Vergleichsverfahren die vollständig benutzergesteuerte Handsegmentierung. Gemessen wurde dabei die Zeit, die Benutzer für die Aufgabe brauchten, die in der Segmentierung unerfahren waren. Die Qualität der Segmentierung wurde durch die mittlere Abweichung in Pixeln von der so ermittelten optimalen Segmentierung gemessen.

Zur Beurteilung der Qualität von Segmentierungsergebnissen der Benutzer, scheint die Verwendung des mittleren Abstandes der gefundenen Objektgrenze von der optimalen ein relativ objektives Maß zu sein. Wir werden sie daher noch einmal in Kapitel 8 für die eigene Evaluation aufgreifen. Dagegen hat sich bei der Evaluation der unten beschriebenen Methoden herausgestellt, dass fortgeschrittenere Verfahren gegenüber einer

Handsegmentierung überdurchschnittlich stark von der Übung des Benutzers profitieren. Daher wurde bei der Evaluation im Rahmen dieser Arbeit die Qualität des Ergebnisses durch die Anzahl der vom Benutzer gesetzten Punkte geteilt, um von der Einflussgröße Zeit unabhängiger zu werden.

Das grundsätzlich sinnvolle Abstandsmaß zur Beurteilung der Qualität konnte hier nicht verwendet werden, da eines der Vergleichsverfahren regionenbasiert arbeitet. Dabei wird einfach jedes Pixel als zum Objekt oder zum Hintergrund gehörend klassifiziert. Eine Kontur im eigentlichen Sinne entsteht gar nicht. Um die Qualität dennoch einschätzen zu können, wurden die Ergebnisse sieben Testpersonen gezeigt, die sie nach subjektiven Maßstäben beurteilen sollten.

5.3.1 Vergleichsverfahren

Die folgenden vier Methoden wurden miteinander verglichen:

Methode 1: Kantenbasierter Detektor

Der Benutzer beschreibt einen geschlossenen Polygonzug durch Definition einzelner Punkte entlang der Objektbegrenzung. Zwischen den Punkten folgt der Verlauf der stärksten Kante. Gegenüber einem einfachen geschlossenen Polygonzug hat dieses Verfahren (oft) den Vorteil, dass nicht unnötig viele Punkte gesetzt werden müssen, da sich der Algorithmus über weite Strecken an einer Kante orientieren kann - so eine solche vorhanden ist. Unangenehm wird die Interaktion mit dem Verfahren an Stellen, an denen die eigentliche Objektgrenze nahe an einer starken, aber dem Objekt fremden Kante verläuft. Hier muss der Algorithmus im ungünstigsten Fall pixelweise zum Verfolgen der richtigen Grenze gezwungen werden. Einen ganz ähnlichen Charakter hat das von Mortensen und Barret vorgeschlagene Verfahren, welches auch gelegentlich "wie ein Magnet" (Zitat der Autoren) von einer fremden Kante angezogen wird.

Methode 2: Regionenbasiertes Verfahren

Hier wird versucht, möglichst große, bezüglich einer Eigenschaft homogene Regionen zu finden. Meist wird einfach ein ähnlicher Grau- oder Farbwert in der Nachbarschaft gesucht. Um einen benutzerdefinierten Punkt herum wird so eine Region geflutet. Offensichtlich funktioniert das Verfahren vor allem dann, wenn sich Objektpixel durch leicht zu identifizierende, ähnliche Eigenschaften auszeichnen, also z. B. ähnliche Grauwerte. Wenn dies nicht zutrifft, kann es für den Benutzer sinnvoll sein, statt dessen den Hintergrund zu selektieren. Verwendet wird später der dazu komplementäre Bereich durch Invertieren der Selektion. Im komplementären Bereich sollte sich nur noch das Objekt selbst befinden. Sind jedoch weitere Objekte enthalten, so hätte man diese im vorhergehenden Schritt zusammen mit dem Hintergrund selektieren müssen.

In Bildern mit ebenso detailreichen Objekten wie Hintergründen kann eine Vorsegmentierung die Arbeit vereinfachen, z. B. unter Verwendung des Watershed-Verfahrens [OLS97].

Methode 3: Vollständig benutzergesteuerte Segmentierung

Hierbei obliegt dem Benutzer selbst das pixelgenaue Segmentieren, wobei zur Vereinfachung auch Geradenstücke definiert werden können. Das in der Praxis zu aufwändige und daher kaum praktikable Verfahren wurde hinzugenommen, um jeweils eine optimale Referenzsegmentierung zu erhalten.

Methode 4: SemiSeg

SemiSeg zeigt seine Stärken vor allem da, wo Methode 1 Schwächen aufweist, da es immer versucht, eine möglichst ähnliche Objektgrenze zu finden. Ein Abdriften zu dominanten benachbarten Kanten ist dem Verfahren daher völlig fremd. Nachteilig ist, dass die Verfolgung der Objektgrenze immer damit endet, dass die Trajektorie in den Hintergrund oder das Objekt abdriftet. Dann muss SemiSeg gestoppt und der letzte Teil des Pfades gelöscht werden. Dem gegenüber hat Live-Wire (mehr dazu in Kapitel 6) den Vorteil, dass der Pfad immer interaktiv angezeigt wird. Zwar neigt Live-Wire eher

noch stärker zum Abdriften, jedoch kann der Benutzer unverzüglich an den letzten, bestmöglichen Punkt zurückgehen, bei dem der Pfad noch am Objekt liegt. Das nächste Kapitel widmet sich deshalb dem Versuch, die Stärken im Bereich der Interaktivität von Live-Wire mit der Unbeirrbarkeit von SemiSeg zu kombinieren.

5.3.2 Testergebnis und Interpretation

Die Bilder am Anfang jeder Serie sind jeweils die Originalbilder, nach rechts folgen die Methoden 1 (kantenbasiert), 2 (regionenbasiert), 3 (vollständig benutzergesteuert) und 4 (SemiSeg). In Tabelle 5.1 kann die Bewertung des Bildes durch die Probanden sowie der Aufwand zur Segmentierung (Anzahl der benötigten Punkte bzw. Mausklicks) abgelesen werden.

Abbildung 5.7: Beispiel *Africa*. Originales Bild (ganz links), kantenbasierte, regionenbasierte, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts).

Beispiel *Africa* zeigt eine Frau mit Kind vor einem unruhigen Hintergrund. Wie aus der Tabelle ersichtlich, war die kantenbasierte Segmentierung *Africa1* zwar sehr aufwändig, jedoch auch erfolgreich. Lediglich im Bereich der Haare und Arme sind einige helle Flecken an den Rändern zu erkennen, die mit dem Verfahren ohne hohen zusätzlichen Aufwand schwer zu entfernen waren.

Bild	Klicks	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	Summe
Africa1	102	6	7	6	5	6	5	6	6	5	6	5	5	68
Africa2	57	2	2	4	2	2	2	1	2	1	2	1	1	22

Africa3	216	6	6	7	6	7	6	7	7	6	6	6	6	76
Africa4	59	3	3	3	3	3	3	2	4	1	4	2	3	34
Fashion1	55	4	5	6	5	5	3	5	5	6	5	5	5	59
Fashion2	2	3	3	5	6	6	6	6	6	5	6	7	6	65
Fashion3	110	4	3	6	5	4	4	6	5	5	5	6	5	58
Fashion4	2	5	6	7	7	7	7	7	5	7	6	7	7	78
Noise1	60	3	3	6	5	5	4	6	4	6	5	5	5	57
Noise2	4	2	1	3	1	2	2	1	2	1	3	1	1	20
Noise3	150	4	5	6	6	6	5	6	5	5	6	6	6	66
Noise4	2	4	7	6	7	7	7	7	6	7	6	7	7	78
Sea1	28	3	3	4	4	4	4	6	3	3	5	5	5	49
Sea2	40	2	1	3	2	4	2	2	2	1	3	3	4	29
Sea3	83	5	7	6	7	7	6	7	7	6	6	7	7	78
Sea4	21	4	5	7	5	6	5	7	7	5	6	7	5	69
Clouds1	54	4	6	7	6	6	6	7	6	5	4	7	7	71
Clouds2	106	2	1	3	3	4	3	1	2	1	6	3	2	31
Clouds3	123	3	7	6	6	3	6	6	7	4	5	5	5	63
Clouds4	9	3	4	7	5	5	5	6	6	4	6	6	6	63
Veget1	22	5	6	5	5	4	3	5	6	4	4	5	4	56
Veget2	73	2	1	3	1	3	2	2	1	1	3	2	2	23
Veget3	56	3	7	6	7	3	5	5	6	4	5	4	5	60
Veget4	12	4	5	7	4	3	4	4	6	5	4	3	4	53
Bear1	17	4	5	5	3	3	5	6	5	2	5	4	6	53
Bear2	48	2	1	2	1	1	1	1	1	1	1	1	1	14
Bear3	49	6	6	7	7	7	7	7	6	7	6	6	7	79
Bear4	10	4	4	6	4	4	5	7	5	3	4	5	6	57

+

Tabelle 5.1: Bewertung der unterschiedlichen Segmentierungsverfahren (1-4) für unterschiedliche Bilder durch zwölf Personen (P1-P12). Die Bewertungsskala reicht von 1=schlecht bis 7=sehr gut.

Im Bild daneben wurde die regionenbasierte Segmentierung angewandt. Da sich weder die Personen noch der Hintergrund durch eine besonders gleichmäßige Struktur auszeichnen, liefert das Verfahren aus Sicht der Testpersonen das schlechteste Ergebnisse. Jede der vielen kleinen Regionen im Hintergrund musste eigens angewählt werden. Dem hohen Aufwand steht hier ein dennoch unbefriedigendes Ergebnis gegenüber. Das Verfahren ist in diesem Fall offensichtlich ungeeignet. Ein Praktiker würde das Ergebnis evtl. durch Auslöschen der kleinen Regionen mit einem passenden Werkzeug retten. Trotzdem bleibt auch danach ein ausgefranster Objektrand übrig. Im vierten Bild von links wurde vollständig benutzergesteuert segmentiert. Obwohl das Ergebnis sehr gut ist, allerdings mit 216 Klicks auch mit Abstand den größten Aufwand verursachte, sind immer noch kleine Fehler enthalten. Im letzten Bild wurde SemiSeg verwendet. Die horizontalen Streifen sind Folge numerischer Ungenauigkeiten einer noch nicht ganz produktisierten Implementierung, wurden aber der Fairness wegen dennoch so in den Test

übernommen, wie dies in der Abbildung zu sehen ist, obwohl die Streifen keine Eigenart des Verfahrens selbst sind.

Trotz dieser Artefakte bekam hier SemiSeg von den Probanden die beste Gesamtnote im Sinne des Quotienten, in dem die Qualität im Zähler und der Aufwand im Nenner steht. Allerdings rettete in diesem Fall nur der geringe Aufwand die mäßige Qualität der Segmentierung. Im Test stellte sich das Beispiel *Africa* für alle Verfahren als eines der problematischsten heraus. Kein Verfahren führte zu einem richtig perfekten Ergebnis.

Abbildung 5.8: Bild *Fashion* (ganz links), kantenbasierte-, regionenbasierte-, benutzergesteuerte Segmentierung, Verwendung von SemiSeg (ganz rechts)

Das zweite Bild des Tests "Fashion" (siehe Abbildung 5.8) war für alle Verfahren eher unproblematisch. Obwohl das Bild keine explizit gespeicherte Objektmaske enthält, ist diese durch den weißen Hintergrund de facto bereits vorgegeben. Die Qualitätsunterschiede sind nur marginal. Dennoch verhielten sich hier einige Verfahren untypisch. So lieferte das regionenbasierte Verfahren eines der exaktesten Ergebnisse, die vollständig benutzergesteuerte Segmentierung schnitt, mit dem deutlichen weißen Rand am schlechtesten ab. Eine Erklärung hierfür könnte sein, dass die Jacke am rechten Rand teilweise etwas weicher in den Hintergrund verläuft und so eine genaue Abgrenzung, zumindest während der Segmentierung, nicht immer ganz offensichtlich ist. Im fertigen Ergebnis stört der Rand dann doch deutlich aufgrund des schwarzen Hintergrundes. Dagegen finden das kantenbasierte Verfahren (zweites Bild von Links) und SemiSeg (Bild rechts) scheinbar genau den richtigen Kompromiss zwischen blau und weiß. Auch das

regionenbasierte Verfahren liefert bis auf wenige kleine Ausreißer am Rand ein sehr gutes Ergebnis. Erwähnenswert ist dabei, dass regionenbasiertes Segmentieren auch immer die Definition eines Schwellwertes beinhaltet. Dieser gibt an, welche Farbtöne noch als Hintergrund aufgefasst werden. Des weiteren konnte man bei der regionenbasierten Methode definieren, ob auf Grundlage des RGB-Wertes, dem Farbton (in der Literatur meist als Hue bezeichnet oder phys. die Wellenlänge des Lichtes) oder der Helligkeit segmentiert werden soll. Im vorliegenden Bild war die Wahl der Helligkeit optimal, da weiß keinem Farbton zugeordnet werden kann.

Besonders das regionenbasierte Verfahren schlägt schnell zwischen optimal und unbrauchbar um. Wäre z. B. der Hintergrund durch einen einzelnen weißen Pixel mit dem ebenso weißen T-Shirt verbunden, so würde der Hintergrund in den falschen Bereich "hineinlaufen". Die einzige Möglichkeit, dies zu verhindern, ist dann, das Bild selbst zu verändern - was in der Praxis unerwünscht ist.

Abbildung 5.9: Bild *Fashion* mit Rauschen (ganz links), kantenbasierte, regionenbasierte, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts)

In Abbildung 5.9, einer künstlich verrauschten Version des Beispiels "Fashion", verhalten sich fast alle Verfahren ähnlich wie im vorherigen Beispiel. Lediglich die auf region-growing basierende Methode ist, wie zu erwarten, verloren, da innerhalb des Hintergrundes überall Pixel vorkommen, die nicht in das Intervall der Hintergrundfarben fallen. Genau so ist das Bild ja auch konstruiert. Trotz des Rauschens ist die Objektkante

noch deutlich zu erkennen, so dass sich hier das kantenbasierte Verfahren ähnlich gut wie SemiSeg schlägt.

Abbildung 5.10a: Bild *Sea* (ganz links), kantenbasierte, regionenbasierte, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts)

In Abb. 5.10a ("Sea") spielt SemiSeg seine volle Stärke aus, sowohl bei der Qualität der Segmentierung als auch im geringen Aufwand. Auf der linken Seite der abgebildeten Person schlägt sich auch noch das kantenbasierte Verfahren gut. Allerdings driftet es im Bereich der rechten Schulter und des Armes mehrfach in den Regenmantel ab. Tatsächlich ist die Kleidung dort aufgrund der Spiegelung der Sonne fast weiß und somit von den Wolken im Hintergrund kaum zu unterscheiden. Das regionenbasierte Verfahren hat an dieser Stelle wie alle anderen ähnliche Probleme, hinterlässt aber zusätzlich viele kleine "Spots", die mit vertretbarem Aufwand kaum zu entfernen sind. Für eine fototechnische Verwertung wäre darüber hinaus der Rand zu ausgefranst. Dass der Hintergrund im unteren Bildbereich nicht in die weiße Hose hineinläuft, ist einer aufwändigen Feinjustierung der Intervallgrenze für die Hintergrundfarbe zu verdanken. Auch wurde der Hintergrund im regionenbasierten Verfahren aus einer Vielzahl von einzelnen Bereichen zusammengesetzt und die inverse Selektion später als Objekt verwendet.

Die letzten beiden Beispiele in Abbildung 5.10b-c zeigen Wolken und Baumwolle, also Objekte, die an den Grenzen weich mit dem Hintergrund verschmelzen. Eine eindeutige Abgrenzung ist hier selbst für den Menschen kaum möglich. Legt der Benutzer jedoch einmal fest, wo im Übergang des Objektes zum Hintergrund die Grenze zu suchen ist, so kann SemiSeg diese Vorgabe sehr verlässlich in eine Segmentierung umsetzen.

Abbildung 5.10b: Bild *Clouds* (ganz links), kantenbasierte, regionenbasierte, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts)

Natürlich hängt immer noch von der Vorgabe ab, ob und wie gut sich die Grenze für eine gegebene Anwendung eignet. Die subjektive Natur dieser Entscheidung wurde besonders in der Evaluation durch die Probanden deutlich. Einige bevorzugten nach eigenen Angaben Objekte mit einer breiten Rand, der immer noch den Hintergrund erkennen lässt, andere wollten diesen lieber vollständig ausgeblendet sehen und waren eher bereit auf einen Teil des Objektes zu verzichten. Auch gab es relativ starke Präferenzen zwischen weichen und kantigen Silhouetten, unabhängig von der tatsächlichen Beschaffenheit der Objekte.

Abbildung 5.10c: Bild *Veget* (ganz links), kantenbasierte, regionenbasierte, benutzergesteuerte Segmentierung, SemiSeg (ganz rechts)

Tabelle 5.2 zeigt ein abschließendes kumuliertes Ergebnis. Dabei wurde für jedes Verfahren in der zweiten Spalte die Summe über alle Noten gebildet, in der dritten Spalte die Summe über die benötigten Interaktionen des Benutzers mit den Verfahren, was dem Aufwand entspricht. Der Quotient aus Qualität und Aufwand steht in der letzten Spalte.

Die Verfahren 1, 3 und 4 erhalten alle ähnlich gute Noten für die Qualität der Ergebnisse, wobei erwartungsgemäß die vollständig benutzergesteuerte Segmentierung mit leichtem Vorsprung am besten abschneidet. Weit abgeschlagen ist dagegen das regionenbasierte Verfahren. Teilt man nun durch die Anzahl der Benutzerinteraktionen mit den jeweiligen Verfahren, so zeigt sich eine deutlich andere Beurteilung der Brauchbarkeit der untersuchten Ansätze. Denn der relativ kleine Unterschied zwischen der manu-

ellen Segmentierung und SemiSeg wurde durch einen fast sieben mal höheren Aufwand für den Benutzer erkauft. Der Grenznutzen, also die zusätzliche Qualität die man bei einer manuellen Segmentierung gegenüber SemiSeg erhält, ist also ausgesprochen gering.

Verfahren	Summe Noten	Anzahl Klicks	S. Noten/ Klicks
Methode 1 (kantenbasiert)	413	338	1.222
Methode 2 (regionenbasiert)	204	330	0.618
Methode 3 (vollst. manuell)	480	787	0.61
Methode 4 (SemiSeg)	432	115	3.76

Tabelle 5.2: Aggregiertes Ergebnis der Auswertung. Dargestellt ist die Qualität (Spalte *Summe Noten*), der Aufwand (Spalte *Klicks*) und der Quotient aus beiden Werten.

Interessanterweise kann das bei der Qualitätsbeurteilung sehr schlecht abschneidende regionenbasierte Verfahren, durch den sehr geringen Aufwand, doch wieder soviel an Boden gutmachen, dass es immerhin den vorletzten Platz erreicht.

Zusammenfassend kann man feststellen, dass SemiSeg nur geringfügig hinter der Qualität einer optimalen Segmentierung zurückbleibt, gleichzeitig aber mit einem Minimum an Aufwand für den Benutzer auskommt.

5.3.3 Kritik an der Evaluation

Eine objektive Bewertung der Verfahren bleibt trotz aller Bemühungen eine kritische Aufgabe. Zwar findet durch die Quotientenbildung eine gewisse Normierung der Qualität auf eine Einheit des Aufwands statt. Dadurch allein wird ein Verfahren aber noch nicht fair bewertet. Denn an welcher Stelle soll der Benutzer mit der weiteren Segmentierung bzw. Verbesserung des Vorgangs aufhören? Jeder weitere Klick könnte das Bild verbessern, jedoch den Aufwand so erhöhen, dass er nicht mehr gerechtfertigt ist.

> Exkurs: In der Betriebswirtschaft gibt es ein analoges Problem: Bis zu welcher Stelle sollte man eine Einheit mehr Aufwand leisten, um noch zusätzlichen Gewinn zu machen? Wenn der funktionale Zusammenhang zwischen Aufwand und Ertrag bekannt ist, so kann man den Punkt berechnen, an dem eine zusätzliche bewertete Einheit Aufwand den zusätzlichen Gewinn übersteigt. Dies ist genau der Punkt, an dem der Quotient aus Ertrag/Aufwand gleich der ersten Ab-

leitung der Kostenfunktion ist. In der halbautomatischen Segmentierung ist allerdings der Zusammenhang zwischen Aufwand und zusätzlicher Genauigkeit der Segmentierung kaum als Funktion zu erfassen.

5.4 Problem unscharfer Objektgrenzen

Zu Anfang des Kapitels wurde bereits erwähnt, dass zwischen Objekten und ihrem Hintergrund halbtransparente Bereiche vorkommen können, z. B. bei Haaren, Fell etc. Grundsätzlich ist das Problem der Behandlung halbtransparenter Objektgrenzen allerdings orthogonal zum Problem der Segmentierung. Denn das Objekt muss unabhängig von der Behandlung und Analyse des Randbereiches zuerst segmentiert werden. Genau genommen gibt es auch bei scharf abgegrenzten Objekten immer einen schmalen Grat der Breite eines Pixels, der teilweise über dem Objekt und teilweise über dem Hintergrund liegt. Denn es ist nicht zu erwarten, dass Randpixel vollständig dem Objekt oder vollständig dem Hintergrund zuzuordnen sind.

Das Problem mit unscharfen Bereichen kann nur gelöst werden, indem man dem Bild einen Alphakanal hinzufügt. Dabei wird für jeden Bildpunkt je ein Wert im Intervall [0, 1] gespeichert. Möchte man das Bild vor einen anderen Hintergrund setzen, so gibt der Wert an, in welchem Maß die Objektfarbe eine Pixelfarbe im neuen Bild dominiert. (1-α) bezeichnet den Anteil der künstlichen Hintergrundfarbe.

$$Pixelfarbe = \alpha * Objektfarbe + (1-\alpha) * Hintergrundfarbe \qquad \text{(Ausdruck 5.3)}$$

Im Inneren eines Objektes werden die Pixel mit α=1 undurchsichtig sein, an der Objektgrenze sind für $\alpha < 1$ Mischungen möglich.

Neben der Schätzung der Durchsichtigkeit eines Objektes muss in den Mischbereichen die Farbe des segmentierten Objektes natürlich von der des Hintergrundes bereinigt werden. Für alle $\alpha < 1$ kann die wahre Objektfarbe ohne sichere Kenntnis des Hintergrundes nur geschätzt werden. Auf Grundlage nur eines Photos ist diese Schätzung nicht exakt möglich, das Problem wird in der Literatur als *natural image matting* be-

zeichnet. Allerdings kann man sich - ebenso wie bei der Schätzung der Hintergrundfarbe in halbtransparenten Bereichen - relativ große Fehler leisten, weil der Mensch gegenüber kleinen Farbfehlern wenig empfindlich ist, insbesondere in den von Natur aus verschwommenen transparenten Grenzbereichen.

Lösungsansätze des Segmentierungsproblemes mit halbtransparenten Objektgrenzen wurden von Berman, Ruzon und Tomasi vorgeschlagen und als US Patent 6.134.345 bzw. 6.134.346 angemeldet [BER00, RUZ00]. Den diversen Varianten ist eigen, dass der Benutzer eine innere und eine äußere Segmentierungsgrenze für jedes Objekt definieren muss. Zwischen den beiden Grenzen liegt der halbtransparente Bereich (siehe Abbildung 5.11), in dem der unbekannte Alphakanal geschätzt werden muss.

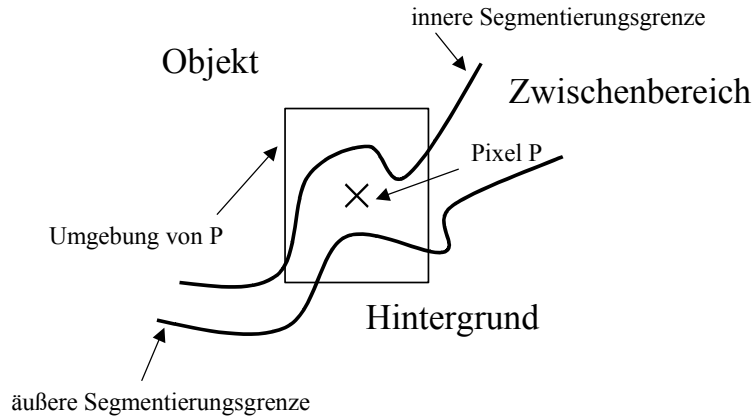

Abbildung 5.11: Pixel P liegt im halbtransparenten Bereich zwischen Objekt und Hintergrund. In der Umgebung von P (das Rechteck) können typische Objekt- und Hintergrundfarben, jenseits der beiden Segmentierungsgrenzen, ermittelt werden.

Wie der Benutzer zu den beiden Segmentierungen des Objektes kommt, ist nicht Gegenstand des natural-image-matting Problems. In Abbildung 5.11 ist das zu berechnende Pixel P von einer Umgebung umschlossen. In dieser kann jenseits der beiden Segmentierungsgrenzen eine typische Vorder- und Hintergrundfarbe ermittelt werden. Die begrenzte Umgebung von P wird verwendet, um ein Stück Vorder- und Hintergrund zu analysieren, das für P möglichst repräsentativ ist, oder mit anderen Worten: Wegen der lokalen Begrenztheit der Umgebung hofft man, eine Vorder- und Hintergrundfarbe erkannt zu haben, aus denen sich P tatsächlich zusammensetzt.

Nun muss eine typische Objektfarbe berechnet werden. Eine Einzelne reicht jedoch nicht aus, vor allem, weil das Objekt unterschiedlich starke Schattierungen aufweisen kann. Sinnvoller ist es, ein Volumen im RGB-Raum zu ermitteln, welches klein sein sollte, gleichzeitig jedoch alle möglichen Objektfarben enthalten muss. In der Art der in Kapitel 2 beschriebenen Hauptachsenanalyse kann man durch eine Wolke von Farbwerten z. B. drei Hauptachsen bestimmen, die ein Ellipsoid möglichst optimal in die Wolke einpassen. Chuang et al. schlagen vor, den Farbraum des Objektes durch eine angepasste Gausskurve (bzw. ein Volumen) anzunähern [CHU01]. Von Mishima [MIS93] wurde auch vorgeschlagen, ein Polyeder zu verwenden, jedoch in Zusammenhang mit dem bereits weiter oben beschriebenen Blue-Box-Verfahren. In jedem Fall muss der Raum aller möglichen Objektfarben irgendwie eingegrenzt werden. In Abbildung 5.12 ist der RGB-Raum dargestellt. Alle Objekt- und Hintergrundfarben bilden je eine Punktewolke, die durch je ein Ellipsoid angenähert wird. Schon jetzt wird klar, dass der Ansatz auf monochrome Bilder kaum übertragbar ist, denn wenn ein Objekt sowohl schwarze als auch weiße Pixel enthält, bleibt kein Raum mehr für mögliche Mischtöne. Enthält z. B. das Foto eines Zebras schwarze und weiße Pixel, so ersteckt sich das entsprechende Histogramm von den maximalen bis zu den minimalen Werten. Evtl. sind auch im Hintergrund ebenso schwarze wie weiße Pixel zu finden. Wenn sich aber beide Histogramme über den gesamten Wertebereich verteilen, so kann man beliebige Pixel kaum eindeutig dem einen oder anderen Histogramm zuordnen. Die Bestimmung des Alphakanales ist daher nicht mehr möglich.

Die Farbe eines Pixels auf der durchsichtigen Objektgrenze entspricht einem einzelnen Punkt im RGB-Raum. Dieser wird auf die Verbindungsgerade zwischen beiden Farbclustern projiziert. Der Anteil der Geraden von der Projektion zum Mittelpunkt der Objekt-Farbwolke gibt gerade den Wert für den Alphakanal an.

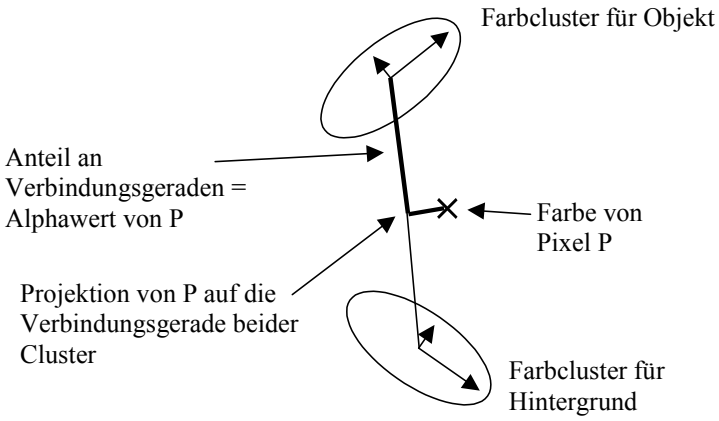

Abbildung 5.12: Objekt- und Hintergrundfarben bilden im RGB-Raum je eine Punktewolke, die durch ja ein Ellipsoid angenähert wird.

Je weiter sich eine Farbe in das Farbcluster des Objektes hineinbewegt, desto eher liegt auch eine solche Objektfarbe vor.

Wie bereits erwähnt, ist die Ermittlung des Alphakanals zum Problem der Segmentierung orthogonal. Man kann aber zum Beispiel mit Hilfe von SemiSeg eine innere und äußere Grenze finden und dazwischen für jedes Pixel den Alphakanal berechnen. Auch wäre denkbar, um die Objektgrenze herum ein mehr oder weniger breites Band zu definieren. Eine weitere Optimierungsaufgabe der Segmentierung wäre es dann, das Band immer so breit zu halten, dass ein vom Benutzer vorgegebenes, typisches Muster für den transparenten Übergang möglichst optimal wiedergefunden wird. Eine stärkere Verknüpfung der Segmentierung mit einem gleichzeitigen Auffinden eines Alphakanales bietet noch breiten Raum für zukünftige Forschung - unter Beachtung der Erschwernis durch die Patentierung der Grundidee.

KAPITEL 6

VERFEINERUNG DES SUCHALGORITHMUS

6.1 Stärken und Schwächen von SemiSeg

Grundsätzlich funktioniert die Verfolgung von Objektgrenzen durch SemiSeg schon recht gut. Bei der interaktiven Benutzung ist es aber immer wieder störend, dass das Verfahren des öfteren die Objektgrenze verliert und entweder in das Objekt oder den Hintergrund hineinläuft. Dies ist kaum zu verhindern, vor allem nicht, wenn sich die Gestalt der Grenze deutlich verändert. Dann finden sich im übrigen Bild Bereiche, die dem benutzerdefinierten Beispiel immer noch ähnlicher sind. Daher muss der Algorithmus vom Benutzer angehalten und der gefundene Pfad im Bereich der falschen Segmentierung gelöscht werden. Der instabile Aspekt des Verfahrens entsteht also nicht durch den eigentlichen Algorithmus der Mustererkennung, sondern eher durch die Art des "gierigen" Fortschreitens, welches seine Entscheidung der Vergangenheit, einen bestimmten Weg einzuschlagen, nie mehr revidiert.

6.2 Das Live-Wire-Verfahren

Eine interessante Lösung dieses Problems bietet der sogenannte Live-Wire Ansatz von Mortensen und Barret [MOR92, MOR95, MOR 99, BAR97]. Dabei ist der eigentliche Ansatz der Mustererkennung eher unspektakulär. In der Hoffnung, dass ein Objekt

durch eine starke Kante vom Hintergrund getrennt ist, wird diese in einer Umgebung gesucht und als Objektgrenze definiert. Der bestechende Aspekt des Verfahrens liegt in der hohen Interaktivität bei der Segmentierung.

Dabei definiert der Benutzer im Bild einen Ausgangspunkt, den sogenannten Seed-Point. Zwischen diesem und der aktuellen Cursorposition im Bild wird in Echtzeit die jeweils optimale Segmentierung eingeblendet. Was dabei optimal ist, muss später noch genauer diskutiert werden. Zum Zweck einer optimalen Segmentierung legt der Benutzer den Seed-Point natürlich auf die Silhouette selbst und fährt sie mit der Maus so weit wie möglich ab (siehe Abb. 6.1). Die tatsächlich gefundene Segmentierung ist zu jedem Zeitpunkt sichtbar. Dabei wird für die interaktive Segmentierung kaum Rechenzeit benötigt, jedoch auf Kosten einer sehr komplexen Vorberechnung.

Abbildung 6.1: Das weiße Kreuz markiert den Seed-Point S, das schwarze Kreuz die interaktive Cursorposition. Während der Benutzer den Cursor im Bild mit der Maus bewegt, richtet sich die weiße Segmentierungslinie (der Live-Wire) an der stärksten Kante aus.

In der interaktiven Phase des Kennzeichnens der Silhouette definiert der Benutzer sozusagen, was das Objekt ausmacht. Dabei hat er durch den Ausgangspunkt und die aktuelle Cursorposition bereits zwei Punkte festgelegt, die mit Sicherheit zur Objektgrenze gehören. Fehler können also nur dazwischen auftreten. Die in Kapitel 5 vorgestellte

Implementierung von SemiSeg kennt dagegen nur den Startpunkt und ist damit während des Fortschreitens größerer Unsicherheit ausgesetzt. Live-Wire weiß dagegen, dass der gefundene Pfad (auch Trajektorie) in jedem Fall auf die aktuelle Mausposition zulaufen muss. So lassen sich z. B. Lücken in der Kontur sicherer überbrücken.

6.2.1 Dynamische Programmierung zur stabilen Segmentierung

Die Interaktivität von Live-Wire, ist nur dadurch möglich, dass anfangs eine rechnerisch aufwändige, vollständige Baumstruktur aufgebaut wird, in der der Startpixel als Wurzel fungiert. Alle anderen Pixel des Bildes sind die Blätter. Ein beliebiger Weg von einer Stelle des Bildes zum Startpixel entspricht dem Weg im Baum vom entsprechenden Blatt zur Wurzel. Die Echtzeitfähigkeit erklärt sich aus der Tatsache, dass das Traversieren des Baumes nur O(log(n)) Zeiteinheiten braucht, wenn n die Anzahl der Pixel des Bildes ist.

Das Optimum, also die Wahl des besten Pfades im Sinne einer guten Segmentierung, wird durch eine Kostenfunktion definiert, die gute Segmentierungen mit geringen Kosten und schlechte mit hohen Kosten bewertet. Ob der durch die Kostenfunktion gefundene Pfad tatsächlich der optimalen Segmentierung entspricht, hängt wesentlich von deren Konstruktion ab. Die genaue Zusammensetzung dieser Funktion wird weiter unten noch erläutert. Zum Aufbau des Baumes verwenden die Autoren die dynamische Programmierung [PUT77]. Im Allgemeinen ist die Wahl einer guten Kostenfunktion, die genau das bewertet, was der Benutzer als die Grenze zwischen Objekt und Hintergrund ansieht, nicht trivial. Wurde sie aber einmal festgelegt, so garantiert die Verwendung der dynamischen Programmierung, dass der global optimale Pfad auch gefunden wird. Die dynamische Programmierung kann unter bestimmten Umständen dazu verwendet werden, die vollständige Enumeration aller Möglichkeiten zu vermeiden, um eine optimale Entscheidung (in diesem Zusammenhang manchmal als Politik bezeichnet) zu erreichen [SMI91, THU72, SNI92]. Dabei muss das Problem in diskrete Stufen unterteilbar sein. Jede Stufe besteht aus einer Anzahl von wählbaren Zuständen. Deren Kardinalität ist oft in allen Stufen gleich, dies ist aber keine notwendige Bedingung. Abbildung 6.2 zeigt eine Skizze des Vorgehens. Das Problem besteht darin, von jedem

Knoten einer Stufe zu einem Knoten der nächsten Stufe zu wechseln, während für den Übergang möglichst geringe Kosten anfallen sollen. Die Summe der Entscheidungen für je einen Knoten in jeder Stufe erzeugt einen einzelnen Pfad durch das Netzwerk, weshalb die dynamische Programmierung auch als ein Problem der Flussoptimierung mit nur einem möglichen Weg verstanden wird.

Manchmal ist der Übergang von jedem Knoten zu jedem anderen möglich. Im Falle der Segmentierung sind die Knoten Pixel im Bild. Da jedes Pixel nur acht Nachbarn hat, kann in jeder Stufe auch nur unter acht Alternativen gewählt werden. Der kostenoptimale gesamte Pfad vom ersten zum letzten Knoten (also vom Start- zum Endpunkt) ist das Ergebnis der dynamischen Programmierung und zugleich die optimale Segmentierung im Kontext dieser Arbeit. In jeder Stufe wird eine optimale Entscheidung für einen Knoten getroffen, die meist auch von den vorhergehenden Entscheidungen abhängig ist. Denn das Pixel, das in der vorhergehenden Stufe gewählt wurde, bestimmt auch, welche in der nächsten gewählt werden können.

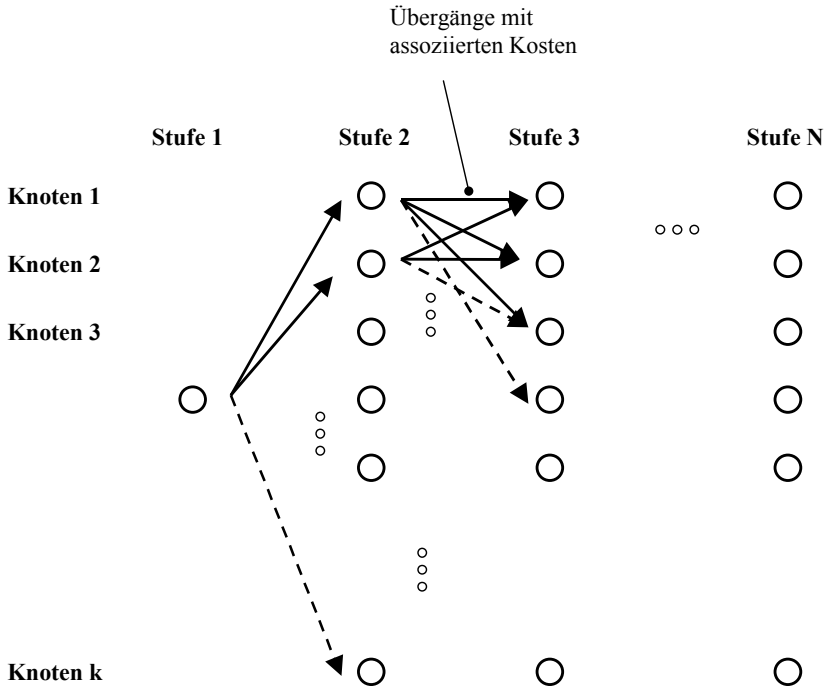

Abbildung 6.2: Dynamische Programmierung. In jeder der N Stufen kann für den Übergang zur nächsten eine Entscheidung getroffen werden. Gesucht sind die N-1 Entscheidungen, die zu einem Pfad minimaler Kosten führen. Dabei ist eine unterschiedliche Anzahl von Knoten pro Stufe möglich.

Als eine Eigenschaft der dynamischen Programmierung muss eine Folge von vergangenen Entscheidungen, also eine Politik, nicht zu jedem Zeitpunkt als optimal erkennbar sein. Vielmehr ist sogar typisch, dass oft zuerst ein scheinbar schlechter Weg beschritten wird, der sich erst später in minimalen Gesamtkosten auszahlt. Ansonsten wäre auch ein Greedy-Ansatz einfacher, der sich in jeder Stufe sofort für die lokal beste Alternative entscheidet.

Kosten entstehen genau genommen für die eigentliche Wahl eines Knotens und zusätzlich für den Übergang zwischen zwei Knoten. Am besten veranschaulicht man sich die dynamische Programmierung am Beispiel: Ausgangspunkt der Betrachtung sei der erste Knoten aus einer Stufe k mit k>1 (Stufe 1 wird generell nicht explizit betrachtet, da hier keine Wegewahl zur vorherigen Stufe möglich ist). Er könnte ausgehend von jedem Knoten der Stufe (k-1) erreicht worden sein. (In der Abbildung 6.2 wurde für die erste Stufe nur ein Knoten gezeichnet, da dies im Kontext der Segmentierung dem Live-Wire-Verfahren entspricht. Allgemein kann aber jede Stufe eine unterschiedliche Anzahl von Knoten enthalten.) Die Summe aus Knoten- und Übergangskosten wird am jeweiligen Knoten aus Stufe k vermerkt. Mit anderen Worten: Nun weiß man, von welchem Knoten aus Stufe (k-1) der erste aus Stufe k am billigsten erreicht werden kann. Für die spätere Verwendung wird ein rückwärts gerichteter Pfeil zum ersten Ausgangsknoten aus Stufe (k-1) gezogen. Diese Sorte Pfeile sind in Abbildung 6.3 fett gedruckt.

Der gleiche Vorgang wird für jeden weiteren Knoten aus Stufe k durchgeführt. Am Ende zeigt je ein Pfeil auf einen Knoten der Stufe (k-1). Es ist jedoch nicht zwingend notwendig, dass auch von jedem Knoten der Stufe (k-1) ein Pfeil ausgeht. Manche Knoten könnten sich als besonders teuer herausstellen und nicht gewählt werden.

Der gleiche Vorgang, der eben mit Stufe k und (k-1) vollzogen wurde, wiederholt sich nun mit Stufe (k+1) und k. Wieder wird für jeden Knoten aus Stufe (k+1) der optimale rückwärts gerichtete Übergang von Stufe k aus ermittelt usw. Am Ende sind nur noch die Gesamtkosten der letzten Stufe von Interesse. Die vorherigen muss man sich nur kurzfristig merken. Aus der letzten Spalte N wählt man nun den Knoten mit den ge-

ringsten Gesamtkosten. Da ja mit jedem Knoten auch ein rückwärts gerichteter Pfeil verbunden ist, ist auch die optimale Vorentscheidung bekannt. Man verfolgt also nur noch die fetten Pfeile, um in jeder Stufe die optimale Entscheidung - im Sinne geringster Gesamtkosten - zu treffen.

Nun wird auch die Aussage klar, warum die dynamische Programmierung besser als ein reiner Greedy-Ansatz ist. Denn beim Übergang von einer Stufe k zu (k+1) kann auch eine lokal schlechte Lösung gewählt werden, um in der nächsten Stufe durch besonders geringe Kosten den Nachteil wieder mehr als auszugleichen.

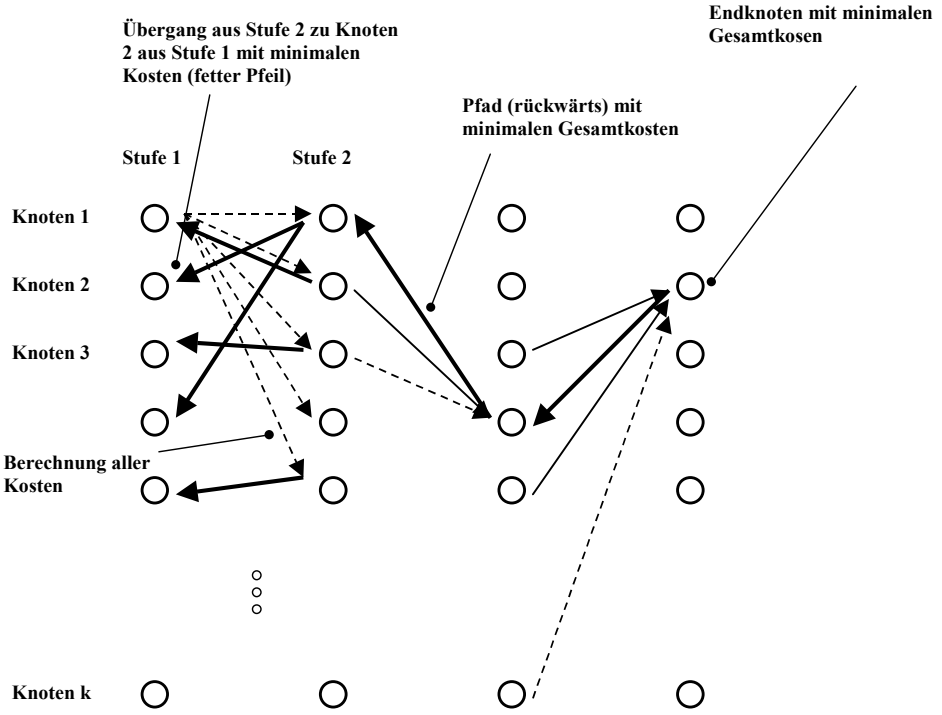

Abbildung 6.3: Dynamische Programmierung im Ablauf. Die fett gedruckten Pfeile markieren den jeweils optimalen Weg von einem Knoten k nach (k-1). In diesem Beispiel haben alle Stufen die gleiche Anzahl Knoten.

6.2.2 Vorgänger halbautomatischer Segmentierungsverfahren

Die Idee, die optimale Objektgrenze mit Hilfe einer Kostenfunktion zu berechnen, geht auf eine Arbeit von Fischler [FIS73] bzw. Chien und Fu aus den Jahren 1973 und 1974 zurück [CHI74]. Dabei gingen sie davon aus, dass aus einer Menge M aller Bildpunkte, eine Punktmenge K entnommen werden sollte, die die Objektgrenze beschreibt. Die

Menge K beschreibt eine mögliche 1-Pixel breite Trajektorie. Diejenige Menge K mit den geringsten Kosten soll die optimale Segmentierung sein. Schon damals war den Autoren klar, dass die Segmentierung nicht durch eine globale Betrachtung des Bildes stattfinden sollte, sondern in der Kostenfunktion lokal und in Bezug zu den bereits gefundenen Punkten berechnet werden muss:

"The decision function should also be dependent on the particular problem at hand, so that a priori knowledge of the types of edges to be extracted can be passed on to the decision rule using parameters in the decision function. The advantage of doing this is to provide flexibility in making changes or improvements for a given problem, as well as in switching from one problem to another." (Zitat aus [CHI74])

Auch war klar, dass eine sorgfältig gewählte Entscheidungs- bzw. Kostenfunktion (decision function) dazu beitragen kann, genau die Struktur zu finden, die im Rahmen einer bestimmten Anwendung gesucht wird. Wie auch bei Mortensen et al. besteht bei Chien die Kostenfunktion aus einer den Gradienten begünstigenden Komponente und einem Glattheitsterm.

Offensichtlich ist das Problem der Auswahl von K aus M Punkten ein bereits für kleine M nicht mehr berechenbares |M| über |K| Problem (|M| bezeichnet die Kardinalität, also die Anzahl der Elemente von M). Chien und Fu nehmen zum Zweck der Vereinfachung an, dass für jeden noch unbekannten, aber optimalen Segmentierungspixel eine Menge C von Kandidatenpunkten existiert, die in Frage kommen. Die Kandidatenpixel C kann man sich wie einen Schlauch vorstellen, der um die Objektgrenze herum gelegt wird. Die Suche nach der optimalen Segmentierung schränkt sich daher auf den Schlauch C ein. Für die Grenzpixel K kann so ein Baum mit |C| über |K| möglichen Kombinationen aufgebaut werden, wobei $|K| << |C|$ gilt. An dieser Stelle ähnelt der Algorithmus den erst später entwickelten aktiven Konturen [BLA98], allerdings mit dem Unterschied, dass es sich bei diesen frühen Ansätzen um keine Fixpunktiteration, sondern um eine vollständige Suche handelt. Auch die aktiven Konturen erfordern eine Initialisierung um

das Objekt herum. Doch auch die |C| über |K| Möglichkeiten sind in der Praxis nicht enumerierbar. Die Autoren dieser frühen Idee schlagen daher vor, zuerst innerhalb der Kandidatenpixel C (also dem Schlauch um das Objekt) eine beliebige Segmentierung zu finden. Bei der Aufzählung weiterer Segmentierungen können alle Lösungsmöglichkeiten verworfen werden, die schon frühzeitig höhere Kosten verursachen, bevor überhaupt eine geschlossene Trajektorie gefunden wurde. Neue, bessere Segmentierungen erhöhen natürlich die Hürde für alle weiteren Möglichkeiten, die immer früher abgebrochen werden können.

Bei der beschriebenen Suche handelt es sich nur insofern um einen halbautomatischen Algorithmus, als dieser von Hand initialisiert werden muss. Die weitere Suche funktioniert automatisch. Im Gegensatz dazu sind SemiSeg und Live-Wire Verfahren, die gerade während der Segmentierung Benutzerinteraktion erwarten.

6.2.3 Der Live-Wire-Algorithmus am Beispiel

Grundsätzlich ist Live-Wire nun nur noch eine spezielle Anwendung der dynamischen Programmierung zum Zweck der Segmentierung. Da man sich dies intuitiv schlecht vorstellen kann, soll das Verfahren hier noch einmal am Beispiel eines typischen Ablaufs beschrieben werden:

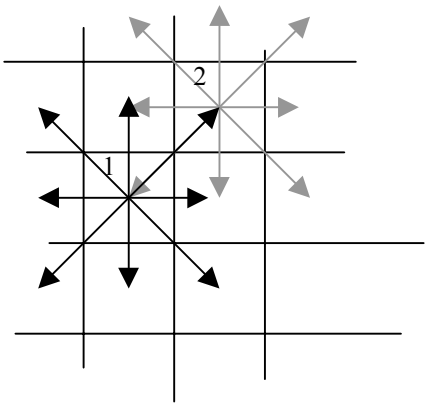

Abbildung 6.4: Nachbarschaftsbeziehungen von Pixeln. Pixel 1 ist der Seed-Point. Unter allen Kandidaten erzeugt der Übergang zu Pixel 2 die geringsten Kosten.

Zu Anfang besteht der Baum nur aus dem vom Benutzer gesetzten Seed-Point. In seiner Umgebung bieten sich acht Kandidaten zur Fortsetzung der Segmentierung an (siehe Abbildung 6.4). Jeder Übergang zu einem der Kandidaten verursacht Kosten, die

weiter unten noch vorgestellt werden. Vorgreifend kann aber schon gesagt werden, dass das Bewegen entlang einer Kante niedrige Kosten verursacht, wogegen das Kreuzen einer Kante teuer sein soll. Alle der acht Möglichkeiten werden in eine Prioritätswarteschlange eingeordnet, die niedrige Kosten bevorzugt. Für jeden eingeordneten Pixel wird auch dessen Elternteil vermerkt, in diesem Fall ist dies der Seed-Point selbst. Danach wird auch gleich das erste Kind mit den geringsten Kosten wieder entnommen und dessen acht Nachbarn (sortiert) in die Warteschlange eingeordnet. Durch das bevorzugte Entnehmen von Pixeln auf Kanten wächst der Algorithmus besonders schnell entlang von Kanten und Texturen. Abbildung 6.5 zeigt die mit dieser Arbeit entstandene Implementierung im Verlauf der Berechnung. Das weiße Kreuz markiert den Seed-Point. Der Bereich im Bild, der bereits vom Algorithmus erfasst wurde, ist heller gefärbt, wobei die aktuelle Grenze fast weiß erscheint. Man erkennt, dass die weißen Bereiche schneller in kanten- und texturreiche Regionen vordringen. Der weitgehend gleichförmig schwarze Grill ist bis zum letzten Bild rechts unten immer noch nicht vollständig eingenommen. Denn ohne Kanten werden diesen Bereichen hohe Kosten zugeordnet. Dementsprechend lange bleiben solche Pixel in der Warteschlange.

Abbildung 6.5: Live-Wire-Algorithmus während der Berechnung des Baumes. Der Seed-Point ist mit einem weißen Kreuz markiert. Die bereits bearbeitete Pixelmenge ist heller dargestellt. Im Ablauf (links oben nach rechts unten) wird deutlich, dass texturreiche Bereiche früher eingenommen werden als glatte.

Der eigentliche Baum entsteht dadurch, dass jedes Pixel eine rückwärts gerichtete Referenz auf seinen Elterpixel erhält (vergleiche mit den rückwärts gerichteten Kanten aus Abbildung 6.3). Dadurch ist nach der Entnahme aus der Warteschlange auch gleich klar, wohin der neue Pixel im Baum gehört. Dies erst ermöglicht das spätere Rückverfolgen des optimalen Pfades. Der bisher beschriebene Algorithmus ermöglicht durchaus, dass ein Pixel zweimal in der Warteschlange eingeordnet wird. So werde z. B. ein Nachbar N des Seed-Points gleich zu Anfang mit hohen Kosten in die Warteschlange eingeordnet. Um den Seed-Point könnte sich eine kleine Schleife aus optimalen Pixeln ergeben, so dass N ein zweites Mal besucht wird, nun jedoch mit geringeren Kosten. Entsprechend darf N nochmals mit einem neuen (besseren) Elternteil in die Warteschlange eingefügt werden, der alte Eintrag muss jedoch gelöscht werden. Dieser spezielle Fall rechtfertigt die Aussage, dass die dynamische Programmierung auch bereit ist, eine frühere Entscheidung zu verwerfen. Abbildung 6.6 zeigt eine Veranschaulichung der aufgebauten Datenstruktur.

Der Seed-Point in der Mitte des Bildes ist mit einem Sternchen markiert. Zu diesem gelangt man von jedem beliebig Punkt des Bildes durch Verfolgen der Pfeile. Der dabei entstehende Weg wird immer möglichst dicht eine Kante in der lokalen Umgebung verfolgen, auch dann, wenn dies einen längeren Weg erfordert. Ein Punkt innerhalb einer Fläche gleicher Grauwerte wird auch immer versuchen, auf dem kürzesten Weg die nächste Kante zu erreichen.

6.2.4 Zusammensetzung der Kostenfunktion

In der ersten Version von Live-Wire aus [BAR97] wird die Kostenfunktion aus drei Komponenten zusammen gesetzt. Die Wahl der Gewichte vor den Kostenkomponenten wurde empirisch ermittelt und wird mit $\omega_Z = 0.43$, $\omega_G = 0.43$ und $\omega_D = 0.14$ angegeben. In einer späteren Erweiterung wurden noch weitere Kostenkomponenten hinzugenommen [MOR92], [MOR95], [MOR99].

$$l(p,q) = \omega_G f_G(q) + \omega_Z f_Z(q) + \omega_D f_D(p,q) \qquad \text{(Ausdruck 6.1)}$$

Der Ausdruck gibt die Kosten zwischen Pixel p und dem unmittelbar benachbarten Pixel q an.

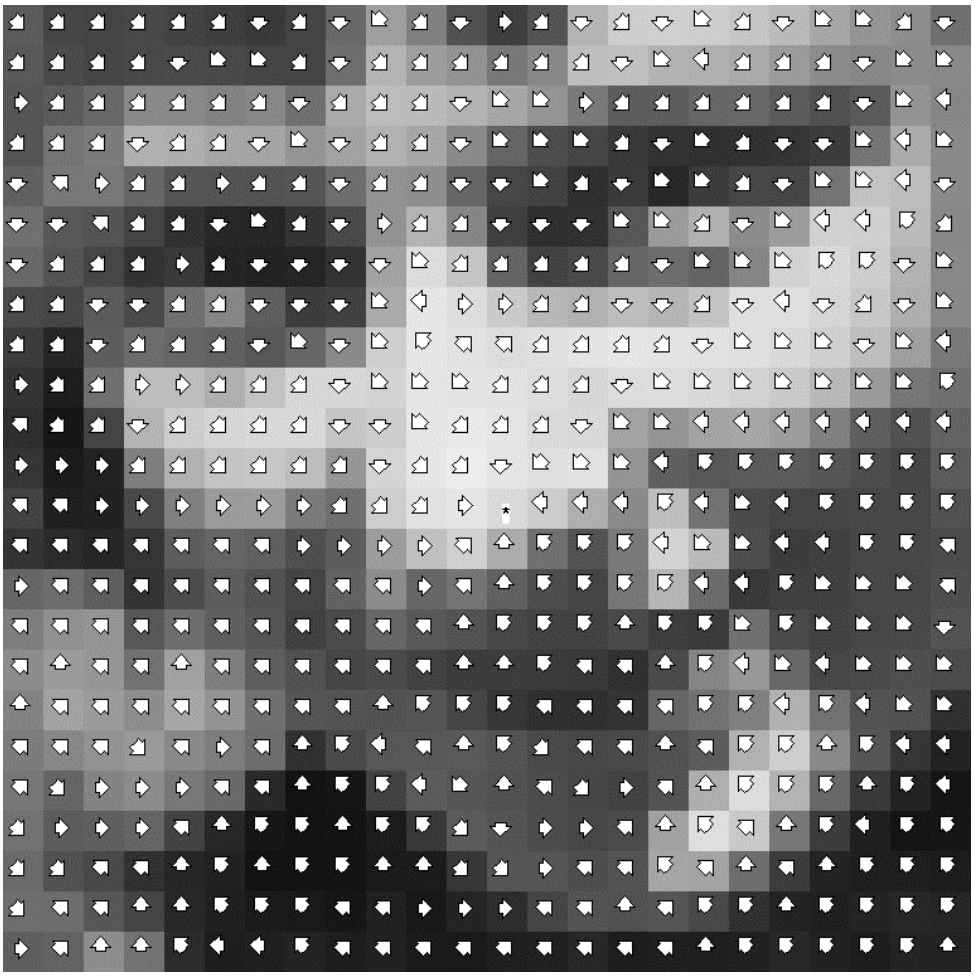

Abbildung 6.6: Das Bild zeigt ein Gesicht in starker Vergrößerung. Der Seed-Point in der Mitte ist mit einem Sternchen markiert. Diesen erreicht man durch Verfolgen der Pfeile von jedem anderen Punkt des Bildes. Dabei versucht jeder mögliche Weg, sich dicht entlang lokaler Kanten zu bewegen.

Mit der ersten Komponente f_G wird die Objektgrenze nach Mortensen und Barret grob positioniert. Sie wird auf Grundlage des Gradienten berechnet. Bewegt sich die Kurve auf einer starken Kante, so sollen die Kosten möglichst gering sein. Daher wird in Ausdruck 6.2 der Gradient G zur Normierung durch sein Maximum geteilt und von 1 abgezogen. Starke Kanten resultieren dadurch in geringen Kosten nahe null.

$$f_G = 1 - \frac{G}{\max(G)} \qquad \text{(Ausdruck 6.2)}$$

Damit sich die Objektgrenze nicht an zu feinen Texturen, sondern an den möglichst dominanten Kanten festsetzt, wird das Bild vor der Berechnung des Gradienten mit Gaussfiltern unterschiedlicher Breite geglättet. Die Filterbreite kann man als eine stärkere oder schwächere Konzentration auf feine Details ansehen. In einem zunehmend grob aufgelösten (bzw. stark gefilterten) Bild treten nur noch die dominantesten Kanten hervor. Den Autoren ist intuitiv klar, dass die Suche nach der optimalen Kante auf einer bestimmten Skala statt finden muss. Trotzdem wird die Filterbreite empirisch ermittelt und für alle Bilder festgelegt. Die in Kapitel 5 beschriebene Verbesserung der Kantensuche durch den waveletbasierten Ansatz von SemiSeg vermeidet diese Festlegung zugunsten einer variablen Analyse auf unterschiedlichen Skalen, die sich auch während der Segmentierung der sich verändernden Kante anpassen kann.

Die zweite Komponente f_Z dient der Feinpositionierung der Trajektorie. In der Tat kann die durch f_G gefundene Kante abhängig vom Objekt relativ breit "verschmiert" sein, so dass eine Feinjustierung nützlich ist.

$$f_Z(q) = \begin{cases} 0; \text{if } I_L(q) = 0 \\ 1; \text{if } I_L(q) \neq 0 \end{cases} \qquad \text{(Ausdruck 6.3)}$$

Gesucht werden mit f_Z alle Nulldurchgänge der zweiten Ableitung I_L. Die Bildfunktion nimmt an einer Kante, ausgehend von einem lokalen Minimum, ein lokales Maximum an oder umgekehrt. Zwischen diesen Extrema gibt es einen Wendepunkt, in dem die Steigung maximal ist. An dieser Stelle hat *die erste Ableitung* (nicht die Bildfunktion selbst) ihr Maximum. Dieser Punkt kann zwischen den beiden lokalen Extrema gewissermaßen als die optimale Mitte der Kante angesehen werden und ist daher von besonderem Interesse. In der Literatur wird der Operator oft als *laplacian zero-crossing feature* bezeichnet. Um das Maximum der ersten Ableitung zu finden, kann man die zweite auf null setzen. Es ist allerdings zu beachten, dass die zweite Ableitung nicht nur in den Wendepunkten null ist, sondern auch überall da, wo die Bildfunktion konstant bleibt. Da aber nur die Mitten von Kanten von Interesse sind, wird zusätzlich geprüft, ob die zweite Ableitung rechts und links der Null unterschiedliche Vorzeichen hat. Erst

dann ist sicher, dass die Nulllinie durchbrochen wurde und wirklich ein Wendepunkt vorliegt (daher der Name *zero-crossing*). In der Praxis kommt eine echte Null genau an den Samplestellen des Bildes eher selten vor, so dass ein kleines Intervall als null aufgefaßt wird. Wurde ein Wert innerhalb eines solchen Intervalls gefunden, für den auch noch die Eigenschaft des Schneidens der Abszisse zutrifft, so hat die Segmentierung genau die Mitte der Kante gefunden. Dies wird mit Kosten von 0 bewertet, alle anderen Punkte erhalten die Maximalkosten von 1. f_Z ist also ein binärer Operator. Die Bildfunktion ist somit per Grundeinstellung auf 1 gesetzt. Lediglich im Bereich der Kanten ziehen sich schmale "Gräben" um das Objekt, in denen die Funktion minimal ist. Eine in diesem Zusammenhang nicht notwendige, aber nützliche Eigenschaft des Laplace-Operators ist, dass er immer geschlossene Konturen beschreibt und somit an keiner Stelle einfach abreißt.

Die dritte Komponente f_D fügt eine Glattheitsbedingung hinzu, die schnellen Oszillationen der Kontur hohe Kosten zuweist.

$$f_D(p,q) = \frac{2}{3\pi}\left\{\operatorname{acos}\left[d_p(p,q)\right] + \operatorname{acos}\left[d_q(p,q)\right]\right\}$$

$$d_p(p,q) = D(p)L(p,q)$$
$$d_q(p,q) = L(p,q)D(q) \qquad \text{(Ausdruck 6.4)}$$

$$L(p,q) = \begin{cases} q-p; \text{if } D(p)(q-p) \geq 0 \\ p-q; \text{if } D(p)(q-p) < 0 \end{cases}$$

D(p) beschreibt einen Einheitsvektor, der an der Stelle p senkrecht auf dem Gradienten steht. D() zeigt also in die Richtung, in der sich das Bild am wenigsten ändert. L(p,q) kann man sich anschaulich als eine Linie zwischen den Pixeln p und q vorstellen, die in einer bestimmten Weise normiert wurde (es geht bei der Normierung nur um die Richtung). d_p beschreibt, als das Produkt dieser beiden Vektoren, wie gut die Gerade zwischen p und q der lokalen Struktur des Bildes folgt. Es gibt keinen Grund, die Berechnung alleine in Abhängigkeit von p durchzuführen. Da auf der anderen Seite der Gerade auch noch der Punkt q beteiligt ist, wird die gleiche Berechnung an dieser Stelle noch-

mals durchgeführt. Zusammenfassend kann man sagen, dass die Glattheitsbedingung nicht bewertet, ob ein Punkt auf einer Kante liegt, sondern nur, wie glatt die Richtung in Bezug auf den nächsten Punkt der Silhouette ist. Der Arcus Cosinus rechnet diesen Sachverhalt nur noch in Winkel um, so dass gut angepasste Punkte (p, q) zu kleinen Werten führen.

Die Glattheitsbedingung trägt nicht a priori zur Verbesserung der Segmentierung bei. Vielmehr muss das Objekt die unterstellte Vermutung auch erfüllen. Dann können allerdings auch Löcher in einer Objektkontur gut überbrückt werden. In der später beschriebenen Evaluation zeigte sich, dass die Einführung solch spezieller Bedingungen, im Mittel über alle Beispiele keine signifikante Verbesserung brachte.

Grundsätzlich würde Live-Wire auch mit nur einer der drei Kostenkomponenten funktionieren, sogar, wenn ausschließlich die binäre Komponente f_Z verwendet würde.

Alleine mit einem binären Operator wären andere Verfahren, darunter auch SemiSeg, schnell überfordert. Der Grund für diese Stabilität und die Robustheit des Live-Wire-Verfahrens liegt also weniger in der geschickten Wahl der Kostenfunktion, als vielmehr in der Verwendung der dynamischen Programmierung. Denn die dynamische Programmierung findet immer einen optimalen Weg von einer Senke zum Ziel. Auf diesem Weg können alle möglichen Kosten aufsummiert und minimiert werden - auch binäre. Ein Verfahren, welches dagegen lokal nach einer optimalen Richtung sucht, könnte auf Grundlage einer Kostenfunktion, die nur zwei Werte liefert, keine Richtung bestimmen.

6.3 Richtungssuche von Live-Wire und Mustererkennung von SemiSeg

Für eine möglichst benutzerfreundliche Segmentierung liegt es nahe, die stabile Suche von Live-Wire mit der robusten Mustererkennung von SemiSeg zu verbinden [HAE03]. Dies ist im Rahmen dieser Dissertation entstandenen Anwendung *SemiSegDP* geschehen. Hinter dem Aufruf des Programmes kann man in der Kommandozeile eine PPM-Bilddatei angeben, in der segmentiert werden soll. Im Programm kann man mit der W-Taste zwischen wavelet- und kantenbasiertem Modus umschalten.

In der Anwendung ging es im Wesentlichen darum, die einfache Kantensuche durch die skalenbasierte Analyse von SemiSeg zu ersetzen und das kombinierte Verfahren auf seine Brauchbarkeit zu testen (siehe auch Kapitel 8). Bei SemiSegDP definiert der Benutzer wie gewohnt einen Seed-Point als Wurzel des Graphen aller Pixel. Gleichzeitig ist der Punkt auch der Anfang des benutzerdefinierten Rechtecks. Ein weiterer Punkt ist nun nötig, um auch das Ende des Rechtecks zu bestimmen. Auf diesem Rechteck wird die gleiche Waveletanalyse wie in Kapitel 5 durchgeführt. Als Ergebnis erhält man wieder für alle Koeffizienten einer Pixelspalte einen Mittelwert und die Varianz.

Die von Live-Wire bekannte Suche in der Umgebung des Seed-Points orientiert sich nun nicht mehr am Übergang zur besten Kante, sondern daran, ob das benutzerdefinierte Muster (mehr oder weniger) wiedergefunden werden kann (Abbildung 6.7, rechts). Dabei wird für jede der acht Richtungen eine zum Pfeil senkrechte Pixelreihe in der gleichen Weise wie das benutzerdefinierte Muster analysiert und die dabei entstandenen Koeffizienten $k_{i,s}$ mit den zuvor berechneten jeweiligen Mittelwerten $\bar{k}_{i,s}$ verglichen. An der Metrik von SemiSeg (siehe Ausdruck 6.5) ändert sich daher nichts.

$$m = \sum_{s=1}^{S} \sum_{i=1}^{I_s} \left| \frac{k_{i,s} - \bar{k}_{i,s}}{1 + var_{i,s}} \right| \qquad \text{(Ausdruck 6.5)}$$

S : Anzahl der Skalen
I_s : Anzahl der Koeffizienten einer Skala
$\bar{k}_{i,s}$: Durchschnittlicher Koeffizient i auf Skala s
$k_{i,s}$: Koeffizient i der Pixelreihe einer Suchrichtung auf Skala s

Für den Benutzer besteht der einzige Unterschied zur Verwendung des reinen Live-Wire-Algorithmus darin, dass am Anfang zwei Klicks auf die Objektsilhouette nötig sind. Beim interaktiven Setzen weiterer Punkte im Laufe der Segmentierung kommt man grundsätzlich auch mit der alleinigen Definition des Seed-Points aus, denn es ist ja bereits ein Stück Objektgrenze gefunden, aus dem der zweite Punkt entnommen werden kann.

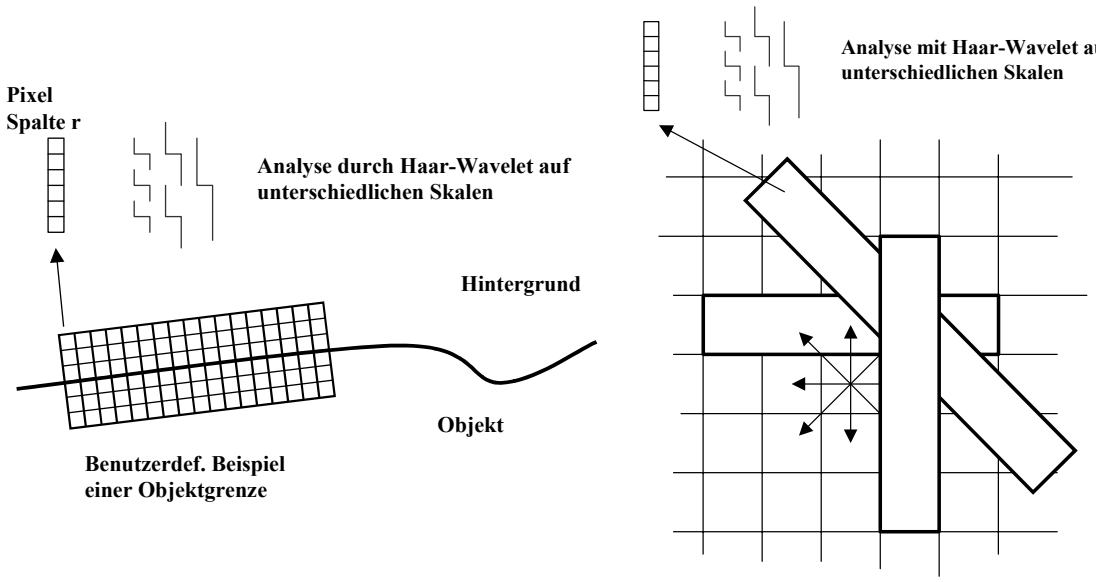

Abbildung 6.7: Skalenbasierte Mustererkennung von SemiSeg (links) in Verbindung mit der Graphensuche von Live-Wire. Bei der Entscheidung für eine der acht Richtungen (rechts) wird nicht mehr nach der stärksten Kante, sondern der besten Übereinstimmung mit dem Muster (links) gesucht.

Die eigentliche Berechnung gestaltet sich aufgrund der acht Wavelet-Analysen (in jede Richtung) für jeden Pixel etwas aufwändiger, ist aber auf aktuellen PCs kein Hindernis bei der interaktiven Segmentierung. Bei der später noch beschriebenen Evaluation entstanden für den Aufbau des Baumes Wartezeiten von 2-5 Sekunden für Bilder bis 2 Mio. Pixel.

In der Implementierung wurde auch ein Verfahren zur weiteren Beschleunigung getestet. Die Idee besteht darin, dass Übergänge zwischen Objekt und Hintergrund selten dort stattfinden, wo das Bild gleichmäßige Farbflächen besitzt. Daher wird vor der Generierung des Baumes eine Übersegmentierung durch das Watershed-Verfahren durchgeführt [OLS97]. Lediglich alle Pixel *zwischen* den homogenen Farbflächen müssen in den Baum aufgenommen werden, so dass sich die Anzahl der betrachteten Pixel je nach Bild auf etwa 30%-50% reduziert. Allerdings ist zu beachten, dass dafür zuvor die Watershed-Segmentierung durchgeführt werden muss, die auch mit sortierten Warteschlangen arbeitet. In der Implementierung wurde für die Sortierung ein schnelles Bucket-Sort-Verfahren genutzt. Trotzdem verbraucht Watershed so viel Zeit, dass der Vorteil des einfacheren Aufbaus des Live-Wire-Baumes zu einem Teil zuvor verbraucht wurde. Da der Aufbau des Baumes ohnehin nur wenige Sekunden benötigt, ist der Vorteil durch die komplexere Implementierung nur bedingt gerechtfertigt.

6.4 Vergleich von Live-Wire und SemiSeg

Die globale Optimierung macht Live-Wire, wie bereits erwähnt, in einigen Fällen sehr robust, so z. B., wenn sich ein Objekt stellenweise vom Hintergrund wenig unterscheidet. Dann wird auch ein bezüglich der Kosten ungünstiges Überbrücken wenig charakteristischer Stellen einer Silhouette in Kauf genommen, um danach wieder von geringen Kosten zu profitieren.
In anderen Fällen ist aber das eher lokale Vorgehen von SemiSeg aus Kapitel 5 besser. Abbildung 6.8 zeigt ein Beispiel. Dort wurde der Versuch gemacht, das Bein der Katze zu segmentieren. Hier nimmt Live-Wire einen ungünstigen Weg in Kauf, ungünstig sogar in Bezug auf die kantenorientierte Kostenfunktion. Trotzdem ist der kurze, schlechte Pfad aufgrund der wenigen Pixel immer noch kostengünstiger als der zwar kantenreichere, jedoch wesentlich längere Pfad.

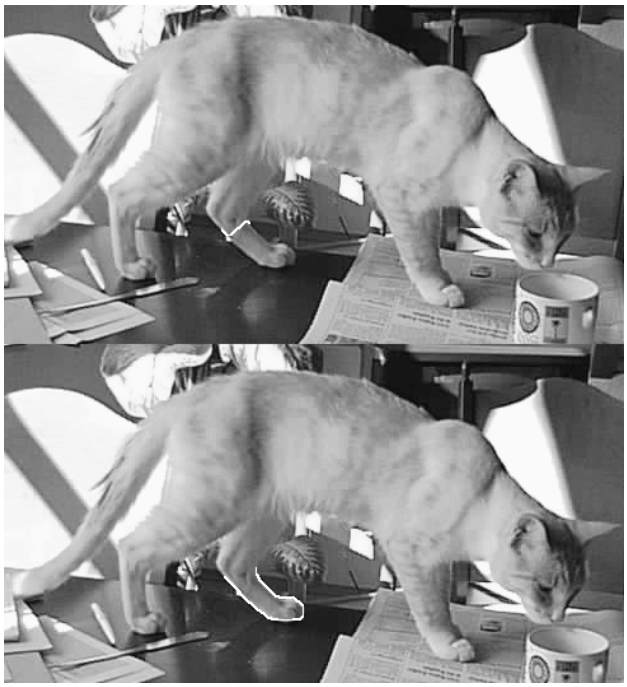

Abbildung 6.8: In einigen Fällen nimmt Live-Wire aufgrund der globalen Optimierung ungewollte Abkürzungen (oben), wogegen SemiSeg auch einen längeren Weg in Kauf nimmt (unten).

Hier zeigt sich das originäre SemiSeg robuster, denn die Länge des Pfades spielt bei der lokalen Richtungssuche, die alle 3-4 Pixel erneut durchgeführt wird, keine Rolle. Ab-

schließend kann man also auch der global optimalen Suche von Live-Wire nicht grundsätzlich die beste Segmentierung bescheinigen. Natürlich ist es möglich den Kostenterm von Live-Wire durch die Länge des gefundenen Pfades zu teilen, um auf Kosten pro Pixel zu normieren. Dieser Versuch führe im Rahmen dieser Arbeit aber zu keinen brauchbaren Ergebnissen, da teilweise extrem lange Pfade zustande kamen, die für eine Segmentierung vollkommen ungeeignet waren.

In Kapitel 8 folgt eine umfangreichere Evaluation unterschiedlicher Verfahren, in der auch SemiSegDP gegen die anderen Verfahren antreten muss.

KAPITEL 7

SEGMENTIERUNG VON VOLUMENDATEN

7.1 Segmentierung im Dreidimensionalen

In den vorangegangenen Kapiteln wurden Objekte aus zweidimensionalen Abbildungen segmentiert. Ebenso gut ist auch Segmentierung von Volumendaten denkbar. Dabei spricht man in der Literatur bei den gesampelten Daten nicht mehr von Pixeln, sondern von sogenannten Voxeln (Pixel = PIcture **EL**ement / Voxel = **VO**lume **EL**ement). Ebenso wie ein Pixel einen Grauwert (an einer implizit definierten Stelle) bezeichnet, gibt ein Voxel eine Materialdichte an, die irgendwo im Raum angenommen wird. Solche dreidimensionalen Datensätze entstehen vor allem in der Medizin durch CT (Computertomographie) oder MRT (Magnet-Resonanz-Tomographie). Die MRT arbeitet mit einem starken Magnetfeld und Radiowellen, wobei die Ausrichtung von Wasserstoffatomen gemessen wird, um die Dichte zu ermitteln. Daher eignet sich diese Methode zur Analyse von wasserreichem Körpergewebe. Knochen sind wegen ihres geringen Wassergehaltes kaum sichtbar. Für sie und bestimmte Anwendungen in der berührungsfreien Materialprüfung ist die mit Röntgenstrahlen arbeitende CT besser geeignet, da Materialien mit einer hohen Dichte die Strahlen stark absorbieren und weiches Gewebe kaum.

In der Operationsvorbereitung kann es sinnvoll sein, bestimmte Regionen des Körpers isoliert zu betrachten, um z. B. schon vor einem Eingriff die genaue Lage eines Tumors im umgebenden gesunden Gewebe zu kennen. Es soll also ein räumliches Objekt virtuell von seiner Umgebung getrennt werden, indem seine Oberfläche gefunden wird.

In dem im Rahmen dieser Dissertation entwickelten und in [HAE03] publizierten Ansatz benötigt der Benutzer mehrere Einzelflächen, um ein räumliches Objekt anzunähern, ebenso wie eine einzelne Trajektorie selten zur Segmentierung eines ganzen Objektes in der Ebene ausreicht. Die Oberfläche eines Objektes wird also durch eine Anzahl von "Flicken" (in der Literatur auch als Patch bezeichnet) zusammengesetzt.

7.2 Vorarbeiten und Stand der Technik

3D-Segmentierung funktioniert nach dem heutigen Stand der Technik meist durch schwellwertbasierte Verfahren, ähnlich der regionenbasierten Methode 2 in Kapitel 5 [BOM90, HOE90, RAY90]. Dabei werden Regionen gleicher Dichte zusammengefasst und durch boolesche Operationen mit anderen Regionen verknüpft und als Objekt definiert. Eine ähnliche Idee verfolgt die Watershed-Segmentierung in Volumendaten, die von G. Kühne implementiert wurde [KUE97]. Für Regionen mit einheitlicher Dichte funktionieren diese Verfahren relativ gut. Probleme machen dagegen Körperteile aus unterschiedlichem Gewebe. So besteht die Nase z. B. aus Haut, Knochen, Muskeln und Hohlräumen und kann daher auf Grundlage einer Regionensuche weniger gut gefunden werden. Das hier vorgeschlagene Verfahren zielt nun darauf ab, eine bestimmte Oberfläche anhand eines kostenbasierten Kriteriums zu finden, ohne auf homogene Dichtewerte angewiesen zu sein. Die Segmentierung einer Nase kann z. B. aus zwei Flächen bestehen, je eine für jeden Nasenflügel. Das Verfahren der 3D-Segmentierung ist an die Graphensuche von Live-Wire angelehnt. Live-Wire kann aber aus den im Folgenden erläuterten Gründen nicht ohne weiteres in die dritte Dimension übertragen werden.

Im Unterschied zur 2D-Segmentierung kann man nun nicht mehr von einem Seed-Punkt sprechen. Das Gebilde, das ein Volumen von seiner Umgebung abgrenzt, ist kein Pfad, sondern eine Fläche. Ebenso wie der Pfad einen Startpunkt (bzw. Seed-Punkt) und ei-

nen Endpunkt hat, ist die Fläche im Raum analog durch eine Start- und End*linie* begrenzt (also jeweils eine Dimension mehr). In Abbildung 7.1 ist ein Beispiel zu sehen, an dem auch gleich die Problematik einer nativen Übertragung von Live-Wire in den Raum deutlich wird.

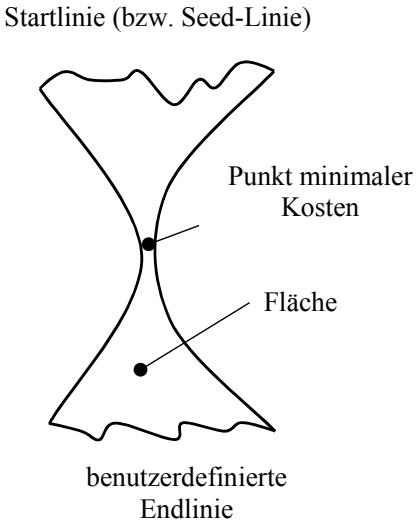

Abbildung 7.1: Eine benutzerdefinierte Startlinie setzt sich in Richtung der Seed-Linie fort. Dabei neigt sie dazu, in der Mitte ihre Kosten zu minimieren.

Der Live-Wire ist in der Fläche bestrebt, die Kosten entlang seines Pfades zu minimieren. In der Praxis bedeutet dies meist, dass der Pfad Umwege eher vermeidet, es sei denn, es bieten sich in der Umgebung besonders starke Kanten an. Trotzdem ist der Pfad per Definition gezwungen den Start- und Endpunkt zu durchlaufen.

Auch der Live-Patch (in Analogie zum Live-Wire) müsste per Definition von der Start- zur Endlinie laufen. Dazwischen hätte er aber mehr Freiheitsgrade. Die Minimierung der Kosten würde daher versuchen, die Fläche möglichst klein zu halten, und sich so wie in Abbildung 7.1 gezeigt in der Mitte zusammen ziehen.

Ein weiteres Problem einer naiven Umsetzung von Live-Wire in die dritte Dimension ist die interaktive Benutzersteuerung. Im 2D-Fall bewegt der Benutzer den Endpunkt der Trajektorie interaktiv mit der Maus. Im Dreidimensionalen ist eine interaktive Definition der Endlinie nicht trivial und mit der Maus schwerer zu leisten. Das eigentliche Problem liegt aber darin, in Analogie zum Baum der kürzesten Wege den Baum kürzes-

ter Flächen für ein gegebenes Volumen zu berechnen. Denn jeder Knoten des Baumes entspricht einem Pfad durch das Volumen.

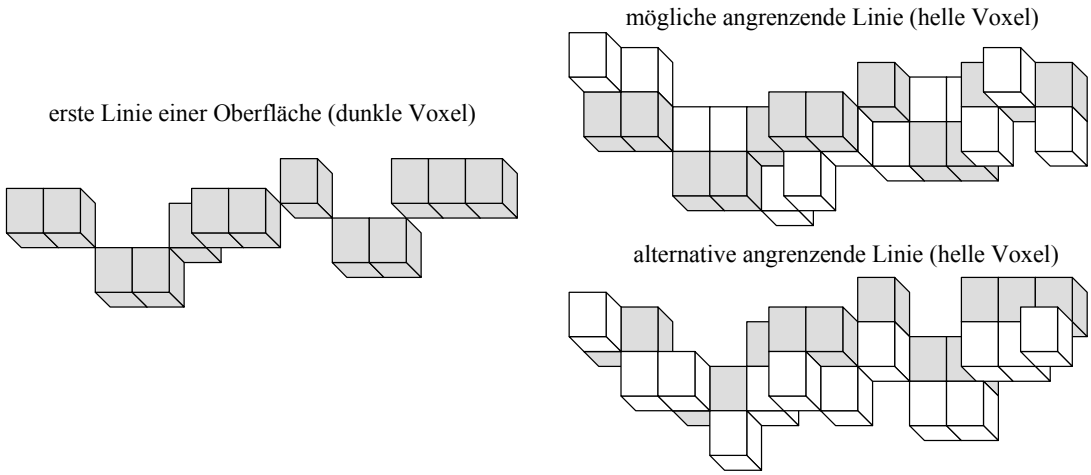

Abbildung 7.2: Initialer Pfad einer Oberfläche durch dunkle Voxel gekennzeichnet (links). Exemplarisch wurden zwei mögliche angrenzende Pfade durch helle Voxel dargestellt (rechts).

Traversiert man den Baum von einem Knoten zum nächsten, so "baut" man, anschaulich gesprochen, einen neuen Pfad, also eine Kette von Voxeln an eine bereits bestehende an. Man kann sich den Vorgang mit einer Hängebrücke über einen Graben veranschaulichen. An beiden Seiten sei ein Stamm befestigt, der eine, analog zu diesem Anwendungsszenario, die Seed-Linie, der andere die Eingabe-Linie des Benutzers. An jedem Stamm (bzw. Pfad durch das Volumen) wird ein weiterer Pfad angehängt, bis eine Brücke zwischen beiden Seiten entstanden ist. Im Gegensatz zu einem Bild, in dem ein Knoten des Baumes einem Pixel entspricht, der seinerseits nicht mehr als acht Nachbarn haben kann, ist die Anzahl der Nachbarn einer Kette von Voxeln zwar abzählbar doch sehr groß. Abbildung 7.2 zeigt ein Beispiel zur Veranschaulichung. An die graue Kette von Voxeln (links) wurden zwei Varianten einer weißen Kette (rechts) angehängt. Selbst mit der Nebenbedingung, dass ein Voxel jeweils mit einem weißen seiner eigenen Kette und einem grauen der vorhergehenden Kette zusammenhängen muss, um eine geschlossene Oberfläche zu definieren, explodiert die Anzahl möglicher Kombinationen exponentiell. Im Gegensatz zu einem Bild, in dem *nur jedes Pixel* ein möglicher Knoten des Baumes sein konnte, können *alle möglichen Pfade* durch ein Volumen praktisch nicht mehr aufgezählt werden, von ihren Nachbarknoten ganz zu schweigen.

Zuletzt stellt sich, abgesehen von den oben genannten Problemen, die Frage, wie sich eine Fläche zwischen zwei gegebenen Pfaden verhalten soll. So könnte man z. B. wie bei LiveWire üblich die Minimierung der Kantenkosten als Ziel postulieren. Eine Oberfläche wäre dann bestrebt, wie in Abbildung 7.1 so schnell wie möglich auf einen Voxel zusammen zu schrumpfen, da ein einzelner Voxel die geringsten Kosten verursacht. Um dies zu verhindern, könnte man die Gesamtkosten entlang des Pfades durch die Anzahl der Voxel teilen, um die Kosten zu normieren und nicht die Verwendung möglichst weniger Voxel zu begünstigen. Trotzdem kann man davon ausgehen, dass einer der Voxel auf einem gegebenen Pfad die geringsten Kosten verursacht. Verengt sich die Fläche so schnell wie möglich auf diesen, so hat sie auch die geringsten mittleren Kosten verursacht, denn die Hinzunahme jedes anderen Voxel brächte eine Verschlechterung.

Aus den genannten Gründen ist eine naive Übertragung der Segmentierung auf den dreidimensionalen Raum, sowohl aus praktischen wie auch theoretischen Erwägungen heraus, nicht implementierbar.

7.3 Das Problem der 3D-Segmentierung mit Live-Wire

Der vorgeschlagene Ansatz aus [HAE03] nutzt zum Ermitteln einer Oberfläche in den Volumendaten mehrere Bäume kürzester Pfade (nicht Flächen) zu unterschiedlichen Seed-Punkten, die vom Benutzer definiert werden. Abbildung 7.3 zeigt einen solchen Graphen.

Jeder Voxel hat im Raum 26 Nachbarn. Demnach enthält der Graph in jedem Knoten ine von 26 möglichen Richtungsangaben. Von jedem Ort kommt man durch Verfolgen der jeweiligen Richtung zum Seed-Punkt zurück. Damit ist jedoch noch nicht das Problem der Segmentierung gelöst, denn wie bereits erwähnt, werden zur Segmentierung im Raum Flächen und nicht Pfade benötigt. Die Graphen kürzester Pfade werden aber vom folgenden Verfahren genutzt, um möglichst optimale Segmentierungsflächen zu erzeugen. Die verwendete Graphensuche unterscheidet sich nicht vom originalen Live-Wire-Ansatz. Sie arbeitet auch auf Grundlage der Minimierung einer Kostenfunktion, die starke Kanten bevorzugt.

Die Graphen in der Art der Abbildung 7.3 werden entsprechend der Anzahl der Voxel sehr groß. Die aus CT- und MRT-Aufnahmen gewonnenen Bilder haben heute typischerweise eine Kantenlänge von 256 Voxeln und damit etwa $256^3 \approx 16$ Mio. Elemente.

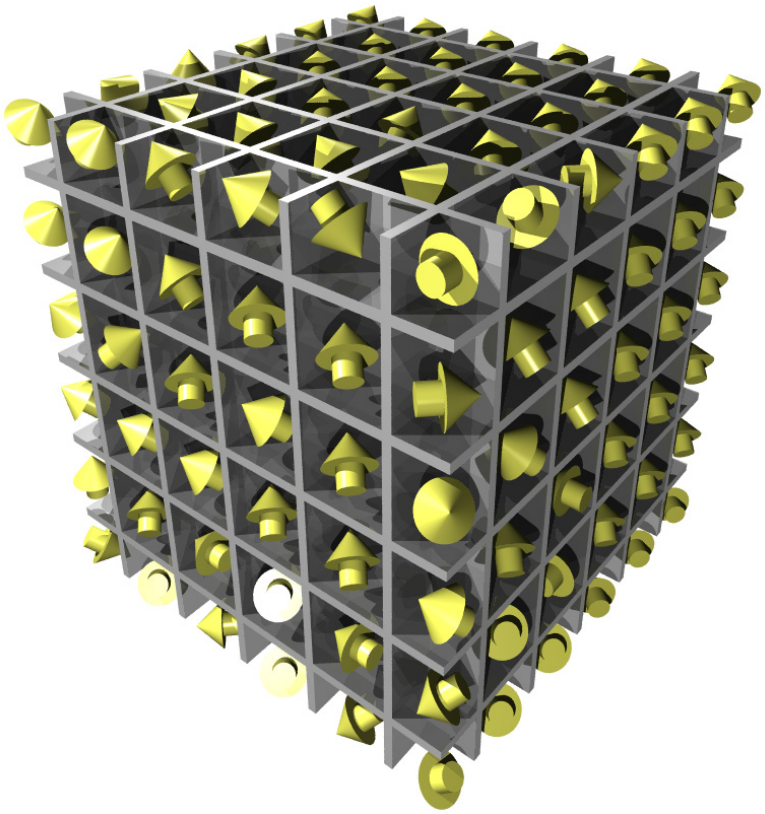

Abbildung 7.3: Live-Wire-Graph im Raum. In jedem Voxel ist eine Referenz gespeichert, die auf einen von 26 möglichen Nachbarn zeigt. Folgt man den Pfeilen konsequent, so erreicht man von jedem Voxel aus den Seed-Punkt. Dabei entsteht auf dem Weg durch das Volumen keine Fläche, sondern nur ein Pfad.

Die daraus entstehenden Bäume sind zwar aufwändig, jedoch speicher- und berechenbar.

7.4 Erzeugung der Oberfläche

Benutzerseitig wird die Oberfläche eines Objektes durch eine beliebige Anzahl von Seed-Punkten abgegrenzt, für die je ein Graph kürzester Pfade berechnet wird. In der Ebene befinden sich Kanten immer in Bereichen starker Gradienten. Dies gilt auch für die Voxeldarstellungen. Es ist also Aufgabe des Benutzers, die Seed-Punkte so im Raum zu positionieren, dass sie tatsächlich auf einer Oberfläche sitzen. Bezogen auf die

Dichtewerte der Volumendaten sollte also auch lokal ein starker Gradient vorhanden sein, und der Bereich starker Gradienten sollte in der Art einer Oberfläche zusammenhängen.

Die Oberfläche sollte der folgenden Bedingung genügen: In der lokalen Umgebung eines Seed-Punktes sollte sich eine Ebene durch den Punkt selbst finden lassen, die den Raum zumindest lokal in einen Halbraum mit höherer und einen Halbraum mit niedrigerer Dichte aufteilt. Man kann sich anschaulich vorstellen, dass ein solcher Punkt auf der Oberfläche eines Objektes sitzt. Sie hat den Charakter einer Tangentialebene. Zwischen den vom Benutzer definierten Seed-Punkten, also auf der Oberfläche (siehe Abbildung 7.6), sollten sich lückenlos weitere Punkte befinden, deren Teilerebenen (in einem Halbraum hoher und niedriger Dichte) sich auch möglichst stetig drehen. In einem solchen Fall kann von einer klar definierten Oberfläche ausgegangen werden. Die Definition, welchen Bedingungen eine Oberfläche genügen soll, ist im Raum offensichtlich weniger einfach als eine Abgrenzung von Regionen in der Ebene. Eindeutig kann man die Oberfläche eines Objektes sehr hoher Dichte in einer Umgebung geringer Dichte finden. So sind die Kirschkerne in einem lockeren Kuchen leicht zu identifizieren. Viel schwieriger ist eine eindeutige Abgrenzung der Quarkfüllung, die sich an den Übergängen zum Teig teilweise mit diesem vermischt. Im ungünstigsten Fall lässt sich eine klare Trennung überhaupt nicht ausmachen, weil das sich durchdringende Gemisch kein erkennbares Objekt enthält.

Um im Endergebnis eine brauchbare Oberfläche zu erhalten, liegt es weitgehend in der Verantwortung des Benutzers, die Seed-Punkte sinnvoll im Raum zu platzieren. Abbildung 7.4 zeigt die Volumendarstellung einer fertigen Segmentierung, die mit dem Programm *3DLiveWire* entstand.

Die Seed-Punkte wurden mit weißen Kreuzen markiert. Sie liegen offensichtlich auf der Oberfläche des Nasenflügels. Die helle Oberfläche dazwischen ist das Ergebnis der Segmentierung.

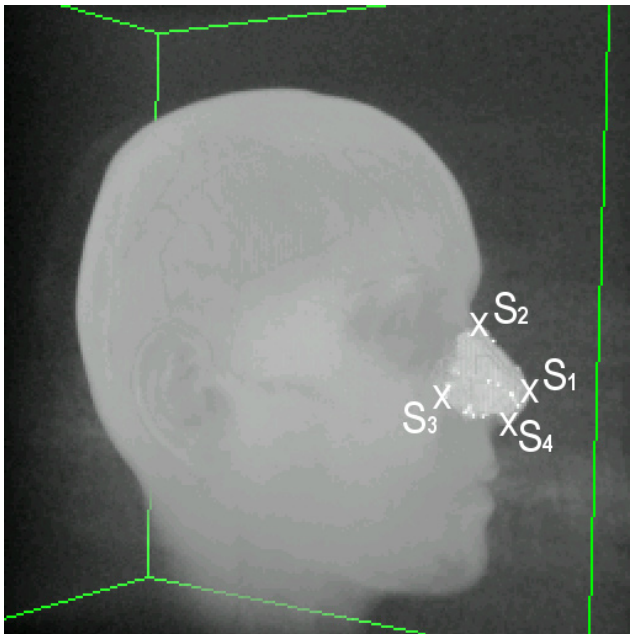

Abbildung 7.4: Darstellung eines Volumenmodells mit segmentierter Nase. Die benutzerdefinierten Seed-Punkte sind durch weiße Kreuze markiert. Die helle Oberfläche ist das Ergebnis der Segmentierung.

Bemerkung: Die Visualisierung des segmentierten Datensatzes in Abb. 7.4 entstand mit der Software VGStudio MAX der Heidelberger Softwarefirma Volumegraphics (www.volumeraphics.de), die im Gegensatz zum Volumenraytracer *VolRay* aus dieser Arbeit besser für die Berechnung von Oberflächen in Volumen optimiert ist. Für die zur Verfügung gestellte Software soll Volumegraphics an dieser Stelle herzlich gedankt werden. Daneben enthält die Anwendung *3DLiveWire* noch einmal die eigene Visualisierung, die mit der R-Taste gestartet werden kann.

Zwischen den Seed-Punkten S sucht 3DLiveWire einen geschlossenen Pfad, der auf der Oberfläche liegt. Die eingeschlossene Fläche (auch als Patch bezeichnet) ist in Abb. 7.6 schematisch in der Ebene dargestellt. Um den Umriss des Patches zu finden, muss nur, ausgehend von einem Seed-Punkt S_1, der Baum optimaler Wege (in der Art von Abb. 7.3) zu S_2 traversiert werden, um zwischen beiden die stärkste Kante zu finden. Danach wird, ausgehend von S_2, der Baum zu S_3 durchlaufen usw. Am Ende entsteht ein geschlossener Pfad zwischen den benutzerdefinierten Seed-Punkten, der auch bereits gut

mit der so abgegrenzten Oberfläche übereinstimmt. Das folgende Verfahren beschäftigt sich damit, das Innere des Patches so zu füllen, dass die gesuchte Oberfläche entsteht.

Abbildung 7.5 zeigt in dunklem Grau ein Stück des eben erwähnten benutzerdefinierten Pfades im dreidimensionalen Raum. Gesucht ist die nächste angrenzende Kette von Voxeln, hier in mittlerem Grau dargestellt - im Folgenden als Kinder der dunkelgrauen Eltern bezeichnet. Jeder neue Voxel muss dabei der Bedingung genügen, unmittelbarer Nachbar eines Elternteils und eines Bruders zu sein. Denn es soll ja eine geschlossene Oberfläche gefunden werden. Iterativ nimmt man sich nun jeden Voxel der Eltern-Kette vor. Aus der Liste aller Nachbarn eines Elternvoxel kann man den Elternteil selbst bereits streichen. Er kann kein zweites Mal gewählt werden. Zusätzlich wurde die Bedingung postuliert, dass sich ein neu generierter Voxel nicht weiter vom Mittelpunkt des Patches entfernen darf als der zuletzt betrachtete Elternteil. Ohne diese Bedingung entstanden in der Implementierung manchmal Flächen, die trichterförmig nach außen wuchsen.

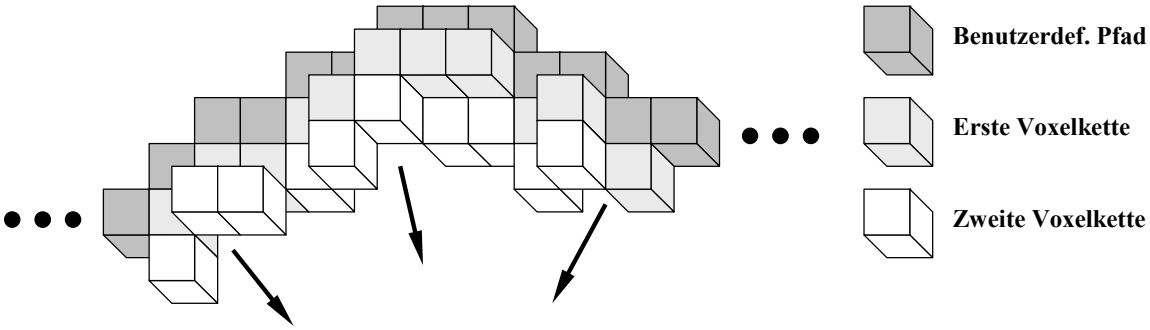

Abbildung 7.5: Der graue benutzerdefinierte Pfad wird sukzessiv durch weitere Ketten von Voxeln erweitert, bis die Oberfläche geschlossen ist.

Wie viele Kandidaten nun noch verbleiben, hängt von der jeweiligen Konfiguration der in Abb. 7.5 dunkel- und mittelgrauen Voxel ab. Im Programm 3DLiveWire werden alle 26 Möglichkeiten iteriert und die jeweils nicht gültigen nicht weiter verfolgt. Im Durchschnitt entstehen 8-9 gültige Kandidaten, also solche, die sowohl mit ihren Eltern und Geschwistern zusammenhängen als auch sich weiter dem Mittelpunkt nähern.

Für jeden dieser Kandidaten wird der optimale Pfad zu jedem Seed-Punkt berechnet (siehe Abbildung 7.6). Die Bäume optimaler Wege sind ja für jeden Seed-Punkt bekannt. Natürlich ist jeder Pfad mit einem Kostenwert verbunden. Je eher der Pfad die Möglichkeit hat, sich auf kantenreichen Voxeln zu bewegen, desto geringer werden auch die Kosten. Es bietet sich nun an, denjenigen Voxel als bestes Oberflächenelement zu wählen, der die geringsten zusammengefassten Kosten verursacht. Er liegt sozusagen auf der besten Kante zu allen Seed-Punkten, und die stärkste Kante zu allen Seed-Punkten ist wahrscheinlich die Oberfläche des gesuchten Objekts.

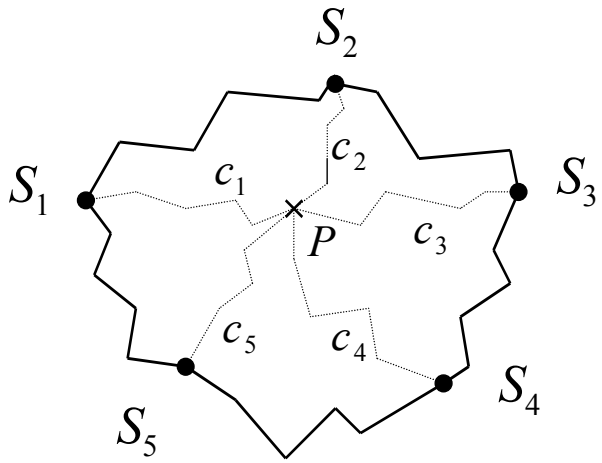

Abbildung 7.6: Die durch einzelne Live-Wires verbundenen Seed-Punkte S grenzen die Oberfläche (den Patch) ein. In der Mitte werden die Kosten für Voxel p berechnet. Diese Kosten setzen sich aus den Teilkosten zusammen, die alle Pfade c verursachen.

Die Auswahl kann gelegentlich auf einen Voxel fallen, der nicht Elternteil ist, jedoch bereits früher gewählt wurde, also auf einen Bruder. Dieser Fall wird einfach ignoriert, und die Umgebungssuche setzt sich mit dem nächsten Elternteil fort. Die doppelte Auswahl eines bereits gefundenen Kindes darf nicht ausgeschlossen werden, da man dem Algorithmus sonst verbieten würde, nach innen zu schrumpfen. Dies ist jedoch nötig, um den initialen Pfad irgendwann im Sinne einer geschlossenen Oberfläche auf einen einzelnen Punkt im Inneren zusammenzuführen.

Oben wurde in der Herleitung etwas unscharf von den *zusammengefassten Kosten* aller Pfade gesprochen. Man kann sich vorstellen, dass jeder Seed-Punkt eine Kostenbeurteilung für den jeweiligen Pfad abgibt. Als besten Kandidaten könnte man denjenigen wählen, der die geringste Kostensumme erzielt. In der Praxis hat sich aber gezeigt, dass

die Kostenbewertungen weit entfernter Seed-Punkte wenig relevant sind. Vielmehr ist wichtig, dass nahe Seed-Punkte dem Kandidaten niedrige Kosten attestieren. Daher wurde eine ungleiche Gewichtung der Teilkosten verwendet, die in der Literatur als generalisiertes baryzentrisches Koordinatensystem bekannt ist und 1990 von Loop und DeRose im Rahmen der geometrischen Modellierung vorgeschlagen wurde [LOO90].

$$c_{gesamt} = \sum c_i l_i \qquad \text{(Ausdruck 7.1)}$$

Die Teilkosten c_i werden in Ausdruck 7.1 mit den Gewichten l_i versehen. Dabei sind die Gewichte positiv und summieren sich zu 1. Das Gewicht l_i zum Seed-Punkt S_i wird zunehmend größer, je stärker sich der Voxel p an S_i annähert. Damit erhält die Kostenbewertung von S_i ein stärkeres Gewicht, die Gewichte der anderen Seed-Punkte konvergieren gleichzeitig gegen null. In der Mathematik bezeichnet man die l_i auch als eine Konvexkombination von p durch die Seed-Punkte S_i. Die genaue Berechnung und Bedeutung der baryzentrischen Koordinaten wird im nächsten Abschnitt erläutert. (Streng genommen gilt die Konvexkombination nur in der Ebene. Siehe dazu den nächsten Abschnitt.)

Das vorgeschlagene Verfahren der 3D-Segmentierung funktioniert in der Praxis dann recht stabil, wenn der Benutzer die Seed-Punkte auf eine tatsächlich vorhandene Oberfläche im Volumen setzt. Ist dies nicht der Fall, so können beliebig geformte und deformierte Oberflächen entstehen - was jedoch auch nicht verwunderlich ist. Es ist zu beachten, dass die erzeugten Oberflächen das theoretische Optimum (im Sinne optimaler Kosten) nicht garantieren können, da wie anfangs beschrieben die dynamische Programmierung wegen der enorm großen Knotenzahl im Dreidimensionalen nicht durchführbar ist. Andererseits wurde die Kandidatenwahl so ausgeführt, dass die Trajektorien zu den einzelnen Seed-Punkten minimale Kosten verursachen. Da diese Trajektorien bereits auf der Oberfläche liegen müssen, sollte die Optimalität auch für den gefundenen Voxel gelten. Es könnte jedoch sein, dass eine größere Anzahl von Voxeln mit schlechten Kosten gewählt werden müsste, um danach durch besonders niedrige Nachfolgekosten doch noch das globale Optimum zu erreichen, so wie dies bei der dynamischen Programmierung öfter der Fall ist. Die geschilderte voxelweise Suche könnte je-

doch theoretisch in ein lokales Minimum laufen. In der Praxis hat sich gezeigt, dass eine Oberfläche, die für den Benutzer offensichtlich ist, auch von 3DLiveWire verlässlich gefunden wird.

7.5 Das generalisierte baryzentrische Koordinatensystem

In der geometrischen Modellierung möchte man unter anderem gekrümmte Oberflächen möglichst einfach, flexibel und numerisch stabil erzeugen. Dazu wurde das sogenannte generalisierte baryzentrische Koordinatensystem von Loop und DeRose vorgeschlagen [LO89], [LO90].

Abbildung 7.7 skizziert die damit verbundene Aufgabe. Links sieht man eine Oberfläche im Raum (auch als Patch bezeichnet), die durch eine Anzahl von Kurven eingeschlossen ist. Dabei schließen die Kurven den Patch zwar ein, sie definieren aber nicht das Innere. Um die Fläche zu füllen, braucht man einen Parameterraum, auf dem man sich über die Fläche bewegen kann. Analog kann man sich z. B. mit nur einem Parameter auf einer Geraden hin und her bewegen. Bei einer evtl. mehrfach gekrümmten Fläche ist es nicht sofort offensichtlich, wie ein solcher Parameterraum aussehen könnte, durch den man jeden Punkt sicher erreichen kann.

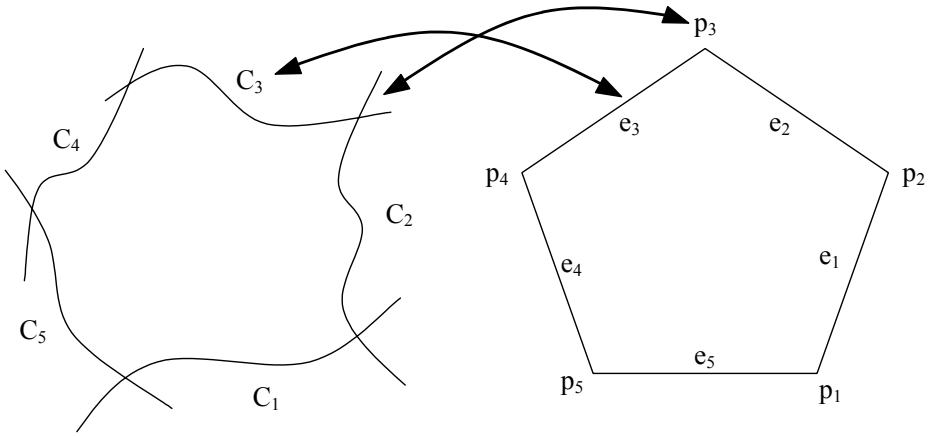

Abbildung 7.7: Eine Oberfläche wird durch fünf Kurven begrenzt (links). Rechts befindet sich der Parameterraum zur Fläche. Gesucht wird nach einer Abbildung, die einem Punkt im gleichseitigen Parameterpolygon einen Punkt auf der Oberfläche zuweist.

Hier brachte der Vorschlag des generalisierten baryzentrischen Koordinatensystems den Durchbruch Anfang der 90er Jahre. Der Parameterraum ist in Abb. 7.7 rechts darge-

stellt. Das Polygon in der Ebene besitzt ebenso viele Eckpunkte wie der Patch Schnittpunkte von Kurven im Raum besitzt. Ansonsten gibt es keine Gemeinsamkeiten, denn das Polygon ist gleichseitig. Im Folgenden geht es darum, einen Punkt p innerhalb des Polygons in den korrespondierenden Patch im Raum zu übertragen. Ein Beispiel ist in Abb. 7.8 gezeigt. Der frei gewählte Punkt teilt das Polygon in eine Anzahl von Dreiecksflächen mit den Flächeninhalten α_i ein. Warum werden diese Flächen berechnet? Letztlich geht es darum, den Punkt p selbst als eine Konvexkombination der umgebenden Ecken des Polygons p_i zu berechnen. Mit anderen Worten: Gesucht sind die einzelnen Gewichtungen l_i passend zu den Ecken p_i, so dass deren Summe, wie in Ausdruck 7.1 formuliert, genau p ergibt.

$$p = \sum_{i=1}^{m} l_i(p) p_i \qquad \text{(Ausdruck 7.1)}$$

Mit Hilfe der Gewichte kann dann im Raum auf dem Patch ein zu p entsprechender Punkt gefunden werden. Eine Lösung hierfür wurde von S. Kurijama [KUR93] vorgeschlagen, ist jedoch nicht mehr Gegenstand dieser Arbeit. Es genügt uns einen Parameterraum für einen gekrümmten Patch zu finden.

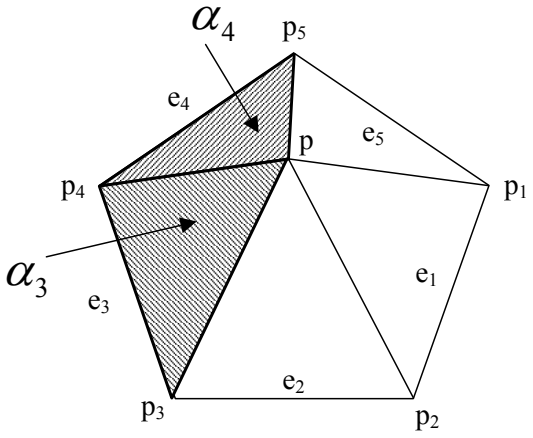

Abbildung 7.8: Ein Punkt p spannt mit jeder Seite des Polygons eine Fläche α_i auf.

$$l_i(p) = \frac{\pi_i(p)}{\sum_{k=1}^{m} \pi_i(p)} \qquad \text{(Ausdruck 7.2)}$$

Die erwähnten Gewichte l_i berechnen sich nach Ausdruck 7.2. Dazu werden die Anteile $\pi_i(p)$ benötigt, die in Ausdruck 7.3 ermittelt werden. π_i versteht sich als ein Maß, welches jeder Ecke des Polygons zugeordnet ist.

$$\pi_i(p) = \frac{\prod_{k=1}^{m} \alpha_k(p)}{\alpha_{i-1}(p)\alpha_i(p)} \qquad \text{(Ausdruck 7.3)}$$

Die α_i sind gerade die bereits erwähnten Flächeninhalte aus Abb. 7.8. Auch für π_i gibt es eine anschauliche Bedeutung. Man kann es sich als Produkt aller Flächen α_i vorstellen, die den Punkt p_i *nicht* berühren.

> Bemerkung: Bei flüchtiger Betrachtung könnte man meinen, dass der Nenner von Ausdruck 7.3 null werden könnte, wenn die beiden α im Nenner null werden. Dies ist auch tatsächlich der Fall, wenn p gegen p_i konvergiert. Es ist aber zu beachten, dass man α_{i-1} und α_i mit dem Zähler kürzen kann. Was verbleibt, ist, wie bereits erwähnt, das Produkt aller Teilflächen, die nicht an p_i angrenzen.

Mit den π_i ist die Konvexkombination eigentlich schon so gut wie berechnet. Denn wenn p gegen p_i konvergiert, so werden die Flächen π_{i-1} und π_i null und alle anderen Flächen nehmen die volle Fläche des Polygons ein. Daher wird das entsprechende Gewicht π_i maximal. Die Berechnung der l_i hat eigentlich nur noch zur Aufgabe, die Gewichte π_i in das Intervall [0, 1] zu normieren.

Nun gelten folgende Eigenschaften:

Liegt p genau auf p_i, so soll auch nur das entsprechende Gewicht eins sein.

$$l_i = 1 \cap l_{j \neq i} = 0 \qquad \text{(Ausdruck 7.4)}$$

Bewegt sich p auf der Linie zwischen p_i und p_{i+1}, so sind auch nur die entsprechenden Gewichte l_i und l_{i+1} ungleich null.

$$l_i + l_{i+1} = 1 \cap l_{j \neq i,i+1} = 0 \qquad \text{(Ausdruck 7.5)}$$

Zuletzt soll die Summe der Gewichte eins betragen.

$$\sum_{i=1}^{m} l_i = 1 \qquad \text{(Ausdruck 7.6)}$$

In Abschnitt 7.4 soll gerade das umgekehrte Problem gelöst werden. Hier ist nicht gefordert, einen Punkt p im Parameterraum auf den Patch zu übertragen. Im Gegenteil ist der Punkt auf dem Patch, nämlich ein Kandidaten-Voxel, bereits gegeben. Gesucht ist vielmehr die Gewichtung aller Seed-Punkte, die den Ecken des Polygons im oben beschriebenen Sinne entsprechen.

Dazu wurde das eben ausgeführte Verfahren nur leicht abgewandelt. Der einzige Unterschied besteht darin, dass p (ein Kandidaten-Voxel) und die Eckpunkte p_i (die Seed-Punkte) nicht in der Ebene liegen, sondern im Raum verteilt sind. Auch sind die Seiten des Polygons nicht mehr gleichseitig. Ansonsten ändert sich an der Berechnung der π_i und l_i nichts. Auch die nützliche Eigenschaft der Gewichte l_i hat sich nicht geändert. Konvergiert p gegen einen Seed-Punkt p_i, so wird dessen Gewicht $l_i \equiv 1$. Genau das sollte ja erreicht werden, denn je weiter sich p an p_i annähert, desto mehr sollten die Kosten des Live-Wire zu p_i ins Gewicht fallen. Die Kostenwerte, die die entfernteren Seed-Punkte melden, sind dabei weniger von Bedeutung.

7.6 Kostenmaß für 3D-LiveWire

Das oben beschriebene Verfahren der Generierung von Oberflächen basiert anschaulich gesprochen auf der wiederholten Anwendung des LiveWire Verfahrens das, jedes einzelne für sich gesehen, lediglich eine optimale Trajektorie durch ein Volumen ermittelt. Dabei stellt sich wie auch für die 2D-Variante die Frage nach dem optimalen Kostenmaß. Im Rahmen dieser Arbeit wurde lediglich der Betrag des Gradienten im Raum für die Kosten verwendet. Ebenso wie LiveWire im Bild die stärkste Kante sucht, geschieht dies im Raum.

Theoretisch könnte man die Kostenfunktion um weitere Terme erweitern, so z. B. um eine Glattheitsbedingung. Insbesondere in Bezug auf die Glattheit gilt aber die analoge Kritik, die auch schon in Abschnitt 6.2.4 zur Anwendung kam. Denn lediglich dann, wenn Oberflächen tatsächlich in der Weise glatt sind, wie es der Formterm begünstigt, kann die Segmentierung gewinnen. Hat man jedoch kein a priori-Wissen über eine Oberfläche, so sollte man keine Annahme begünstigen, die der Charakteristik der gesuchten Fläche evtl. gar nicht entspricht.

7.7 Waveletanalyse im Raum

In Zusammenhang mit der Kostenfunktion stellt sich zwangsläufig die Frage, ob das Segmentierungsverfahren im Raum von einer Wavelet-Zerlegung in der gleichen Weise profitieren würde, wie dies in der 2D-Segmentierung der Fall war (vergleiche hierzu auch Kapitel 8).

Gerade bei Volumendaten aus CT-Aufnahmen spielt das Rauschen eine mitunter erhebliche Rolle. Auch hier ist anzunehmen, dass Störungen nicht gleichmäßig über alle Skalen verteilt sind, sondern dass sie sich auf bestimmten Frequenzbändern konzentrieren. So gesehen wäre eine Analyse der Verlässlichkeit jeder Skala einer möglichen Wavelet-Zerlegung durchaus sinnvoll. Für die Implementierung wäre eine naive Wavelet-Transformation keine große Erschwernis, da alle sogenannten separierbaren Wavelets dazu verwendet werden können das Signal entlang der Achsen des Koordinatensystems nacheinander und in beliebiger Reihenfolge zu filtern.

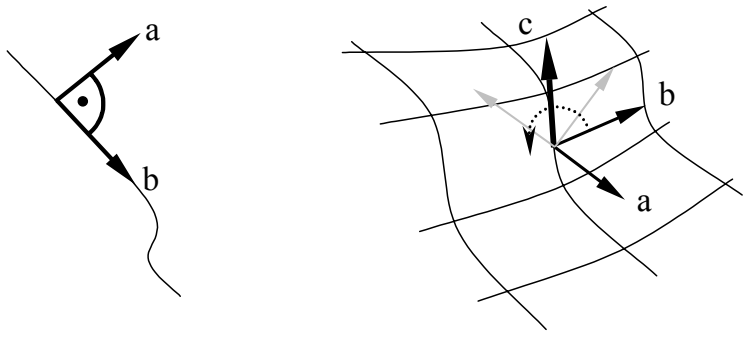

Abbildung 7.9: Im zweidimensionalen Fall ist die Richtung der Wavelet-Transformation mit der Objektgrenze eindeutig festgelegt. Im Dreidimensionalen kann das lokale Koordinatensystem dagegen frei um den Gradienten a gedreht werden. Die Oberfläche selbst legt diesen Freiheitsgrad nicht fest.

Ein jedoch nicht triviales Problem stellt die Richtung dar, in der die Transformation durchgeführt werden soll. Abbildung 7.9 veranschaulicht den Sachverhalt. Schon im zweidimensionalen Fall wird in Abschnitt 5.2.3 darauf hingewiesen, dass die Transformation entlang der Struktur der Objektgrenze durchgeführt werden muss. Die Erklärung hierfür ist, dass die Richtung der zu verfolgenden Struktur keine Rolle spielen darf. Statt dessen soll immer entlang der Objektgrenze transformiert werden. Der Vektor a ist in Abbildung 7.9 durch den Gradienten festgelegt. Der senkrechte Vektor b ist damit eindeutig festgelegt, sieht man vom Vorzeichen ab. Die analoge Überlegung müsste nun ins Dreidimensionale übertragen werden (Abbildung 7.9, rechts). Denn auch hier müsste eine Oberfläche die gleichen Koeffizienten erzeugen, unabhängig davon in welche Richtung sie verläuft. In der Ebene erfordert die Definition eines lokalen Koordinatensystems (das eine Richtung für das Filtern im Rahmen der Transformation festlegt) lediglich einen Winkel. Analog kann man auch nur eine Achse definieren, die zweite muss bereits senkrecht dazu stehen und stellt somit keinen weiteren Freiheitsgrad dar.

Zwar definiert der Gradient auch im Dreidimensionalen eine Richtung (also eine Achse), es fehlen aber zwei weitere Achsen zur Definition eines vollständigen lokalen Koordinatensystems. Man kann sich die beiden Verbleibenden als frei drehbar, am Gradienten befestigt, vorstellen. Jede mögliche Drehung entspricht einem gültigen Koordinatensystem. Und für jedes würde die Wavelet-Transformation eine eigene Zerlegung erzeugen, deren Koeffizienten vollkommen andere Charakteristika besitzen. Das Auffinden eines bestimmten Musters würde dies erheblich erschweren.

Weitere Forschung in Richtung einer transformationsbasierten Kostenfunktion könnt dennoch zu einer besseren Mustererkennung für die Segmentierung von Volumendaten führen. So wurden in der Literatur eine Reihe rotationsinvarianter Maße vorgeschlagen, unter anderem die sogenannten Hu-Momente [HU62]. Letztere findet man im Bereich von Bilddatenbanken und in der Texterkennung. Die Idee der Hu-Momente ist, jeden Pixel in Bezug zum Schwerpunkt S eines Objektes zu setzen. So kann man z. B. für jeden Objektpixel die Entfernung zu S berechnen und den entsprechenden Wert mit dem Grauwert des Pixels gewichten. Summiert man über alle Pixel des Objektes, so entsteht

ein charakteristischer Wert. Da sich der Schwerpunkt in Bezug auf die Pixel des Objekts auch unter Translation oder Rotation niemals verändert, erhält man die nützliche Eigenschaft der Translations- und Rotationsinvarianz. Nachteilig wäre im Rahmen der Segmentierung, dass der Schwerpunkt erst berechnet werden kann, wenn ein Objekt bereits identifiziert ist. Trotzdem kann man vermuten, dass rotationsinvariante Maße und Zerlegungen für das Auffinden einer Oberfläche mit ähnlichen Eigenschaften, nützlich sein könnten.

KAPITEL 8

VERGLEICH VERSCHIEDENER TRANSFORMATIONEN

8.1 Aufbau des Vergleichsverfahrens

Nach der aufwändigen Einführung verschiedener Transformationen in den ersten Kapiteln sollen diese nun abschließend auf ihre Brauchbarkeit für die Segmentierung untersucht werden. Grundlage wird das in Kapitel 6 beschriebene hybride Verfahren sein, welches die Graphensuche von Live-Wire mit SemiSeg kombiniert. Jedoch werden nicht nur Wavelets, sondern unterschiedliche Transformationen eingesetzt. Jeder Transformation ist dabei eigen, dass das Bildsignal entlang der Silhouette zeilenweise in Koeffizienten zerlegt wird. Danach wird jeder Koeffizient, wie in Kapitel 5 beschrieben, anhand der Varianz auf seine Nützlichkeit zum Finden der Objektkontur bewertet. Die eigentliche Suche nach ähnlichen Strukturen strebt danach, in der Umgebung diejenigen Koeffizienten mit der niedrigsten Varianz wieder zu finden. Ob es überhaupt Koeffizienten mit einer besonders niedrigen Varianz gibt, hängt natürlich ganz von der Transformation ab und soll im Folgenden evaluiert werden.

Zu jedem Testbild existiert eine vollständig benutzergesteuerte Segmentierung, die als optimal gilt. Jedem Algorithmus wird eine maximale mittlere Abweichung von der optimalen Grenze eingeräumt. Im folgenden Test wurde eine mittlere Abweichung von der

optimalen Silhouette von zwei Pixeln erlaubt. Unterscheiden konnten sich die einzelnen Verfahren nur noch in der Anzahl der (Seed-) Punkte, die zur Einhaltung der mittleren Abweichung nötig waren. Dies entspricht der Anzahl an Klicks, die der Benutzer auf die Objektsilhouette hätte tätigen müssen. Deshalb werden kleinere Werte bevorzugt.

8.2 Der Testalgorithmus

Die eigentliche automatische Evaluation wurde ebenfalls mit dem Programm Semi-SegDP gemacht und funktioniert wie folgt: In der Kommandozeile muss hinter dem Bild noch die .ASC Datei angegeben werden, die eine pixelweise Beschreibung der optimalen Silhouette in absoluten Koordinaten enthält. Eine solche Datei kann im interaktiven Modus von Hand erzeugt werden, eine genauere Beschreibung zu diesem Modus und einer weiteren möglichen Realisierung eines Vergleichstests findet sich im Anhang. Der dritte Parameter der Kommandozeile gibt die tolerierte mittlere Abweichung in Pixeln an.

Der Testmodus wird mit der S-Taste gestartet. Dabei setzt der Algorithmus den ersten Seed-Punkt auf den ersten Punkt der Silhouette. Danach wird ein (sehr kurzer) Live-Wire zum zweiten Punkt der Silhouette berechnet. Bis jetzt ist der mittlere Abstand von der Referenzsegmentierung zwangsläufig null. Danach entfernt sich der zweite Punkt immer weiter (auf der Objektsilhouette) vom Seed-Point. Bei jedem Fortschreiten wird erneut der mittlere Fehler in Pixeln zwischen dem nun immer länger werdenden Live-Wire und der optimalen Segmentierung berechnet. Überschreitet der Fehler die tolerierte Grenze, so wird ab dem entsprechenden Endpunkt ein neuer Seed-Point gesetzt. Das Verfahren setzt sich nun analog fort, bis das Objekt letztlich vollständig segmentiert ist. Entscheidend für den Vergleich ist die dabei benötigte Anzahl von Seed-Punkten, wie sie in Tabelle 8.1 für unterschiedliche Varianten und Bilder eingetragen ist.

8.3 Versuch einer repräsentativen Bildauswahl

Zum Test wurden aus einer Sammlung von Bildern, die in ganz unterschiedlichem Kontext aufgenommen wurden, 20 Photos ausgewählt und die interessierenden Objekte von Hand pixelweise segmentiert. An dieser Referenzsegmentierung musste sich jedes Verfahren messen lassen. Hier sei noch einmal betont, dass es in der folgenden Evaluation um den Vergleich unterschiedlicher Transformationen ging, bzw. um deren Eignung für die Segmentierung oder noch genauer um deren Fähigkeit die Bildinformation in der best-möglichen Weise zu dekorrelieren.

An einigen Stellen ist dabei die Abgrenzung der Objekte vom Hintergrund nur für den menschlichen Betrachter aufgrund seines Vorwissens nachvollziehbar. Es ist also nicht grundsätzlich gewährleistet, dass offensichtliche Bildeigenschaften eine gute Abgrenzung des Objektes erlauben. Für die Evaluation ist dies jedoch nicht von Bedeutung, denn diesem Problem sehen sich alle Verfahren in gleicher Weise gegenüber gestellt.

Die Auswahl der Bilder selbst wurde vollkommen zufällig erzeugt. Man könnte hier einwenden, dass eine zufällige Auswahl auch keine Gleichverteilung über unterschiedliche Bildtypen gewährleistet. Aber selbst wenn man versucht hätte, eine bestimmte Gewichtung unterschiedlicher Bildtypen zu erzeugen, wäre auch diese Gegenstand der Diskussion gewesen. Beliebte Motive können sich im Laufe der Zeit auch verändern, so dass ein allgemeingültiger, typischer Mix von Bildern kaum möglich ist. Die getroffene Auswahl versteht sich also als Stichprobe einer zu Anfang des dritten Jahrtausends typischen Sammlung von Photos.

Deshalb wird es bei der Bewertung der Brauchbarkeit einzelner Transformationen besonders darauf ankommen festzustellen, für welchen Typ von Bild sie sich eignet. Der Benutzer könnte dann für unterschiedliche Bilder auf einen Baukasten verschiedener Segmentierungsverfahren zurückgreifen. Auch wäre für eine kommerzielle Implementierung denkbar, je nach Beschaffenheit des Bildes automatisch zwischen verschiedenen Verfahren umzuschalten, um das jeweils beste Ergebnis zu erzielen. Abbildung 8.1 zeigt die 20 handsegmentierten Bilder. Die Referenzsegmentierung wurde durch einen weißen Rand dargestellt.

8.4 Untersuchte Transformationen

Als Zerlegung wird die *Wavelettransformation*, die *Cosinus*- bzw. *Hadamard-Transformation*, die Suche im *Ortsraum* und das originale *Live-Wire* Verfahren verwendet. Die Wavelettransformation wurde in Kapitel 5 ausführlich behandelt. Das reine Live-Wire Verfahren zählt streng genommen nicht zu den Transformationen, weil nur nach einer Kante gesucht wird. Es wurde aber zum Zweck der Vergleichbarkeit in die Untersuchung mit aufgenommen.

Die Cosinus-Transformation wurde in Kapitel 3 ausführlich behandelt. Die Hadamard-Transformation (HT) ist eigentlich nichts anderes als die auf ihr Vorzeichen reduzierte DCT: ist der Cosinus über der Abszisse, so ist die HT (+1), sonst (-1).

Die Suche im Ortsraum ist einfach. Hier wurde durch einen pixelweisen Vergleich versucht, Objektgrenzen mit ähnlichen Eigenschaften zu finden, also ganz ohne Drehung.

Alle verwendeten Bilder und die ausführbaren Programme sind auf der CD enthalten, die dieser Arbeit beigefügt ist, so dass der Test durch einfaches Aufrufen der jeweiligen Programmversionen, mit einem Bild und der Referenzsegmentierung (der .ASC-Datei) in der Kommandozeile, nachvollzogen werden kann. Der Tabelle 8.1 liegt ein tolerierter mittlerer Abstand von 2.0 Pixeln zugrunde.

Abbildung 8.1a: Testbilder - Seemann (1), Ente (2), Waschbär (3), Kaffeetasse (4), Photograph (5), Pärchen (6) , Katze (7), Bär (8), Gründerzeit (9), Halloween(10)

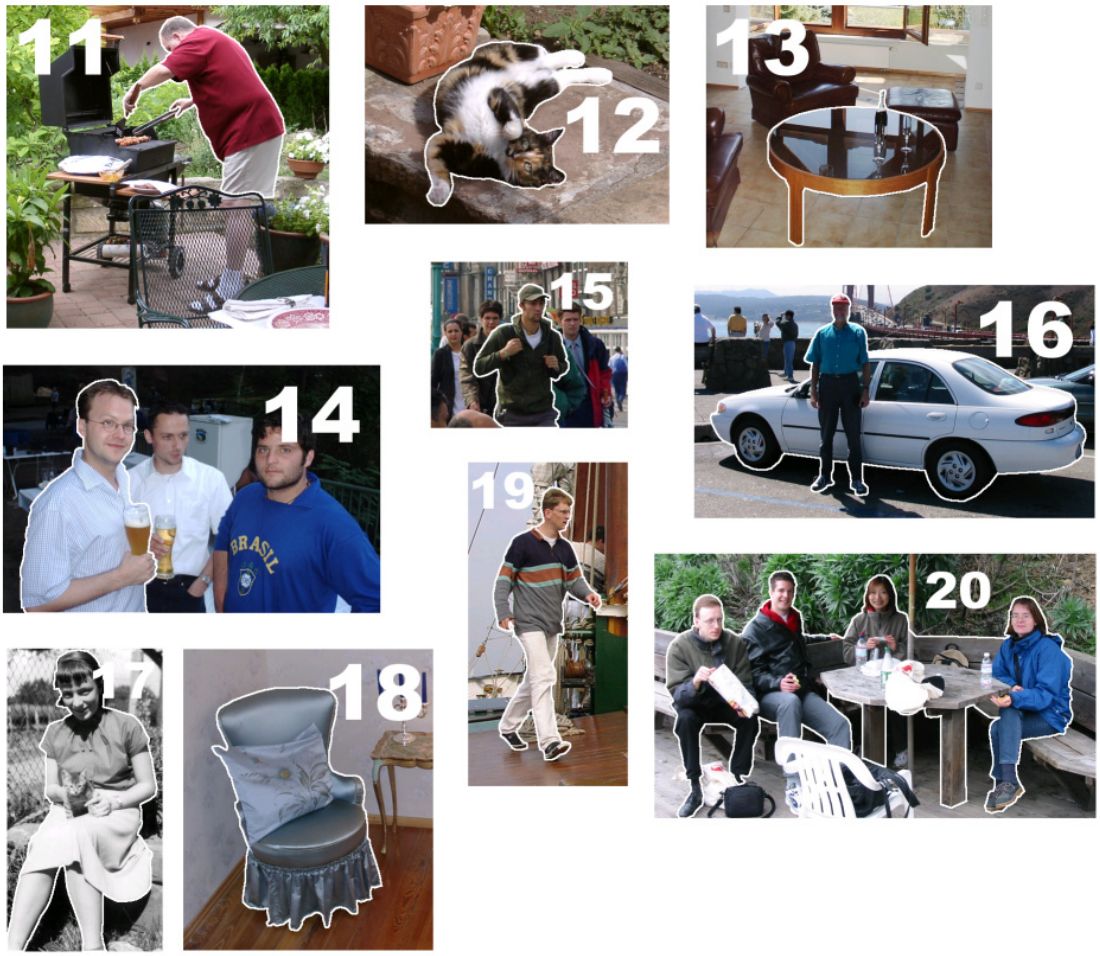

Abbildung 8.1b: Testbilder, Fortsetzung - BBQ (11), Souris (12), Tisch (13), Gelage (14), Fußgänger (15), Golden Gate Bridge (16) , Mama (17), Sessel (18), Matrose (19), Mittag (20)

Bild/Verfahren	Wavelets	Live-Wire	Hadamard	DCT	Ortsraum
Gründerzeit	26	31	50	45	29
Ente	20	19	27	28	25
Bär	21	15	28	33	14
Waschbär	34	63	46	39	37
Kaffeetasse	38	30	60	57	39
Katze	13	18	19	21	14
Pärchen	31	41	48	31	29
Seemann	36	32	58	38	47
Halloween	29	15	42	29	31
Photograph	24	21	36	30	24
Mama	26	34	51	43	28
Matrose	35	39	42	41	34
BBQ	24	21	24	27	21
Fußgänger	13	17	22	18	20
Sessel	22	24	37	26	22
Souris	20	23	30	28	19
Tisch	13	14	19	17	18
Gelage	17	20	30	28	22
Mittag	85	83	104	108	86
G.G. Bridge	40	41	51	26	45
Summe	567	601	824	712	604

Tabelle 8.1: Die 20 Bilder aus Abbildung 8.1 wurden mit fünf verschiedenen Transformationen segmentiert. Die Einträge entsprechen der Anzahl von Seed-Punkten (bzw. benötigten Maus-Klicks) für Segmentierungen vergleichbarer Qualität.

8.5 Evaluation und Interpretation

Insgesamt kann man an der Summe der Seed-Punkte am Ende der Tabelle feststellen, dass das Verfahren SemiSegDP, also die dynamische Programmierung, kombiniert mit einer Wavelet-Analyse, mit 567 Punkten am besten abgeschnitten hat. Danach folgt mit 601 Punkten der originale Live-Wire Algorithmus, der, wie bereits erwähnt, insofern aus der Rolle fällt, als er durch seine Kantensuche gar keine Transformation durchführt. Mit nur marginalem Unterschied folgt dann die Suche im Ortsraum. Die beiden Transformationen Hadamard und DCT folgen erst in erheblichem Abstand.

Neben den zwar unterschiedlichen, jedoch größenordnungsmäßig ähnlichen Gesamtergebnissen fällt auf, dass sich einzelne Segmentierungen eher unerwartet verhalten. So schneidet Live-Wire, trotz der insgesamt etwas schlechteren Bewertung, im Beispiel *Kaffeetasse* etwa 20% besser ab als das waveletbasierte SemiSegDP (siehe Kapitel 6)

aus Spalte 1. Andererseits braucht die reine Kantensuche Live-Wire im Beispiel *Waschbär* fast doppelt so viele Seed-Punkte. Betrachtet man Bild 3 in Abb. 8.1, so fällt auf, dass die Pflanzen im Hintergrund relativ viele Kanten verursachen. Das gleiche gilt für das stark texturierte Fell des Waschbären. Man kann sich leicht vorstellen, dass eine rein kantenbasierte Suche in Texturen schnell fehlgeleitet wird. Hier ist eindeutig Semi-SegDP im Vorteil, weil Kanten bei der Analyse dieses Beispiels keine große Rolle spielen sollten. Es ist zu vermuten, dass die Koeffizienten der tiefen Frequenzen ein verlässlicheres Verfolgen der Objektgrenze zulassen.

Bild 4 ist dagegen für eine Kantensuche eher geeignet. Man kann dies in Abbildung 8.2 sehen. Die Katze ist von einem deutlichen weißen Rand umgeben, vor allem im Bereich des Rücken. Im Vergleich dazu ist eine Kantensuche im Waschbär-Bild vor allem im Bereich der Haare der Frau aussichtslos. Auch der Waschbär ist kaum noch zu erkennen. In diesem Bereich kann die skalenbasierte Wavelet-Analyse ihre Stärken ausspielen. Im Bereich der Jacke leisten dagegen beide Verfahren eine vergleichsweise gute Arbeit.

Abbildung 8.2: Kantenbild Kaffeetasse (links) und Waschbär (rechts)

Abschließend kann man zum Vergleich zwischen Live-Wire und der Wavelet-Analyse sagen, dass beide oft ähnliche Ergebnisse liefern, die reine Kantensuche jedoch in einigen wenigen Fällen recht deutlich versagt. Nur durch diese Ausreißer entsteht im Wesentlichen der Unterschied in der Gesamtsumme.

Wie aber kann es sein, dass die mit 712 Punkten vom Trend her erheblich schlechtere DC-Transformation im Bild *G. G. Bridge* mit annähernd der Hälfte der Punkte aus-

kommt, die alle anderen Verfahren brauchen? Auch die schlechteste Transformation, *Hadamard,* nimmt in einigen Bildern zumindest einen Platz im Mittelfeld ein. ie Begründung lässt sich in der mathematischen Statistik aus Kapitel 2 finden. Denn die in Pixelspalten zerlegten Objektgrenzen (siehe Kapitel 5) der unterschiedlichen Bilder sind nichts anderes als Punktewolken in einem Raum hoher Dimension. Welche Drehung bzw. welche Transformation eine solche Wolke so dreht, dass die Charakteristik der Objektgrenze besonders offensichtlich wird, kann nicht a priori für alle Bilder festgelegt werden, und dies war auch nicht zu erwarten. So ist es nicht mehr verwunderlich, dass die Charakteristik einer Kante vielleicht ganz ohne Drehung bereits im Ortsraum besonders offensichtlich ist. Dies ist z. B. in den Bildern *Bär*, *Pärchen*, *Matrose* und *Souris* der Fall. Etwas lax gesprochen könnte man sagen, dass das gewöhnliche Betrachten des Bildes im Ortsraum hier die besten Ergebnisse liefert. Die relevante Information ist bereits in den Pixeln des Bildes (vermutlich weitgehend) optimal dekorreliert. Die Anwendung einer Drehung vermischt und verschleiert hier die Charakteristik nur unnötig. Bei den meisten Bildern scheint aber die Wavelettransformation die Punktewolken so zu drehen, dass die charakteristischen Aspekte erst danach besonders deutlich werden. Vielleicht würde man für andere als den Haar-Filter noch bessere Ergebnisse bekommen, zumindest auf einer Auswahl von Bildern. an kann also resümieren, dass keine Drehung von vornherein alle Typen von Objektgrenzen am besten dekorreliert. Gäbe es eine solche optimale Drehung, so könnte man einfach eine Objektgrenze konstruieren, die die Charakteristik nach der Transformation gleichmäßig über alle Koeffizienten vermischt, und hätte so schon den Gegenbeweis angetreten. In der Sprache der Statistik würde man sagen, dass man die Daten immer so manipulieren kann, dass Faktorladungen gleichmäßig über alle Faktoren verteilt sind. In den Worten der linearen Algebra kann man die Daten so manipulieren, dass alle Eigenvektoren den gleichen Eigenwert besitzen (streng genommen dürfte man dann von Eigenvektoren nicht mehr sprechen).

Vielleicht konnte diese Arbeit dazu beitragen, die Lücke zwischen der einfachen, univariaten Statistik und den modernen Transformationen etwas zu schließen. Der Behauptung "Wavelets eignen sich grundsätzlich besser zur Analyse oder Kompression" darf man nun skeptisch gegenüberstehen, denn dreht man eine Punktewolke einfach aus dem angeblich besten Koordinatensystem heraus, beweist man damit das Gegenteil.

KAPITEL 9

FAZIT UND AUSBLICK

9.1 Fazit

Das Hauptziel dieser Arbeit bestand darin, benutzerunterstützte Verfahren der Segmentierung zu untersuchen und über den aktuellen Stand der Forschung hinaus zu erweitern. Damit die implementierten Verfahren ein Objekt möglichst selbständig abgrenzen können, wird ein vom Benutzer definiertes Beispiel der Objektgrenze auf Regelmäßigkeiten untersucht. Dabei wird in der Regel eine gewisse Form der Kontinuität bzw. ein bestimmtes Muster entlang der Objektgrenze gefunden. Um weiter zu segmentieren wird ermittelt, in welche Richtung sich dieses Muster am deutlichsten fortsetzt.

Eine wesentliche Erkenntnis im Rahmen dieser Arbeit ist, dass diese Regelmäßigkeit entlang einer Objektgrenze oft erst nach einer Transformation offensichtlich wird. Eine schwarze Linie (auch Trajektorie) im oberen Teil der Abbildung 9.1 grenze ein Objekt vom Hintergrund ab. Alleine durch Detektieren der Linie lässt sich das Objekt segmentieren. Die Regelmäßigkeit der sich wiederholenden dunklen Pixel ist im Ortsraum leicht zu beobachten.

Im unteren Teil der Abbildung hat die Trajektorie links einen anderen Charakter. Sie besteht nun aus Paaren von je einem hellen und dunklen Pixel, gefolgt von einem inversen Pixelpaar.

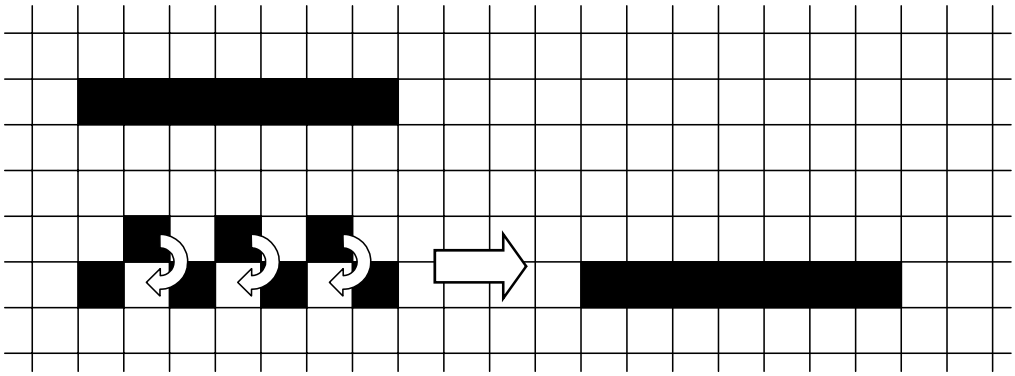

Abbildung 9.1: Offensichtliche Trajektorie (oben). Die ebenfalls korrelierte Struktur unten wird erst nach einer Vertauschung jedes zweiten Pixelpaares als ebenso kontinuierlich wahrgenommen. Dabei ist die Vertauschung die Transformation, die die Struktur noch deutlicher hervorhebt.

Die Kontinuität besteht nun nicht mehr allein aus sich wiederholenden Pixeln, sondern aus der alternierenden Formation. Diese Regelmäßigkeit ist für den Betrachter immer noch deutlich, jedoch nicht mehr ganz so offensichtlich, wie die einfache Trajektorie im oberen Teil der Abbildung. Sie besteht eben genau darin, dass benachbarte Pixel negativ korreliert sind. Man kann schnell eine Transformation angeben, die die Regelmäßigkeit wieder trivial macht, nämlich die Vertauschung jedes zweiten Pixelpaares.

Ausgehend von diesem Beispiel kann man sich vorstellen, dass es auch Korrelationen mit einer größeren Anzahl beteiligter Pixel gibt, die durch bloßes Betrachten nicht mehr leicht erkennbar sind. Existiert eine solche Korrelation jedoch, so kann man auch eine Transformation finden, die die Kontinuität wieder offensichtlich macht (siehe Kapitel 2). In der Sprache der Statistik spricht man von der Dekorrelation der Information. Nach der Dekorrelation hat jeder Koeffizient nur noch eine einzigartige Information, die keine Querbezüge zu anderen Koeffizienten mehr enthält. Auf das Bild bezogen soll ein Verhältnis der Art "Pixel A hat den inversen Grauwert von Pixel B" nicht mehr existieren. Erst dann kann die Trajektorie wieder zuverlässig verfolgt werden. Zum Schluss der Arbeit werden in Kapitel 8 erneut unterschiedliche Transformationen aufgegriffen und auf ihre Brauchbarkeit zur Dekorrelation von Bildinformation, konkret zum Zweck der Segmentierung, untersucht.

Der zweite große Beitrag der Arbeit besteht darin, die halbautomatische Bildsegmentierung auf Volumendaten auszudehnen, die ihrerseits im dreidimensionalen Raum definiert sind. Im 2D-Bild konnte ein Objekt durch eine Trajektorie von seiner Umgebung abgegrenzt werden. Im Raum ist das abgrenzende Objekt selbst eine gekrümmte Fläche und hat mehr Freiheitsgrade bei der Generierung. Die sich daraus ergebende Komplexität ist aus den in Kapitel 7 beschriebenen Gründen grundsätzlich nicht mehr handhabbar, zumindest dann nicht, wenn man eine naive Übertragung der für Bilder geeigneten Verfahren anstrebt. Daher wurde eine Heuristik entwickelt, die eine möglichst optimale Oberfläche auf Grundlage von vielen nachweislich optimalen 1D-Trajektorieren zusammensetzt. Die Berechnung einer 1D-Trajektorie ist im Raum nicht aufwändiger als im Bild, allerdings fordert die Vorbereitung zur Berechnung eines Baumes kürzester Wege einen höheren Aufwand.

9.2 Ausblick auf Nahziele

Dem Evaluationskapitel 8 könnte man neben den fest definierten Transformationen wie DCT, Wavelet-Transformation etc. eine weitere, in der Art der KLT dynamisch erzeugte hinzufügen. Gewissermaßen würde man damit selbst die Analyse erzeugen, die die vom Benutzer beispielhaft definierte Grenze zwischen Objekt und Hintergrund am besten dekorreliert. Man müsste dann sozusagen nicht mehr darauf hoffen, dass dies eine fest definierte Transformation am besten tut. Geht man davon aus, dass sich die Charakteristik von Objektgrenzen über das gesamte Objekt hinweg treu bleibt, so wäre damit die beste Segmentierung möglich. In der Regel ist jedoch mit einer unveränderlichen Kantencharakteristik nicht zu rechnen, so dass a priori auch nicht klar ist, ob vielleicht die kontinuierliche Wavelet-Transformation, für Kanten im Allgemeinen, nicht doch besser geeignet ist. Es ist jedoch zu beachten, dass eine einfache Anwendung der KLT (siehe Kapitel 2) in der Regel zu einer DC-ähnlichen Transformation führt. Die genaue Begründung hierzu wurde in Kapitel 3 gegeben. Eine nutzbringende Anwendung der KLT ist daher im Rahmen einer naiven Implementierung nicht zu erwarten, könnte aber in einer (noch zu erforschenden Anpassung) durchaus auch zu Verbesserungen bei der Mustererkennung der Objektgrenze führen.

Eine weitere Verbesserung könnte man evtl. dadurch erreichen, dass man die optimale Transformation einer sich leicht, jedoch kontinuierlich verändernden Kante anpasst. Konkret würde dies bedeuten, dass nicht stur nach der benutzerdefinierten Struktur gesucht wird, sondern eine langsame Anpassung der Kantencharakteristik zugelassen wird und so eine längere Verfolgung des Musters möglich ist, welches das Objekt vom Hintergrund trennt.

In der 3D-Segmentierung hat sich bereits bei der Implementierung in Kapitel 7 gezeigt, dass vor allem im Raum nicht allein das Auffinden der Oberfläche die Problematik ausmacht. Vielmehr spielen hier Fragen der Realisierung einer einfachen und intuitiven Navigation im Raum eine zunehmende Bedeutung. In der Ebene ist diese Navigation durch die üblichen Mittel grafischer Benutzeroberflächen bereits gut gelöst, im Raum stellen sich aber neue Herausforderungen, die zusätzliche Hardware, aber vielleicht auch neue Paradigmen für die Interaktion mit den Daten erfordern.

9.3 Der größere Bogen

Eine neue Herausforderung wird in Zukunft sein, den Segmentierungsansatz für Volumendaten auf Videos zu übertragen. Existierende Ansätze betrachten lediglich Einzelbilder und versuchen die Segmentierung eines Bildes auf die darauf folgenden zu übertragen und die Objektgrenze dabei nicht zu verlieren. Auch in der Videoinhaltsanalyse werden heute ebenso wie bei der Kompression meist nur Folgen von Einzelbildern betrachtet. Denn der Mensch sieht in erster Linie ein Bild als Einheit. Ein Video wird dagegen als Folge von Bildern aufgefasst. Trotzdem ist jeder Grau- oder Farbwert eines Videos eigentlich im dreidimensionalen Raum (des Videos) definiert. Der Mensch neigt grundsätzlich dazu Multimediadaten in Ebenen zu projizieren, die orthogonal zur Zeitachse verlaufen. Dies ist selten statistisch oder mathematisch begründet, sondern folgt mehr den Sehgewohnheiten und der Wahrnehmung.

Im Raum (und nicht in der Ebene) sollte aus unserer Sicht die Analyse und Kompression und im speziellen die Segmentierung stattfinden. Dabei wäre besonders interessant

keine achsenparallelen Ebenen zu betrachten, sondern von vornherein alle Richtungen vorurteilsfrei gleich zu priorisieren. Es würde auch niemand auf die Idee kommen, dass die Pixelzeile eine zu betrachtende Einheit darstellt, die sich nach oben und unten mehr oder weniger gleichartig fortsetzt. Es ist intuitiv sofort klar, dass beide Dimensionen eines Bildes als Einheit zu sehen sind. Eine zeilenweise Betrachtung macht keinen Sinn, da der Ausrichtung der X- und Y-Achse keine Bedeutung zukommt. Ebenso könnte der Fotograph die Kamera neigen oder die abgebildeten Objekte könnten gedreht sein. Zeilen und Spalten bekämen durch eine Drehung inhaltlich eine ganz andere Bedeutung.

Die Zeitachse wird aber als fest vorgegeben angesehen. Bilder müssen nach dieser Anschauung gewissermaßen auf die Zeitachse "gestapelt" sein. Kaum jemand würde auf die Idee kommen einen anderen Schnitt durch das Video-Volumen zu betrachten.

Ob diese Festlegung auch der Natur der Daten gerecht wird, wurde bisher nur selten hinterfragt. Vielleicht könnte man bessere Kompressionsraten erzeugen oder animierte Objekte besser segmentieren, wenn man Videos nicht als zur Zeitachse parallelen Bildstapel auffassen würde, sondern die im Raum definierten Daten ohne Vorbedingung und Annahmen analysieren würde.

ANHANG

TESTMETHODE ZUM VERGLEICH VON SEGMENTIERUNGSVERFAHREN

A.1 Der Faktor *Benutzer* in der Evaluation

Im Rahmen dieser Arbeit soll ein neuer Vorschlag gemacht werden, wie sich unterschiedliche halbautomatische Segmentierungsverfahren objektiv und unabhängig von der Geschicklichkeit der Benutzer miteinander vergleichen lassen. Im Gegensatz zum Verfahren aus Kapitel 8 wird hier die Anzahl der Knoten festgehalten und die Güte der Segmentierung ermittelt, die ein Verfahren erzeugen kann. Im Folgenden wird lediglich auf das eigentliche Verfahren zur Evaluation eingegangen. Als Beispiel für einen zu testenden Segmentierungsalgorithmus findet der einfache Live-Wire Ansatz ohne Skalenanalyse Verwendung, der sich ausschließlich am Gradienten orientiert. Der eigentliche Vergleich unterschiedlicher Verfahren unter Einsatz der neuen Bewertungsstrategie erfolgt in Kapitel 8.

Ein analoges Problem existiert bei der selbständigen Segmentierung. Dabei sollen vollautomatisch erzeugte Umrisse eines Objektes mit den tatsächlichen verglichen werden.

Die Bewertung von Segmentierungsergebnissen vollautomatischer Verfahren ist insofern objektiver, als jeder Algorithmus einem Eingabebild eindeutig eine zugehörige

Segmentierung zuordnet. Dagegen werden halbautomatische Verfahren durch den Benutzer gesteuert. Das gleiche Verfahren kann unter verschiedenen Benutzern unterschiedlich gut abschneiden. Dabei hängt die Qualität der Segmentierung wesentlich davon ab, wie gut sich der Mensch an den Algorithmus anpasst, d. h. wie gut die Vorgaben der Person sind. Besonders problematisch wirkt sich aus, wenn der Entwickler eines neuen Algorithmus sein eigenes Verfahren testet.

Einige Autoren versuchen die Bewertung durch Probandentests zu objektivieren. Mortensen und Barret [MOR92], [BAR97] lassen acht Personen jeweils fünf Objekte mit zwei unterschiedlichen Segmentierungswerkzeugen ausschneiden. Obwohl der Aufwand mit 2x40 segmentierten Objekten insgesamt bereits groß ist, kann man bei dieser Zahl noch mit einer Varianz rechnen, die das Ergebnis signifikant beeinflussen könnte. Noch aufwendigere Tests wie z. B. zum Optimieren von Filterbreiten wären wegen der grossen Anzahl möglicher Parameterkombinationen mit Benutzertests nicht mehr zu leisten.

Abbildung A.1: Die weiße Objektgrenze (links) wurde vollständig durch den Benutzer definiert. Die schwarzen Punkte werden durch den Live-Wire Algorithmus verbunden. Das Ergebnis der Segmentierung zwischen den Punkten wird als überlagerte schwarze Linie (rechts) dargestellt.

A.2 Die Referenzsegmentierung

Es liegt daher nahe auch einen halbautomatischen Algorithmus ohne Zutun eines Benutzers zu evaluieren. Wie bei der vollautomatischen Segmentierung benötigt man eine optimale Referenzsegmentierung, in der ein Mensch Objekte eindeutig vom Hintergrund abgrenzt. Im optimalen Fall sollte der Benutzer pixelweise entscheiden, wo die Objektgrenze liegt. Unterschiedliche Benutzer werden vermutlich leicht unterschiedlichen Silhouetten erzeugen. Die zum heutigen Zeitpunkt vorliegenden Referenzsegmentierungen der MPEG4 Gruppe zeigen, dass die zur manuellen Segmentierung nötige Konzentration und Ausdauer eine große Herausforderung darstellen und nicht immer stringent durchgehalten werden. Aber auch im Bereich ausgefranster, faseriger oder halbdurchsichtiger Objektgrenzen ist auch bei gutem Willen eine eindeutige Abgrenzung nicht möglich. Da sich aber jeder getestete Algorithmus dem gleichen Problem gegenüber sieht, eine zumindest teilweise fragwürdige Objektgrenzen finden zu müssen, bleibt die Vergleichbarkeit trotzdem erhalten. Auch die Referenzsegmentierungen in dieser Arbeit wurden Pixel für Pixel durchgeführt. Der naheliegenden Verwendung eines Segmentierungswerkzeuges sollte man dringend widerstehen, da dadurch das verwendete Verfahren oder ein ähnliches bei der Evaluation im Vorteil wäre.

A.3 Problem der optimalen Verteilung der Stützstellen

Zum Test unterschiedlicher halbautomatischer Verfahren wird zuvor eine Anzahl von n Stützstellen vorgegeben. Im Abbildung A.1 wird hierfür ein Beispiel gezeigt. Die weiße Umrandung um die Person ist pixelweise vom Benutzer definiert. An den meisten Stellen sollte die gewählte Grenze intuitiv als die wahre Objektgrenze zu erkennen sein. Im Bereich der Haare gibt es einige Stellen, an denen das Bild unterschiedlichen Benutzer evtl. kleine Interpretationsspielräume lässt. Die schwarzen Punkte auf der Linie markieren sogenannte Stützstellen.

Jeder Algorithmus soll nun die n Stützstellen verteilen, die jeweils exakt auf der vorgegebenen Referenzsegmentierung liegen müssen. Im Prinzip simuliert man damit den

Benutzer, der Punkte auf der wahren Objektgrenze definieren würde. Alle Pixel zwischen je zwei Stützstellen werden mit dem zu testenden Segmentierungsverfahren berechnet. Je besser sich die vom jeweiligen Verfahren gefundene Grenze der tatsächlichen anpasst, desto besser hat sich der Algorithmus bewährt. Auf der rechten Seite von Abbildung A.1 ist das Ergebnis der Segmentierung zwischen den links abgebildeten Stützstellen zu sehen.

Abbildung A.2: Der unkorrigierte Live-Wire sucht sich entlang des Beines die stärkste Kante als Pfad, nämlich den Übergang von Licht zu Schatten. Tatsächlich liegt die echte Grenze des Objektes aber bereits selbst im Schatten.

Problematisch ist bei der Evaluation, wie die n Punkte auf der Referenzsilhouette verteilt werden sollen. Verteilt man diese zufällig, so entstehen Segmentierungen mit sehr unterschiedlicher Qualität. Der Algorithmus wird grundsätzlich dann besser, wenn sich die zur Verfügung stehenden Punkte auf die Stellen der Objektgrenze konzentrieren, an denen das Verfahren die Grenze nicht gut findet. Je enger die Stützstellen an Problemstellen zusammenrücken, desto kleiner werden die Fehlermöglichkeiten dazwischen. Unterschiedliche Algorithmen oder auch gleiche Verfahren mit verschiedenen Filterbreiten und Schwellwerten, können dann am besten objektiv miteinander verglichen werden, wenn für jeden Algorithmus die global optimale Verteilung der n Stützstellen

auf der Referenzsegmentierung gefunden wird. Mit anderen Worten: Ein Vergleich verschiedener Verfahren ist dann besonders fair, wenn die Verteilung der n Stützstellen für ein bestimmtes Verfahren keine Benachteiligung darstellt. Dies ist genau dann der Fall, wenn die optimale Verteilung gefunden wurde.

Die Vergrößerung in Abbildung A.2 macht deutlich, warum ein bestimmtes Verfahren manchmal schlecht mit der wahren Objektgrenze zurecht kommt. Im gezeigten Bild ist es für den Betrachter auf den ersten Blick unverständlich, warum die Segmentierung gerade an der kaum texturierten Fläche des linken Beines so schlecht abschneidet. In Abbildung A.1 kann man erkennen, dass gerade an dieser Stelle 1/3 aller Stützstellen aufgewendet wurde. Die Erklärung liegt darin, dass für den Menschen offensichtlich ist, dass auch der im Schatten liegende Teil des Beines mit zur Person gehört. Für einen nur auf Kanten basierenden Algorithmus kann zwischen dem Schatten und dem Hintergrund kaum noch unterschieden werden. Die stärkere, jedoch falsche Kante liegt in der Mitte zwischen Licht und Schatten, so dass der Live-Wire in dieser Region fast pixelweise auf die gewünschte Bahn gerückt werden muss. Schon hier deutet sich an, dass ein nicht nur auf Kanten basierendes Verfahren hier besser abschneiden könnte.

Bei Bildgrößen von heute 500.000 bis 4 Mio. Pixeln und in Zukunft noch höheren Auflösungen können auch die Objektgrenzen 10.000 oder mehr Pixel lang werden. Der Benutzer ist evtl. bereit bis zu 100 Stützstellen zu setzen. Dies führt zu einem bekannten kombinatorischen Problem von 100 Ziehungen (also Verteilung der Stützstellen) aus einer Grundgesamtheit von 10.000 Elementen (dem Umriss). Unter allen 10.000 über 100 Kombinationen ist die global optimale zu erwarten. Die vollständige Enumerierung ist offensichtlich wenig aussichtsreich:

z. Z. kursierende, großzügigste Schätzung für die Anzahl der Atome im Weltall: ~ 6e79
10.000 über 100: ~ 6e399

Durch Verwendung der dynamischen Programmierung kann die Komplexität in Bezug auf die Rechenzeit aber auf $n * m^2$ und auf $n * m$ für den Speicher reduziert werden, wenn

n: die Anzahl der Stützstellen und

m: die Anzahl der Pixel für die Umrandung

sind. Warum kann für die Berechnung des globalen Optimums die dynamische Programmierung verwendet werden?

A.4 Dynamische Programmierung zur Suche des globalen Optimums

Seien die Randpixel des Objektes mit der Folge p_n, für eine Anzahl n aufeinander folgender Punkte, bezeichnet. Offensichtlich beeinflusst die Verteilung von Stützstellen in einem Teil der Folge p_h bis p_i nicht die Kosten einer Verteilung von Stützstellen in einem disjunkten Teil der Folge zwischen p_j bis p_k, wobei gelten soll, dass

$$0 \leq h < i < j < k \qquad \text{(Ausdruck A.1a)}$$

Die Bedingung soll lediglich sicherstellen, dass zwei überlappungsfreie Bereiche in der Folge miteinander verglichen werden. Intuitiv bedeutet das, dass die Wahl der Stützstellen in einem Intervall die lokalen Kosten der Wahl der Stützstellen in einem anderen, überlappungsfreien Intervall, nicht beeinflussen kann. Dies ist insofern einleuchtend, als die Kosten allein durch die lokale Konfiguration der Pixel im Bild bestimmt sind (also ob z. B. ein ausreichend starker Gradient wirklich die Objektgrenze markiert). Die vorherige Wahl der Stützstellen hat auf das Bild keinerlei Einfluss. Man mag an dieser Stelle einwenden, dass die frühere Wahl der Stützstellen natürlich die Anzahl der verbleibenden beeinflusst. Doch dieser Einfluss wird schon durch die Kostenoptimierung der dynamischen Programmierung berücksichtigt.

Wie im nächsten Abschnitt erläutert wird, lässt sich die Anzahl m der Randpixel weiter reduzieren, so dass nur solche Verteilungen der Stützstellen verloren gehen, die zum gleichen oder zu sehr ähnlichen Qualitätsmaßen führen.

A.5 Erzeugen der manuellen Segmentierung

Am Anfang steht die zuvor erwähnte Definition einer benutzerdefinierten Umrandung des Objektes. Programm *SemiSegDP* von der CD bietet zum Festlegen dieser Definition einen interaktiven Modus, in dem der Benutzer die Objektgrenze als geschlossenen Polygonzug pixelgenau definieren kann.

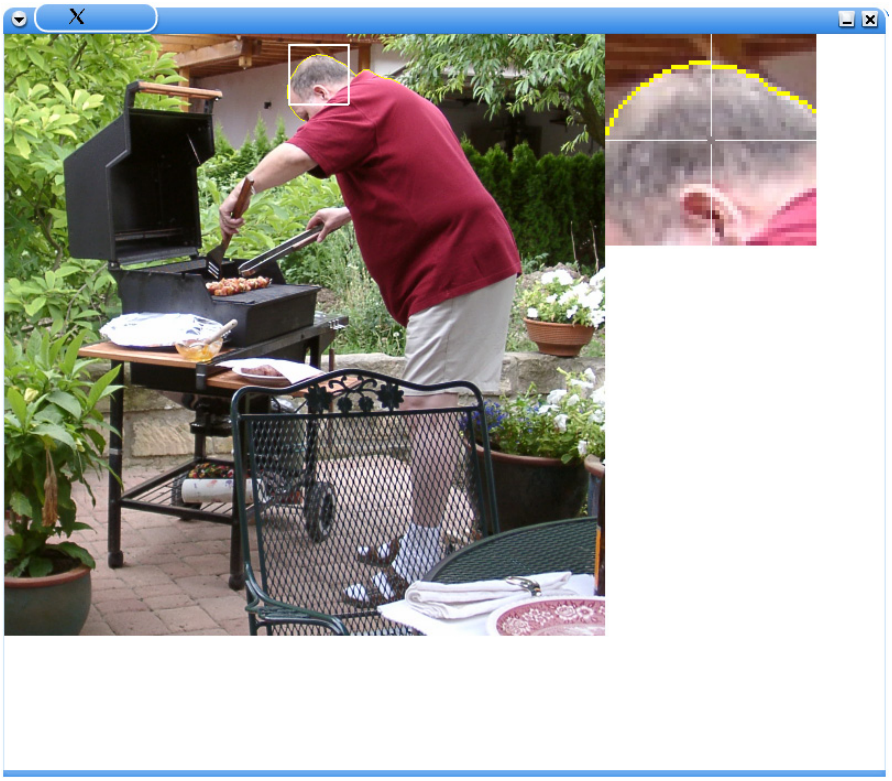

Abbildung A.3: Im interaktiven Modus von SemiSegDP muss der Benutzer zuerst eine aus menschlicher Sicht optimale Referenzsegmentierung erzeugen. An dieser lassen sich im nachfolgenden Schritt unterschiedliche Verfahren messen. Das Fadenkreuz in der Vergrößerung hilft bei der pixelgenauen Positionierung der Objektgrenze.

Der Polygonzug wird dann als lückenlose Folge von (x, y) Werten zur späteren Wiederverwendung in eine Datei geschrieben.

Im zweiten Schritt läuft die Anwendung in einem automatischen Evaluationsmodus. Dabei wird auf der Umrandung ein bestimmter Startpunkt S festgelegt. Von diesem Startpunkt ausgehend wird der für Live-Wire übliche Graph kürzester Wege zu jedem anderen Pixel des Bildes erstellt. Nun wird der Live-Wire ausgehend von S zu den benachbarten Pixeln auf der Segmentierung berechnet. Zwischen S und dem unmittelbar

benachbarten Pixel, hat der Live-Wire noch gar keine Entscheidungsmöglichkeit, da Start- und Zielpunkt die gesamte Trajektorie ausmachen. Je weiter sich P von S auf der Objektgrenze entfernt, desto größer werden die Spielräume und auch die Fehlermöglichkeiten des Segmentierungsverfahrens. Für jeden Schritt wird also der Weg berechnet, für den sich Live-Wire zwischen S und P entscheiden würde. Dieser Weg wird mit dem gespeicherten optimalen Weg zwischen S und P verglichen, indem der mittlere Abstand in Pixeln zwischen beiden Pfaden berechnet wird. Bis hierher gleicht das Verfahren noch dem aus Kapitel 8.

A.6 Abstand zweier Kurven

Der im Folgenden verwendete mittleren Abstandes zweier Kurven in Pixel ähnelt etwas dem Charakter der Hausdorff-Distanz, ist aber nicht das gleiche.

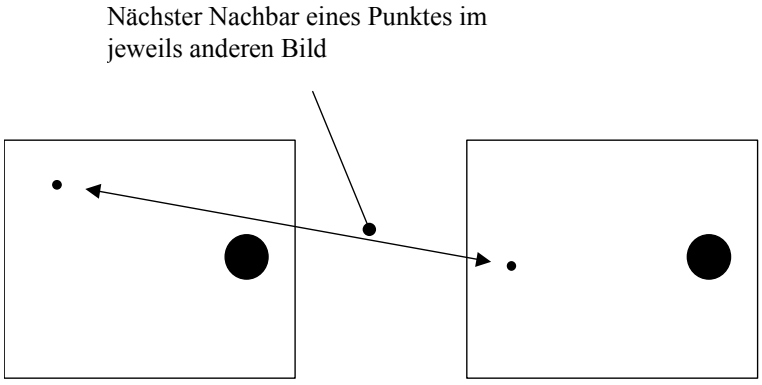

Abbildung A.4: Beide Bilder entsprechen sich bis auf den kleinen Punkt. Daher kann der Hausdorff Abstand beider Bilder nicht geringer als der Abstand dieser beiden Punkte werden.

Auch die Hausdorff-Distanz ist zwischen zwei schwarz-weiß Bildern definiert. Bei ihr wird für jeden Punkt P_1 eines Bildes im anderen Bild der jeweils nächstgelegene P_2 gesucht und Abstand $|P_1 - P_2|$ berechnet. Letztendlich zählt unter allen Abständen jedoch nur der größte, alle anderen fallen aus dem Maß heraus.

Es ist auch zu beachten, dass der bisher beschriebene Abstandsbegriff nicht kommutativ ist, d. h. $|P_1 - P_2| \neq |P_2 - P_1|$ und daher noch nicht die Bedingungen einer Metrik erfüllt. Zur Metrik wird die Berechnung erst durch

$$m = \max\left(|P_1 - P_2|, |P_2 - P_1|\right) \qquad \text{(Ausdruck A.1b)}$$

Je stärker zwei Bilder gegeneinander konvergieren, desto kleiner muss auch der maximale Abstand werden, der sich zwischen zwei beliebigen Punkten finden lässt. Man bemerkt, dass dieses Maß kaum dem menschlichen Empfinden des Abstandes zweier Bilder entsprechen kann, denn nur zwei konstant gewählte, jedoch ungleiche Punkte in beiden Bilder verhindern, dass diese über eine feste Grenze hinaus konvergieren können, weil dies von den konstant gewählten Punkten verhindert wird (siehe Abbildung A.4).

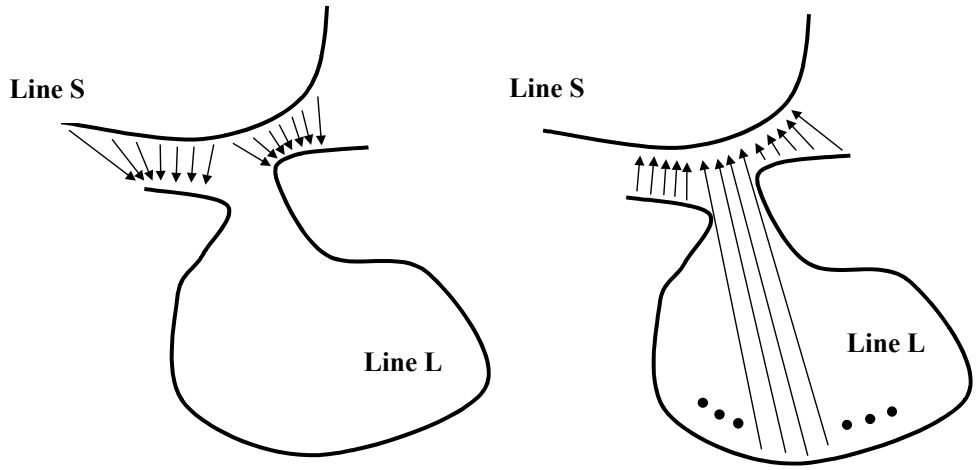

Abbildung A.5: Der mittlere Abstand der Linie S zu L ist kleiner als der Abstand zwischen L und S.

Zur Beurteilung der Qualität einer Segmentierung soll im Folgenden der mittlere Abstand d zweier Kurven S und L in Pixel gemessen werden.

$$S : A \to R^2 \text{ mit } A \in R^+$$
$$L : B \to R^2 \text{ mit } B \in R^+$$

$$d = \max\left(\int_{x \in A} \min_{y \in B} |S(x) - L(y)| dx, \int_{y \in A} \min_{x \in B} |L(y) - S(x)| dy \right) \qquad \text{(Ausdruck A.2)}$$

Ausdruck A.2 lässt sich auch prozedural beschreiben: Zuerst wird die gesamte Linie S punktweise abgelaufen. Für jeden Punkt wird der kürzeste Abstand zu einem Punkt der

Linie L gesucht. Über alle so ermittelten Abstände wird das arithmetische Mittel gebildet. In Abbildung A.5 ist das Verfahren symbolisiert. Von jedem Punkt auf der Linie S geht ein Pfeil zum nächsten Nachbarn auf L aus. Dabei ist die durchschnittliche Pfeillänge der mittlere Abstand von S zu L im Sinn des oben gewählten Begriffes. Abbildung A.5 zeigt auch, dass die umgekehrte Berechnung von L zu S offensichtlich zu einem anderen mittleren Abstand führt. Deshalb wird in Ausdruck A.2 das Maximum beider Varianten ermittelt.

In der praktischen Implementierung sollte die Abstandsberechnung sehr schnell erfolgen. Zur Erinnerung: Ein variabler Punkt P soll sich auf der optimalen Silhouette des Objektes von einem Startpunkt S wegbewegen.

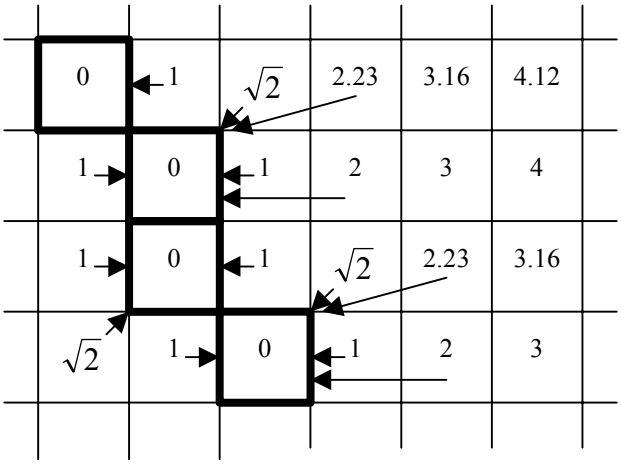

Abbildung A.6: Die stark umrandeten Felder kennzeichnen die Pixel des Objektsilhouette. Jeder Pixel versteht sich nicht als Grauwert, sondern als Cache mit dem Eintrag des kürzesten Abstands zum nächsten Pixel des Umrisses.

Für jedes Paar aus unterschiedlichen Punkten S und P soll getestet werden, wie weit der dazwischen liegende Live-Wire (oder die Segmentierung durch ein anderes Verfahren) von der Optimalline entfernt ist. Aufgrund der Vielzahl von Kombinationen darf für die Berechnung des Abstandes nur wenig Zeit aufgewendet werden. Um für jeden Punkt den Abstand zur Referenzkurve möglichst schnell zu ermitteln, wurde daher ein Cache mit der Größe des Bildes angelegt (siehe Abbildung A.6). Die Werte des Caches bezeichnen jeweils den Abstand zum nächstgelegenen Pixel des Objektrandes. Die Abstandsberechnung kann dann in O(n) erfolgen, wenn n die Anzahl der Pixel des Live-Wires bezeichnet.

Leider muss wie in Ausdruck A.2 sowohl der Abstand von S nach L, als auch der umgekehrte Abstand berechnet werden. Somit werden auch zwei Caches benötigt. Der zur Referenzkurve gehörende muss nur einmal berechnet werden und kann während der ganzen weiteren Berechnung genutzt werden. Dagegen ist der Cache des Live-Wires insofern ein Problem, als er für jeden Punkt P auf der Silhouette neu berechnet werden muss und seine Initialisierung im Vergleich zu den wenigen Werten, die tatsächliche abgefragt werden, viel zu aufwendig ist. In der aktuellen Implementierung wurden daher alle Cache-Werte auf (-1) gesetzt und der Wert erst bei Bedarf berechnet. Erst bei einem erneuten Lookup kann Rechenzeit gespart werden.

Abbildung A.7: Bild 1 (links oben) bis Bild 4 (rechts unten)
zeigt unterschiedliche Stadien des Segmentierungsverlaufes

Je weiter sich P und S auf der Optimallinie voneinander entfernen, desto größer wird die Wahrscheinlichkeit, dass sich der Live-Wire einen kürzeren, als den optimalen Weg sucht. Interessanterweise wächst der Fehler mit zunehmender Entfernung nicht stetig, sondern weist fast immer eine Sprungstelle auf. Vor einer solchen Sprungstelle laufen Live-Wire und optimale Objektgrenzen den gleichen oder einen ähnlichen Pfad ab. Ab einer bestimmten Stelle "schnappt" der von Live-Wire gefundenen Pfad auf einen anderen Pfad und erzeugt so eine deutliche Sprungstelle in der Kostenfunktion. Abbildung A.7 zeigt den Sachverhalt: In Bild 1 läuft der Live-Wire vom schwarzen Startpunkt bis

zum Endpunkt des Drahtzaunes. Wie zu erwarten, verläuft der optimale Weg entlang der Baumkronen im Hintergrund. Im zweiten Bild wurde der Endpunkt P auf die weiße Optimallinie verlegt. Man erkennt, dass die Segmentierung die Optimallinie relativ zuverlässig überdeckt. Im dritten Bild wurde P vom Evaluationsalgorithmus weiter auf der Ideallinie verschoben. Tatsächlich ist der mittlere Fehler im Vergleich zu Bild 2 sogar noch etwas gesunken, da in Summe noch mehr Pixel auf der Ideallinie liegen und so der durchschnittliche Fehler verkleinert werden konnte. Die Fehlerfunktion ist also weder stetig noch monoton.

Erst in Bild 4 wurde P über den nächsten, schwarz markieren Punkt hinaus verschoben. In diesem Moment schnappt Live-Wire nach unten und sucht sich einen für die Segmentierung ungeeigneten Weg durch das Gesicht der Person. Die automatische Evaluation geht dann an den letzten optimalen Punkt zurück, und speichert diesen als den nächsten sogenannten neuen Seed-Punkt. Nach dem gleichen Verfahren wird auch der Rest des Objektes abgelaufen. Dabei entstehen immer an den Stellen, an denen die Fehlerfunktion einen Schwellwert überschreitet ein neuer Seed-Punkt, der wieder abgespeichert wird. (Bemerkung: In Abbildung A.1 (links) sind diese Punkte zur besseren Erkennbarkeit der Stellen, an denen der Live-Wire einen starken Fehler macht, bereits eingezeichnet).

Nachdem alle Punkte ermittelt wurden, entstand das Bild A.1 (rechts). In dieser Abbildung wurde ein mittlerer Fehler von 2.0 Pixeln Abstand zur Ideallinie toleriert. Dieser Schwellwert ist so zu interpretieren, dass kleinere Fehler als nicht signifikante Abweichung angesehen werden. Wenn das Testbild nicht gerade synthetisch erzeugt wurde, lässt sich ohnehin nicht mit einer Subpixelgenauigkeit begründen, warum die Grenze genau an der benutzerdefinierten Stelle liegen muss. Natürlich kann man den Wert je nach Genauigkeit der eigenen Segmentierung beliebig verkleinern. Eine kleinere Toleranz erzeugt zwangsläufig mehr Seed-Punkte.

An einigen Stellen wie z. B. im Bereich des Kopfes sind trotz des Zaunes im Hintergrund erstaunlich wenige Punkte nötig. An anderen Stellen wie dem linken Bein wird trotz einer scheinbar klaren Kante eine Vielzahl von Punkten benötigt.

Das Auffinden der n Seed-Punkte in Abhängigkeit der Strenge des tolerierten Fehlers dient der Evaluation eigentlich nur zur Reduktion der Komplexität. Zuletzt soll nämlich aus den verbleibenden n Punkten eine kleinere optimale Auswahl getroffen werden. Ebenso könnte man für die Auswahl theoretisch auch alle Randpixel zulassen. Der große Aufwand hierfür lohnt sich jedoch kaum. Denn wie oben beschrieben, sind die n möglichen Stützstellen ja per Definition gerade so gewählt, dass die dazwischen liegende Segmentierung besonders gut zur Referenzkurve passt. Demnach gibt es auch keinen Grund, gerade zwischen zwei der oben ermittelten Punkte weitere einzufügen. Zur Verdeutlichung soll folgendes Beispiel dienen: Zwischen zwei Punkte P_1 und P_2 hat der Live-Wire exakt die Optimallinie getroffen. In vielen praktischen Beispielen kommt dies zumindest über kurze Strecken tatsächlich vor. Wenn ohnehin keine Abweichung stattfindet, kann die Segmentierung auch nicht durch Hinzunahme eines zweiteren Punktes auf der Kurve zwischen P_1 und P_2 verbessert werden. Hat man dies akzeptiert, so macht auch die weitere Relaxierung der Anforderung (z. B. auf einen erlaubten mittleren Fehler von 1-2 Pixeln) durchaus Sinn, vor allem, wenn man bedenkt, dass auch die benutzerdefinierte Segmentierung nicht beliebig genau ist.

Nun wurden die n wesentlichen Kandidaten für Seed-Punkte (auch als Stützstellen bezeichnet) gefunden. Auch ist gesichert, dass der Live-Wire zwischen allen Punkten einen maximalen mittleren Fehler von (hier) 2.0 Pixeln aufweisen kann. Die nächste Aufgabe besteht nun darin, eine beliebige aber feste Anzahl k < n zu suchen, die unter allen möglichen k Ziehungen aus n Möglichkeiten, zur optimalen Segmentierung führt.
Um dies zu erreichen wird eine Übergangsmatrix der Größe n x n aufgebaut. Ein Beispiel einer solchen Matrix M ist in Tabelle A.1 angegeben. Das Bild (Abbildung A.8) zur den Daten entspricht dem bisher verwendeten. Allerdings wurde die Anzahl der Stützstellen reduziert um keine unübersichtlich große Datenmenge zu erhalten. Bei einem maximal erlaubten mittleren Fehler von acht Pixeln, erzeugte das Verfahren nur noch zwölf Stützstellen. Wie zu erwarten, stehen an jeder Stelle (i, j) der Matrix die Kosten um von Punkt P_i zu P_j zu gelangen (in der Abbildung wurde jeder mögliche Weg zwischen je zwei Knoten schwarz eingezeichnet). Wie in Abbildung A.8 zu sehen ist, findet der Live-Wire den optimalen Pfad meist in einer lokalen Umgebung am si-

chersten. Daher sind auch die Einträge in der Matrix um die Diagonale eher mit kleinen Werten besetzt. Wie in der Abbildung ebenfalls zu sehen ist, weicht der Pfad mit zunehmender Entfernung zwischen Start- und Endpunkt meist schlagartig vom gewünschten Pfad ab. Daher werden die weiter von der Diagonalen entfernten Werte typischerweise sprunghaft größer.

> Bemerkung: Das sprunghafte Verhalten ist bei genauerer Analyse nicht verwunderlich. Trägt man ein Kantenbild (insbesondere die zweite Ableitung) als Höhengraf im dreidimensionalen Raum ab, so enthält dieser im Bereich von Kanten Höhenzüge, ähnlich eines Gebirges. Da bei Live-Wire Kanten in eine Kostenfunktion einfließen, werden die Höhenzüge als Täler interpretiert, weil ein Verfolgen der Kante positiv bewertet wird. Aufgrund der vielen Kanten die vor allem texturierte Bilder besitzen, gibt es immer eine Vielzahl von Tälern, die mögliche Wege für die Segmentierung bieten. Entfernen sich zwei Punkte voneinander, so kann die Wahrscheinlichkeit zunehmen, dass plötzlich an einer Gabelung ein anderer Graben geringere Kosten verursacht. Kanten bilden fast immer ein Netz von unterschiedlich tiefen Gräben, so dass es kaum zu einer stetigen Verschiebung der Kosten kommen kann, weil der optimale Pfad nur an Gabelungen mehrerer Täler seinen Verlauf ändern wird.

Es fällt auf, dass die untere Dreiecksmatrix mit (-1) besetzt ist. Grundsätzlich gilt $M^T = M$, d. h. die obere Dreiecksmatrix entspricht der unteren, weil die Richtung auf einem festen Pfad im Bild nicht in die Kosten einfließt. Bei der Segmentierung macht es aber keinen Sinn die Silhouette des Objektes ab einer bestimmten Stelle plötzlich in die entgegengesetzte Richtung zu verfolgen. Hat man den Stützknoten P_j erreicht, so macht nur noch die Auswahl zwischen P_{j+1} und dem letzten Knoten P_n Sinn. Ein Zurückgehen wird daher mit unendlich hohen Kosten bewertet - hier mit (-1) symbolisiert. In der Implementierung wurde die dynamische Programmierung gleich so angepasst, dass immer nur nachfolgende Knoten in Betracht gezogen werden und damit die Komplexität und der Speicheraufwand halbiert wurde.

Rein rechnerisch ergibt sich für alle Diagonalelemente eine Null, da das Verbleiben in der gleichen Stützstelle keine Kosten verursachen kann.

	zu 1	zu 2	zu 3	zu 4	zu 5	zu 6	zu 7	zu 8	zu 9	zu 10	zu 11	zu 12
von 1	0	5,586	16,97	14,14	23,75	19,16	39,89	42,08	55,21	138,4	225,5	278,7
von 2	-1	0	3,958	14,72	24,7	19,47	40,85	43	56,25	140,6	180,4	268,3
von 3	-1	-1	0	1,861	26,03	19,4	42,6	44,66	58,21	144,8	185,3	238
von 4	-1	-1	-1	0	1,418	12,14	44,23	46,37	60,79	144,2	159,6	166,7
von 5	-1	-1	-1	-1	0	0,691	49,37	50,88	65,63	62,7	144,9	162,3
von 6	-1	-1	-1	-1	-1	0	3,093	32,62	69,28	61,05	56,77	58,67
von 7	-1	-1	-1	-1	-1	-1	0	3,01	35,08	62,56	56,8	59,06
von 8	-1	-1	-1	-1	-1	-1	-1	0	3,633	71,53	63,5	63,65
von 9	-1	-1	-1	-1	-1	-1	-1	-1	0	5,437	20,03	43,57
von 10	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	2,789	49,7
von 11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	6,457
von 12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Tabelle A.1: Jedes Element der Matrix M stellt die Kosten dar,
die der Live-Wire von Knoten i zu Knoten j erzeugt.

Die global optimale Lösung wäre dann überhaupt keinen Knoten bzw. immer den ersten zu wählen. Um die triviale Lösung auszuschließen, muss bei der späteren Berechnung des Optimums festgelegt werden, dass der letzte Knoten auf jeden Fall erreicht werden muss.

Damit ist auch bei schlechter Wegewahl zumindest ein geschlossener Pfad gewährleistet. Nun könnte der Algorithmus immer noch bis zuletzt mit Kosten 0 im Knoten 1 verharren, müsste dann aber zum "Preis" von 278,7 (Element [1,12]) den teuren Weg von 1 zu 12 wählen, so dass sich bessere Lösungen finden lassen.
Auf weitere Elemente kann man die Prüfung aber nicht ausdehnen, da der Live-Wire beim Überspringen eines Knoten, als z. B. für (i, i+2) gleich einen ganz anderen Weg wählen könnte und Maximalkosten nur zwischen zwei benachbarten Knoten definiert wurden.

A.7 Suche der optimalen Knotenmenge

Die zuletzt folgende Suche nach einer optimalen Auswahl von k Knoten aus den zur Verfügung stehenden n Stützstellen kann nun mit Hilfe der Übergangsmatrix mit dem

normalen dynamischen Programmierungsansatz geleistet werden, wie er auch schon in Kapitel 6 beschrieben wurde.

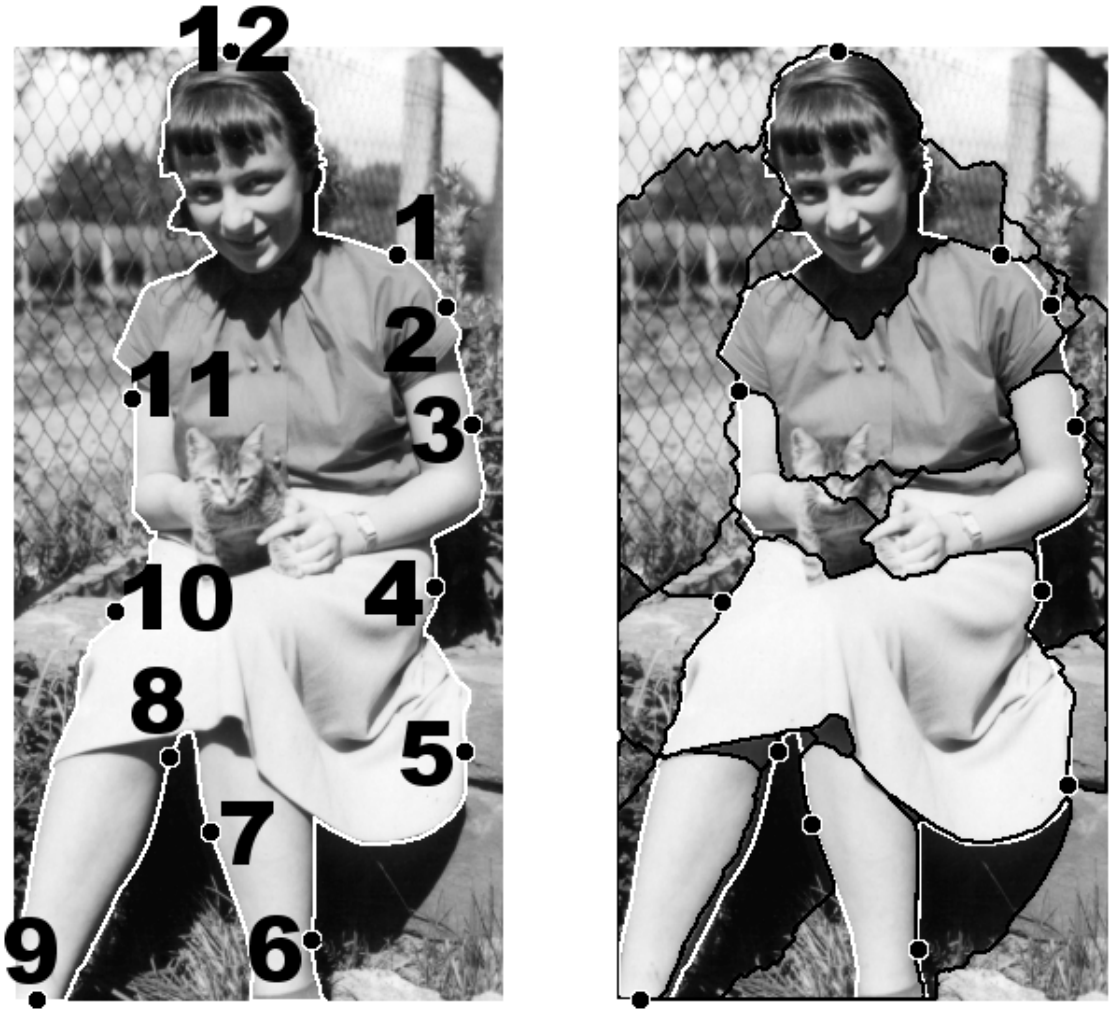

Abbildung A.8: Links wurden lediglich zwölf Stützstellen um das Objekt herum festgelegt, um Tabelle A.1 übersichtlich zu halten. Im rechten Bild wurde der Live-Wire von jedem Knoten zu jedem anderen eingezeichnet.

In allen bisher abgebildeten Beispielen war die Anzahl der Stützstellen relativ gering. Dies ist lediglich darin begründet, dass in den Abbildungen keine unübersichtliche Menge von Punkten dargestellt werden soll. Für die eigentlich Bewertung zweier Algorithmen soll aber ein mittlerer Fehler von weniger als zwei Pixeln postuliert werden, so dass zwangsläufig viel mehr Stützstellen erzeugt werden.

Mit dem Wissen, dass die maximal tolerierte Abweichung acht Pixel betragen darf, kann man die Matrix einer einfachen Plausibilitätsprüfung unterziehen. Jedes Element (i, i+1) darf maximale Kosten von acht enthalten.

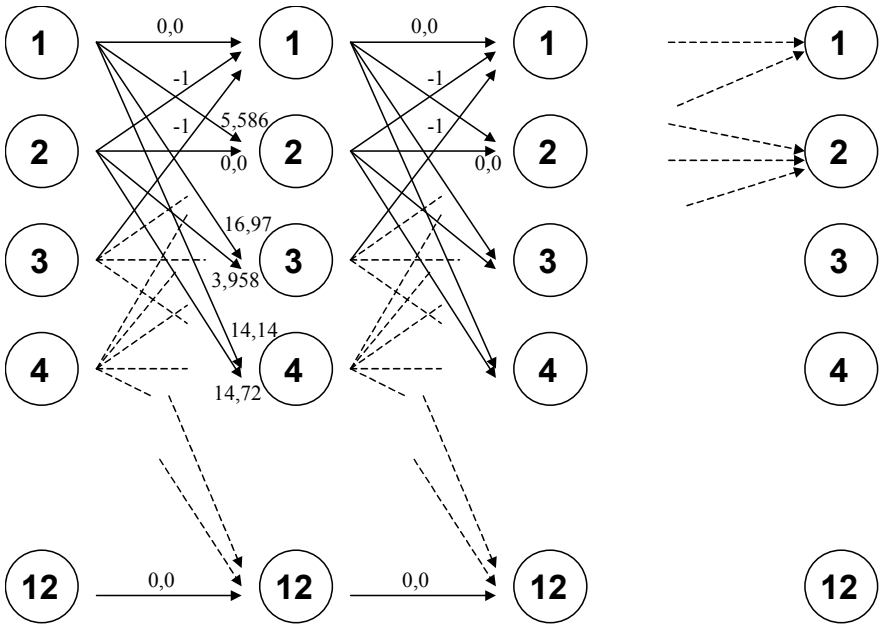

Abbildung A.9: Suche nach einer optimalen Auswahl von k Knoten aus einer Menge von 12 mit Hilfe der dynamischen Programmierung

Abbildung A.9 zeigt noch einmal den bereits in Kapitel 6 beschriebenen Algorithmus zur dynamischen Programmierung. Die Anzahl der Knoten pro Spalte entspricht der Anzahl verfügbarer Stützstellen, die Anzahl der Spalten selbst entspricht der Auswahl von k optimalen Knoten.

In der Regel wird k << n gelten, da der Benutzer die Segmentierung nur dann als sinnvoll empfindet, wenn er zwischen zwei Knoten ausreichend viele Pixel auslassen kann. Weiter oben wurde bereits erwähnt, dass aus der Reihe der Stützstellen nur fortlaufend weitere entnommen werden, da die Segmentierung ihren Weg an keiner Stelle umkehren darf. Dies wurde auch durch die auf unendlich gesetzte untere Dreiecksmatrix berücksichtigt.

Da unendlich große Kosten nie gewählt werden, kann man sich den Sachverhalt in der Optimierung gleich zunutze machen. In Abbildung A.10 auf der rechten Seite sind bereits die Knoten und Übergänge entfernt, die ohne Zurücklaufen nicht erreichbar sind. Der Graph ist dadurch im Vergleich zur linken, nicht optimierten Seite deutlich dünner besetzt. In Abbildung A.11 wurde noch eine zusätzliche Erweiterung eingeführt. In der

Praxis kommt es niemals vor, dass eine Stützstelle mehrfach erwählt wird. Dies ist darin begründet, dass dadurch für den restlichen Weg weniger Stützstellen zur Verfügung stehen.

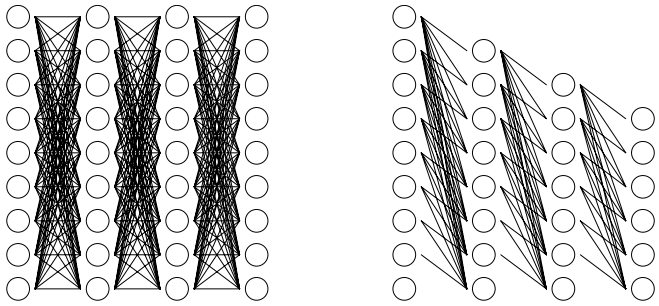

Abbildung A.10: Fügt man dem Optimierungsproblem (links) die Nebenbedingung hinzu, dass die Segmentierung einen bereits gefundenen Pfad nicht zurücklaufen darf, so reduziert sich das Problem wesentlich (rechts).

Wäre es möglich diese einfach auszulassen, so hätte der ganz am Anfang beschriebene Algorithmus zum Auffinden der Stützstellen diese erst gar nicht erzeugt. Ein Verbleiben im Graphen auf dem gleichen Knoten macht also kaum Sinn. Der Algorithmus wird wenigstens den nächsten oder einen weiter entfernten Knoten wählen. Daher kommen die untersten Knoten im Graphen nicht in Frage. Hierzu ein Beispiel: Würde man bei der ersten Entscheidung gleich den letzten Knoten wählen, so müsste die weitere Wahl ausschließlich auf den letzten Knoten fallen, denn die Umkehr der Segmentierungsrichtung macht aus den besprochenen Gründen keinen Sinn. Überträgt man diese Überlegung rekursiv auf den letzten Knoten, so ist klar, dass in der vorletzten Stufe auch maximal der vorletzte Knoten gewählt werden darf, usw.

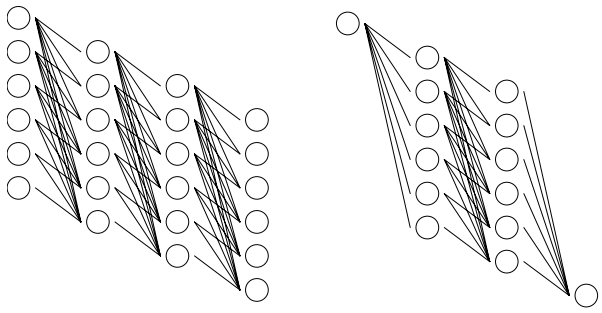

Abbildung A.11: Schließt man aus, dass ein Seed-Point nicht zweimal gewählt werden darf, so können die untersten Knoten zu Anfang nicht gewählt werden. Als weitere Nebenbedingung wurde festgelegt, dass der letzte Seed-Point in jedem Fall erreicht werden muss (rechts). Der Graph dünnt sich jeweils weiter aus.

Dadurch entsteht die rautenförmige Struktur in Abbildung A.11 (links). Alle bisherigen Bedingungen verfolgten nur den Zweck den Graph zu vereinfachen. Die letzte Bedingung ist aber eine notwendige (siehe den rechten Teil der Abbildung). Hier gibt es anfangs nur die Möglichkeit den ersten Knoten zu wählen, am Ende wird die Wahl des letzten Knoten erzwungen. Ansonsten könnte der optimale Weg bei einer Auswahl von vier aus neun Knoten mit Knoten 3 beginnen und mit Knoten 6 enden. In der Regel ist aber nur eine geschlossene Silhouette sinnvoll. Der Vollständigkeit wegen soll erwähnt werden, dass im Abbildung A.11 o. B. d. A. der Pfad von Knoten 1 zu Knoten 12 verlaufen sollte. Natürlich kann die Reihenfolge der Knoten auf der Silhouette auch um einen oder mehrere Knoten rotiert werden.

REFERENZEN

[AHM74] N. Ahmed, T. Natarajan, K. R. Rao, "Discrete Cosine Transform", *IEEE Transations on Computers*, C-23:90-3, January 1974

[ALL92] A. Gershoe, R. M. Fray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Bosten, Dordrecht, Londen, 1992

[BAC96] K. Backhaus, B. Erichson, *Multivariate Analysemethoden*, Spinger, 1996

[BAR88] M. F. Barnsley, *Fractals everywhere*, Academic Press, Boston, 1988

[BAR93] M. F. Barnsley, L. P. Hurd, *Fractal image compression*, A K Peters, Wellesley (MA), USA, 1993

[BAR96] M. F. Barnsley, W. Gisbert, *Bildkompression mit Fraktalen*, Vieweg, Braunschweig, 1996

[BAR97] W. A. Barrett, E. N. Mortensen, "Interactive Live-Wire Boundary Extraction", *Medical Image Analysis* (Elsevier), vol. 1, no. 4, pp. 331-341, September 1997

[BER00] A. Berman, A. Dadourian, P. Vlahos, "Method for removing from an image the background surrounding a selected object", *U.S. Patent 6,134,345 and 6,134,346*, year 2000

[BHA97] V. Bhaskaran, K. Konstantinides, *Image Video Compression Standards - Algorithms and Architectures*, Kluwer Academic Publishers, Boston, Dordrecht, London, 1997

[BLA98] A. Blake, M. Isard, *Active contours*, Springer Verlag, Berlin, Heidelberg, 1998

[BOE94] W. Boehme, H. Prautzsch, *Geometric Concepts for Geometric Design*, A K Peters, 1994

[BOM90] M. Bomans, K. H. Höhne, U. Tiede, M. Riemer, "3D-Segmentation of MR-Images of the Head for 3D-Display", *IEEE Transactions on Medical Imaging*, vol. MI-9, no. 2, pp. 177-183, 1990.

[BUR84] P. J. Burt, *The Pyramid as a Structure for Efficient Computation*, Springer Verlag, New York, 1984

[CHA93] T. Chang, C. C. J. Kuo, "Texture Analysis and Classification with Tree-structured Wavelet Transform", *IEEE Transactions on Image Processing*, vol. 2, no. 4, pp. 429-441, 1993

[CHE01.1] H. Chen, J. Hesser, and R. Männer, "Volume Rendering of Deformed Objects Combined with Algorithmic Optimization", *Proceedings of GRAPHICON 2001*, Nizhny Novgorod, Russia, September 2001

[CHE01.2] H. Chen, J. Hesser, and R. Männer. "Fast Free-Form Volume Deformation Using Inverse-Ray-Deformation", *Proceedings of VIIP 2001*, Marbella, Spain, September 2001

[CHE01.3] H. Chen, J. Hesser, B. Vettermann, R. Männer, "An Adaptive Distance-coding Based Volume Rendering Accelerator", *Proceedings of the 1st International Game Technology Conference*, Hongkong, January 2001

[CHI74] Y. P. Chien, K. S. Fu, "A Decision Function Method for Boundary Detection", *Computer Graphics and Image Processing*, pp. 125-140, 1974

[CHU01] Y. Chuang, B. Curless, D. Salesin, "A Bayesian Approach to Digital Matting", *IEEE Computer Vision and Pattern Recognition 2001*, vol. II, pp. 264-271, Kauai, Hawaii, December 2001

[CHU02] Y. Chuang, A. Agarwala, B. Curless, "Video matting of complex scenes", *ACM SIGGRAPH*, session "Images and video", pp. 243-248, San Antonio, Texas, 2002

[EFF98] W. Effelsberg, R. Steinmetz, *Video Compression Techniques*, dpunkt Verlag, Heidelberg, 1998

[FAR02] D. Farin, T. Haenselmann, S. Kopf, G. Kühne, W. Effelsberg, "Segmentation and Classification of Moving Video Objects", *Handbook of Video Databases*, CRC Press, 2002

[FIS73] M. A. Fischler, R. A. Elschlager, "The representation and matching of pictorial structures", *IEEE Transactions on Computers*, vol. C-22, pp. 67-92, 1973

[FOL90] J. Foley, A. van Dam et al, *Computer Graphics – Principles and Practice* (second edition), Addison-Wesley, 1990

[FOR83] O. Forster, *Analysis I - Differential- und Integralrechnung einer Veränderlichen*, 4. Auflage, Vieweg Verlag, 1983

[GAB46] D. Gabor, "Theory of Communication", *J. Inst. Electr. Engrg.*, 93, pp. 429-457, 1946

[GUT50] L. Guttmann, "The basis of scalogramm analysis", S. A. Stouffer et al. *Studies on Social Psychology in World War II*, vol. IV, Princeton University Press, Princeton, New York

[GUT97] M. Gutzmann, R. Scholz, "Wavelets - Grundlagen und Anwendung in der Bildkompression", Friedrich-Schiller-Universität Jena, Lehrstuhl für Rechnerarchitektur und –kommunikation, *TB-97-BR-3,11*, 1997

[HAE00] T. Haenselmann, C. Schremmer, W. Effelsberg, "Wavelet-based Semi-automatic Segmentation of Image Objects", *Signal and Image Processing* (SIP), pp. 387-392, Las Vegas, Nevada, USA, November 2000

[HAE01] T. Haenselmann, "MOTIONWAVE - A Tool for Wavelet-Based Video Transformation and Visualization", *IEEE Workshop on Multimedia Signal Processing* (SIP), Cannes, France, October 2001

[HAE02] T. Haenselmann, W. Effelsberg, "Texture Resynthesize using Principle Component Analysis", *SPIE Human Vision and Electronic Imaging VII*, San Jose (CA), USA, January 2002

[HAE03] T. Haenselmann, W. Effelsberg, "Wavelet based semi-automatic live-wire segmentation", *SPIE Human Vision and Electronic Imaging VII*, San Jose (CA), USA, January 2003

[HAE96] T. Haenselmann, *Raytracing - Grundlagen, Implementierung, Praxis,* Addison-Wesley Publishing Company, Bonn, Paris, New York, 1996

[HAR89] H. Harashima, K. Aizawa, T. Saito, "Modelbased Analysis Synthesis Coding of Video-Telephone Images – conception and basic study of intelligent image coding", *Transactions IEICE*, vol. E72, no. 5, pp. 452-458, 1989

[HAR92] J. Hartung, B. Elpelt, *Multivariate Statistik*, 4. Auflage, R. Oldenbourg Verlag, München, Wien, 1992

[HEI79] J. Heinhold, K. Gaede, *Ingenieur-Statistik*, Oldenbourg Verlag, München, 1979

[HES98] J. Hesser, C. Poliwoda, *Volume Visualization. Computer Vision Handbook*, 1998

[HES99] J. Hesser, A. Kugel, H. Singpiel, B. Vettermann, R. Männer, "Volume Rendering FPGA Hardware", *11th Symposium on Computer Architecture and High Performance Computation*, SBAC 99, Natal, Brazil, October 1999

[HOE90] K. H. Höhne, W. A. Hanson, "Interactive 3D-Segmentation of MRI and CT volumes using morphological operations", *Journal of Computer Assisted Tomography*, vol. 16, no. 2, pp. 285-294, 1992.

[HU62] M-K. Hu, " Visual pattern recognition by moment invariants ", *IRE Transactions on Information Theory*, IT-8, pp. 179-187, 1962

[HUB98] B. B. Hubbard, *The World According to Wavelets*, second edition, A K Peters, Massachusetts, USA, 1998

[ISO10918] DIS (Draft International Standard), 10918 Part 1, ISO - International Organization for Standardization

[JAE97] B. Jähne, *Digitale Bildverarbeitung*, 4. Auflage, S. 482ff, Springer Verlag, Heidelberg, 1997

[JAI98] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989

[KAI81] A. K. Jain, "Advances in mathematical models for image processing", *Proceedings of the IEEE*, vol. 69, no. 5, pp. 502-528, May 1981

[KIR90] M. Kirby, L. Sirovich, "Application of the karhunen-loeve procedure for the characterization of human faces", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, 1990

[KUE97] G. Kühne, C. Poliwoda, "Interactive segmentation and visualization of volume data sets", *Proceedings IEEE Visualization 97*, Late Breaking Hot Topics, pp. 9-12, Phoenix (AZ), USA, 1997

[KUH01] C. Kuhmünch, G. Kühne, C. Schremmer, T. Haenselmann, "A Video-Scaling Algorithm Based on Human Perception for Spatio-Temporal Stimuli", *Proceedings of SPIE, Multimedia Computing and Networking*, pp. 13-24, San Jose (CA), USA, January 2001

[KUN92] A. Kundu, J. L. Chen, "Texture classification using qmf bank-based subband decomposition CVGIP", *Graphical Models and Image Processing*, vol. 54, no. 5, pp. 369-384, 1992

[KUR93] S. Kuriyama, "Surface Generation from an Irregular Network of Parametric Curves", *Modelling in Computer Graphics*, IFIP Series on Computer Graphics, pp. 256-274, 1993

[LAN95] A. Lanitis, C. J. Taylor, T. F. Cootes, "A unified approach to coding and interpreting face images", *Proceedings of the 5th International Confer-*

ence on Computer Vision, pp. 368-373, 1995.

[LOO98] C. T. Loop, T. D. DeRose, "A Multisided Generalization of Bézier Surfaces", *ACM Transactions on Graphics* (TOG), 8(3): pp 204-234, July 1989

[LOO90] C. T. Loop, T. D. DeRose, "Generalized B-spline Surfaces of arbitrary Topology", *Computer Graphics (SIGGRAPH)*, ACM Press, 24(4): pp. 347-356, 1990

[LOU98] A. K. Louis, P. Maaß, A. Pieder, *Wavelets - Theorie und Anwendungen*, Teubner Verlag, Stuttgart, 1998

[MAR82] D. Marr, **Vision**, W.H. Freeman and Co., San Francisco, 1982

[MES89] R. Mester, *Regionenorientierte Bildsegmentierung unter Verwendung stochastischer Bildmodelle*, VDI Verlag, Düsseldorf, 1989

[MEY97] K. Meyberg, P. Vachenauer, *Höhere Mathematik 1*, Springer Verlag, Berlin, Heidelberg, New York, 1997

[MIS93] Y. Mishima, "Soft edge chroma-key generation based upon exoctahedral color space", U.S. patent 5,355,174, year 1993

[MOR76] D. F. Morrison, *Multivariate Statistical Methods*, second edition, McGraw-Hill Book Company, New York, 1976

[MOR92] E. N. Mortensen, B. S. Morse, W. A. Barrett, "Adaptive Boundary Detection Using 'Live-Wire' Two-Dimensional Dynamic Programming", *IEEE Proceedings of Computers in Cardiology* (CIC '92), pp. 635-638, Durham (NC), October 1992

[MOR95] E. N. Mortensen, W. A. Barrett, "Intelligent Scissors for Image Composition", *Computer Graphics* (SIGGRAPH '95), pp. 191-198, Los Angeles (CA), USA, August 1995

[MOR99] E. N. Mortensen, W. A. Barrett, "Tobboggan-Based Intelligent Scissors with a Four Parameter Edge Model", *Proc. IEEE: Computer Vision and Pattern Recognition* (CVPR '99), vol. II, pp. 452-458, Fort Collins (CO), USA, June 1999

[NIS88] B. Niss, Prediction of AC Coefficients from the DC Values, ISO/IEC - International Organization for Standardization JTC1/SC2/WG8 N745, May 1988

[OLS97] O. F. Olsen, "Multiscale watershed segmentation", J. Sporring et al., *Gaussian Scale-Space Theory*, Kluwer Academic Publishers, Dordrecht, 1997

[PEN93] W. Pennebaker, J. Mitchell, *JPEG - Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993

[PRE02] W. H. Press, *Numerical recipes in C++ : the art of scientific computing,* 2nd edition, Cambridge University Press, 2002

[PUT78] M. Puterman, "Dynamic Programming and Its Applications", *International Conference on Dynamic Programming and Its Applications 1977*, Vancouver, British Columbia, Proceedings by Academic Press, 1978

[RIN00] H. Rinne*, Statistische Analyse Multivariater Daten*, R. Oldenourg Verlag, München, Wien, 2000

[RAY90] S. P. Raya, J. K. Udupa, "Low-level segmentation of 3-D magnetic resonance brain images: A rule-based system", *IEEE Transactions on Medical Imaging*, vol. MI-9, no. 3, pp. 327-337, 1990

[ROS71] A. Rosenfeld, M. Thurston, "Edge and curve detection for visual scene analysis", *IEEE Transactions on Computers*, C-29, 1971

[ROS82] A. Rosenfeld, A. Kak, *Digital picture processing*, 2nd edition, vol. I/II, Academic Press, Orlando, 1982

[RUZ00] M. A. Ruzon, C. Tomasi, "Alpha estimation in natural images", *Procedures of the IEEE: Computer Vision and Pattern Recognition* (CVPR 2000), pp. 18-25, June 2000

[SCH00] C. Schremmer, T. Haenselmann, F. Bömers, "Wavelets in Real-Time Digital Audio Processing: A Software for Understanding Wavelets in Applied Computer Science", *Workshop on Signal Processing Applications* (WoSPA), Brisbane, Australia, December 2000

[SCH00] T. Schenk, N. Mai, J. Zihl, "The catching performance of a motion-blind patient", *The Society for the Neural Control of Movement* (NCM), Santa Barbara (CA), USA, 2000

[SCH01] C. Schremmer, T. Haenselmann, F. Bömers, "A Wavelet Based Audio Denoiser", *IEEE International Conference on Multimedia and Expo* (ICME 2001), pp. 145-148, Tokyo, Japan, August 2001

[SCH96] W. Schroeder, *The visualization tollkit: An object-oriented approach to 3D graphics*, Prentice Hall, pp. 209ff, 1996

[SER97] M. P. Servais, G. de Jaher, "Video Compression Using the Three Dimensional Discrete Cosine Transform (3D-DCT)", *Proceedings of the 1997 South African Symposium on Communications and Signal Processing* (COMSIG ’97), pp. 27-32, Rhodes University, September 1997

[SMI91] D. K. Smith, *Dynamic Programming*, Ellis Horwood, New York, London, Toronto, 1991

[SNI92] M. Sniedovich, "Dynamic Programming", Marcel Dekker, Inc., New York, Basel, Hong Kong, 1992

[TORR00] L. Torres, E. J. Delp, "New Trends in Image and Video Compression", *X European Signal Processing Conference* (EURASIP), Tampere, Finland, September 2000

[TSU94] K. Tsunashima, J. B. Stampleman, V. M. Bove, "A Scalable Motion-Compensated Subband Image Coder", *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, February 1994

[THU72] R. Thurner, *Dynamische Programmierung unter besonderer Berücksichtigung der Lösung diskreter deterministischer Probleme auf Computern* (Dissertation), Rodenbusch, Bamberg, 1972

[TOR00] L. Torres, E. J. Delp, "New Trends in Image and Video Compression", *X European Signal Processing Conference* (EURASIP), Tampere, Finland, September 2000

[TOR01] L. Torres, L. Lorente, J. Vilià, "Face recognition using self-eigenfaces", *Proceedings of the International Symposium on Image/Video Communications Over Fixed and Mobile Networks*, pp. 44-47, Rabat, Marocco, April 2000

[TUR91] M. Turk, A. Pentland, "Eigenfaces for recognition", *Journal of Neuroscience*, 3(1): pp. 71-86, 1991

[UNS96] M. Unser, "Texture classification and segmentation using wavelet frames", *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1549-1560, 1996

[WAN99] A. Wang, Z. Xiong, P. A. Chou, "Three-Dimensional Wavelet Coding of Video with Global Motion Compensation", *Data Compression Conference* (IEEE Computer Society), Snowbird, Utah, USA, 1999

[WAT91] J. Waterworth, *Multimedia technology and application*, Ellis Horwood Ltd., 1991

[WET67] W. Wetzel, M. Jöhnk, P. Naeve, *Statistische Tabellen*, Walter de Gruyter, Berlin, 1967

[YAN97] X. Yang, K. Ramchandran, "Hierarchical Backward Motion Compensation for Wavelet Video Coding using Optimized Interpolation Filters", *International Conference on Image Processing* (ICIP), vol. 1, Washington, USA, 1997

[ZEK91] C. Zeki, *Brain - Cerebral akinetopsia (motion blindness)*, Oxford University Press, vol. 114, Issue 2811-824, 1991