Bedarfsgesteuerte Bildübertragung mit Regions of Interest und Levels of Detail für mobile Umgebungen

Dissertation

zur
Erlangung des akademischen Grades

Doktor–Ingenieur (Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften
der Universität Rostock

vorgelegt von
Uwe Rauschenbach aus Rostock

Rostock, 5. Mai 2000

urn:nbn:de:gbv:28-diss2009-0081-5

Gutachter:
Prof. Dr.–Ing. habil. Heidrun Schumann, Universität Rostock
Prof. Dr.–Ing. Dr. h.c. Dr. E.h. José Luis Encarnação, TU Darmstadt
Prof. Dr. rer. nat. Dieter Schütt, SIEMENS AG

Datum der Verteidigung: 5. Oktober 2000

Vorwort

Die vorliegende Arbeit entstand während meiner Zeit als Kollegiat im Graduiertenkolleg *„Verarbeitung, Verwaltung, Darstellung und Transfer multimedialer Daten – technische Grundlagen und gesellschaftliche Implikationen“* an der Universität Rostock und meiner gleichzeitigen Arbeit im Projekt *„Mobile Visualisierung“* (MoVi, [EFK95, EKS96]). Das MoVi-Projekt wurde vor nunmehr sechs Jahren von Prof. Encarnação initiiert. In dieser Zeit entwickelte sich Mobile Computing von einer Vision zur etablierten Forschungs- und Anwendungsdisziplin. Es war eine Herausforderung, an solch einer innovativen, dynamischen Thematik mitzuarbeiten.

Meine Mentorin, Prof. Heidrun Schumann, verdient besondere Erwähnung. Sie ermutigte mich zum Schreiben der Arbeit, begleitete sie mit konstruktiven Anregungen, schuf mir die nötigen Freiräume und schärfte meinen Blick für das Wesentliche. In der von ihr geleiteten Arbeitsgruppe am Institut für Computergraphik fand ich ein gutes Umfeld nicht nur für das Gelingen dieser Arbeit. Mein Dank gilt den Kolleginnen und Kollegen am Institut und am Fachbereich für die freundschaftliche und kreative Atmosphäre und ihre Hilfe bei kleinen und großen Problemen der täglichen Arbeit.

Die Studenten Mathias Balzer, Dirk Bösche, Roland Göcke, Stefan Jeschke, Ralf Jubin, Joachim Lechelt, René Rosenbaum, Christian Schmidt, Randolf Schultz und Tino Weinkauf trugen im Rahmen ihrer Studien- und Diplomarbeiten bzw. als studentische Hilfskräfte mit Implementationen und durch das sich daraus ergebende Feedback zum Gelingen der Arbeit bei. Das große Engagement von Tino Weinkauf möchte ich besonders hervorheben.

Thomas Kirste und Tilo Strutz danke ich für die inspirierenden fachlichen Diskussionen und Holger Theisel für die Einführung in das Programm Maple. Karina Oertel, Sabine Radtke, Doritt Linke und Sabrina Duda halfen mit ihren Tests im Usability-Lab, die Benutzbarkeit des in Kapitel 5 beschriebenen RECHTECKIGEN FISHEYE-VIEWS zu verbessern.

Mein Dank gilt weiterhin der Deutschen Forschungsgemeinschaft, die sowohl das MoVi-Projekt als auch das Graduiertenkolleg finanzierte. Die Firma Mannesmann ermöglichte durch Überlassung eines Mobilfunkvertrages die Zeitmessungen zur Bildübertragung über GSM, die Firma LuRaTech stellte unentgeltlich eine Lizenz ihrer Software LuRaWave ebenfalls für Performancemessungen zur Verfügung, und der Verkehrsverbund Warnow sowie das Werbestudio Nitschke gestatteten mir, ihr Kartenmaterial für die Evaluierung der FISHEYE-Darstellung zu verwenden.

Besonders bedanke ich mich bei Prof. Encarnação und Prof. Schütt für die Übernahme der Gutachten und die langjährige Begleitung meiner Arbeit.

Schließlich danke ich meinen Eltern, die immer für mich da waren und mich in jeder Hinsicht unterstützten.

Diese Arbeit wurde nach den Regeln der neuen Rechtschreibung verfasst.

Inhalt

1 Einführung **1**
- 1.1 Motivation . . . 1
- 1.2 Mobile Umgebungen . . . 1
- 1.3 Bildkodierung, adaptive und bedarfsgesteuerte Bildübertragung . . . 2
- 1.4 Aufbau und Ziele der Arbeit . . . 3

2 Stand der Forschung **5**
- 2.1 Überblick über die Teilaspekte . . . 5
- 2.2 Klassifizierung von Rasterbildern . . . 6
- 2.3 Adaptive Steuerung der Bildübertragung . . . 7
- 2.4 Progressive Bildkodierung und -übertragung . . . 8
  - 2.4.1 Einordnung und geschichtlicher Überblick . . . 8
  - 2.4.2 Klassifikation von Bildübertragungsverfahren . . . 9
  - 2.4.3 Ausgewählte Teilschritte der Kodierungspipeline . . . 12
  - 2.4.4 Bewertung progressiver Kodierverfahren . . . 18
  - 2.4.5 Ausgewählte progressive Bildübertragungsverfahren . . . 20
- 2.5 Regions of Interest . . . 31
  - 2.5.1 Übersicht und Einordnung . . . 31
  - 2.5.2 Ausgewählte Verfahren . . . 31
  - 2.5.3 Zusammenfassende Wertung . . . 36
- 2.6 Progressive Verfeinerung von Farbtabellenbildern . . . 37
  - 2.6.1 Kompression von Farbtabellenbildern . . . 37
  - 2.6.2 Progressive Verfeinerung der Farbinformation . . . 39
- 2.7 FishEye-Techniken zur Platz sparenden Bilddarstellung . . . 39
  - 2.7.1 Prinzip verzerrungsorientierter Displays . . . 39
  - 2.7.2 FishEye-Darstellungen als grafische Ausgabetechnik . . . 40
  - 2.7.3 Beispiele . . . 40
- 2.8 Schlussfolgerungen und weiteres Vorgehen . . . 41

3 Regions of Interest und Levels of Detail in der Bildübertragung **43**
- 3.1 Grundidee . . . 43
- 3.2 Formale Beschreibung . . . 44
- 3.3 Integrierbarkeit von RoIs/LoDs in verschiedene Bildkodierungsverfahren . . . 48
  - 3.3.1 Verfahren für Farbtabellenbilder . . . 48
  - 3.3.2 Quadtree-Verfahren . . . 49
  - 3.3.3 Verfahren mit Transformationskodierung . . . 49

4 Regions of Interest und Levels of Detail mit Wavelets **51**
- 4.1 Überblick . . . 51
- 4.2 Umsetzung des LoD-Raumes mit Wavelets . . . 51
- 4.3 Das Basis-Kodierverfahren . . . 52
  - 4.3.1 Auswahl und Umsetzung . . . 52

4.3.2 Performance von MOVIWAVECODEC . . . 53
4.4 Ein neues Wavelet-Dekompositionsschema . . . 54
4.4.1 Problem . . . 54
4.4.2 Lösungsprinzip . . . 55
4.4.3 Ungenauigkeiten und Kompressionsartefakte . . . 55
4.4.4 Das resultierende Zerlegungsschema . . . 56
4.4.5 Anpassung der Zerotree-Kodierung an das neue Zerlegungsschema 57
4.4.6 Performance des neuen Schemas . . . 58
4.5 MOVIROILOD: Waveletbasierte Umsetzung von RoIs und LoDs . . . 61
4.5.1 Anpassung von Quantisierung und Traversierung für RoIs und LoDs 61
4.5.2 Steuerung der Übertragung . . . 68
4.6 Partielle Dekodierung . . . 73
4.7 LIBROILOD: Ein System zur bedarfsgesteuerten Bildübertragung . . . 74

5 Anwendungen 77
5.1 Client-Server-Architektur der Anwendungen . . . 77
5.1.1 Die Clientkomponenten . . . 77
5.1.2 Die RoI/LoD-Serverkomponente . . . 78
5.1.3 Client-Server-Kommunikation . . . 78
5.1.4 Latenzzeit . . . 79
5.2 RoI-Unterstützung für Bildautor und Bildbetrachter . . . 81
5.2.1 Problemstellung . . . 81
5.2.2 Realisierung . . . 81
5.2.3 Übertragungsbeispiel . . . 82
5.2.4 Diskussion . . . 83
5.3 Übertragung und Anzeige großer Bilder . . . 83
5.4 Die Fokus-und-Kontext-Technik „RECHTECKIGER FISHEYE-VIEW“ . . . 84
5.4.1 Problemstellung . . . 84
5.4.2 Funktionsweise des RECHTECKIGEN FISHEYE-VIEWS . . . 85
5.4.3 Der RECHTECKIGE FISHEYE-VIEW als reine Darstellungstechnik 96
5.4.4 Ergebnisse . . . 98

6 Verfeinerung der Farbinformation von Farbtabellenbildern 103
6.1 Motivation und Grundidee . . . 103
6.2 Unterstützte Dimensionen des LoD-Raumes . . . 104
6.3 Kodierung . . . 105
6.3.1 Sortierung der Farbtabelle . . . 105
6.3.2 Vorverarbeitungsverfahren . . . 106
6.3.3 Berechnung der Mischfarbtabelle . . . 107
6.3.4 Bitebenenweise Kodierung . . . 108
6.4 Evaluierung . . . 109
6.4.1 Die Testdatenbasis . . . 110
6.4.2 Visuelle Vergleiche . . . 110
6.4.3 Kompressionsraten . . . 112
6.4.4 Einfluss von Farbanzahl und Dithering auf die Kompressionsrate . . 113
6.5 Verbesserungsmöglichkeiten . . . 114

7 Schlussbetrachtungen 117
7.1 Zusammenfassung der Ergebnisse . . . 117
7.2 Ausblick . . . 118

A Publikationen zum Thema RoI 119

B Bilder und Diagramme 121

B.1 Abhängigkeit der Kompressionsrate von der Farbtiefe bei Nutzung von Standardverfahren . . . 121
B.2 Graustufen-Testbilder . . . 122
B.3 Qualitätsvergleiche komprimierter Bilder . . . 123
B.3.1 Spektrale Selektion vs. Sukzessive Approximation . . . 123
B.3.2 JPEG vs. Wavelets . . . 123
B.4 Kompressionsbeispiele zum BCQ-Verfahren . . . 125
B.5 Latenzzeit der Client-Server-Kommunikation . . . 127
B.6 Ghost-Feedback beim RECHTECKIGEN FISHEYE-VIEW . . . 129
B.7 Der RECHTECKIGE FISHEYE-VIEW als reine Anzeigetechnik . . . 130
B.8 Platzersparnis durch den RECHTECKIGEN FISHEYE-VIEW . . . 131
B.9 Progressive Übertragung von Farbtabellenbildern . . . 132
B.9.1 Dateigröße vs. Lauflängeninkrement der RLR-Kodierung . . . 134
B.9.2 Abhängigkeit der komprimierten Dateigröße von der Anzahl der Farben . . . 135
B.9.3 Vergleich von Y-Sortierung und CP-Sortierung . . . 137
B.9.4 Bildbeispiele . . . 138

C Messwerttabellen **145**
C.1 Client-Server-Kommunikation . . . 145
C.2 Zeiten für den Bildaufbau beim RECHTECKIGEN FISHEYE-VIEW als reine Anzeigetechnik . . . 145
C.3 Progressive Übertragung von Farbtabellenbildern . . . 146
C.3.1 Bitebenen-Histogramme . . . 146
C.3.2 Kompressionsratenvergleiche . . . 146
C.3.3 Dynamische Auswahl des Kodierverfahrens . . . 147

D Spezifikation des ASCII–RoI–Formates **149**
D.1 Zweck . . . 149
D.2 Syntax . . . 149
D.3 Bemerkungen . . . 150
D.4 Beispiel . . . 151

Literatur **153**

Abbildungen

2.1 Kodierungspipeline 10
2.2 Zweistufige dyadische Wavelet-Transformation (oben) und ihre Invertierung (unten) 14
2.3 Interlacing-Schema von PNG 22
2.4 Traversierung der DCT-Koeffizienten 25
2.5 Zerlegung der DCT-Koeffizienten in Spektralbandgruppen (spektrale Selektion) 26
2.6 Zerlegung der DCT-Koeffizienten in Gruppen von Bits (sukzessive Approximation) kombiniert mit spektraler Selektion 26
2.7 Traversierungsreihenfolge (links) und Zerotree-Struktur (rechts) für eine dreistufige dyadische Wavelet-Zerlegung 27

3.1 Grafische Notation der Constraints 46

4.1 Repräsentation des LoD-Raumes mit Wavelets 52
4.2 Performancevergleich existierender Bildkompressoren mit dem implementierten Basisverfahren MOVIWAVECODEC 53
4.3 XY-Ebene des LoD-Raumes (links: dyadisches, rechts: neues Dekompositionsschema) 54
4.4 Neues Zerlegungsschema für größere Unabhängigkeit der Skalierungsfaktoren in X- und Y-Richtung. Oben: Dyadische Zerlegung. Unten: Grundidee des neuen Schemas 55
4.5 Kodierungsartefakte mit neuem Zerlegungsschema und deren Reduktion . . 56
4.6 Vergleich der Standardabweichungen und Erwartungswerte der Koeffizientenbeträge der einzelnen Bänder für verschiedene Filteralternativen 56
4.7 Resultierendes hybrides Zerlegungsschema zur Artefaktreduktion. Oben: Dyadische Zerlegung. Unten: Beispiel für hybrides Schema mit $l = 3$ und $il = 2$ 57
4.8 Modifizierte Zerotree-Kodierung für neues Zerlegungsschema. Links: Traversierungsreihenfolge. Mitte: Generische Zerotree-Struktur. Rechts: Umgesetzte Zerotree-Struktur für hybrides Dekompositionsschema am Beispiel von $l = 3$ und $il = 2$ 58
4.9 Anpassung der sukzessiven Approximation und der Traversierung des Wavelet-Koeffizientenfeldes für lokale LoDs 61
4.10 Anpassung der Traversierung des Wavelet-Koeffizientenfeldes für RoIs . . 62
4.11 Abweichung der kodierten Bitrate von der Zielbitrate bei der Kodierung des Testbildes „lena“ (Abbildung B.2(b)) mit einer RoI, die das gesamte Bild umfasst ($l = 4$, $il = 1$) 63
4.12 Mehrfachauflösungsrepräsentation eines Dreiecks 65
4.13 Mögliche Fälle bei der Berechnung der X-Koordinate x_S des Schnittpunktes einer Spanne mit einer nicht-senkrechten Kante 66

4.14 Mögliche Zustände und Zustandsübergänge einer RoI während der Übertragung . . . 72
4.15 Architektur von LIBROILOD . . . 74
4.16 Testumgebung für die RoI/LoD-Bibliothek . . . 76

5.1 Beispiel flexibler RoI-Unterstützung für Bildautor (links) und Bildbetrachter (rechts) . . . 83
5.2 Originalbild mit überlagertem Skalierungsgitter (links) und resultierender RECHTECKIGER FISHEYE-VIEW (rechts) . . . 85
5.3 Geometrische Struktur des RECHTECKIGEN FISHEYE-VIEW-Gitters . . . 87
5.4 Automatische Berechnung der Kontextierung bei einer Gesamtbreite aller Ringe von 1000 Pixeln im Originalbild und von 1 bis 535 Pixeln in der FISHEYE-Darstellung . . . 91
5.5 Redundante lokale Bilddatenverwaltung . . . 95
5.6 Beispiel für einen multifokalen RECHTECKIGEN FISHEYE-VIEW mit 4 Fokusregionen . . . 96
5.7 Gegenseitige Beeinflussungen zweier Foki . . . 96
5.8 Funktionsprinzip der glatten Kontextierung . . . 97
5.9 Übertragungssimulation mit einem RECHTECKIGEN FISHEYE-VIEW der Größe 512x512 Pixel bei einer Originalbildgröße von 1024x1024 Pixeln . . . 100
5.10 Qualitätsvergleich nach der Übertragung von 24562 Bytes für den RECHTECKIGEN FISHEYE-VIEW aus Abbildung 5.9(a) (rechts) und das ganze Bild (links) . . . 101

6.1 Vergleich der Bildqualität nach Übertragung der ersten (oben) und dritten (unten) Bitebene bei Nutzung der Y- (rechts) und der CP-Sortierung (Mitte) mit dem Original (links) . . . 106
6.2 Übertragungssimulation einer Navigationsbitmap mit vier Techniken: MOVICOLORQ (iMCQ und MCQ), iPNG sowie iGIF . . . 111
6.3 Farbverfälschungen in frühen Übertragungsstufen bei MOVICOLORQ . . . 112

B.1 Kompressionsrate von JPEG und PNG für Bilder unterschiedlicher Farbtiefe 121
B.2 Graustufen-Testbilder . . . 122
B.3 Vergleich von spektraler Selektion allein (links, 0.276 bpp, DC, AC1, AC2) mit spektraler Selektion und sukzessiver Approximation (rechts, 0.275 bpp, DC Bits 7-1, AC1-AC5 Bits 7-2) . . . 123
B.4 Vergleich der mit JPEG- (links) und Wavelet-Kodierung (rechts) erzielbaren Bildqualität bei 0.05bpp (1:160) . . . 123
B.5 Vergleich der mit JPEG- (links) und Wavelet-Kodierung (rechts) erzielbaren Bildqualität bei 0.128bpp (1:62.5) . . . 124
B.6 Vergleich der visuellen Bildqualität bei gleicher PSNR von 27.32 dB. Links: JPEG-Kodierung. Rechts: Wavelet-Kodierung . . . 124
B.7 BCQ-Zerlegung des Testbildes „logos“ (Abbildung B.2(a)) . . . 125
B.8 Vergleich der visuellen Bildqualität des Testbildes „lena“ bei der Kodierung mit BCQ (links) und mit dem MOVIWAVE-Format (rechts) bei jeweils gleicher Bitrate . . . 126
B.9 CPU-Belastung eines ansonsten unbelasteten Servers (INTEL PENTIUM II, 450 MHz, 256 MB RAM) bei einer dreiminütigen Übertragung einer 1024x1024 Pixel großen RoI im LAN . . . 127
B.10 Latenzzeiten der Kommunikationsalternativen HTTP+CGI und Sockets im Übertragungsverlauf . . . 128
B.11 Ghost-Feedback zur Verbesserung des Antwortverhaltens beim Bewegen des Fokus . . . 129

B.12 FISHEYE-Darstellung einer Vektorgrafik durch Clipping-Kontextierung mit drei Kontextringen . . . 130
B.13 FISHEYE-Darstellung durch glatte Kontextierung . . . 130
B.14 Platzersparnis durch den RECHTECKIGEN FISHEYE-VIEW. . . . 131
B.15 Gesamtgrößen aller komprimierten Bilder des Testsets *All* bei Nutzung von iGIF, iPNG, PNG im Vergleich zu MOVICOLORQ (MCQ) mit unterschiedlichen Sortierverfahren, mit und ohne Interlacing sowie mit und ohne Optimierung der Parameter k_{inc} und k_{dec} . . . 132
B.16 Gesamtgröße der komprimierten Dateien des Testsets *All* bei Benutzung von MOVICOLORQ (MCQ und iMCQ) im Vergleich zu iGIF ($\widehat{=}$100%) abhängig von der Anzahl der Bitebenen . . . 132
B.17 Gesamtgröße der komprimierten Dateien des Testsets *All* bei Benutzung von MOVICOLORQ (MCQ und iMCQ) im Vergleich zu iPNG ($\widehat{=}$100%) abhängig von der Anzahl der Bitebenen . . . 133
B.18 Gesamtgröße der komprimierten Dateien des Testsets *All* bei Benutzung von MOVICOLORQ (MCQ und iMCQ) im Vergleich zu PNG ($\widehat{=}$100%) abhängig von der Anzahl der Bitebenen . . . 133
B.19 Die Testbilder „innenst18“ (18 Farben, links) und „innenst256“ (244 Farben, rechts) . . . 134
B.20 Komprimierte Dateigröße in Abhängigkeit von den RLR-Parametern k_{inc} und k_{dec} für das Testbild „innenst18“ für verschiedene Sortierverfahren mit und ohne Interlacing . . . 134
B.21 Komprimierte Dateigröße in Abhängigkeit von den RLR-Parametern k_{inc} und k_{dec} für das Testbild „innenst256“ für verschiedene Sortierverfahren mit und ohne Interlacing . . . 135
B.22 Komprimierte Dateigröße in Abhängigkeit von der Anzahl der Farben im Bild für das Beispiel aus (Abbildung B.19 rechts) . . . 135
B.23 Komprimierte Dateigröße in Abhängigkeit von der Anzahl der Farben im Bild für das Beispiel aus (Abbildung B.19 rechts) – unterste 64 Farben dargestellt . . . 136
B.24 Komprimierte Dateigröße in Abhängigkeit von der Anzahl der Farben im Bild für das Beispiel aus Abbildung B.19 rechts – Bitebenenwechsel in der Farbanzahl dargestellt . . . 136
B.25 Vergleich der Bildqualität nach Übertragung von einer (oben), zwei (Mitte) und drei (unten) Bitebenen bei Nutzung der Y- (rechts) und der CP-Sortierung (Mitte) für das Testbild „innenst18“. Das Originalbild ist jeweils links dargestellt . . . 137
B.26 256-Farben-Version des Lena-Testbildes nach Übertragung des ersten Passes der ersten Bitebene mit Interlacing . . . 138
B.27 256-Farben-Version des Lena-Testbildes in einer späteren Stufe der Übertragung . . . 138
B.28 Bild mit wenigen Bitebenen (16 Farben, 4 Bitebenen). Für iMCQ (unten links) wurde der ersten Interlacing-Pass der ersten Bitebene vollständig übertragen. . . . 139
B.29 Bessere Erkennbarkeit geringer Farbunterschiede mit größerer räumlicher Ausdehnung (Straßennetz) bei iGIF und iPNG; Streifen durch schlechtere Kompression von iMCQ im Vergleich zu MCQ in späteren Übertragungsschritten . . . 139
B.30 Bild mit 224 Farben nach Abschluss der Übertragung der ersten Bitebene bei MOVICOLORQ mit Interlacing (unten links) . . . 140
B.31 Bild mit 224 Farben nach Abschluss der Übertragung der ersten Bitebene bei MOVICOLORQ ohne Interlacing (oben links) . . . 140

B.32 Vollständig übertragenes GIF-Bild mit 224 Farben (unten rechts) und geringe visuelle Unterschiede zu den noch nicht vollständig übertragenen MOVICOLORQ-Pendants (links) 141
B.33 Erste Übertragungsstufe eines Bildes mit Farbverlauf 141
B.34 Spätere Übertragungsstufen eines Bildes mit Farbverlauf 142
B.35 Bildqualität eines Bildes mit Farbverlauf nach Übertragung der kleinsten erreichbaren Dateigröße (MOVICOLORQ mit Interlacing) im Vergleich zu MOVICOLORQ ohne Interlacing, iPNG und iGIF 143

Tabellen

2.1 Koeffizienten einiger biorthogonaler Wavelet-Filter 15
2.2 Kodierungspipelines der JPEG-Operationsmodi 23
2.3 Anzahl der Veröffentlichungen in der gängigen wissenschaftlichen Literatur zu RoIs in Bildkodierung und -übertragung pro Jahr von 1983 bis 1999 . 31

3.1 Koordinatenmodell der Constraints 47
3.2 Integrierbarkeit von RoIs und LoDs in verschiedene Bildformate 48
3.3 Vergleich der komprimierten Dateigrößen bei der verlustfreien Kodierung mit Bit-Quadtree, BCQ, GIF und Wavelets 49

4.1 MOVIWAVE-Kodierungs- und -Dekodierungszeiten für dyadisches und neues Dekompositionsschema bei verschiedenen Bitraten 60
4.2 Vergleich der durchschnittlichen Kodierungs- und Dekodierungszeiten des monolithischen MOVIWAVE-Codecs und des spannenbasierten MOVIROILOD-Codecs 67
4.3 Beispiele für Steuerkommandos 70

5.1 Erwartungswert, Standardabweichung, Minimum und Maximum der Latenzzeit abhängig von der benutzten Kommunikationsmethode im GSM-Netz 80
5.2 Erwartungswert, Standardabweichung, Minimum und Maximum der Latenzzeit abhängig von der RoI-Größe im Highspeed-LAN 80
5.3 Übertragungszeiten verschiedener Dekompositionsalternativen für den RECHTECKIGEN FISHEYE-VIEW 99

6.1 Vergleich der komprimierten Gesamt-Dateigröße für verschiedene Kompressionsmethoden. Testset: *All* 113
6.2 Der Einfluss von Dithering-Mustern im Bild auf die Kompressionsleistung . 114

C.1 Ankunftsreihenfolge der Kommandos bei einer HTTP+CGI-Übertragung über GSM 145
C.2 Performance des RECHTECKIGEN FISHEYE-VIEWS als reine Darstellungstechnik 145
C.3 Bitebenenhistogramme der Testsets 146
C.4 Vergleich der komprimierten Gesamt-Dateigröße für verschiedene Kompressionsmethoden. Testset: *Cartoons* 146
C.5 Vergleich der komprimierten Gesamt-Dateigröße für verschiedene Kompressionsmethoden. Testset: *Internet* 146
C.6 Steigerung der Kompression von MOVICOLORQ durch dynamische Auswahl des Kodierverfahrens (ungeditherte Bilder) 147
C.7 Steigerung der Kompression von MOVICOLORQ durch dynamische Auswahl des Kodierverfahrens (geditherte Bilder) 147

Abkürzungen

AC	Alternating Current; hier: die → DCT-Koeffizienten eines Blockes außer dem → DC-Koeffizienten
BCQ	Bitwise Condensed Quadtree [DK91]
bpp	Bits pro Pixel; Maß für die Bitrate einer komprimierten Bilddatei
bps	Bits pro Sekunde; Maß für die Übertragungsgeschwindigkeit eines Kanals
CIE	Commission Internationale de l'Éclairage; Standardisierungsgremium
CGI	Common Gateway Interface; Schnittstelle zur Erweiterung eines WWW-Servers [CGI]
CP	Closest Pair; Sortierverfahren für Farbtabellenbilder (6.3.1)
CQT	Compact Quadtree; Verfahren zur Kodierung von Waveletkoeffizienten (2.5.2.5)
DC	Direct Current; hier: → DCT-Koeffizient, der den Gleichanteil eines Blockes repräsentiert
DCT	Diskrete Cosinus-Transformation (2.4.3.1)
DWT	Dyadische Wavelet-Transformation (2.4.3.1)
DPCM	Differential Pulse Code Modulation (2.4.5.3)
EBCOT	Embedded Block Coding with Optimized Truncation; eingebettetes waveletbasiertes Kodierverfahren für → JPEG2000 (2.4.5.8)
EZW	Embedded Zerotree Wavelet [Sha93]; eingebettetes waveletbasiertes Kodierverfahren (2.4.5.6)
EOF	End of file; Endesymbol bei der arithmetischen Kodierung nach [WNC87]
GDI	Graphics Device Interface; Grafisches Subsystem der WINDOWS-Systeme von Microsoft
GIF	Graphics Interchange Format; Grafikaustauschformat des Online-Dienstes Compuserve [GIF89] (2.4.5.2)
GSM	Global System for Mobile communication; Mobilfunkstandard
GSTP	Generalized Self-Similarity Tree with Packetization; Verfahren zur Kodierung von Waveletkoeffizienten (2.5.2.5)
HTML	Hypertext Markup Language; Strukturierungssprache für Dokumente im World Wide Web
HTTP	Hypertext Transfer Protocol; das Übertragungsprotokoll des World Wide Web
IDCT	Inverse → DCT
IDWT	Inverse → DWT
iGIF	interlaced → GIF
IIP	Internet Imaging Protocol (2.4.5.11)
iMCQ	interlaced → MCQ
iPNG	interlaced → PNG

ISO	International Organization for Standardization; Standardisierungsgremium
ITU	International Telecommunication Union; Standardisierungsgremium
ISDN	Integrated Services Digital Network
IZ	Isolated Zero; Symbol zur Kodierung von Insignifikanz einzelner Koeffizienten im → EZW-Algorithmus
JBIG	Joint Bi-Level Image Experts Group; Kompressionsstandard für Schwarz-Weiß-Bilder [JBG93]
JFIF	JPEG File Interchange Format; Format einer → JPEG-Datei
JPEG	Joint Photographic Experts Group; → DCT-basierter Kompressionsstandard für Standbilder [PM93] (2.4.5.5)
JPEG2000	Joint Photographic Experts Group; neuer Wavelet-basierter Kompressionsstandard für Standbilder [JPG99a]
LAN	Local Area Network
LoD	Level of Detail
LZ77	Lempel-Ziv-77 [ZL77]; verlustfreies Datenkompressionsverfahren, nicht patentiert, im → PNG-Format verwendet (2.4.3.2)
LZW	Lempel-Ziv-Welch; verlustfreies Datenkompressionsverfahren, patentiert, z.B. im → GIF-Format verwendet (2.4.3.2)
MCQ	MoviColorQ-Format; das in dieser Arbeit entwickelte Bilddateiformat zur progressiven Übertragung von Farbtabellenbildern (Kapitel 6)
MOS	Mean Opinion Score; subjektives Qualitätsmaß für verlustbehaftete Bildkodierverfahren (2.4.4.1)
MoVi	Mobile Visualisierung; DFG-gefördertes Projekt zur Untersuchung von Konzepten zu Computergraphik und Multimedia in mobilen Umgebungen [K+98]
MPEG	Motion Picture Experts Group; Familie von → DCT-basierten Kompressionsstandards für Video
MRL	MoviRoILoD-Format; das in dieser Arbeit entwickelte Bilddateiformat zur progressiven Übertragung von Graustufenbildern mit → RoI und → LoD
MVW	MoviWave-Format; das in dieser Arbeit entwickelte Bilddateiformat zur progressiven Übertragung von Graustufen- und Echtfarbbildern ohne → RoI und → LoD (4.3.1 und 4.3.2)
NEG	Symbol zur Kodierung von „negativer Signifikanz“ im → EZW-Algorithmus
PDA	Personal Digital Assistant
PGM	Portable Gray Map; sehr einfaches, portables Graustufen-Bilddateiformat [P+]
PNG	Portable Network Graphics [PNG] (2.4.5.3)
POS	Symbol zur Kodierung von „positiver Signifikanz“ im → EZW-Algorithmus
PROTRAC	Progressive Transmission Capability; experimentelles Bildübertragungsverfahren (2.6.2)
PSNR	Peak Signal to Noise Ratio; objektives Qualitätsmaß u.a. für verlustbehaftete Bildkodierverfahren (2.4.4.1)
PTI	Progressive Transmission of Images; experimentelles progressives Bilddateiformat (2.6.2)
RGB	Red-Green-Blue; Farbsystem mit additiver Farbmischung für den Einsatz auf Displays
RLR	Run-Length / Rice; Lauflängenkodierungsverfahren (2.4.3.2)
RoI	Region of Interest (2.5)
SPIHT	Set Partitioning In Hierarchical Trees [SP96b]; eingebettetes waveletbasiertes Kodierverfahren (2.4.5.7)

TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator
WAN	Wide Area Network; Weitverkehrsdatennetz
WWW	World Wide Web
XOR	Exclusiv-Oder
YCP	Y+Closest Pair; in dieser Arbeit entwickeltes hybrides Sortierverfahren für Farbtabellenbilder (6.3.1)
YIQ	Farbsystem der japanischen und amerikanischen Fernsehnorm NTSC mit einem Helligkeitskanal und zwei Farbdifferenzkanälen
YUV	Farbsystem der europäischen Fernsehnormen PAL und SECAM mit einem Helligkeitskanal und zwei Farbdifferenzkanälen
YC_bC_r	Helligkeit – Chrominance blue – Chrominance red; im → JPEG-Standard verwendetes Farbsystem mit einem Helligkeitskanal und zwei Farbdifferenzkanälen (2.2)
ZTR	Zerotree Root: Symbol zur Kodierung von Insignifikanz ganzer Koeffizientenbäume im → EZW-Algorithmus

Kapitel 1

Einführung

1.1 Motivation

In den letzten Jahren gewann das Internet große Popularität und leitete eine neue Ära der Computernutzung ein. Dieser Erfolg ist zu einem großen Teil auf die Möglichkeit zurückzuführen, statt reiner Texte auch Bilder und Grafiken zur Informationspräsentation und zum Strukturieren der Inhalte zu nutzen. Die notwendige Übertragung dieser Bilder im Internet erfordert jedoch eine größere Bandbreite als die Übertragung reiner Textinformation.

Ein weiterer Trend ist der Übergang von stationären zu mobilen Rechnern, was – beispielsweise unter Nutzung des Datenmodus von GSM-Telefonen – einen Zugriff auf die gewünschte Information *jederzeit* und *überall* [IK96] ermöglicht, jedoch auf Grund knapper Ressourcen der mobilen Rechner und der entsprechenden Datenübertragungstechnik neue Anforderungen an die Aufbereitung der Information stellt. Diese Aufbereitung ist vor allem für die bandbreitenintensive Informationsklasse „Bild“ erforderlich und offeriert ein erhebliches Reduktionspotenzial.

Die vorliegende Arbeit befasst sich mit der Problematik der Übertragung und Darstellung von Rasterbildern in mobilen Umgebungen und stellt Ansätze vor, große Bilder auch über Netze mit niedriger Bandbreite zu übertragen sowie diese auf kleinen Displays anzuzeigen.

1.2 Mobile Umgebungen

Unter einer mobilen Umgebung soll im Kontext dieser Arbeit ein portabler Rechner verstanden werden, der über ein Weitverkehrsnetz (WAN) mit einem stationären Rechner im Internet verbunden ist. Dieser portable Rechner wird auch als *mobiles Endgerät* bezeichnet und kann sowohl ein Notebook als auch ein PDA (Persönlicher Digitaler Assistent) sein. Charakteristisch für mobile Umgebungen sind häufige Ortswechsel, limitierte Ressourcen und ein großes Spektrum verschiedener Endgeräte.

Im Vergleich zu stationären Rechnern mit hoher Rechenleistung und großen Displays führt bei mobilen Endgeräten der notwendige kleine Formfaktor zu Kompromissen zwischen Displaygröße, Verarbeitungsleistung und Energieverbrauch einerseits und Größe bzw. Gewicht des Gerätes andererseits. Mobile Endgeräte bieten auf Grund dieses Kompromisses in der Regel geringere Rechenleistung und kleinere Displays mit zum Teil nur geringer Farbtiefe bis hin zu rein monochromer Darstellung. Ein weiteres Problem liegt in der relativ geringen Batteriekapazität.

Der Zugriff auf Weitverkehrsnetze erfolgt zurzeit vor allem über analoge Modems (was das Vorhandensein eines Telefonanschlusses voraussetzt) bzw. über den Datenmodus des GSM. Aktuelle analoge Modems bieten Bandbreiten von 33.6 bis 56 Kilobit pro Sekunde (kbps). Die Geschwindigkeit der Datenübertragung über GSM ist noch geringer, 9.6 oder – zunehmend anzutreffen – 14.4 kbps. Im Vergleich zu analogen Modems treten bei GSM häufigere Verbindungsabbrüche auf. Die Qualität des Übertragungskanals ist nicht gleich bleibend, sondern weist eine variable Fehlerrate auf. Charakteristisch ist das wiederholte Auftreten so genannter Fehlerbündel, das sind viele Fehler in einem kurzen Zeitintervall.

Bei der Nutzung von Standardverfahren zur Bildübertragung in solchen mobilen Systemen entstehen oft lange Wartezeiten. Außerdem sind der Darstellung großer Bilder durch kleine Displays Grenzen gesetzt. Im Rahmen der vorliegenden Arbeit sollen daher vor allem die Problemkomplexe langsamer Netzverbindungen und kleiner Displays bezogen auf die Übertragung und Darstellung von Rasterbildern adressiert werden. Andere Probleme im Vergleich zu stationären Systemen, wie die geringere Batteriekapazität oder Rechenleistung, stehen dabei nicht im Mittelpunkt.

1.3 Bildkodierung, adaptive und bedarfsgesteuerte Bildübertragung

Ausgangspunkt der Bildkodierung ist ein zweidimensionales *Rasterbild*. Dieses besteht aus in Zeilen und Spalten angeordneten Bildpunkten, die als Pixel (picture element) bezeichnet werden. *Bildkodierung* ist die Umwandlung eines zweidimensionalen Bildes in einen eindimensionalen (also sequenziellen) Datenstrom. Dieser Datenstrom kann als Datei abgespeichert bzw. über einen Kanal übertragen werden. Kodierung ist somit die Voraussetzung für die *Bildübertragung*. Zur Übertragung ist ein *Protokoll* notwendig, das neben der Übertragung optional auch den Kodierungsprozess steuern kann. Die *Präsentation* des übertragenen Bildes erfordert eine *Dekodierung* auf dem Endgerät, die aus dem Datenstrom wieder ein Pixelfeld erzeugt. In mobilen Umgebungen setzt die geringe verfügbare Bandbreite den Einsatz eines Kodierungsverfahrens voraus, das eine möglichst starke Kompression der Datenmenge ermöglicht. Hierzu existieren vielfältige Ansätze.

Auf Grund der heterogenen Ressourcencharakteristik mobiler Umgebungen muss ein Bildübertragungssystem ein zu übertragendes Bild an die aktuelle Ressourcensituation, den *Kontext*, anpassen können. Diese Anpassung, die im Allgemeinen durch die Adaption von Kodierparametern erfolgt, wird als *Adaptivität* bezeichnet. Parameter können sowohl die Auswahl des am besten geeigneten Kodierers beeinflussen als auch diesen Kodierer selbst steuern. Die Steuerung erzeugt eine Detaillierungsstufe (*Level of Detail, LoD*) eines Bildes, deren Datenvolumen und Detaillierungsgrad dem aktuellen Kontext angepasst sind. Adaptive Bildübertragung bedingt also entweder eine – zumindest teilweise - Neukodierung des Bildes auf Anforderung oder die Möglichkeit, aus einer Menge vorkodierter Detaillierungsstufen die Passendste auszuwählen. Durch die Steuerung des Datenvolumens lässt sich direkt die Übertragungszeit kontrollieren.

Wenn mehrere hierarchische Levels of Detail existieren, wobei höhere Detaillierungsstufen auf niedrigeren aufbauen, kann automatisch eine progressive Verfeinerung (*Progressive Refinement*) des Bildes realisiert werden. Dabei werden nach abgeschlossener Übertragung eines LoD weitere Daten übertragen, um den nächst höheren LoD zu erreichen. Die laufend aktualisierte Anzeige des übertragenen Bildes erlaubt dem Nutzer den Abbruch der Übertragung, wenn der Detaillierungsgrad ausreichend ist oder die Übertragungszeit die Vorstellungen des Nutzers übersteigt.

Adaptive Bildübertragung heißt *bedarfsgesteuert*, wenn der Kontext zusätzlich zur Res-

sourcensituation die aktuellen Nutzerwünsche berücksichtigt und eine Reaktion auf Kontextänderungen während der Übertragung möglich ist. Dabei wird es dem Nutzer ermöglicht, die gewünschte Detaillierungsstufe des Bildes zu spezifizieren, um die Übertragung zu steuern. Bei großen Bildern ist der Nutzer häufig nur an einem Teil des Bildes interessiert. *Regions of Interest* (*RoI*) gestatten es, unterschiedlichen Bereichen eines Bildes verschiedene Detaillierungsstufen zuzuweisen. So können unwichtige Bildteile in einem niedrigen Detaillierungsgrad übertragen werden, und die frei werdende Bandbreite kommt der Übertragung wichtigerer Bildbereiche in höherer Detailliertheit zugute. Wird ein in geringer Detaillierung vorliegender Bildteil später in einem höheren LoD benötigt, so kann der Benutzer Daten zur Verfeinerung nachfordern. Diese Vorgehensweise wird *Detail on Demand* genannt und erfordert ebenfalls die Existenz aufeinander aufbauender LoDs.

Das Konzept der Levels of Detail stammt aus dem Gebiet der virtuellen Realität (vgl. z.B. [AP94, FS93]). Hierbei wird ein Objekt redundant in mehreren Komplexitätsstufen gespeichert. Zur Bildberechnung wird diejenige Komplexitätsstufe ausgewählt, die eine Zielfunktion (z.B. Framerate, Darstellungsqualität) am besten erfüllt. Dieses Konzept wurde bisher nur auf Vektorgrafiken, jedoch nicht auf Rasterbilder angewendet. Regions of Interest werden häufig in Bildgebungs- und Bildverarbeitungssystemen genutzt, um rechenintensive Operationen nur auf interessanten Teilen des Bildes durchzuführen und sie so zu beschleunigen. Zunehmend werden RoIs auch für die Bildkompression interessant.

1.4 Aufbau und Ziele der Arbeit

Für viele Teilprobleme des skizzierten Problembereiches (wie Kompression, Übertragung, RoI und LoD) existieren also bereits Lösungen, die zum Teil mit anderen Zielsetzungen oder für andere Anwendungsbereiche erarbeitet wurden. Durch Kombination, Vervollständigung bzw. Übertragung dieser Lösungskonzepte auf die bedarfsgesteuerte Bildübertragung in mobilen Umgebungen sollen die in 1.2 skizzierten Probleme gelöst bzw. deren negativer Einfluss auf das Antwortzeitverhalten interaktiver mobiler Systeme verringert werden.

Ziel der vorliegenden Arbeit ist es folglich, die oben dargelegten Konzepte speziell für die bedarfsgesteuerte Übertragung und Darstellung von Rasterbildern in mobilen Umgebungen zu konkretisieren. Es sollen hybride Lösungsansätze erarbeitet und exemplarisch umgesetzt werden, die es besonders im Kontext limitierter Displayressourcen und geringer Übertragungskapazität ermöglichen, Übertragungsbandbreite durch eine Kombination von Bildkompressionsverfahren mit Regions of Interest und Levels of Detail zu sparen. Dabei wird Wert auf flexible Steuerungsmöglichkeiten gelegt, die eine Spezifizierung des Bedarfs ermöglichen.

Zunächst werden in Kapitel 2 ausgewählte Arbeiten auf den Gebieten der progressiven Bildkodierungsmethoden, der Bilddatenkompression und der Integration von RoIs in diese Verfahren diskutiert. Weiterhin wird auf die Besonderheiten bei der Kodierung von Farbtabellenbildern eingegangen sowie FishEye-Views als Bildschirmplatz sparende Anzeigetechniken vorgestellt.

Kapitel 3 schlägt ein neues Übertragungsschema mit RoIs und LoDs vor und gibt eine formale Beschreibung. Auf verschiedene Umsetzungsmöglichkeiten des Schemas wird eingegangen.

In den Kapiteln 4 und 5 wird die Umsetzung des entwickelten Modells auf der Grundlage waveletbasierter Bildkompressionstechniken beschrieben und verschiedene Anwendungen diskutiert. Hierbei wird besonderer Wert auf eine Platz sparende Bilddarstellung mithilfe einer neu entwickelten Fokus-und-Kontext-Technik sowie auf die Unterstützung der beiden

Benutzerrollen Bildautor und Bildbetrachter gelegt.

In Kapitel 6 werden Besonderheiten der Bildklasse „Farbtabellenbilder“ diskutiert und ein neuer Ansatz zur Erzeugung mehrerer Detaillierungsstufen der Farbinformation dieser Bildklasse entwickelt.

Kapitel 7 fasst die wichtigsten Ergebnisse der Arbeit zusammen. Dazu gehören:

- ein formales Modell für Regions of Interest und Levels of Detail in der Rasterbildübertragung,
- ein neues Wavelet-Zerlegungsschema zur Unterstützung von Levels of Detail,
- Konzepte zur Umsetzung dieses Modells auf der Grundlage eines waveletbasierten Bildkodierungsverfahrens und eine entsprechende Implementierung,
- Anwendungskonzepte auf der Basis dieser Umsetzung; insbesondere der RECHTECKIGE FISHEYE-VIEW als kombinierte Übertragungs- und Anzeigetechnik für große Rasterbilder, die Bandbreite und Bildschirmplatz spart sowie
- ein Verfahren zur progressiven Übertragung von Farbtabellenbildern, bei dem feine Details im Bild durch Verfeinerung in der Farbtiefe eher erkennbar sind als bei den etablierten Standardverfahren interlaced GIF und interlaced PNG, das aber vergleichbare bzw. bessere Kompressionsraten liefert

Kapitel 2

Stand der Forschung

2.1 Überblick über die Teilaspekte

Für viele Teilprobleme des in Kapitel 1 skizzierten Problembereiches (wie Kompression, Übertragung, RoI und LoD) existieren bereits Lösungen, die zum Teil mit anderen Zielsetzungen oder für andere Anwendungsbereiche erarbeitet wurden und die für die Bildübertragung in mobilen Umgebungen angepasst und erweitert werden müssen. Dieses Kapitel stellt solche Lösungsansätze vor.

Da viele Methoden von der Bildklasse abhängig sind, wird zunächst in 2.2 eine häufig verwendete Klasseneinteilung für Rasterbilder beschrieben.

Die Vielzahl der für verschiedene Bildklassen entwickelten Kodierungsverfahren legt die Idee nahe, eine Auswahl des am besten geeigneten Kompressionsverfahrens automatisch vorzunehmen und dieses ebenfalls automatisch zu parametrisieren. Abschnitt 2.3 stellt einige existierende Ansätze dazu sowie ihre Schwachstellen vor.

Zur besseren Ausnutzung der geringen Übertragungsbandbreite ist die Verwendung von progressiven Bildkompressionstechniken und flexiblen Übertragungsverfahren notwendig. Progressive Kodierung heißt, am Anfang des kodierten Datenstromes die „wichtigste“ Bildinformation zu verschlüsseln und diese sukzessive um weitere Details zu ergänzen. Wird ein solcher Datenstrom über einen schmalbandigen Kanal übertragen und gleichzeitig dargestellt, ist durch wiederholte Aktualisierung der Anzeige während der Übertragung eine frühe Erkennbarkeit des groben Bildinhaltes sichergestellt, kombiniert mit der Darstellung von immer mehr Details bei zunehmender Übertragungszeit. Dieses Prinzip der *progressiven Bildübertragung* wurde erstmals 1979 von Tanimoto [Tan79][1] beschrieben. Abschnitt 2.4 diskutiert relevante progressive Bildkodierungs- und -übertragungsverfahren aus der wissenschaftlichen Literatur und stellt ein im Rahmen dieser Arbeit entwickeltes Klassifizierungsschema zur Einordnung dieser Verfahren vor.

Zusätzlich zur Kompression können weitere Bandbreiteneinsparungen durch den Einsatz von Regions of Interest erzielt werden. Existierende Arbeiten auf diesem Gebiet werden in 2.5 zusammengefasst.

Farbtabellenbilder sind eine Bildklasse, deren progressive Übertragung bisher eher vernachlässigt wurde. Als Voraussetzung für die Entwicklung eines eigenen alternativen Ansatzes zur progressiven Kodierung von Bildern dieser Klasse in Kapitel 6 werden in 2.6 relevante Verfahren aus der Literatur zur Kodierung von Farbtabellenbildern vorgestellt.

[1] Titel der Arbeit: „Image transmission with gross information first“.

Neben der geringen Bandbreite bieten mobile Geräte nur eine eingeschränkte Displayfläche. FishEye-Techniken ermöglichen eine Fokus-und-Kontext-Darstellung, die wichtige Bildteile detailliert und unwichtige, nur für einen Überblick notwendige Bildbereiche Platz sparend darstellt. Für mobile Umgebungen bietet es sich an, eine solche Anzeigetechnik mit dem Übertragungsmechanismus zu kombinieren, um nur für die Anzeige benötigte Daten zu übertragen. Abschnitt 5.4 schlägt eine solche Technik vor; als Voraussetzung werden ausgewählte Arbeiten zu FishEye-Views in 2.7 vorgestellt.

Abschnitt 2.8 fasst die Fähigkeiten existierender Ansätze zusammen, zeigt Lücken auf und motiviert so die in den folgenden Kapiteln dargestellten eigenen Arbeiten.

2.2 Klassifizierung von Rasterbildern

Rasterbilder wurden bereits allgemein als zweidimensionale Pixelfelder eingeführt. Sie werden häufig nach dem Typ der Pixelwerte als *Binärbilder*, *Graustufenbilder*, *Farbtabellenbilder* oder *Echtfarbbilder* klassifiziert.

Binärbilder unterscheiden je Pixel nur die beiden Werte „schwarz“ und „weiß“. Je Pixel wird dazu ein Bit gespeichert. Eine Erweiterung stellen *Graustufenbilder* dar, die einen Grauwert pro Pixel speichern, wobei dieser Grauwert die Helligkeit des Pixels angibt. In Multimediasystemen erfolgt die Speicherung meist mit 256 Graustufen (8 Bits pro Pixel), im medizinischen Bereich haben sich 4096 Graustufen (12 Bits pro Pixel) durchgesetzt.

Digitale *Farbbilder* können auf zwei verschiedene Arten repräsentiert werden, als *Echtfarbbilder* oder als *Farbtabellenbilder*. Die Kodierungsformen unterscheiden sich darin, wie die Farbe der Pixel gespeichert wird. Echtfarbbilder (True-Color-Bilder) bestehen meist aus 3 *Farbkanälen* (z.B. *RGB*, siehe 2.2). Pro Pixel wird im True-Color-Format für jeden Farbkanal ein Byte gespeichert. Daneben gibt es das so genannte HiColor-Format, das in 16 Bits pro Pixel zwei Farbkanäle mit 5 Bits und einen Kanal mit 6 Bits speichert. Bei Bildern, die viele Farben enthalten, ist die Speicherung als Echtfarbbilder angebracht.

Bilder mit wenigen (in der Regel bis zu 256) Farben können effizienter als Farbtabellenbilder gespeichert werden. Dabei wird die Farbinformation jeder im Bild vorkommenden Farbe (in der Regel als *RGB*-Tripel) in einer Farbtabelle abgelegt. Für jedes Pixel wird mit 4 oder 8 Bits ein Index in diese Tabelle gespeichert. Farbtabellenbilder benötigen daher weniger Speicherplatz. Diese Bildklasse wird z.B. bei Cartoons, Logos, Bildschirmabzügen von Computerprogrammen, technischen Zeichnungen und Geschäftsgrafiken häufig verwendet. Auf preiswerter, älterer bzw. mobiler Display-Hardware sind häufig nur Farbtabellenbilder darstellbar, da diese Geräte maximal 256 Farben gleichzeitig anzeigen können. Echtfarbbilder müssen zur Anzeige auf solchen Geräten in Farbtabellenbilder umgewandelt werden. Zur Konvertierung eines Echtfarbbildes in ein Farbtabellenbild mit N Farben kommen Farbquantisierungsverfahren [Hec82, PG88, GP90, Ver95] zum Einsatz. Dabei wird zunächst die Menge aller Farben nach verschiedenen Kriterien (z.B. Histogramminformationen [Hec82], Lage der Farben im Farbraum [PG88, GP90] oder durch Nutzung von Lernalgorithmen [Ver95]) in N Teilmengen unterteilt. Für jede dieser Teilmengen wird dann eine repräsentative Farbe ausgewählt, die in die Farbtabelle eingetragen wird. In einem dritten Schritt wird jedem Pixel der entsprechende Indexwert zugewiesen.

Unabhängig davon, ob es sich um Farbtabellen- oder Echtfarbbilder handelt, werden die Farben in der Regel im *RGB*-Farbsystem beschrieben. Hierbei wird pro Pixel ein Tupel von drei Intensitätswerten für die Primärfarben Rot, Grün und Blau gespeichert. Der *RGB*-Farbraum ist aber aus zwei Gründen für die Bildkompression ungünstig: Zum einen bestehen starke Korrelationen zwischen den Farbkanälen, und zum anderen kann der Fakt nicht ausgenutzt werden, dass das menschliche Auge für Farbinformationen nicht so empfindlich

ist wie für Helligkeitsinformationen.

Daher wird für die Kompression von Echtfarbbildern häufig das YC_bC_r-System verwendet. Hierbei stellt der Y-Kanal den Helligkeits- oder *Luminanz*-Anteil des Bildes dar, die *Chrominanz*-Kanäle C_b und C_r sind *Farbdifferenzsignale*. Die drei Kanäle YC_bC_r sind weitgehend dekorreliert, und die Datenmenge in den Chrominanz-Kanälen kann durch Unterabtastung reduziert werden, ohne dass die wahrgenommene Bildqualität signifikant leidet. Weitere dem YC_bC_r-System ähnliche Farbsysteme sind YIQ und YUV.

Die Umrechnung zwischen *RGB* und YC_bC_r ist wie folgt definiert [L$^+$]:

$$Y = 0.29900 \cdot R + 0.58700 \cdot G + 0.11400 \cdot B \quad (2.1)$$
$$C_b = -0.16874 \cdot R - 0.33126 \cdot G + 0.50000 \cdot B \quad (2.2)$$
$$C_r = 0.50000 \cdot R - 0.41869 \cdot G - 0.08131 \cdot B \quad (2.3)$$

$$R = Y + 1.40200 \cdot C_r \quad (2.4)$$
$$G = Y - 0.34414 \cdot C_b - 0.71414 \cdot C_r \quad (2.5)$$
$$B = Y + 1.77200 \cdot C_b \quad (2.6)$$

2.3 Adaptive Steuerung der Bildübertragung

Hier sollen Verfahren kurz dargestellt werden, die Standard-Bildkompressionsmethoden auswählen, steuern und parametrisieren, um die Übertragung zu optimieren. Weiterhin wird auf die Probleme bei der Nutzung solcher Standardkomponenten eingegangen.

Fox und Brewer [FB96] schlugen das Konzept der *Destillation* zur Anpassung von Bildern und anderen speicherintensiven Dokumenttypen an langsame Übertragungskanäle und ressourcenlimitierte Displays vor. Dabei wird Destillation als datentypabhängige, verlustbehaftete Kompressionsmethode aufgefasst, die einen großen Teil der Dokumentsemantik beibehält und in einem so genannten *refinement space* (für Bilder: Skalierung und Farbtiefe) arbeitet. Das Aufwenden zusätzlicher Rechenzeit für den Destillationsprozess spart Bandbreite ein. Das Konzept ist im WWW-Proxy `TranSend` realisiert, der bei Anforderung einer WWW-Seite eingebettete Bilder automatisch „destilliert“. Die Steuerung der Kompressionsrate erfolgt mithilfe von Heuristiken, die sich aber nach Angaben in [FB96] in vielen Fällen als relativ ungenau erweisen. Eine eventuell durch den Nutzer gewünschte Verfeinerung „destillierter“ Bilder kann weder abgestuft noch redundanzfrei erfolgen, sie wird durch Nachladen des Originalbildes realisiert.

Ein ähnliches Verfahren wurde 1998 von Intel unter dem Namen „Intel QuickWeb“ [IQW98] kommerziell angeboten, jedoch nach einem Jahr wieder eingestellt.

Ortega et al. [WVOC97, OCAV97] schlugen das Prinzip des so genannten *Soft Caching* für Bilder vor, das eine Erweiterung des Destillationsprinzips darstellt. Während bei der Destillation das reduzierte Bild erst bei Anforderung des Clients erzeugt wird, kann ein Bild beim Soft Caching in mehreren Auflösungsstufen im Cache des Proxys gespeichert werden. Abhängig von Randbedingungen wird die für den Client am besten geeignete Version übertragen. Eine nachfolgend mögliche Verfeinerungsanforderung des Clients liefert dann wie bei [FB96] das unreduzierte Bild. Bei Nutzung progressiver Kodierverfahren würde es sich anbieten, nur differenzielle Daten zu übertragen; das ist jedoch mit den heute verfügbaren Standardtechnologien GIF und JPEG noch nicht möglich.

2.4 Progressive Bildkodierung und -übertragung

2.4.1 Einordnung und geschichtlicher Überblick

Bildübertragung kann allgemein als Folge der Prozesse *Bildkodierung*, *Datenstromübertragung*, *Bilddekodierung* und *Präsentation* beschrieben werden. Von progressiver Bildkodierung spricht man, wenn der Datenstrom so organisiert ist, dass die Dekodierung eines zusammenhängenden Teils des Datenstromes, angefangen von dessen Beginn, bereits ein vollständiges Pixelfeld in geringerer Detailliertheit liefert. Progressive Übertragung erfordert einen solchen progressiv organisierten Datenstrom und wird dadurch charakterisiert, dass die Prozesse Dekodierung und Präsentation parallel zur Datenstromübertragung mehrfach durchlaufen werden. Dadurch ist der wesentliche Bildinhalt früh im Übertragungsverlauf erkennbar, längere Übertragungszeiten führen zu einem detaillierteren oder qualitativ besseren Bild.

Das Problem der progressiven Bildkodierung und -übertragung ist bereits von einer ganzen Reihe von Autoren behandelt worden und eine Anzahl verschiedener Methoden wurde entwickelt.

In Arbeiten der Siebziger- und Achtzigerjahre (vgl. [Tan79, Kno80, Dre87, DK91, Dür93]) wurden Methoden für die progressive Übertragung von Mehrfachauflösungsbildern benutzt, um Bilder über langsame Verbindungen zwischen Mainframe und Terminals zu übertragen. Bereits 1984 benutzte Lohscheller [Loh84] die progressive Kodierung von DCT-Koeffizienten in einem Bildübertragungssystem.

Mit dem Aufkommen von Multimedia-Systemen verschob sich die Aufmerksamkeit zu Gunsten von Bildkompressionsverfahren. Parallel zur Entwicklung neuer *verlustfreier Bildkompressionsmethoden* (GIF [GIF87], PNG [PNG], Lossless JPEG [JPGa, JPGb] und JBIG [JBG93]) wurden in den letzten Jahren *verlustbehaftete* Verfahren entwickelt, die höhere Kompressionsraten erzielen. Dazu gehören neben dem *JPEG-Standard* [PM93, JPG96] die Kompressionsmethoden *Waveletbasierte Kompression* [Mal89] und *Fraktale Kompression* [Fis94].

Als das WWW in den Neunzigerjahren Popularität erlangte, stieg das Interesse an progressiver Bildübertragung wieder. Vor allem waveletbasierte Kompressionstechniken sind ein populäres Forschungsgebiet. Waveletkodierung ermöglicht eine feinstufige progressive Bildübertragung durch Erzeugung eines so genannten *eingebetteten Datenstromes*, der erstmals von Shapiro [Sha93, S. 3445] wie folgt definiert wurde:

> *Alle Kodierungen desselben Bildes mit niedrigeren Bitraten sind in den Anfang des Bitstromes für die Zielbitrate eingebettet.*

Diese Möglichkeit der eingebetteten Kodierung ermöglicht neben der progressiven Bildübertragung auch die Erzeugung einer komprimierten Bilddatei mit vorgegebener Größe durch einfaches Abschneiden des Datenstromes. Eingebettete Kodierung bedeutete einen großen Fortschritt in der progressiven Bildkodierung und wird in den meisten modernen waveletbasierten Kodierern eingesetzt. Beispiele sind die Zerotree-Weiterentwicklung SPIHT [SP96b], die sich als Referenz für die Beurteilung neuer Kompressionsmethoden etabliert hat, oder die seit einigen Jahren auf dem Markt erhältliche Kompressionssoftware LuRaWave [LWF]. Der aktuell in der Entwicklung befindliche Bildkompressionsstandard ITU-T.800 (besser bekannt als JPEG2000, siehe [JPG99b, JPG99a, SC99]) wird die Forschungsergebnisse zu waveletbasierter Bildkompression integrieren und die Vorteile eingebetteter Kodierung allgemein verfügbar machen.

Im Gegensatz dazu verfügen viele der heute benutzten Bildkompressionsverfahren über weniger leistungsfähige progressive Übertragungsschemata, die von einfachem Interlacing

(GIF und PNG [GIF87, GIF89, PNG]) bis zu Scan-basierter (d.h., teilweise eingebetteter) progressiver Kodierung (JPEG, [PM93]) reichen.

2.4.2 Klassifikation von Bildübertragungsverfahren

Bevor auf einzelne progressive Kodierungsverfahren eingegangen wird, soll zunächst eine Klassifikation zur Einordnung der Verfahren entwickelt werden. Die hierbei verwendeten Merkmale ermöglichen einen Vergleich der Verfahren sowohl bezogen auf ihre Arbeitsweise als auch hinsichtlich bestimmter Leistungsparameter.

Zusätzlich ist anzumerken, dass manche Kodierungsverfahren (wie PNG, GIF oder JPEG) neben progressiven auch nicht-progressive Modi bieten, die sich durch höhere Kompressionsraten oder niedrigere Komplexität auszeichnen. Wegen des Fokus vorliegender Arbeit auf die Übertragung mit gleichzeitiger Darstellung finden diese Modi jedoch nur am Rand Berücksichtigung.

2.4.2.1 Klassifikation nach der Übertragungssteuerung

Eine Möglichkeit, progressive Bildübertragungsverfahren zu klassifizieren, ist eine Einteilung nach den Steuermechanismen während der Übertragung. Folgende prinzipielle Einteilung ist möglich:

Nicht-interaktive Verfahren. Hierbei erfolgt die Kodierung offline, der kodierte Datenstrom wird als Datei auf einem Server abgelegt und auf Anforderung übertragen. Das Übertragungsprotokoll kann einfach gehalten werden und muss nur die sequenzielle Übertragung eines Datenstromes ermöglichen. Diese Verfahren unterstützen in der Regel ausschließlich progressive Verfeinerung. Ein Beispiel ist die Einbindung von interlaced-GIF-Bildern oder progressive-JPEG-Dateien in HTML-Dokumente und deren Übertragung über das `HTTP`-Protokoll. In Verbindung mit einer verfügbaren Caching-Infrastruktur ist auch ein rudimentäres Detail-on-Demand möglich: eine abgebrochene Übertragung kann mit dem ersten noch nicht übertragenen Byte des Datenstromes fortgesetzt werden. Das wird beispielsweise bei LuRaWave [LWF] realisiert.

Interaktive Verfahren. Im Gegensatz zu nicht-interaktiven Verfahren erfolgt hier die Kodierung auf einem Server online, also parallel zur Übertragung. Über einen Rückkanal vom Endgerät zum Server kann der Bildbetrachter direkt die Bildkodierung steuern. Zur Übertragung ist ein Protokoll erforderlich, das diese Steuerinformationen übertragen und zwischen Enkoder und Dekoder konsistent halten kann. Das Schema ermöglicht komplexe Detail-on-Demand-Szenarien und damit eine sehr effiziente Nutzung der Übertragungs- und Displayressourcen. Dieser Vorteil wird jedoch durch einen höheren Rechenaufwand auf dem Server erkauft. Vertreter dieser Klasse sind beispielsweise das System von Lohscheller [Loh84] oder die in Kapitel 5 dieser Arbeit vorgeschlagenen Anwendungen.

Zwischenformen. Neben der interaktiven Steuerung des Kodierungsprozesses zur Realisierung von Detail-on-Demand ist auch eine Steuerung der Übertragung eines entsprechend strukturierten Datenstromes denkbar. Hierbei wird der Datenstrom bei der Kodierung mit Informationen angereichert, die ohne eine Dekodierung die Zuordnung von Datenstromteilen zu Detaillierungsstufen bzw. Bildbereichen ermöglichen. Da keine Neukodierung auf Anforderung während der Übertragung erfolgt, ist die Serverlast geringer. Das benötigte Übertragungsprotokoll ist jedoch nicht so flexibel wie bei der interaktiv gesteuerten Kodierung. Diese Klasse von Übertragungsver-

fahren kann beispielsweise auf der Basis von JPEG2000 [JPG99b, JPG99a, SC99] realisiert werden.

2.4.2.2 Klassifikation nach der Enkoderarchitektur

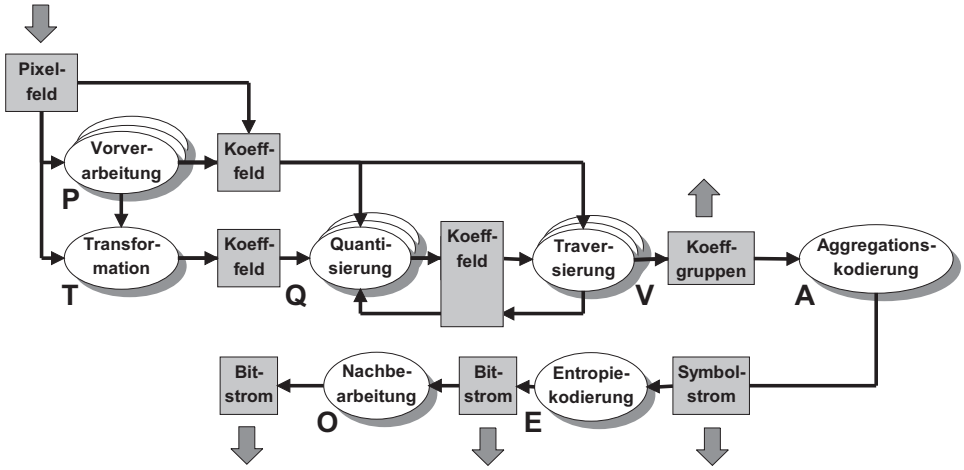


Abbildung 2.1: Kodierungspipeline. Die Buchstaben an den Teilprozessen werden in den nachfolgenden Abschnitten zur Klassifizierung der einzelnen Verfahren verwendet.

Abbildung 2.1 veranschaulicht die bei der Bildkodierung beteiligten Prozesse und ihre möglichen Datenflüsse. Diese Pipeline wurde so allgemein gehalten, dass sich alle im Folgenden vorzustellenden progressiven Kodierverfahren dort einordnen lassen. Nicht jedes Kodierverfahren durchläuft dabei alle möglichen Teilprozesse oder Datenflüsse. Die Dekodierung muss für jeden Kodierungsprozess den inversen Prozess durchlaufen, außer der Traversierung, die bei Enkoder und Dekoder identisch sein muss.

Ein Pixelfeld kann zunächst entweder eine *Transformation* durchlaufen, in einem *Vorverarbeitungsschritt* für die Kodierung vorbereitet werden oder unverändert bleiben. Bei der Transformation (z.B. Diskrete Cosinus-Transformation oder Wavelet-Transformation) wird das Bild vom Ortsraum in eine andere Repräsentation transformiert, wobei eine m:n-Zuordnung zwischen Pixeln und Koeffizienten besteht. Bei der Vorverarbeitung (z.B. Prädiktion, Quadtree-Darstellung) bleibt demgegenüber die Zuordnung zwischen Pixeln im Bild und Koeffizienten, also der Ortsbezug, erhalten. Das Ergebnis des ersten Schrittes wird als *Koeffizientenfeld* bezeichnet[2]. Gängige Transformationsverfahren werden in 2.4.3.1 beschrieben.

Ein *Quantisierungsprozess* kann auf das Koeffizientenfeld angewendet werden, um die Koeffizienten mit geringerer Genauigkeit darzustellen. Hierbei tritt immer ein *Informationsverlust* auf, das Ergebnis der Quantisierung ist aber auf Grund seiner niedrigeren Entropie besser komprimierbar. Es gibt verschiedene Quantisierungsverfahren. Die einfachste Möglichkeit besteht in der *skalaren* Quantisierung. Dabei wird bei der Kodierung jeder Koeffizient durch einen skalaren Wert dividiert; die Dekodierung erfolgt durch Multiplikation mit demselben Wert. Diese *Quantisierungsfaktoren* können für verschiedene Koeffizienten unterschiedliche Werte annehmen. Aufwändigere Quantisierungsverfahren verwenden *Vektorquantisierung* [CGV96], wobei ein Vektor von Koeffizientenblöcken in einen Vektor von Indizes umgewandelt wird, indem für jeden Block der ähnlichste Block aus einem

[2] Das soll zur Vereinfachung auch gelten, wenn dieser Schritt übersprungen wurde.

so genannten Kodebuch selektiert und dessen Index ausgegeben wird. Ein dekodierter Koeffizientenblock ergibt sich aus dem Kodebucheintrag, der durch den entsprechenden Indexwert adressiert wird. Eine Abwandlung der skalaren Quantisierung ist die *sukzessive Approximation*, bei der die Koeffizienten bitebenenweise kodiert werden. Mit jeder neuen Bitebene wird die Quantisierungsschrittweite halbiert. Sukzessive Approximation wird häufig für die progressive Bildkodierung angewendet, da sie eine Verfeinerung der Genauigkeit der Koeffizienten bzw. Pixelwerte erlaubt.

Zum Zwecke der Kodierung wird das Koeffizientenfeld *traversiert*, das heißt, die Koeffizienten werden in einer bestimmten Reihenfolge besucht. Die Traversierung realisiert oft eine Umordnung der Koeffizienten, die für die Kodierung günstig ist. Beispielsweise werden im JPEG-Standard aufgrund der Eigenschaften der Koeffizienten diese in einer Zick-Zack-Folge besucht, um möglichst lange Folgen von Nullen zu erzeugen. Auch die Traversierung ist ein wichtiger Prozess in der progressiven Bildkodierung, da hier beispielsweise durch mehrfaches Durchlaufen des Pixelfeldes eine Verfeinerung in der räumlichen Auflösung möglich ist (z.B. [GIF89], [PNG]).

Bei der Traversierung werden Gruppen von Koeffizienten ermittelt. Ziel der darauf folgenden *Aggregationskodierung* (einige Methoden werden im 2.4.3.2 näher vorgestellt) ist die Erzeugung eines Symbolstroms mit niedriger Entropie durch Zusammenfassen mehrerer Koeffizienten zu einem Symbol unter Ausnutzung von Kontextwissen oder von Wissen über statistische Zusammenhänge bzw. über Korrelationen zwischen Koeffizienten. Dieser Schritt wird in vielen Publikationen auch als statistische Modellierung [PM93], Semi-Kodierung [SR93], Vor- oder Präkodierung [Str97] bezeichnet. Hier soll stattdessen zur Einordnung von Verfahren der Begriff „Aggregationskodierung“ verwendet werden, da dies bei einigen Verfahren bereits der letzte Schritt der Pipeline sein kann. Dieser Symbolstrom wird danach teilweise noch mit einem Entropiekodierer komprimiert. Dabei wird ein Strom von Symbolen mit einer möglichst geringen Anzahl von Bits repräsentiert. Die untere Grenze der Bitanzahl ist dabei durch die Entropie dieses Symbolstromes bestimmt. Entropiekodierungsverfahren sind Huffman-Kodierung [Huf52] und arithmetische Kodierung [WNC87]. Auf sie wird in 2.4.3.3 näher eingegangen.

Das Schema in Abbildung 2.1 wurde entwickelt, um für jede der in 2.4.5 vorzustellenden Bildkodierungsmethoden die an der Kodierung beteiligten Teilprozesse in kompakter Form darstellen zu können. Dazu wird eine Folge von Kürzeln der Teilprozesse angegeben, wobei ein +-Zeichen nach einem Kürzel das mehrmalige Durchlaufen des entsprechenden Teilprozesses zum Zwecke der progressiven Kodierung repräsentiert. Beispielsweise ist der Ablauf des interlaced-PNG-Verfahrens (siehe 2.4.5.3) durch die Schritte *Vorverarbeitung*, *Traversierung* (in mehreren Durchläufen), *Aggregationskodierung* und *Entropiekodierung* charakterisiert und kann daher durch die Folge PV^+AE dargestellt werden.

2.4.2.3 Klassifikation nach dem Informationsverlust

Eine übliche Klassifikation teilt Bildkodierverfahren nach dem Informationsverlust in *verlustfreie* und *verlustbehaftete* Methoden ein. Bei der verlustfreien Kodierung wird versucht, durch geeignete Verfahren die Redundanzen zwischen den Pixeln (*Interpixelredundanz*) auszunutzen, um eine möglichst hohe Kompressionsrate zu erreichen. Zu den verlustfreien Verfahren zählen GIF [GIF89], PNG [PNG], JBIG [JBG93] und JPEG-LS [WS99]. Dekodiert man ein derart kodiertes Bild wieder, so ist es mit dem Original identisch.

Verlustbehaftete Kodierverfahren nutzen die *psycho-visuelle Redundanz* in Bildern aus, um nicht wahrnehmbare bzw. „unwichtige“ Information aus dem Bild zu entfernen. Das erfolgt in der Regel durch Quantisierung von Koeffizienten, die durch Anwendung einer Transformation auf das Bild berechnet wurden. Ein Beispiel ist JPEG [PM93]. Verluste treten nicht nur durch Quantisierung auf, sondern können auch durch schnelle Näherungsimplementie-

rungen von Transformationen entstehen.

Progressive Kodierverfahren (siehe unten) nehmen hier eine Sonderstellung ein. Wenn der gesamte kodierte Datenstrom auch dekodiert wird, sind diese Verfahren unter den folgenden Bedingungen verlustfrei:

- verlustlos implementierte Transformation oder Fehlen eines Transformationsschrittes,
- kein separater Quantisierungsschritt vor der sukzessiven Approximation,
- Kodierung sämtlicher Bits aller Koeffizienten bzw. Pixel.

Bricht man Kodierung bzw. Dekodierung früher ab, so erhält man ein verlustbehaftetes Verfahren. Beispiele sind EZW [Sha93] und SPIHT [SP96b] auf der Basis der Wavelet-Transformation oder das BCQ-Verfahren [DK91] ohne Transformation.

2.4.2.4 Klassifikation nach der Domäne des Koeffizientenfeldes

Aus der Kodierungspipeline ergibt sich, ob das Koeffizientenfeld im *Ortsraum* oder als *transformierte Repräsentation* vorliegt. Im ersten Fall erfolgt keine Transformation, und die Pixel bilden direkt das Koeffizientenfeld. Der zweite Fall benutzt eine Transformation, um die Pixel in Koeffizienten einer transformierten Repräsentation umzurechnen.

Im Ortsraum können die Pixelwerte *direkt* die Farbe des entsprechenden Pixels oder *indirekt* einen Zeiger auf einen Eintrag einer Farbtabelle speichern.

Somit gibt es drei Formen der Darstellung von Koeffizientenfeldern:

- Ortsraum mit direkter Pixelkodierung,
- Ortsraum mit indizierter Pixelkodierung,
- Transformierte Repräsentation.

2.4.3 Ausgewählte Teilschritte der Kodierungspipeline

Dieser Abschnitt stellt ausgewählte Teilprozesse der Kodierung vor, die zum besseren Verständnis der folgenden Abschnitte benötigt werden, aber teilweise auch als Bausteine der in den Kapiteln 4 und 6 entwickelten Verfahren dienen.

2.4.3.1 Transformationen

Transformationsverfahren dienen dazu, Bilddaten in eine andere Repräsentation umzuwandeln, die besser für eine – in der Regel verlustbehaftete – Kodierung geeignet ist. Dabei wird eine Dekorrelation erreicht, die es ermöglicht, „unwichtige“ (*psycho-visuell redundante*) Teile der Bildinformation zu isolieren, um sie wegzulassen oder stark zu quantisieren und durch diesen Informationsverlust eine bessere Komprimierbarkeit des Bildes zu erreichen. In einem eingebetteten Datenstrom sind solche Informationen am Ende des Stromes enthalten.

Die am häufigsten eingesetzten Methoden Diskrete Cosinus-Transformation und Dyadische Wavelet-Transformation sollen im Folgenden näher vorgestellt werden. Daneben besitzt die Fraktalzerlegung [Fis94] eine gewisse Bedeutung für Spezialanwendungen, wie z.B. die Texturkomprimierung.

Diskrete Cosinus-Transformation. Die diskrete Cosinus-Transformation (DCT) stellt ein Signal als Linearkombination von Cosinusfunktionen dar. Als Basis wird eine Anzahl von Funktionen mit unterschiedlichen Ortsfrequenzen in horizontaler und vertikaler Richtung ausgewählt, so dass die DCT ein Signal aus dem Orts- in den Frequenzraum transformiert. Viele Standardverfahren zur Stand- und Bewegtbildkodierung (z.B. JPEG, MPEG, H.26x) verwenden $8 \cdot 8 = 64$ Basisfunktionen.Vor der Transformation wird das Bild daher in Blöcke der Größe 8x8 Pixel zerlegt.

Jeder Bildblock wird separat transformiert. Ergebnis ist ein 8x8-Block von *DCT-Koeffizienten*. Dieser enthält im linken oberen Feld (*DC-Koeffizient*) den Gleichanteil des Blockes. Alle weiteren Koeffizienten werden *AC-Koeffizienten* genannt und repräsentieren jeweils eine Kombination von horizontalen und vertikalen räumlichen Frequenzkomponenten. Koeffizienten, die auf einer parallel zur Nebendiagonale liegenden Gerade angeordnet sind, bilden ein so genanntes *Spektralband*. Durch die Transformation in den Frequenzraum können Komponenten mit unterschiedlicher Frequenz z.B. durch einen nachfolgenden Quantisierungsschritt verschieden stark quantisiert werden.

Die DCT kann durch die inverse DCT (IDCT) rückgängig gemacht werden. Verwendet man Koeffizienten mit unendlicher Genauigkeit, so ist dieser Prozess verlustfrei. Die folgenden Gleichungen definieren DCT ((2.7)) und IDCT ((2.8)) bei einer Blockgröße von 8x8 (Quelle: [PM93, S. 40f]). Dabei steht $s(y,x)$ für einen 8x8-Pixel-Block im Bild und $S(y,x)$ für den entsprechenden Block von DCT-Koeffizienten.

$$
\begin{aligned}
F(x,y,u,v) &= \cos[(2x+1)u\pi/16]\cos[(2y+1)v\pi/16] \\
C(w) &= \begin{cases} 1/\sqrt{2} & \text{wenn} \quad w=0 \\ 1 & \text{wenn} \quad w>0 \end{cases} \\
S(v,u) &= \frac{C(v)}{2}\frac{C(u)}{2}\sum_{y=0}^{7}\sum_{x=0}^{7} s(y,x)F(x,y,u,v) \qquad (2.7) \\
s(y,x) &= \sum_{v=0}^{7}\frac{C(v)}{2}\sum_{u=0}^{7}\frac{C(u)}{2}S(v,u)F(x,y,u,v) \qquad (2.8)
\end{aligned}
$$

Dyadische Wavelet-Transformation. Die *dyadische Wavelet-Transformation* (DWT) wurde 1989 von Mallat [Mal89] vorgestellt und erlangte schnell allgemeines Interesse (siehe z.B. [Fou95]). Erste Arbeiten zu dieser Thematik [Haa10] reichen jedoch bis zum Anfang unseres Jahrhunderts zurück. Bei der Wavelet-Transformation wird ein Signal durch Basisfunktionen endlicher Länge (Wavelets) beschrieben, die durch Skalierung und Verschiebung aus einem so genannten *Mutterwavelet* erzeugt werden. Sweldens gibt als intuitive Definition für Wavelets folgendes an:

> *„Wavelets are building blocks that can quickly decorrelate data."* [Swe95]

Diese Definition impliziert eine hervorragende Eignung der Wavelet-Transformation zur Kompression stark korrelierter Daten, wie z.B. Bilddaten.

Die Wavelet-Transformation diskreter Signale erfolgt durch *Filterbänke*, die aus jeweils einem Tiefpass und einem Hochpass bestehen. Eine *Analysefilterbank* erzeugt eine Mehrfachauflösungsrepräsentation eines Signals, indem es dieses in *Oktavbänder*[3] zerlegt. Durch die inverse DWT (IDWT) ist daraus mithilfe einer *Synthesefilterbank* das Originalsignal rekonstruierbar. Zahlreiche Arbeiten befassten sich damit, Filterbänke für die Wavelet-Transformation zu entwerfen. Stellvertretend seien hier [CDF92, ABMD92, CDSBL98] genannt. Obwohl bereits vor dem Aufkommen der Wavelet-Theorie Arbeiten zur Bild-

[3] Durch Halbierung der Frequenz abgestufte Bänder.

kodierung mit Filterbänken erfolgten (z.B. [AJ81, BA83, ASH87, GT88]), erwiesen sich Wavelets als ein Werkzeug zum Design effizienterer Filter.

Abbildung 2.2 illustriert die Prozesse DWT und IDWT mithilfe einer Filterbank, die aus den Analysefiltern Lo und Hi sowie den Synthese- oder Rekonstruktionsfiltern Lo' und Hi' besteht. Die DWT ist separierbar, das heißt, ein zweidimensionales Signal kann durch aufeinander folgende Anwendung eindimensionaler Filter in Spalten- und Zeilenrichtung transformiert werden.

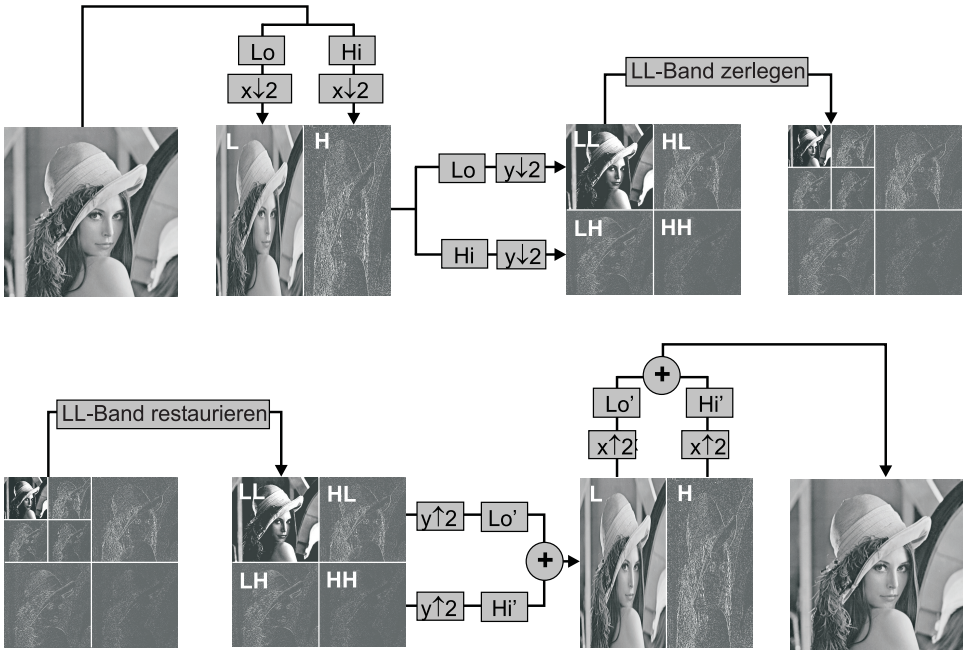


Abbildung 2.2: Zweistufige dyadische Wavelet-Transformation (oben) und ihre Invertierung (unten).

Bei der Transformation wird zunächst jede Zeile des Ausgangsbildes sowohl tiefpass- als auch hochpassgefiltert. Die gefilterten Signale werden danach mit halber Frequenz unterabgetastet. In einem nächsten Durchlauf wird dieser Schritt analog spaltenweise angewendet. Ergebnis ist eine *einstufige Wavelet-Zerlegung* des Bildes, die vier Bänder enthält: das LL-Band, das einem verkleinerten Originalbild entspricht, sowie drei Differenzbänder LH, HL und HH. Diese Zerlegung wird nun mehrmals rekursiv für das LL-Band wiederholt. Für die Bildkodierung haben sich 4 bis 7 Zerlegungsstufen als günstig erwiesen. Die Rekonstruktion erfolgt analog durch stufenweises Rückgängigmachen der DWT. Die Teilbänder werden zeilen- und spaltenweise durch Einfügen von Nullwerten nach jedem Abtastwert aufgetastet, mit den entsprechenden Analysefiltern der Filterbank gefiltert und sampleweise aufaddiert.

Eine Filterbank muss in der Lage sein, das Signal perfekt zu rekonstruieren. Dazu existieren mit orthogonalen [Dau88] und biorthogonalen [CDF92] Wavelets verschiedene Filterfamilien. In [SN97, Str97] sind ausführliche Informationen zu diesem Thema zu finden. Tabelle 2.1 listet die Filterkerne der biorthogonalen Wavelets *9/7* [CDF92, S. 551] und *5/3* [CDF92, S. 542] auf, die als Basis für das in Kapitel 4 entwickelte Bildübertragungssystem genutzt werden. Nach Untersuchungen verschiedener Filter in [Str97] offeriert das 9/7-Wavelet das beste Verhältnis zwischen PSNR und Kompressionsrate, ist aber relativ rechenaufwändig. Das 5/3-Wavelet erzielt ein geringfügig schlechteres PSNR-Kompressionsraten-Verhältnis, bietet aber eine bessere Rechenperformance.

Bei der Umsetzung der DWT mit biorthogonalen Wavelets ist zu beachten, dass Hochpass und Tiefpass um eine Position gegeneinander verschoben sind. In der Praxis bedeutet das, dass bei der Rekonstruktion Lo' auf die Abtastwerte des aufgetasteten Tiefpasssignals mit gerader Position angewendet wird, Hi' auf die Samples mit ungerader Position. Eine ausführliche Beschreibung der Implementierung der DWT mit biorthogonalen Wavelets findet der interessierte Leser in [Str97, Kapitel 5]. Dort und in [LCK95] wird auch beschrieben, wie die *Randerweiterung* zu erfolgen hat, die den Fall behandelt, dass der Filterkern über den Rand des zu transformierenden Bildes „hinausragt".

Filter	n	0	±1	±2	±3	±4
5/3	Lo	3/4	1/4	-1/8		
	Hi	1/2	-1/4			
	Lo'	1/2	1/4			
	Hi'	3/4	-1/4	-1/8		
9/7	Lo	0.60294902	0.26686412	-0.07822327	-0.01686412	0.02674876
	Hi	0.55754353	-0.29563588	-0.02877176	0.04563588	
	Lo'	0.55754353	0.29563588	-0.02877176	-0.04563588	
	Hi'	0.60294902	-0.26686412	-0.07822327	0.01686412	0.02674876

Tabelle 2.1: Koeffizienten einiger biorthogonaler Wavelet-Filter[4].

Die in Tabelle 2.1 angegebenen Koeffizienten realisieren Analysefilter, die das Tiefpasssignal nicht verstärken. Das impliziert jedoch nach [Str97, Abschnitt 6.2.2], dass die angegebenen Koeffizienten der Rekonstruktionsfilter verdoppelt werden müssen und somit eine Verstärkung von 2 besitzen. Dies führt beim Einsatz sukzessiver Approximation (also bitebenenweiser Kodierung) dazu, dass der resultierende Quantisierungsfehler in niederfrequenten Teilbändern bei der Rekonstruktion verstärkt wird und sich negativ auf die Bildqualität auswirkt. Multipliziert man die Koeffizienten aller Filter mit $\sqrt{2}$, statt die Koeffizienten der Synthesefilter zu verdoppeln, so haben Analyse- und Synthesefilter dieselbe Verstärkung, und Qualitätsprobleme bei der sukzessiven Approximation treten nicht mehr auf.

Neben biorthogonalen Wavelets findet in dieser Arbeit die Haar-Transformation [Haa10] Anwendung, die durch folgende Formeln definiert ist:

$$\begin{aligned} i &= 0(1)n/2-1 \\ l_i &= (s_{2i}+s_{2i+1})/\sqrt{2} && (2.9)\\ h_i &= (s_{2i}-s_{2i+1})/\sqrt{2} && (2.10)\\ s'_{2i} &= (l_i+h_i)/\sqrt{2} && (2.11)\\ s'_{2i+1} &= (l_i-h_i)/\sqrt{2} && (2.12) \end{aligned}$$

Dabei bezeichnet s das Eingangssignal, s' das rekonstruierte Signal, l das Tiefpasssignal, h das Hochpasssignal und n die Anzahl der Abtastwerte von s. Unterabtastung und Auftastung werden durch die Laufvariable i realisiert.

Nach der Wavelet-Transformation ist in allen Bändern, die mit einem Hochpassfilter bearbeitet wurden, die Redundanz zwischen benachbarten Koeffizienten relativ gering (zur Ausnutzung von Restredundanzen siehe 2.4.5.8). Im LL-Band bestehen jedoch weiterhin starke Redundanzen zwischen benachbarten Pixeln. Daher wird im Anschluss an die

[4] Die nach der Länge der Filterkerne als 5/3-Wavelet bezeichneten Filter wurden erstmals in [CDF92] beschrieben und dort als „Wavelet mit $N=2,\widetilde{N}=2$" eingeführt, das 9/7-Wavelet analog als „Wavelet mit $N=4,\widetilde{N}=4$". Andere Autoren (z.B. [Str97]) geben diese Filter ebenfalls an. Bei biorthogonalen Wavelets reicht die Bestimmung der Koeffizienten für Lo und Lo' aus, Hi und Hi' können daraus durch Umstellung einiger Vorzeichen berechnet werden (siehe [Str97]).

Transformation der Gleichanteil (Durchschnitt aller Koeffizienten) des LL-Bandes ermittelt und von allen Koeffizienten dieses Bandes subtrahiert. Dieser Durchschnittswert wird zum Rückgängigmachen der Kodierung benötigt, muss also als Seiteninformation übertragen werden.

Zahlreiche Optimierungen der Wavelet-Transformation wurden vorgeschlagen. Dazu gehören in erster Linie Integer-Transformationen wie die S+P-Transformation [SP96a] oder das Lifting-Schema [Swe95], die gegenüber reellwertigen Koeffizienten Rechenzeit und Speicherplatz sparen. Arbeiten zu dieser Thematik siehe [CF97, CDSBL98, Swe95, Swe96a, Swe96b]. Ein weiteres Konzept sind *Wavelet Packets* [Mey93, Wic94, GR98], bei denen auch die Hochpassbänder weiter zerlegt werden können, um Restkorrelationen zu eliminieren. Eine solche Zerlegung heißt Basis, das Finden der günstigsten Basis ist ein Optimierungsproblem. Wavelet Packets ermöglichen durch die bessere Dekorrelation eine geringfügig bessere Kompression als das dyadische Schema, konnten sich aber durch den höheren Rechenzeitbedarf bisher in der Praxis nicht durchsetzen. Für Hardware-Implementierungen und spezielle Anwendungen z.B. in der Raumfahrt wurden Varianten der Transformation entwickelt, die sich durch einen sehr geringen Speicherplatzbedarf auszeichnen, siehe z.B. [CO98, CO99a, CO99b, OTW+99a, OTW+99b, R+99].

2.4.3.2 Aggregationskodierungsverfahren

Dieser Abschnitt stellt einige gebräuchliche Aggregationskodierverfahren vor. Hierbei soll zwischen *generischen* und *quellenspezifischen* Verfahren unterschieden werden.

Zu den generischen Verfahren gehören Lauflängen-, Golomb-, LZW- und LZ77-Kodierung. Die ersten beiden Methoden sind besonders für Symbolfolgen geeignet, die lange Teilfolgen gleicher Symbole enthalten; die letzten beiden Verfahren kodieren Folgen mit häufiger Wiederholung von Teilfolgen effizient.

Quellenspezifische Verfahren setzen Wissen über spezielle Eigenschaften der zu kodierenden Quelle voraus. Zu dieser Gruppe gehören die Kontextmodellierung (2.4.5.8) sowie die Kodierverfahren EZW (2.4.5.6) und SPIHT (2.4.5.7) für Wavelet-Koeffizienten.

Quellenspezifische Verfahren werden in den entsprechenden Abschnitten diskutiert; die generischen sollen im Folgenden kurz beschrieben werden.

Lauflängenkodierung. *Lauflängenkodierung* fasst Folgen gleicher Symbole zu einem neuen Symbol zusammen, das die Länge der Folge sowie das mehrfach wiederholte Symbol repräsentiert. Sie erfolgt beispielsweise im JPEG-Standard, um die durch die Traversierung erzeugten langen Folgen von Nullen effizient zu verschlüsseln.

Golomb-Kodierung. Eine abgewandelte Form der Lauflängenkodierung ist die *Golomb-Kodierung*. Sie wird in dem in Kapitel 6 dieser Arbeit entwickelten Kodierverfahren verwendet. Malvar [Mal99] beschreibt den Einsatz des RLR[5]-Enkoders zur Kodierung der Signifikanzinformation von Wavelet-Koeffizienten. Dieser Enkoder arbeitet bitebenenweise und repräsentiert lange Folgen von Null-Bits effizient durch Erzeugung eines so genannten *elementaren Golomb-Kodes* [Gol66].

Der RLR-Enkoder benutzt einen adaptiven Parameter k, der vor Beginn der Kodierung mit 1 initialisiert wird. Als Kodewort wird ein 0-Bit ausgegeben, wenn eine Folge von 2^k Nullen zu kodieren ist. Es wird ein 1-Bit, gefolgt von der k-bittigen binären Repräsentation der Zahl n ausgegeben, wenn eine Folge von n ($n < 2^k$) zu kodierenden Null-Bits durch

[5] RLR: Run-Length / Rice Enkoder, [LJ83].

ein 1-Bit abgeschlossen wird. Um den Parameter k an die Charakteristik der Datenquelle anzupassen, schlägt Malvar vor, k nach der Ausgabe einer 0 zu inkrementieren und nach Ausgabe eines mit 1 beginnenden Kodewortes zu dekrementieren.

LZ77-Kodierung. Dieses Verfahren zur Kompression von eindimensionalen Symbolfolgen wurde 1977 von Lempel und Ziv [ZL77] vorgeschlagen. Es bildet die Grundlage des unter UNIX verbreiteten komprimierten `gzip`-Formates [Deu96b] und wird im PNG-Bilddateiformat verwendet. Die Grundidee besteht darin, beim erstmaligen Auftreten eines Symbolmusters im Eingabestrom dessen Position zu speichern und nachfolgende Wiederholungen dieses Musters durch dessen Länge sowie einen Zeiger auf die Position von dessen ersten Auftreten zu ersetzen. Damit bilden bereits kodierte bzw. dekodierte Teile des Datenstromes in Form eines gleitenden Fensters ein *Wörterbuch* für die Kodierung bzw. Dekodierung nachfolgender Teile. Vielfach wird der entstandene Symbolstrom in einem nachfolgenden Schritt Huffman-kodiert, um die Kompressionsrate zu erhöhen.

LZW-Kodierung. Welch [Wel84] schlug eine Verbesserung des LZ77-Verfahrens vor. Die resultierende Methode wird nach ihm LZW (Lempel-Ziv-Welch) genannt und ist patentiert. Wie bei LZ77 handelt es sich um ein wörterbuchbasiertes Verfahren, das Strings aus einem Wörterbuch durch Kodes ersetzt. Die Erweiterung betrifft die Erzeugung des Wörterbuches. Dieses wird derart initialisiert, dass allen Strings der Länge 1 als Kode der ASCII-Kode des entsprechenden Zeichens zugeordnet ist. Im Verlauf der Kodierung wird das Wörterbuch sukzessive um Einträge größerer Länge wie folgt ergänzt: Enthält der Eingabestrom die Zeichenfolge $c_1c_2...c_{n-1}c_n$, so wird geprüft, ob sich dieser bereits im Wörterbuch befindet. Ist das der Fall, so wird der Kode für diesen Eintrag ausgegeben. Fehlt der Eintrag, so wird $c_1c_2...c_{n-1}c_n$ zum Wörterbuch hinzugefügt, und der Kode für $c_1c_2...c_{n-1}$, gefolgt vom Kode für c_n ausgegeben. In [BR95] wird der Algorithmus einschließlich Implementierung detailliert beschrieben.

2.4.3.3 Entropiekodierungsverfahren

Entropiekodierungsverfahren nutzen die so genannte *Kodierungsredundanz* aus, um einen Strom von Symbolen durch einen Bitstrom mit einer möglichst geringen Anzahl von Bits zu repräsentieren. Ein Maß für die zur Kodierung eines Symbols s_i mindestens notwendige Bitanzahl ist die *Entropie* $H(s_i)$, die für ein Alphabet von n Symbolen wie folgt aus der relativen Häufigkeit p der Symbole im Symbolstrom berechnet werden kann [Ohm95, S. 189]:

$$H(s_i) = -\sum_{i=1}^{n} p(s_i) \cdot \log_2 p(s_i) \tag{2.13}$$

Huffman-Kodierung und *arithmetische Kodierung* sind die am weitesten verbreiteten Verfahren. Die Kodierung kann statisch oder dynamisch erfolgen. Bei der *statischen* Variante wird die Symbolhäufigkeit p im Vorfeld der Kodierung ermittelt und mit dem kodierten Datenstrom als Seiteninformation übertragen, oder es werden Standardhäufigkeiten angenommen. Diese Werte von p bleiben dann im gesamten Verlauf der Kodierung fest. Im Gegensatz dazu passt die *dynamische* oder adaptive Variante die Häufigkeit p ständig während der Kodierung bzw. Dekodierung an die aktuellen Symbolhäufigkeiten an. Das ermöglicht eine höhere Kompressionsrate, da keine Seiteninformation übertragen werden muss und die Entropie immer auf Basis der aktuellen Symbolverteilung ermittelt wird.

Huffman-Kodierung. Bei der Huffman-Kodierung [Huf52] wird jedes Symbol durch eine Bitfolge ganzzahliger Länge dargestellt. Häufig auftretende Symbole werden durch kur-

ze, seltene Symbole durch lange Bitfolgen repräsentiert. Ein Huffman-Kode wird erzeugt, indem alle Symbole anhand ihrer relativen Häufigkeit in einen binären Baum einsortiert werden. Die Kantenfolge des Baumes von der Wurzel zu einem Symbol kann dann in den Kode dieses Symbols übersetzt werden. Eine ausführliche Beschreibung des Algorithmus befindet sich in [Ohm95, S. 199ff].

Huffman-Kodierung ist einfach zu implementieren und erfordert wenig Rechenzeit, benötigt jedoch auf Grund der ganzzahligen Kodewortlänge mehr Bits zur Kodierung, als gemäß der Entropie zu erwarten wäre. Die meisten Huffman-Kodierer arbeiten statisch[6]. Sie müssen daher den Kodebaum als Seiteninformation übertragen und können sich nicht an wechselnde Symbolhäufigkeiten anpassen.

Arithmetische Kodierung. Die arithmetische Kodierung stellt eine Symbolfolge als eine Real-Zahl im Intervall $[0,1]$ mit einer großen Anzahl von Stellen dar. Die Kodierung erfolgt durch sukzessive Intervallschachtelung gemäß der kumulativen relativen Häufigkeit der Symbole. Eine Beschreibung des Algorithmus befindet sich wiederum in [Ohm95, S. 201ff]. Witten et al. [WNC87] stellen eine Implementierung vor.

Arithmetikkodierung kann ein Symbol durch eine nicht-ganzzahlige Anzahl von Bits repräsentieren und nähert somit die Entropie besser an als die Huffman-Kodierung. Das Prinzip erlaubt die einfache Realisierung eines dynamischen Enkoders, der keine Übertragung von Seiteninformation erfordert und eine gute Anpassung an wechselnde Symbolhäufigkeiten ermöglicht. Damit erreicht die arithmetische Kodierung höhere Kompressionsraten als die Huffman-Kodierung. Sie erfordert jedoch für Alphabete mit vielen Symbolen mehr Rechenleistung, und die Algorithmen sind z.T. patentiert.

2.4.4 Bewertung progressiver Kodierverfahren

2.4.4.1 Leistungsparameter

Wichtige Bewertungskriterien für die Leistungsfähigkeit von Bildkodierverfahren sind *Kompressionsrate* und *Bildqualität*. Das Maß „Bildqualität“ ist nur bei verlustbehafteten Kodierverfahren sinnvoll. Die hierbei aus dem Bild entfernte Information äußert sich in verschiedenen *Kompressionsartefakten* (wie z.B. dem „Klötzchenartefakt“ bei DCT-basierten Verfahren, vgl. Abbildung B.4 links), die subjektiv mehr oder weniger störend in Erscheinung treten.

Die Messung der Kompressionsrate erfolgt entweder als *Verhältnis* zwischen den Größen von unkomprimierter und komprimierter Bilddatei oder als *Bitrate*. Letztere gibt an, wie viele Bits pro Pixel (*bpp*) des Bildes in den kodierten Datenstrom geschrieben bzw. aus ihm gelesen werden.

Für die Bewertung der Bildqualität existieren objektive und subjektive Kriterien. Das wichtigste objektive Kriterium ist das *Spitzensignal-Rausch-Verhältnis* PSNR[7]. Es wird in dB gemessen und gibt an, wie stark die durchschnittliche Verfälschung in Bezug auf den größten Pixelwert ist. Für ein Graustufenbild I der Größe n_x x n_y Pixel mit dem maximal möglichen Pixelwert I_{max} sowie ein durch verlustbehaftete Kodierung und nachfolgende Dekodierung aus I entstandenes Bild I' berechnet sich die PSNR wie folgt (in Anlehnung an [Ohm95, S. 186]):

[6] Es existieren jedoch auch dynamische Varianten, z.B. [Knu85].
[7] Englisch: *peak signal to noise ratio*.

$$PSNR(I,I')[dB] = 10 \cdot \log_{10} \frac{n_x \cdot n_y \cdot (I_{max})^2}{\sum_{y=1}^{n_y} \sum_{x=1}^{n_x} (I_{xy} - I'_{xy})^2} \tag{2.14}$$

Die subjektive Bildqualität MOS[8] [Ohm95] wird in Benutzertests ermittelt, bei denen eine Anzahl von Testpersonen komprimierte Bilder betrachten und deren Qualität anhand einer Skala beurteilen muss. Problematisch ist, dass kein einfacher Zusammenhang zwischen PSNR und MOS herstellbar ist. Abbildung B.6 illustriert dies. Dasselbe Bild wurde mit verschiedenen Kompressionsverfahren, die unterschiedliche Artefakte erzeugen, derart komprimiert, dass die PSNR in beiden Fällen gleich ist. Die subjektive Qualität beider Bilder unterscheidet sich. Nach Meinung des Autors ist die des rechten Bildes besser.

Es wurden Methoden entwickelt, die subjektive Qualitätsmaße durch Bildanalyse approximieren [MKA96, Lub95, Mül98]. Diese Methoden sind jedoch häufig komplex oder benötigen Angaben über erst während der Dekodierung verfügbare Parameter wie z.B. den Betrachtungsabstand. Sie haben sich daher zur Steuerung der Kodierung bisher nicht breit durchgesetzt. Qualitätsmaße auf der Basis visueller Maskierung (z.B. [HK97, Wat93]) könnten jedoch Bedeutung zur Steuerung aktueller Bildkodierungsverfahren erlangen. Taubman [Tau99a, Tau99b] demonstrierte die einfache Integrierbarkeit eines solchen Maßes in JPEG2000.

2.4.4.2 Funktionsprinzip der Progression

Bei der progressiven Bildkodierung wird aus einem Bild ein Datenstrom erzeugt, der bereits nach Dekodierung eines Teils der Daten für jedes Pixel im Bild einen Näherungswert liefern kann. Während der Übertragung eines solchen Datenstromes ist es deshalb möglich, durch wiederholte Dekodierung mehrere Zwischenstufen des Bildes in wachsender Auflösung oder Qualität anzuzeigen.

Die Progression kann in mehreren *Dimensionen* erfolgen. Im Ortsraum sind das Farbtiefe und Auflösung; in transformierten Räumen Koeffizientengenauigkeit sowie Frequenzband (diskrete Cosinus-Transformation) oder Auflösung (Wavelet-Transformation). Weiterhin ist eine Aufsplittung der Daten in Grau- und Farbanteil möglich. In 3.2 wird ein formales Modell auf der Basis dieser Dimensionen entwickelt.

Progression kodiert also die Bildinformation in mehreren Stufen. Abschneiden eines verlustfrei kodierten Datenstromes führt zu verlustbehafteter Kodierung. Bei einem eingebetteten Datenstrom (siehe 2.4.1) ist jede Progressionsstufe relativ klein (im Idealfall ein Byte groß), womit eine feinstufige Skalierbarkeit der Kompressionsrate möglich ist.

Benutzt man ein Übertragungsprotokoll, das abhängig vom Bedarf des Nutzers gezielt die Richtung der Progression innerhalb der verfügbaren Dimensionen steuert, so kann man auf der Basis progressiver Bildkodierung die bedarfsgesteuerte Bildübertragung realisieren.

2.4.4.3 Anforderungen an progressive Kodierverfahren

Ein progressives Bildkodierverfahren sollte die folgenden Anforderungen erfüllen:

- hohe verlustfreie Kompressionsrate,
- gutes Verhältnis zwischen Kompressionsrate und Qualität, d.h., möglichst hohe Qualität bereits bei starker Kompression,
- Erzeugung eines eingebetteten Datenstromes für feinstufige Skalierbarkeit,

[8] Englisch: *mean opinion score*.

- keine Expansion der Bilddatenmenge für ungünstige Spezialfälle,
- möglichst geringer Speicher- und Rechenzeitbedarf.

2.4.5 Ausgewählte progressive Bildübertragungsverfahren

2.4.5.1 Frühe Ansätze zur progressiven Bildübertragung

Tanimoto [Tan79, SJT79] stellte 1979 das erste progressive Verfahren zur Bildübertragung zwischen Mainframe und Terminal vor. Ziel war es, das Antwortverhalten des Systems bei einer verfügbaren Übertragungsgeschwindigkeit von nur 1200 Bits pro Sekunde dadurch zu verbessern, dass die Grobstruktur des Bildes bereits nach Übertragung eines Teils der Datenmenge erkennbar ist. Zwei Methoden wurden vorgeschlagen. Das erste Verfahren verwendet eine *Durchschnittspyramide* des Bildes und überträgt eine Folge von Bildern mit jeweils verdoppelter Auflösung. Dadurch wird die Datenmenge jedoch um $1/3$ vergrößert. Die zweite Methode nutzt eine *Summenpyramide*, um diesen Overhead zu verringern. Da jedes Pixel eines grob aufgelösten Bildes als Summe von 4 Pixeln des nächst feiner aufgelösten Bildes übertragen wurde, müssen zur Verfeinerung nur noch 3 dieser 4 Pixelwerte transferiert werden. Allerdings erhöht sich die erforderliche Genauigkeit der Pixelwerte durch die Summenbildung mit jeder Stufe um 2 Bits, so dass sich hier ebenfalls ein Overhead ergibt.

Knowlton [Kno80] entwickelte eine Methode, die bei der progressiven Übertragung ganz ohne Redundanz auskommt. Dies wird durch Erzeugung eines Datenstromes erreicht, der für ein Bild mit n Pixeln mit einem so genannten *Kompositwert* beginnt und danach eine Folge von $n-1$ *Differenziatoren* zur Verfeinerung enthält. Kompositwerte und Differenziatoren werden mithilfe von in Lookup-Tabellen gespeicherten invertierbaren Funktionen ermittelt. Durch fortlaufende Zusammenfassung benachbarter Zellen, welche wechselseitig in X- und Y-Richtung erfolgt, wird der progressive Datenstrom erzeugt. Initiale Zellen sind die Pixel des Bildes. Jeder Zusammenfassungsschritt berechnet aus den Kompositwerten der zusammenzufassenden Zellen einen neuen Kompositwert und einen Differenziator. Dabei stellt der Kompositwert eine Näherung für den Durchschnitt dar. Die Differenziatoren wurden Huffman-kodiert.

Eine Anzahl von Erweiterungen zu Knowltons Technik wurde beschrieben. Hill et al. [HJWJG83] erweitern die Methode für Echtfarbbilder und beschreiben einen flexiblen Algorithmus zur Berechnung der zugrunde liegenden invertierbaren Funktion. Sie stellen eine Umsetzung der Methode für den Zugriff auf eine Bilddatenbank vor, die bereits die Verfeinerung von RoIs unterstützt. Malak und Baker [MJ91] schlagen auf dieser Basis ein progressives Bildformat für Farbbilder vor, das Echtfarbbilder vor der Übertragung in ein spezielles Farbtabellenformat umwandelt. Die Farbtabelleneinträge liegen im *YIQ*-Farbraum vor. Zur Vorbereitung der progressiven Übertragung wird die Farbtabelle derart sortiert, dass die Pixelwerte in eine Y-Komponente (3 Bits), eine I-Komponente (2 Bits) und eine Q-Komponente (3 Bits) gesplittet werden können. Die drei Kanäle werden unabhängig voneinander mit Knowltons Methode kodiert. Als vierte Stufe der Farbinformation kann die Differenz zwischen der Farbtabellenversion und der Echtfarbversion des Bildes übertragen werden. Durch dieses Kodierungsschema ist eine unabhängige Verfeinerung von Auflösung und Farbe möglich. RoIs werden unterstützt. Knowltons Berechnungsvorschrift für die Differenziatoren wurde verbessert, um die Kompressionsrate der Huffman-Kodierung zu erhöhen.

Sanz et al. [SMG84] schlagen eine redundanzfreie Kodierungsmethode vor, die erstmals nicht nur in der Auflösung, sondern auch in der Genauigkeit der Pixelwerte progressiv ist. Die Auflösungsreduktion für die frühen Übertragungsstufen erfolgt durch Unterabtastung, deren Qualität durch Nutzung von Interpolationsmethoden nach der Dekodierung verbes-

sert wird. Es wird eine nicht-uniforme Verfeinerung benutzt, um „wichtige“ Bildbereiche bevorzugt zu verfeinern. Die Steuerung der Verfeinerung geschieht durch Aufteilung des Bildes in 16 quadratische Teilbereiche und deren Priorisierung.

Dreizen [Dre87] beschreibt ein quadtreebasiertes Verfahren, das statt einer gleichmäßigen Verfeinerung Bildbereiche mit hohem Informationsgehalt bevorzugt verfeinert. Als Informationsmaß dient dabei die Differenz der Werte aller Pixel des durch einen Knoten im Quadtree repräsentierten Teilbildes. Die Auflösungsverringerung geschieht durch Unterabtastung. Zur Steuerung des Dekoders wird Seiteninformation zur Auswahl des als Nächstes zu verfeinernden Knotens in den Datenstrom eingebettet.

2.4.5.2 Interlaced GIF

Pipeline	Domäne	interaktiv	verlustbehaftet
V^+A	Ortsraum, indirekt	nein	nein
V^+: Traversierung in 4 Durchläufen gemäß Interlacing-Schema A: LZW-Kodierung			

Das interlaced-GIF-Format [GIF87] wurde 1987 vom Online-Dienst Compuserve zur Übertragung von Bitmaps eingeführt. GIF unterstützt ausschließlich Farbtabellenbilder mit bis zu 256 Farben. Das Bild wird zur Kodierung zeilenweise traversiert, und die Pixelwerte werden LZW-kodiert. 1989 [GIF89] wurde eine einfache Transparenz-Unterstützung hinzugefügt, die durch Markierung einer Farbe als „transparent“ realisiert ist.

Das Interlacing-Schema von GIF funktioniert durch Vertauschen der Reihenfolge von Pixelzeilen. Bei der Traversierung, die in vier Durchläufen erfolgt, wird zunächst jede achte Zeile besucht. Der folgende Pass besucht jede vierte der noch nicht traversierten Zeilen, der dritte Pass jede zweite und der letzte Durchlauf schließlich die verbleibenden Zeilen. So kann nach Dekodierung von einem Achtel aller Pixel eine erste grob aufgelöste Approximation des Bildes angezeigt werden.

2.4.5.3 Interlaced PNG

Pipeline	Domäne	interaktiv	verlustbehaftet
P^+V^+AE	Ortsraum, direkt/indirekt	nein	nein
P^+: Prädiktion in 7 Durchläufen gemäß Interlacing-Schema V^+: Traversierung in 7 Durchläufen gemäß Interlacing-Schema A: LZ77-Kodierung E: Huffman-Kodierung			

Das PNG-Format (*Portable Network Graphics*) [PNG] wurde von der Internet-Community als patentfreie Alternative zu GIF entwickelt. Es kann sowohl Farbtabellen- als auch Echtfarbbilder kodieren und verfügt zur progressiven Bildübertragung über einen leistungsfähigeren interlaced Mode als GIF.

Zur Ausnutzung der Interpixelredundanz wird ein Prädiktionsverfahren eingesetzt, um die Effizienz der nachfolgenden Kodierung zu steigern. Dabei wird der Wert des *aktuellen* Pixels durch die Werte vorher übertragener (*vorhersagender*) Pixel prädiziert; nur die Abweichung von dieser Vorhersage muss kodiert werden. Das verwendete Prädiktionsverfahren ist DPCM (*Differential Pulse Code Modulation*), bei dem nach dem ersten Abtastwert eines digitalisierten Signals statt der folgenden Abtastwerte nur die Differenz zum vorherigen Abtastwert übertragen wird. Zur Effizienzsteigerung wird dieses Verfahren mit der Auswahl eines *Prädiktors* aus den folgenden fünf Alternativen kombiniert:

- der Wert 0 (keine Prädiktion),

- der Wert des Pixels *links* vom aktuellen Pixel,
- der Wert des Pixels *oberhalb* vom aktuellen Pixel,
- der Durchschnittswert der Pixel *links* sowie *oberhalb* vom aktuellen Pixel,
- der Wert eines basierend auf einem Abweichungsmaß [Pae91] ausgewählten Pixels entweder *links* oder *oberhalb* oder *diagonal links oberhalb* vom aktuellen Pixel.

Die Auswahl erfolgt für jede Pixelzeile separat, und ein Kode für die Alternative, welche die höchste Kompressionsrate für die Zeile liefert, wird als Seiteninformation im Header der komprimierten Bilddatei übertragen.

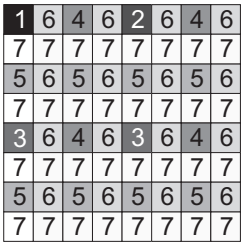


Abbildung 2.3: Interlacing-Schema von PNG.

Zur Nutzung des Interlacing-Schemas sind sieben Durchläufe durch die Teilprozesse *Prädiktion* und *Traversierung* erforderlich. Dazu wird das Bild in Blöcke der Größe 8x8 Pixel unterteilt, und je Block werden pro Durchlauf bestimmte Pixel besucht, was einer Unterabtastung des Bildes entspricht. In aufeinander folgenden Durchläufen werden Unterabtastungsgitter mit den Schrittweiten 8x8, 4x8, 4x4, 2x4, 2x2, 1x2, 1x1 über das Bild gelegt. Abbildung 2.3 zeigt für jedes Pixel eines Blockes die Nummer des Durchlaufes, in dem es besucht wird. Als Ergebnis kann bereits nach der Verarbeitung von 1.56% (1/64) aller Pixel eine erste, sehr grobe Approximation des Bildes angezeigt werden.

2.4.5.4 Quadtrees und BCQ

Pipeline	Domäne	interaktiv	verlustbehaftet
$P^+[Q^+]VE$	Ortsraum, direkt	nein	optional
P^+: Quadtree erstellen (mehrere Durchläufe durch Bild erforderlich) Q^+: Traversierung der Bitebenen im Bit-Quadtree V: *breadth-first*-Traversierung des Quadtree E: Arithmetische Kodierung			

In frühen Arbeiten zur progressiven Bildübertragung [Sam84, SMG84] kamen Quadtrees häufig zur Anwendung. Bei der Quadtree-Zerlegung wird ein Bild in vier Teilbilder (Quadranten) unterteilt und diese Zerlegung mit den Teilbildern rekursiv so lange fortgeführt, bis ein Teilbild einen homogenen Bereich repräsentiert. Ein Teilbild wird durch einen Knoten im Baum dargestellt. In jedem Knoten des Baumes werden Informationen über die Pixelwerte des Teilbildes gespeichert. Das kann im einfachsten Fall der durchschnittliche Pixelwert sein; dabei wird jedoch bis zu $33.\overline{3}\%$ redundante Information kodiert. Traversiert man den Quadtree *breadth-first* und gibt die in den Knoten gespeicherten Werte aus, so erhält man einen Datenstrom, der eine progressive Verfeinerung der Auflösung ermöglicht.

Eine redundanzfreie Möglichkeit besteht darin, für jede Bitebene einen separaten Quadtree zu kodieren und den Wert des entsprechenden Bits der Pixelwerte nur in den Blattknoten zu speichern. Dieser Typ wird als *Bit-Quadtree* oder *Bitebenen-Quadtree* bezeichnet. Da hier nur in den Blattknoten Werte gespeichert werden, existiert für den Wert von Nicht-Blatt-Knoten ein Unsicherheitsintervall, aus dem der Dekoder willkürlich einen Wert auswählen

kann. Dürst [DK91] schlägt dazu verschiedene Möglichkeiten vor. Meist wird der Wert in der Mitte des Unsicherheitsintervalls genutzt.

Der Bit-Quadtree nutzt zwar räumliche Kohärenzen[9] aus, ignoriert aber Kohärenzen zwischen Bitebenen. Dürst [DK91, Dür93] entwickelte mit den *Bitwise Condensed Quadtrees (BCQ)* eine abgewandelte Methode, die diesen Nachteil nicht aufweist. Bei der Erzeugung des Baumes werden in jedem Knoten die Bits gespeichert, die – beginnend mit dem höchstwertigsten Bit – für alle durch den Knoten repräsentierten Pixel identisch sind und die noch nicht in höheren Ebenen des Quadtrees gespeichert wurden. Ein solcher Knoten wird weiter unterteilt, falls die entsprechenden Pixel nicht bis in die unterste Bitebene homogen sind, das heißt, wenn noch nicht alle Bits gespeichert werden konnten. Im Anschluss an die Erstellung der BCQ-Struktur wird diese arithmetisch kodiert [WNC87].

BCQ- und Bit-Quadtree-basierte Bildkodierung ist progressiv; eine Verfeinerung kann sowohl in der Auflösung als auch in der Genauigkeit der Pixelwerte erfolgen.

2.4.5.5 Der JPEG-Standard

Pipeline	Domäne	interaktiv	verlustbehaftet
modusabhängig siehe Tabelle 2.2	Transformation (DCT)	nein	ja

JPEG wurde 1992 als ISO-Standard 10918 [JPGa, JPGb] sowie als ITU-Standard T.81 verabschiedet. Die Entwicklung dieses Standards erfolgte seit 1982 durch die *Joint Photographic Experts Group* bei der ISO. Ziel war es, einen Kompressionsstandard für Bilder mit kontinuierlichen Farbverläufen zu schaffen, der den Anforderungen das Multimedia-Zeitalters (Speicherung großer Bilder auf CD-ROM und Festplatten sowie deren Übertragung über ISDN) gerecht wurde. [PM93] enthält sowohl eine ausführliche Erklärung von JPEG als auch den Text des Standards selbst.

Pipeline	Modus
$PTQVAE$	baseline
$PTQV^+AE$	progressiv, nur spektrale Selektion
$PTQQ^+VAE$	progressiv, nur sukzessive Approximation
$PTQQ^+V^+AE$	progressiv, spektrale Selektion und sukzessive Approximation
P: Farbraumkonvertierung	
T: Diskrete Cosinus-Transformation	
Q: Quantisierung mit Quantisierungstabellen	
Q^+: Sukzessive Approximation	
V: Zick-Zack-Traversierung	
V^+: Zick-Zack-Traversierung mit spektraler Selektion	
A: Lauflängenkodierung	
E: Arithmetische oder Huffman-Kodierung	

Tabelle 2.2: Kodierungspipelines der JPEG-Operationsmodi.

Der JPEG-Standard sieht verschiedene Operationsmodi vor. Jeder Dekoder muss den *Baseline-Modus* dekodieren können, der eine einzige Detaillierungsstufe unterstützt und damit keine Progression erlaubt. Eine Erweiterung stellt der *progressive Modus* dar, der mehrere aufeinander aufbauende Qualitätsstufen (so genannte Scans) enthält und somit progressive Übertragung unterstützt. Dieser Modus ist in allen aktuellen Webbrowsern implementiert. Ein dritter *hierarchischer Modus* ermöglicht mehrere Auflösungsstufen, erlangte jedoch keine allgemeine Verbreitung. In Erweiterungen des JPEG-Standards (ISO-IS 10918-3 bzw. ITU-T.84 [JPG96]) wurden zusätzlich Verfeinerungsrechtecke definiert, mit denen

[9] Räumliche Kohärenz: Die Annahme, dass sich benachbarte Pixel nur geringfügig unterscheiden.

eine rudimentäre RoI-Unterstützung realisiert werden kann. Auch diese Features erlangten keine weite Verbreitung.

Für die progressive Bildübertragung ist vor allem der progressive Modus interessant, der auf dem Baseline-Modus basiert.

Der Baseline-Modus von JPEG. Der Baseline-Modus bildet die Grundlage für alle weiteren JPEG-Modi. Die Komprimierung läuft in folgenden Schritten ab:

Präprozess. Bei Farbbildern erfolgt zunächst die Konvertierung in den YC_bC_r-Farbraum (vgl. 2.2). Wegen der im Vergleich zur Helligkeitsempfindlichkeit niedrigeren Farbempfindlichkeit des menschlichen Auges können die Farbkomponenten C_b und C_r durch Unterabtastung in ihrer Auflösung reduziert werden. Die folgenden Schritte werden auf jeden Farbkanal separat angewendet.

Transformation. Die Bilddaten werden in 8x8 Pixel große Blöcke unterteilt und mittels der diskreten 2D-Cosinustransformation (DCT, siehe 2.4.3.1) in den Frequenzraum umgerechnet. Da die DC-Koeffizienten benachbarter Blöcke meist ähnlich sind, werden sie durch die Differenz zum DC-Koeffizienten des vorherigen Blockes ersetzt (DPCM).

Quantisierung. Nach der Transformation werden die DCT-Koeffizienten skalar quantisiert. Durch diesen Schritt wird die JPEG-Kodierung verlustbehaftet. Wegen der geringeren Empfindlichkeit des Auges für Bildanteile mit hohen räumlichen Frequenzen können die zu diesen Frequenzen gehörenden AC-Koeffizienten stärker quantisiert werden. Der JPEG-Standard empfiehlt Quantisierungstabellen, die für jeden Koeffizienten einen auf dieses subjektive Empfinden abgestimmten Quantisierungsfaktor enthalten. Zur Steuerung des Informationsverlustes (und damit der Kompressionsrate) verwenden viele Implementierungen einen Parameter, mit dem die Faktoren in den Quantisierungstabellen multipliziert werden. Der Enkoder der Independent JPEG Group [L+] benutzt einen so genannten *Qualitätsparameter*, aus dem der Skalierungsfaktor errechnet werden kann.

Traversierung. Die DCT-Koeffizienten werden nachfolgend spektralbandweise traversiert. Ziel dieser Traversierungsfolge ist es, für die nachfolgende Lauflängenkodierung lange Folgen von zu Null quantisierten Koeffizienten zu erzeugen. Der Fakt, dass Koeffizienten hoher Frequenzen stärker quantisiert werden, wird durch die für JPEG charakteristische Zick-ZackTraversierung (siehe Abbildung 2.4) ausgenutzt.

Lauflängenkodierung. In diesem Aggregationskodierungsschritt werden AC-Koeffizientenfolgen der Form $x_1, x_2, \ldots, x_n, y (x_i = 0, y \neq 0)$ durch ein Paar (n, y) dargestellt. Dadurch werden die speziellen Eigenschaften der DCT-Koeffizienten ausgenutzt und eine effizientere Entropiekodierung ermöglicht.

Entropiekodierung. Als Entropiekodierungsverfahren kommt wahlweise Huffman-Kodierung oder arithmetische Kodierung zum Einsatz. Da letzteres Verfahren patentrechtlich geschützt ist, wird meist Huffman-Kodierung verwendet. Würde man jedes mögliche Paar (n, y) als ein Symbol auffassen, so wäre die Anzahl der Symbole zu groß für eine effiziente Kodierung. Daher bildet man sowohl für n als auch für y logarithmisch gestaffelte Gruppen von Werten und kodiert für jedes Paar die Zugehörigkeit zu einer Gruppe. An jeden Kode wird dann eine Anzahl von Bits angehängt, die die exakten Werte innerhalb der Gruppe spezifizieren.

Der progressive Modus von JPEG. Der progressive JPEG-Modus zerlegt die DCT-Koeffizienten in aufeinander aufbauende Detaillierungsstufen, so genannte *Scans*. Ein Scan enthält für jeden 8x8-Block ein oder mehrere Bits der Koeffizienten eines oder mehrerer

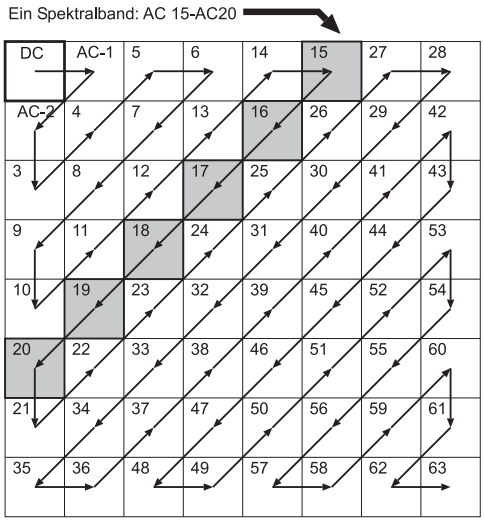


Abbildung 2.4: Traversierung der DCT-Koeffizienten.

Spektralbänder. Die Schritte Vorverarbeitung, Transformation und Quantisierung laufen dabei exakt wie im Baseline-Modus ab. Im Anschluss an die Quantisierung erfolgt in mehreren Durchläufen die Erzeugung der Scans durch eine modifizierte Traversierung (*Spektrale Selektion*) sowie eine *sukzessive Approximation*.

Die *spektrale Selektion* unterteilt die DCT-Koeffizienten in Gruppen von Spektralbändern (siehe Abbildung 2.5). Dabei werden die DC-Koeffizienten nicht zusammen mit den AC-Koeffizienten übertragen, sondern bilden eine eigene Gruppe. Bei der *sukzessiven Verfeinerung* werden die DCT-Koeffizienten in Bits unterteilt, die – beginnend bei den höherwertigen Bits – einzeln oder in Gruppen übertragen werden. Abbildung 2.6 zeigt den Ablauf. Während der Kodierung einer Bitebene eines Koeffizienten wird unterschieden, ob der Koeffizient dem Dekoder noch als Null oder bereits als signifikant bekannt ist. Im ersten Fall erfolgt die Ausgabe von Position und Vorzeichen analog zur im Baseline-Modus verwendeten Lauflängenkodierung, im zweiten wird das aktuelle Bit des Koeffizienten unkodiert in den Datenstrom geschrieben. JPEG verwendet somit verschiedene Kodierungsstrategien für die Signifikanz- und die Verfeinerungsinformation, behandelt beide jedoch – im Unterschied zum nachfolgend beschriebenen Zerotree-Algorithmus – in einem Pass.

Die spektrale Selektion kann allein verwendet werden, da sie einen einfacheren Dekoder ermöglicht. Diese geringere Komplexität wird jedoch durch schlechtere Bildqualität in den ersten Scans erkauft, wie Abbildung B.3 an Hand des zweiten Scans eines im progressiven Modus komprimierten Bildes zeigt. Beim Vergleich der spektralen Selektion mit der Kombination beider Modi ist zu erkennen, dass bei alleiniger Verwendung der spektralen Selektion stärkere Fehler entstehen, da hier in den ersten Scans viele höherfrequente Anteile fehlen.

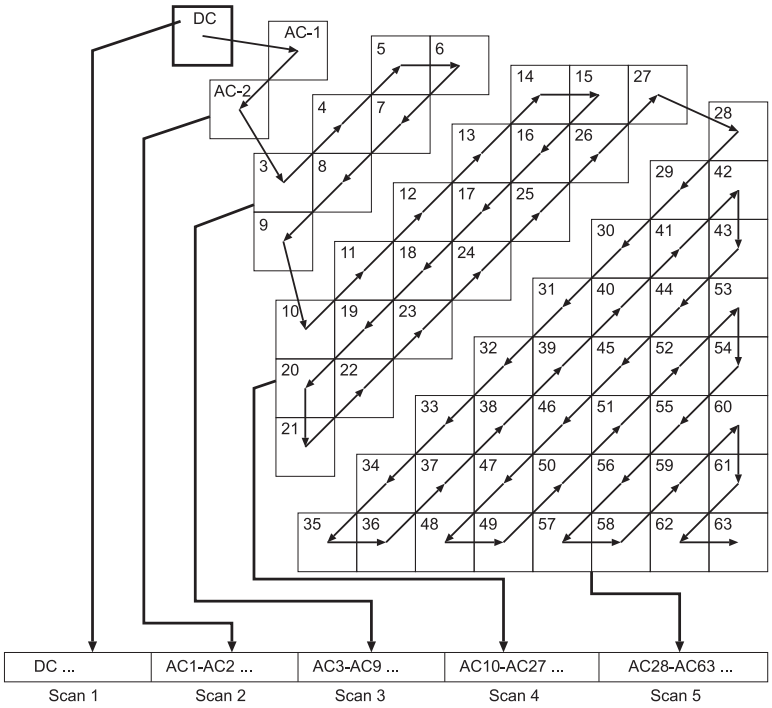


Abbildung 2.5: Zerlegung der DCT-Koeffizienten in Spektralbandgruppen (spektrale Selektion).

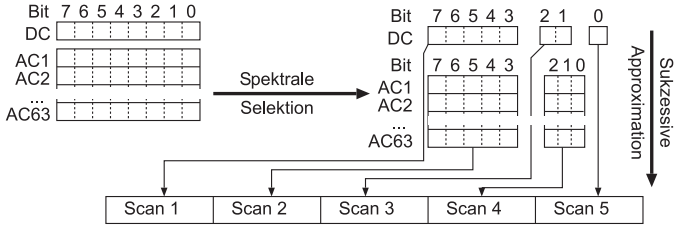


Abbildung 2.6: Zerlegung der DCT-Koeffizienten in Gruppen von Bits (sukzessive Approximation) kombiniert mit spektraler Selektion.

2.4.5.6 Der Embedded-Zerotree-Wavelet-Algorithmus (EZW)

Pipeline	Domäne	interaktiv	verlustbehaftet
TQ^+V^+AE	Transformation (DWT)	nein	meist
T: Wavelet-Transformation Q^+: Bitebenenweise Traversierung V^+: Traversierung der Koeffizienten in zwei Durchläufen A: Zerotree-Kodierung E: Arithmetische Kodierung			

Das Konzept der Zerotrees wurde 1993 von Shapiro [Sha93] entwickelt und 1995 patentiert [Sha95]. Es gilt als das erste Kodierverfahren für eine Subband-Bildrepräsentation, das durch sukzessive Quantisierung einen voll-eingebetteten Datenstrom (vgl. Seite 8) generiert. Dabei erfolgt die Kodierung in mehreren Durchläufen durch das Koeffizientenfeld. Jeder Durchlauf behandelt eine Bitebene, die Quantisierungsschrittweite wird also vor jedem Durchlauf halbiert.

Bei der Dekodierung eines eingebetteten Datenstromes wird auf der Dekoderseite zwischen signifikanten und nicht signifikanten Koeffizienten unterschieden. Signifikante Koeffizienten werden in ihrer Genauigkeit verfeinert, insignifikante Koeffizienten als Null betrachtet und bei der Verfeinerung ignoriert. Somit erfolgt auch die Kodierung in zwei Teilen: *Signifikanzinformation* und *Verfeinerungsinformation*. Für jede Bitebene wird die Signifikanzinformation in einem *dominanten Pass*, die Verfeinerungsinformation in einem darauf folgenden *nachgeordneten Pass* kodiert.

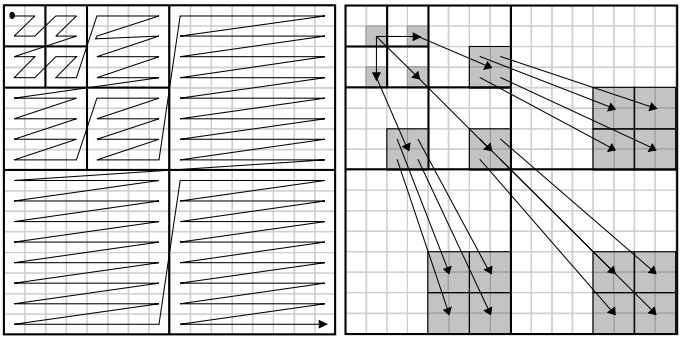


Abbildung 2.7: Traversierungsreihenfolge (links) und Zerotree-Struktur (rechts) für eine dreistufige dyadische Wavelet-Zerlegung.

Während die Verfeinerungsinformation schlecht komprimierbar ist und daher einfach unkodiert ausgegeben wird, kann die Signifikanzinformation durch Ausnutzen ihrer speziellen Charakteristik stark verdichtet werden. Shapiro schlägt dazu für den dominanten Pass ein Alphabet vor, das aus den vier Symbolen POS, NEG, IZ und ZTR besteht. Während jedes dominanten Passes werden bereits signifikante Koeffizienten übersprungen. Für die restlichen Koeffizienten wird jeweils eines der o.g. Symbole wie folgt ausgegeben:

Besitzt ein Koeffizient in der aktuell kodierten Bitebene sein höchstwertigstes Bit mit dem Wert 1, so wird das Symbol POS (positiv signifikant) oder NEG (negativ signifikant) kodiert, um das Vorzeichen des Koeffizienten zu verschlüsseln. In nachfolgenden Durchläufen wird dieser Koeffizient dann als signifikant betrachtet.

Die beiden verbleibenden Symbole IZ (isolated zero) und ZTR (zerotree root) dienen zur effizienten Kodierung von Null-Bits. Dabei wird der Fakt ausgenutzt, dass vor allem in höherwertigen Bitebenen beim Auftreten eines Koeffizienten mit dem Wert Null in einem Subband häufig alle Koeffizienten in höherfrequenten Subbändern mit gleicher Orientierung ebenfalls gleich Null sind (*Hypothese des schwindenden Spektrums*[10]). Eine solche hierarchische Struktur von Koeffizienten, die alle den Wert Null haben, kann effizient als Baum (so genannter *Zerotree*) repräsentiert werden. Im niedrigsten Teilband hat ein Knoten eines Zerotrees drei Kinder, in allen anderen Teilbändern vier. Abbildung 2.7 (rechts) zeigt die Zerotree-Struktur für eine Dekomposition mit drei Stufen. Während das ZTR-Symbol die Wurzel eines Zerotrees repräsentiert und somit effizient Information über die Insignifikanz u.U. sehr vieler Koeffizienten kodiert, dient das IZ-Symbol zum Darstellen von „Löchern“, also von Koeffizienten, die Null sind, aber in deren Baumstruktur Kinder mit Werten ungleich Null existieren.

Die Entscheidung, ob bei einem Koeffizientenwert von Null ein IZ- oder ein ZTR-Symbol kodiert werden muss, erfordert die Auswertung aller Kinder des aktuellen Koeffizienten. Das kann effizient erfolgen, indem auf eine so genannte Zerotree-Map zurückgegriffen wird. Diese speichert bitebenenweise Information über die Zugehörigkeit der Koeffizienten zu Zerotrees und kann effizient in einem Präprozess vor der Kodierung durch rekur-

[10] Die Hypothese des schwindenden Spektrums (englisch: decaying spectrum hypothesis) besagt, dass die Energie eines Signals in niedrigen Frequenzen konzentriert ist und mit ansteigender Frequenz abnimmt.

sive OR-Verknüpfung der potenziellen Eltern-Koeffizienten im Zerotree mit den Werten ihrer Kinder berechnet werden (siehe z.B. [Sha96b]). Hierbei werden die höchstfrequenten Teilbänder ausgeklammert, da dort das ZTR-Symbol nicht genutzt wird. Somit belegt die Zerotree-Map zusätzlich ein Viertel des für das Koeffizientenfeld erforderlichen Speicherplatzes.

Das Koeffizientenfeld wird traversiert, indem zunächst die Koeffizienten des LL-Bandes besucht werden, und danach die Besuchsfolge bis zum höchsten HH-Band wie in Abbildung 2.7 (links) gezeigt fortgesetzt wird. Shapiro schlägt zur Steuerung der Traversierung die Nutzung einer *dominant list* vor. Diese enthält die Koordinaten aller insignifikanten Koeffizienten, die nicht Element eines Zerotrees sind. Das ist für große Bilder speicheraufwändig, senkt aber vor allem in frühen Kodierphasen mit vielen Nullen die Rechenzeit, da nur solche Koeffizienten getestet werden müssen, die in der Liste stehen. Analog werden die Koordinaten signifikanter Koeffizienten in einer *subordinate list* vermerkt.

Das Alphabet zur Beschreibung der Signifikanzinformation ist wegen seiner wenigen Symbole gut für die adaptive arithmetische Kodierung [WNC87] geeignet, die als Entropiekodierungsschritt der Zerotree-Kodierung nachgeschaltet ist und zusätzliche Kompressionsgewinne liefert.

2.4.5.7 Set Partitioning in Hierarchical Trees (SPIHT)

Pipeline	Domäne	interaktiv	verlustbehaftet
$TQ^+V^+A[E]$	Transformation (DWT)	nein	meist
T: Wavelet-Transformation Q^+: Bitebenenweise Traversierung V^+: Traversierung der Koeffizienten in zwei Durchläufen A: Ausgabe der Signifikanzinformation im *sorting pass* E: Optionale arithmetische Kodierung			

Die Methode *Set Partitioning in Hierarchical Trees* (SPIHT) von Said und Pearlman [SP96b] zur progressiven Kodierung von Wavelet-Koeffizienten stellt eine Weiterentwicklung des Zerotree-Verfahrens dar und wird häufig als Referenzverfahren für die Performance-Beurteilung neu entwickelter Kodierverfahren benutzt. Wie EZW arbeitet SPIHT bitebenenweise mit zwei Durchläufen durch das Koeffizientenfeld zur getrennten Kodierung von Signifikanz- und Verfeinerungsinformation. Die Grundidee bei SPIHT besteht darin, durch eine Partitionierung des Koeffizientenfeldes in hierarchisch strukturierte disjunkte Teilmengen die Position signifikanter Koeffizienten effizient zu kodieren. Dazu wird versucht, möglichst große Teilmengen von insignifikanten Koeffizienten zu bilden.

Eine ähnliche Baumstruktur wie bei EZW wird verwendet, um die Ähnlichkeiten zwischen gleich orientierten Teilbändern in der Wavelet-Repräsentation auszunutzen. Sind alle Koeffizienten in einem solchen Baum insignifikant, so wird für den gesamten Baum ein 0-Bit ausgegeben. Anderenfalls erfolgt die Ausgabe eines 1-Bits, und der Baum wird in Unterbäume aufgeteilt. Das erfolgt rekursiv so lange, bis ein insignifikanter Baum gefunden wird oder die Unterteilung nicht mehr fortgesetzt werden kann. Im Anschluss an die Ausgabe der Signifikanzinformation werden die Vorzeichen aller in diesem Pass als signifikant erkannten Koeffizienten ausgegeben. Dieser Durchlauf wird als *sorting pass* bezeichnet, da er die Koeffizienten nach ihrem Betrag sortiert kodiert. Im Anschluss an diesen Pass erfolgt die Ausgabe der Verfeinerungsbits analog zum EZW-Verfahren.

Die Koordinaten der signifikanten bzw. insignifikanten Teilmengen werden in drei Listen gespeichert. Jeder Durchlauf wertet diese Listen rekursiv aus; es erfolgt keine direkte Traversierung des Koeffizientenfeldes wie bei EZW.

Der Signifikanz-Kode, den SPIHT erzeugt, ist sehr kompakt. Eine nachfolgende arithme-

tische Kodierung ergibt nach [SP96b] für viele Bilder nur noch einen PSNR-Gewinn von 0.3 bis 0.6 dB bei gleicher Bitrate. Somit kann die Entropiekodierung weggelassen werden, wenn wenig Rechenleistung zur Verfügung steht.

2.4.5.8 JPEG2000 und EBCOT

Pipeline	Domäne	interaktiv	verlustbehaftet
TQ^+VAEO^+	Transformation(DWT)	nein	meist
T: Wavelet-Transformation Q^+: sub-bitebenenweise Traversierung in 4 Durchläufen V: Traversierung der Koeffizienten A: Kombinierte Ausgabe von Signifikanz und Verfeinerung E: Arithmetische Kodierung O^+: Formatierung der Daten in Layers			

Embedded Block Coding with Optimized Truncation (EBCOT) [Tau99a, Tau99b] ist eine neuartige progressive Kodierungsmethode für Wavelet-Koeffizienten, die im JPEG2000-Standard zum Einsatz kommen wird. Sie unterscheidet sich von den bisher vorgestellten Methoden durch einen Zwei-Ebenen-Kodierungsansatz: Die Struktur des Bitstroms ist nicht von der Struktur des Enkoders abhängig, sondern der Strom ist in aufeinander aufbauende Schichten (so genannte *quality layers*) unterteilt, die die kodierten Daten enthalten und die analysiert werden können, ohne den Datenstrom zu dekodieren. Das erlaubt zusätzlich zur progressiven Dekodierbarkeit des Stromes auch einen wahlfreien Zugriff auf Auflösungs- und Qualitätsstufen. Durch Kachelung des Koeffizientenfeldes in Blöcke und separate Kodierung jedes Blocks ist zusätzlich auch ein wahlfreier Zugriff auf durch das Kachelgitter definierte Bildbereiche möglich.

Die Kodierung erfolgt in zwei Schritten: Eingebettete Kodierung und Layer-Erzeugung. Der erste Schritt besteht aus einer sukzessiven Approximationsquantisierung mit vier Durchläufen je Bitebene und einer blockweisen Traversierung und Kodierung der Teilbänder pro sich ergebender Sub-Bitebene. EBCOT verwendet Kontextmodellierung in jedem der vier Durchläufe, um Redundanzen zwischen räumlich benachbarten Wavelet-Koeffizienten (so genannte *Intra-Band-Redundanzen*) auszunutzen. Das Ignorieren von Inter-Band-Redundanzen (wie z.B. in EZW verwendet) ermöglicht eine weitgehend unabhängige Behandlung der einzelnen Blöcke. Für jeden Block werden Strukturinformationen gespeichert.

Beim folgenden Schritt, der Erzeugung der Layers, werden Teile des kodierten Datenstromes neu zusammengesetzt, so dass das Verhältnis zwischen Bitrate und Qualität optimal ist. Dies wird als *optimized truncation* bezeichnet. Jedes Layer enthält somit die kodierten Daten ausgewählter Sub-Bitebenen aus ausgewählten Teilbändern des Koeffizientenfeldes. Taubman setzte zur Ermittlung der Qualität ein visuelles Maß ein, das die subjektive Bildqualität annähert. Er erzielte damit eine Reduktion der Bitrate gegenüber SPIHT von 50% bei gleicher visueller Qualität.

2.4.5.9 Weitere progressive waveletbasierte Methoden

Progressive Bildkodierung mit Wavelets erfuhr seit Shapiros Arbeit großes Interesse. Dabei wurden in Forschungsarbeiten drei große Gruppen von Aggregationskodierungsverfahren untersucht: Ausnutzung der Interband-Relationen (z.B. *saWave* [SSM97, Str97]), Kontextmodellierung (z.B. *ECECOW* [Wu97]), Lauflängenkodierung (z.B. *PWC* [Mal99]) oder Kombinationen dieser Ansätze (z.B. *PACC* [MC97]).

Zhang und Vuong [ZV97, Zha97] beschreiben das prototypische interaktive Übertragungssystem ANNA, das Bilder in verschiedenen Formaten für die Übertragung und Anzeige auf

mobilen Endgeräten bereitstellen kann. Auf der Basis von Wavelets wurde ein hinsichtlich Auflösung und Dekodierzeit skalierbarer Modus realisiert. Er ermöglicht es, ein Bild zunächst in geringer Auflösung anzufordern und diese dann später zu erhöhen. Weiterhin werden in Abhängigkeit von der Prozessorleistung des Endgerätes verschiedene Wavelet-Filter mit unterschiedlichem Rechenleistungsbedarf ausgewählt. Die Unterstützung einer RoI wird erwähnt, jedoch nicht näher ausgeführt.

Außerdem sind seit einigen Jahren verschiedene progressive Wavelet-Codecs kommerziell verfügbar, wie z.B. *LuRaWave* [LWF], das *WI*-Format von Summus [WI], *Lightning Strike* von Infinop [CF97, LSI], *AV Still Image* von Algovision [AV], *PICTools* von Pegasus Imaging [PIC] sowie – speziell für medizinische Anwendungen – *MT-WICE* von MeVis [MeV].

2.4.5.10 Lohschellers Bildübertragungssystem

Pipeline	Domäne	interaktiv	verlustbehaftet
$TQV^+[AE]$	Transformation (DCT)	ja	ja
T: DCT Q: Quantisierung V^+: Übertragung ausgewählter Koeffizienten $[AE]$: Keine Angaben zu diesen Pipeline-Schritten			

Lohscheller [Loh84] stellte bereits 1984 ein interaktives System zur progressiven Bildübertragung vor, das auf der DCT basiert und RoIs unterstützt. Die Übertragung der DCT-Koeffizienten erfolgt nach einem ähnlichen Schema wie bei der spektralen Selektion von progressive JPEG. Koeffizienten, die die höchste Qualitätsverbesserung erwarten lassen, werden zuerst übertragen. Zusätzlich erfolgt eine Klassifizierung der Koeffizientenblöcke. Dazu wird als näherungsweise subjektives Maß für die Bildqualität eine Sichtbarkeitsschwelle für die DCT-Basisfunktionen verwendet. Pro Klasse werden nur die Koeffizienten übertragen, die über der Schwelle liegen. Die interaktive Steuerung des Systems ermöglicht es, die Übertragung bei Erreichen einer befriedigenden Bildqualität abzubrechen oder eine rechteckige RoI zu definieren, auf die sich dann die weitere Datenübertragung konzentriert.

2.4.5.11 FlashPix und das Internet Imaging Protocol

Pipeline	Domäne	interaktiv	verlustbehaftet
$P^+[PTQVAE]V^+$	Ortsraum, direkt / Transformation (DCT)	ja	optional
P^+: Auflösungspyramide erstellen $PTQVAE$: optionale JPEG-Baseline-Kompression V^+: Wahlfreier Zugriff auf die Kacheln			

Das hierarchische FlashPix-Bilddateiformat [FPX97, Lee96, Lee97] wurde von einem Industriekonsortium unter Leitung von Kodak entwickelt, um die Übertragung und Bearbeitung von digitalen Fotografien über das Internet zu unterstützen. Neben Echtfarbbildern in verschiedenen Farbsystemen werden auch Graustufenbilder unterstützt, jedoch keine Farbtabellenbilder.

Das Bild wird mit $33.\overline{3}\%$ *Redundanz* als so genannte Gauß-Pyramide gespeichert. Eine neue Pyramidenstufe wird aus einer existierenden durch Tiefpassfilterung und nachfolgende Unterabtastung mit der Schrittweite 2 sowohl in X- als auch in Y-Richtung erzeugt. Dieser Prozess beginnt mit dem Bild in maximaler Auflösung und endet mit einem Bild, dessen Breite und Höhe kleiner als oder gleich 64 Pixel sind.

Jede Stufe wird in Kacheln der Größe 64x64 Pixel unterteilt, auf die wahlfrei zugegriffen werden kann. Eine Kachel kann optional mit Baseline-JPEG verlustbehaftet komprimiert werden. Das erlaubt jedoch keine progressive Verfeinerung der Bildqualität.

Zur Übertragung von FlashPix-Dateien über das Netz wurde das Internet Imaging Protocol IIP [IIP97] entwickelt. Dabei wird dem Client-Computer ein virtuelles FlashPix-File zur Verfügung gestellt, auf das wahlweise über Sockets oder `HTTP` zugegriffen werden kann. Das Protokoll erlaubt es, beliebige Bildkacheln in einer wählbaren Auflösungsstufe anzufordern. Auf dieser Basis können Applikationen entwickelt werden, die eine progressive Verfeinerung der Auflösung bzw. Detail on Demand in Form der Vergrößerung ausgewählter Bildteile durch Nachladen der nächsten Auflösungsstufe ermöglichen.

Die Redundanz der Auflösungsstufen bedingt hierbei, dass bereits übertragene Daten verworfen und durch ein vollständig neues Pixelfeld ersetzt werden.

2.5 Regions of Interest

2.5.1 Übersicht und Einordnung

Regions of Interest bieten die Möglichkeit, die zur Übertragung eines Bildes erforderliche Bitrate deutlich zu senken. Wenn dem System bekannt ist, welche Bildregionen den Benutzer interessieren, können die Bildteile außerhalb dieser Regionen stärker komprimiert werden und beanspruchen so weniger Bandbreite. RoI-Unterstützung stellt eine Erweiterung von Bildkodierungsverfahren dar. Als Grundvoraussetzung müssen die Verfahren die Möglichkeit einer adaptiven Quantisierung bieten. Bei progressiven Kodierverfahren wird in der Regel zusätzlich die Traversierungsreihenfolge modifiziert, so dass Koeffizienten, die zu RoIs beitragen, vor den Koeffizienten besucht werden, die keinen Beitrag zu RoIs leisten.

Jahr	1983	1984	1985-94	1995	1996	1997	1998	1999	gesamt
Anzahl	1	1	0	1	3	7	3	2	18

Tabelle 2.3: Anzahl der Veröffentlichungen in der gängigen wissenschaftlichen Literatur zu RoIs in Bildkodierung und -übertragung pro Jahr von 1983 bis 1999.

RoIs erfuhren in den letzten Jahren im Zusammenhang mit Wavelet-Kodierung in Vorbereitung der JPEG2000-Standardisierung sowie insbesondere im medizinischen Bereich großes Forschungsinteresse, wie Tabelle 2.3 an Hand der Anzahl der Veröffentlichungen zu diesem Thema pro Jahr illustriert.

2.5.2 Ausgewählte Verfahren

Bildkodierungsverfahren mit RoI-Unterstützung aus der wissenschaftlichen Literatur sollen nachfolgend vorgestellt werden. Hierbei werden [Zha97] und [ZV97] nicht berücksichtigt, da die RoI-Unterstützung in diesen Veröffentlichungen nicht detailliert genug beschrieben ist, um einer Analyse zugänglich zu sein. Zur besseren Übersicht wurden die Verfahren nach Anwendungsbereichen getrennt.

2.5.2.1 Bildübertragung in Multimedia und Visualisierung

Bereits früh wurde das Potenzial erkannt, das RoIs zur Bandbreiteneinsparung bei der Bildübertragung über schmalbandige Kanäle bieten.

Die Erweiterungen zu Knowltons Verfahren von Hill et al. [HJWJG83] unterstützen die Definition mehrerer rechteckiger RoIs und deren Verfeinerung in der Auflösung. Jede RoI wird exklusiv verfeinert; danach können jedoch weitere RoIs definiert werden.

Lohschellers DCT-basiertes Bildübertragungssystem [Loh84], das bereits in 2.4.5.10 beschrieben wurde, unterstützt die Definition einer rechteckigen RoI während laufender Übertragung. Nach Definition einer RoI werden nur noch Daten übertragen, die die Qualität von Pixeln der RoI verbessern, der Rest des Bildes wird nicht mehr weiter verfeinert.

Shapiro [Sha96a] schlägt einen Vorverarbeitungsschritt für das EZW-Verfahren[11] vor, mit dessen Hilfe eine RoI bevorzugt kodiert werden kann. Dazu wird vor der Kodierung der Durchschnitt aller Pixelwerte berechnet und dieser von allen Pixelwerten subtrahiert. Danach werden alle Pixel innerhalb einer RoI mit einem Skalierungsfaktor größer als 1 multipliziert. Nachfolgend wird auf das so modifizierte Bild die Wavelet-Transformation und ein eingebettetes Kodierverfahren angewendet. Mit dem kodierten Datenstrom muss der Durchschnittswert, der Skalierungsfaktor sowie eine Repräsentation der RoI (die in [Sha96a] nicht näher beschrieben wird) übertragen werden, damit der Dekoder die Kodierung rückgängig machen kann. Dieses Verfahren kann realisiert werden, ohne in das Kodierverfahren selbst eingreifen zu müssen. Es gestattet jedoch keine interaktive Definition der RoI, da diese vor Beginn des Kodierungsprozesses feststehen muss.

Eine Erweiterung des SPIHT-Algorithmus um RoIs wurde von Frajka et al. [FSZ97] vorgeschlagen. Hierbei werden für eine bestimmte Anzahl von Kodierungsdurchläufen nur solche Koeffizienten traversiert, die zur RoI beitragen. Die kreisförmigen RoIs können während laufender Kodierung durch Angabe einer Mittelpunktposition und einer Wichtigkeitsfunktion definiert werden. Diese Funktion hat die Form einer zweidimensionalen Gauß-Glocke und ist im Bild an der Mittelpunktposition zentriert, so dass sie für jedes Pixel eine Wichtigkeit spezifiziert, die im Zentrum der RoI am größten ist. Die Wichtigkeit wird nachfolgend in den Wavelet-Raum transformiert. Jedem Wavelet-Koeffizienten wird abhängig von seinem Wichtigkeitswert ein „Inaktivitätszähler“ zugewiesen, der die Anzahl der Kodierungsdurchläufe festlegt, in denen der Koeffizient nicht berücksichtigt wird. Bei jedem Durchlauf werden die Inaktivitätszähler um 1 verringert. Diese Methode bietet eine große Flexibilität, weil RoIs während laufender Übertragung spezifiziert werden können. Es ist jedoch ein großer Verwaltungsoverhead dadurch erforderlich, dass pro Wavelet-Koeffizient ein Inaktivitätszähler, ein Wichtigkeitswert und die Anzahl bereits übertragener Bitebenen gespeichert werden muss.

Atsumi und Farvardin [AF98] beschreiben ebenfalls eine RoI-Erweiterung für SPIHT. Diese unterstützt die Definition einer einzelnen RoI entweder vor Beginn der Kodierung durch einen Bildautor oder interaktiv im Kodierungsverlauf durch einen Bildbetrachter. Der RoI kann eine relative Wichtigkeit im Vergleich zum Rest des Bildes zugewiesen werden, auf deren Basis ein Bit-Offset n berechnet wird. Die Bits aller Wavelet-Koeffizienten, die zur RoI beitragen, werden um diesen Offset nach links verschoben. Danach erfolgen n Kodierungsdurchläufe nur für die zur RoI beitragenden Koeffizienten, so dass die RoI schnell verfeinert wird. Im Anschluss werden die verbleibenden Bitebenen aller Wavelet-Koeffizienten kodiert. Dabei werden die Koeffizienten der RoI in den untersten n Bitebenen ignoriert, da sie dort generell gleich 0 sind. Zur Form der RoI und zur Verwaltung der Information über die RoI-Zugehörigkeit der Koeffizienten erfolgen keine Angaben. Sollen beliebig geformte RoIs unterstützt werden, erfordert dies eine Bitmap-Darstellung der RoI-Zugehörigkeit. Überlappende RoIs werden nicht unterstützt.

Das kommerzielle waveletbasierte System LuRaWave unterstützt so genannte rechteckige „Fokusregionen“ im Bild, die nicht so stark komprimiert werden wie die restlichen Bildbereiche. Es können mehrere Rechtecke definiert werden. Die Kompression der Regionen im

[11] Dieser Schritt kann beliebigen progressiven Bildkodierungsverfahren vorgeschaltet werden, bei denen Pixelwerte mit höherer Amplitude vor Pixelwerten mit niedrigerer Amplitude kodiert werden.

Vergleich zum Hintergrund wird durch einen „Enhancement-Faktor“ gesteuert, der jedoch für alle Regionen gleich ist.

2.5.2.2 Bildkodierungsstandards

Der Teil III des JPEG-Standards [JPG96] definiert u.a. Verfeinerungsrechtecke für den progressiven Modus. Ein solches Rechteck kann einen Scan auf eine rechteckige Bildregion beschränken. Dabei ist die Übertragung von mehr spektralen Komponenten (spektrale Selektion), von mehr Bits (sukzessive Approximation) bzw. eine Kombination dieser beiden Verfeinerungsdimensionen möglich. Die Verfeinerung kann weiterhin auf einzelne Farbkanäle beschränkt werden. Eine Priorisierung von RoIs ist im Standard nicht vorgesehen, da dieser lediglich die Datenstromsyntax festlegt. Sie kann jedoch auf der Basis dieser Syntax implementiert werden. Verfeinerungsrechtecke dürfen sich nicht überlappen, was Probleme bei der interaktiven Definition der Regionen verursachen kann.

Li, Kasner und Boliek [LKB98] sowie Skodras und Christopoulos [SC99] stellen Möglichkeiten zur RoI-Kodierung im neuen Bildkodierungsstandard JPEG2000 vor. Der Committee Draft des Standards [JPG99a] wurde Ende 1999 veröffentlicht und schreibt die Features fest, die von allen standardkonformen Enkodern und Dekodern unterstützt werden müssen. Optionale Erweiterungen des Standards sind für die Zukunft geplant. Zur internen RoI-Repräsentation dient stets eine Bitmaske. In einem ersten Schritt wird diese Bitmaske analog zum Wavelet-Dekompositionsschema in den Wavelet-Raum transformiert, indem der Filterungsschritt weggelassen und nur der Unterabtastungsschritt ausgeführt wird. Weil die bei der inversen Wavelet-Transformation verwendeten Filterkerne eine Länge größer als 1 aufweisen, kann bei Nutzung einer solchen Maske die RoI in Randbereichen verfälscht werden. Daher wird der Rand der RoI-Maske in einem zweiten Schritt durch Hinzufügen der Position aller Wavelet-Koeffizienten erweitert, die von den Rekonstruktionsfiltern erfasst werden und so zur inversen Transformation beitragen. Für die Kodierung der Koeffizienten der RoI werden verschiedene Kodierungsalternativen vorgeschlagen:

- Im Fall einer *impliziten RoI* wird die RoI vor den restlichen Bildteilen mit einer höheren Bitrate kodiert. Dies ist der einzige Modus, der bereits im Committee Draft des Standards [JPG99a] spezifiziert ist. Die Grundidee dieses als *MaxShift*-Methode bezeichneten Verfahrens besteht darin, dass alle Wavelet-Koeffizienten ungleich Null, die zur RoI beitragen, einen größeren Wert besitzen als der größte Koeffizient, der nicht zur RoI beiträgt. Dadurch kann die Form beliebiger RoIs implizit aus dem Betrag der Koeffizientenwerte abgeleitet werden. Als Seiteninformation muss lediglich die Bitverschiebung übertragen werden, mit der die RoI-Koeffizienten skaliert wurden. Dieses Schema gestattet jedoch keine flexible Priorisierung von RoIs und kann zu Bereichsüberschreitungen der Koeffizienten führen. Es ist vor allem für die Archivierung medizinischer Bilder geeignet, zur interaktiven Steuerung der Bildübertragung in mobilen Umgebungen jedoch nicht einsetzbar.
- Eine flexiblere Möglichkeit stellt die *Skalierungsmethode* dar, bei der die Bitverschiebung der Wavelet-Koeffizienten frei wählbar ist. Dadurch ist eine abgestufte Priorisierung von RoIs möglich, jedoch muss die Form der RoI als Seiteninformation übertragen werden. Bei der Signalisierung werden gemäß [LKB98] zwei RoI-Formen unterstützt. Rechteckige RoIs werden als Paar von Punkten und beliebig geformte RoIs als Bitmaske signalisiert. Zur Kodierung überlappender RoIs sind mehrere RoI-Masken erforderlich, damit im Fall von Überlappungen jedem Teilgebiet die richtige Verschiebung zugewiesen werden kann. Dieser RoI-Modus ist nicht im aktuellen Committee Draft enthalten, sondern als Erweiterung in einem später erscheinenden optionalen Teil des Standards geplant.
- Der dritte Fall der *Transkodierung* geht vom Vorhandensein einer Serverkomponente

aus, die den EBCOT-Datenstrom (vgl. 2.4.5.8) analysiert und mithilfe einer Umordnung der Blöcke Daten für die von der RoI überlappten Blöcke bevorzugt überträgt. Dies ermöglicht nur eine relativ grobe Approximation der RoI-Form. Dafür kommt die Methode jedoch ohne Änderungen an Enkoder- und Dekoder-Software aus, funktioniert also bereits auf der Basis des aktuellen Committee Draft. Eine geeignete Serverkomponente vorausgesetzt, ermöglicht dieser Fall die interaktive Definition der RoI während laufender Übertragung.

2.5.2.3 Active-Vision-Systeme

Active-Vision-Systeme nutzen die mit zunehmendem Abstand zum Blickpunkt (der so genannten *Fovea*) fallende Auflösungsempfindlichkeit des menschlichen visuellen Systems zur Verringerung der Datenrate bei der Bildübertragung, ohne dass die wahrgenommene Bildqualität leidet. Der Enkoder wird dazu durch den Blickpunkt gesteuert, der z.B. von einem Eye-Tracker erfasst werden kann.

Kortum und Geisler [KG96] führten Untersuchungen zur Betrachtung von Bildern mit räumlich variierender Auflösung durch. Bei einem Blickwinkel zwischen 20 und 50° ist dadurch eine Bandbreiteneinsparung zwischen 84 und 98% erzielbar, die durch die Nachschaltung von verlustfreien Kompressionstechniken weiter erhöht werden kann. Als problematisch für die Nutzung dieses Schemas zur Bildübertragung könnten sich Latenzzeiten erweisen.

Aufbauend auf [KG96] beschreiben Chang et al. [CYY97, CY97] ein waveletbasiertes Übertragungssystem für sehr große Bilder (4096x4096 Pixel), das den Waveletkoeffizienten Prioritäten abhängig von ihrem Abstand zu so genannten *fovealen Punkten* zuweist. Diese Punkte könnten in Anlehnung an die RoI-Terminologie als „Points of Interest“ bezeichnet werden. Koeffizienten mit hoher Priorität werden bevorzugt übertragen, so dass die Auflösung mit zunehmender Entfernung von den Points of Interest fällt. Zusätzlich zur räumlich variablen Auflösung ist es möglich, Zooming und Panning auf das Bild anzuwenden, was der Definition von rechteckigen RoIs mit konstanter Auflösung entspricht. Diese Methode erfordert es, Information über die bereits übertragenen Waveletkoeffizienten in einer Maske zu speichern, was speicheraufwändig ist. Weiterhin benutzt das Verfahren bisher keine Datenkompression, so dass nur Erfahrungen zur Nutzung in lokalen Netzen vorliegen, obwohl das Konzept ausdrücklich zur Datenübertragung über *„Thinwires“*[12] entwickelt wurde.

Duchowski [Duc97] beschreibt eine Methode zur waveletbasierten Erzeugung kreisförmiger RoIs in Bildern nach dem Fovea-Prinzip. Ähnlich zu [KG96] und [CY97] werden Bilddaten nahe der Points of Interest (PoI) in hoher Auflösung dargestellt; mit wachsendem Abstand von diesen Punkten in Übereinstimmung mit der Auflösungsempfindlichkeit des menschlichen visuellen Systems wird die Auflösung verringert. Jeder PoI hat somit einen Einflussbereich, der eine kreisförmige RoI definiert. Die variable Auflösung wird durch Multiplikation der Wavelet-Koeffizienten mit Faktoren aus dem Intervall [0,1] im Vorfeld der Kodierung erreicht. Die Kodierung der Wavelet-Koeffizienten wird jedoch nicht betrachtet. Befindet sich ein Pixel im Einflussbereich von zwei oder mehr PoIs, so wird die Auflösung anhand des PoIs bestimmt, der näher an diesem Pixel liegt. Dazu werden überlappende kreisförmige RoIs durch eine Voronoi-Zerlegung in disjunkte polygonale RoIs konvertiert.

[12] Titel der Arbeit: „Realtime Visualization of Large Images over a Thinwire“.

2.5.2.4 Videokodierung

Auch der hohe Bandbreitenbedarf für die Videoübertragung kann in geeigneten Applikationsszenarien durch die Nutzung von RoIs reduziert werden.

Reeves und Robinson [RR96] beschreiben einen Ansatz zur Integration von RoIs in einen MPEG-2-Videostrom mit diesem Ziel. Die RoI ist eine Ellipse oder ein Rechteck voreinstellbarer Größe, deren Mittelpunkt durch den Nutzer auf dem Bildschirm verschoben werden kann. Außerhalb der RoI wird die zeitliche und räumliche Auflösung des Videos verringert; der Unterschied wird über das Bitratenverhältnis zwischen RoI und restlichem Bildbereich gesteuert. Trotz der erreichbaren Reduktion liegt der Bandbreitenbedarf in der Größenordnung von mehreren 100 kBits/s und ist damit zu hoch für mobile Geräte. Als Anwendungen kommen somit z.B. stationäre Überwachungssysteme in Frage.

2.5.2.5 Medical Imaging

In medizinischen Bildern ist die verlustbehaftete Kompression umstritten, da durch sie eventuell diagnostisch relevante Informationen verloren gehen können. Die Nutzung von RoIs ermöglicht es, durch verlustfreie Kompression diagnostisch wichtige Regionen nicht zu verfälschen und durch verlustbehaftete Kompression der verbleibenden Bildbereiche trotzdem eine hohe Kompressionsrate zu erzielen.

Kim et al. [KCKH95] untersuchen verschiedene Alternativen zur Kodierung medizinischer Bilder mit dem oben genannten Ziel. Die RoI-Definition erfolgt hier in Form einer Bitmaske statisch bei der Erzeugung des Bildes. Die Methode kodiert zunächst das gesamte Bild verlustbehaftet mit einer DCT-basierten Kompression. Zusätzlich wird für die Pixel der RoI die Differenz zwischen dem dekodierten Bild und dem Originalbild mit LZ77 kodiert.

Signoroni und Leonardi [SL97] stellen ein System zur EZW-basierten Kodierung medizinischer Bilder vor, das RoIs unterstützt. Die RoIs besitzen eine elliptische Form mit einem unscharfen Rand. Wavelet-Koeffizienten, die zu einer RoI beitragen, werden mit Skalierungsfaktoren multipliziert, wobei diese Faktoren für verschiedene RoIs unterschiedlich sein können, um die RoIs zu priorisieren. Auf das skalierte Koeffizientenfeld wird der unmodifizierte EZW-Algorithmus angewendet, wobei die skalierten Koeffizienten nun bereits in früheren Durchläufen der sukzessiven Approximationsquantisierung kodiert werden. Da keine Seiteninformation über die skalierten Koeffizienten in den Kodierungsprozess einfließt, ist jedoch eine suboptimale Kompressionsrate zu erwarten. Die RoI-Definition erfolgt im Einklang mit dem Einsatzgebiet vor der Speicherung des Bildes, somit ist die Methode nicht interaktiv.

Shin, Wu und Liu [SWL97] entwickelten ein System zur Kompression medizinischer Bilder, bei dem kreisförmige Regionen von diagnostischem Interesse durch waveletbasierte Bildverarbeitungsverfahren erkannt und mit höherer Bitrate kodiert werden. Das Erkennungsverfahren wurde speziell zur Früherkennung von Brustkrebs entwickelt. Als Basis kommt ein eingebettetes Kompressionsverfahren auf der Grundlage der Wavelet-Packet-Transformation zum Einsatz, das als Quantisierung die Methode der sukzessiven Approximation und als Aggregationskodierung eine Quadtree-Kodierung der Waveletkoeffizienten benutzt. Dieses als *Compact Quadtree (CQT)* bezeichnete Verfahren nutzt Intra-Band-Kohärenzen aus, im Gegensatz zu Zerotrees, wo Inter-Band-Kohärenzen effizient kodiert werden. Ähnlich zu [SL97] werden Koeffizienten, die zu RoIs beitragen, vor der Kodierung skaliert und so in früheren Durchläufen der sukzessiven Approximationsquantisierung kodiert. Auch dieses Verfahren ist nicht interaktiv.

Yu et al. [YLLC97] schlagen ein RoI-basiertes Übertragungsprotokoll für waveletkodierte medizinische Bilder vor. Eine als *GSTP* bezeichnet Generalisierung von SPIHT wird zur

fehlerrobusten Kodierung verwendet. Für jede RoI kann die Anzahl von Koeffizientenbits spezifiziert werden, die übertragen werden sollen. Form und Repräsentation der RoIs werden nicht beschrieben, in den Beispielbildern sind jedoch rechteckige RoIs abgebildet.

Nister und Christopoulos [NC98] lösen das Problem der verlustfreien Kodierung von RoIs in einem verlustbehaftet kodierten Bild durch Kombination der S+P-Transformation [SP96a] mit einer modifizierten Version von SPIHT. Die RoI kann eine beliebige Form haben und wird durch eine Bitmaske repräsentiert. Das Verfahren zur Transformation dieser Bitmaske in den Wavelet-Raum ähnelt dem in [LKB98] (siehe Seite 33) für JPEG2000 vorgeschlagenen. Die Kodierung erfolgt in zwei Schritten: Zunächst wird das gesamte Bild mit SPIHT mit einer vorgegebenen Bitrate kodiert. In einem zweiten Schritt werden die verbleibenden Bits der zur RoI beitragenden Koeffizienten kodiert, so dass die RoI verlustfrei rekonstruiert werden kann. Dabei kommt ein modifizierter SPIHT-Algorithmus zum Einsatz, der nicht zur RoI gehörende Koeffizienten überspringt.

2.5.3 Zusammenfassende Wertung

Anhang A gibt eine tabellarische Einordnung der besprochenen Verfahren anhand der folgenden zehn Merkmale:

Basis-Algorithmus. Charakterisiert den Bildkodierungsalgorithmus, der um RoI-Unterstützung erweitert wurde.

RoI-Form. Dieses Merkmal gibt die mögliche Form der RoI an.

RoI-Signalisierung. Mit RoI-Signalisierung wird der Datentyp beschrieben, mit dem der Enkoder dem Dekoder die Information mitteilt, welche Pixel des Bildes zur RoI gehören.

Speicher-Overhead. Die Integration von RoIs erfordert die Verwaltung zusätzlicher Information auf Enkoder- und Dekoderseite. Dieses Merkmal gibt in drei Klassen (niedrig, mittel, hoch) an, wie viel Speicher dafür benötigt wird.

Verfeinerungsdimensionen. RoIs werden in „besserer“ Qualität übertragen als nicht zur RoI gehörige Bildteile. Das Merkmal *Verfeinerungsdimensionen* gibt an, worin diese höhere Qualität bestehen kann.

Priorisierung. Dieses Merkmal ist bei progressiven Verfahren sinnvoll. Es gibt an, auf welche Art einer RoI eine höhere Priorität zugewiesen werden kann als anderen Bildteilen. Dadurch ist es möglich, die verfügbare Bandbreite zwischen dieser RoI und den restlichen Bildteilen (nicht-RoI-Gebieten oder anderen RoIs) variabel aufzuteilen. Ein Sonderfall ist die exklusive Übertragung der RoI: Hierbei wird nach Definition der RoI die Übertragung von Daten für die restlichen Teile des Bildes gestoppt; ausschließlich die RoI wird verfeinert.

Interaktivität. Ein RoI-Verfahren ist *interaktiv*, wenn neue RoIs nach Beginn der Kodierung definierbar sind oder der kodierte Datenstrom auf Anforderung nach Beginn von dessen Übertragung noch so umgeordnet werden kann, dass eine Unterstützung von RoIs möglich ist.

Multiple RoIs. Dieses Merkmal gibt an, ob das Verfahren die Definition und damit die gleichzeitige Verfeinerung mehrerer RoIs unterstützt. Gekoppelt mit der Priorität „exklusiv“ bedeutet dieses Merkmal, dass nur die zuletzt definierte RoI verfeinert wird.

Überlappende RoIs. Dieses Merkmal sagt aus, ob das RoI-Verfahren die redundanzfreie Kodierung überlappender RoIs zulässt.

Kompression. Hier wird angegeben, ob das Verfahren Methoden zur Datenkompression integriert.

Zusätzlich wird in der letzten Zeile der Tabelle die wünschenswerte Charakteristik eines RoI-basierten Verfahrens zur bedarfsgesteuerten Bildübertragung angegeben.

2.6 Progressive Verfeinerung von Farbtabellenbildern

Die bisher betrachteten progressiven Verfahren gehen von Graustufen- oder von Echtfarbbildern aus, wenn sie eine Verfeinerung in der Genauigkeit der Pixelwerte (Verfahren ohne Transformationskodierung) bzw. Koeffizientenwerte (transformationsbasierte Verfahren) zulassen.

Zur progressiven Kodierung von Farbtabellenbildern werden aktuell ausschließlich solche Verfahren benutzt (wie z.B. interlaced PNG [PNG] bzw. interlaced GIF [GIF87, GIF89]), die nur in der Auflösung verfeinern und dadurch in frühen Übertragungsphasen eine Blockstruktur im Bild erzeugen. Das führt in diesen Stufen zu einer schlechten Erkennbarkeit feiner Details. Häufig sind diese aber für die Erfassung des Bildinhaltes essenziell, zum Beispiel, wenn Schriftzüge in einer Navigationsbitmap auf einer WWW-Seite lesbar sein müssen, damit der Nutzer durch Anklicken des entsprechenden Bildbereiches die gewünschte Information abrufen kann. Die Auflösungsverfeinerung ist hier insbesondere bei der Verwendung langsamer drahtloser Verbindungen nachteilig, da die Navigation dann erst nach einer relativ langen Übertragungszeit benutzbar ist. Daher wäre es wünschenswert, auch für Farbtabellenbilder nicht nur eine Verfeinerung in der Auflösung, sondern ebenfalls in der Genauigkeit der Farbinformation zu ermöglichen.

Bisher gibt es kein Verfahren, welches diese Verfeinerung in der Farbtiefe unterstützt. Problematisch bei Farbtabellenbildern ist, dass je Pixel statt eines Farbwertes nur ein Index in eine Farbtabelle gespeichert wird und daher keine Korrelation zwischen den Pixelwerten und den Farbwerten besteht. Diese fehlende Korrelation stellt bei der verlustbehafteten Kompression von Farbtabellenbildern ebenfalls ein Problem dar. Daher existieren hier Arbeiten, die sich mit der Herstellung einer solchen Korrelation für verlustbehaftete Transformationskodierungsverfahren befassen. Eine progressive Verfeinerung der Farbinformation ermöglichen diese Ansätze allerdings nicht. Hierzu erfolgten jedoch Arbeiten für Graustufen- und Echtfarbbilder, wo die geforderte Korrelation vorhanden ist.

Im Folgenden werden speziell auf die verlustbehaftete Kodierung von Farbtabellenbildern zugeschnittene Kompressionsverfahren aus der aktuellen Fachliteratur beschrieben. Danach wird auf existierende Ansätze zur Verfeinerung der Farbinformation von Graustufen- bzw. Echtfarbbildern eingegangen.

2.6.1 Kompression von Farbtabellenbildern

Zur Kompression von Farbtabellenbildern können verschiedene Verfahren angewendet werden. Eine Möglichkeit ist es, die *Indizes in die Farbtabelle* als Folge von Zeichen zu betrachten und diese mit einem verlustfreien Verfahren zu komprimieren, wie es z.B. bei PNG und GIF geschieht. Bei Bildern mit wenigen Farben (vgl. auch eigene Ergebnisse in Anhang B.1) erlaubt dies hohe Kompressionsraten. Die zweite Möglichkeit besteht darin, den Farbindex jedes Pixels durch den entsprechenden Farbtabelleneintrag zu ersetzen, also das Farbtabellenbild *in ein Echtfarbbild umzuwandeln* und danach Kompressionsverfahren für Echtfarbbilder (z.B. JPEG) zu verwenden. Dabei werden allerdings wieder drei Farbkanäle (YC_bC_r) erzeugt, die einzeln komprimiert werden müssen, so dass die Komprimierungseffizienz leidet. Da die DCT-Koeffizienten quantisiert werden, enthält das deko-

dierte Bild mehr Farben als das Ausgangsbild. Für eine Darstellung auf einem Endgerät mit 256-Farben-Display muss es erneut farbquantisiert werden. Dieser Quantisierungsschritt ist häufig sehr rechechenaufwändig und sollte daher auf Geräten mit geringer Prozessorleistung möglichst vermieden werden.

Verschiedene Autoren [ZL93, CPB93, WR94, PTC94, PT94] haben dieses Problem untersucht und Lösungen dafür vorgeschlagen. In allen dieser Veröffentlichungen wird zunächst die Farbtabelle so sortiert, dass ähnliche Farben in der Farbtabelle möglichst eng benachbart auftreten (also ähnliche Indizes aufweisen) und dass die Redundanz zwischen benachbarten Pixeln möglichst hoch ist.

Die einzelnen Arbeiten unterscheiden sich in der verwendeten Sortiermethode. Zaccarin und Liu [ZL93] verwenden die Ordnung der Farben entlang der *Luminanz-Achse* des YC_bC_r-Farbraumes zur Sortierung. Waldemar und Ramstad [WR94] benutzen eine Farbkorrelationsmethode, die die Ähnlichkeit von Farben an Hand von Nachbarschaftsbeziehungen im Bild bestimmt. Zwei Farben sind umso ähnlicher, je häufiger sie im Bild in benachbarten Pixeln auftreten. Po, Tan und Cham [PTC94, PT94] verwenden die so genannte *Closest-Pair-Methode* im *RGB*-Raum. Dabei wird in eine anfangs leere Farbtabelle zunächst die Farbe eingefügt, deren euklidischer Abstand von der Farbe Schwarz ($R = G = B = 0$) am geringsten ist. Danach wird iterativ jeweils die Farbe zur Farbtabelle hinzugefügt, deren euklidischer Abstand von der zuvor eingefügten Farbe minimal ist, bis alle Farben eingefügt worden sind. Die Abstandsberechnung erfolgt im *RGB*-Raum. Chen, Peterson und Bender unterteilen die Farben der Palette in Gruppen ähnlicher Farben, wobei die Ähnlichkeit im CIE L*a*b*-Farbraum berechnet wird. Obwohl in [GAW90] darauf hingewiesen wird, dass perzeptuell uniforme Farbräume nur für kleine, gerade wahrnehmbare Farbdifferenzen uniform sind und dass bei der Konvertierung Kalibrierungsschritte notwendig sind, erzielen Chen, Peterson und Bender nach eigener Aussage durch die Nutzung eines uniformen Farbraumes im Vergleich zu [ZL93] bessere Ergebnisse.

Das Pixelfeld enthält nach der Sortierung die Indizes in die sortierte Farbtabelle. Dieses Pixelfeld (oft als Pseudo-Graustufenbild bezeichnet) wird nun wie ein Graustufenbild mit JPEG [ZL93, CPB93] bzw. einem Subband-Koder [WR94] verlustbehaftet kodiert. Im folgenden Prozess der Dekodierung können Indizes entstehen, denen kein Farbtabelleneintrag zugeordnet ist. Diese werden durch den nächsten gültigen Index ersetzt. Um die dabei auftretenden Farbverfälschungen zu minimieren, unterteilen einige Autoren [ZL93, WR94] das Bild zusätzlich in Blöcke und kodieren je Block in einem Bitvektor, welche Farben in diesem Block vorkommen. Chen, Peterson und Bender [CPB93] kodieren je Pixel zusätzlich eine Marke, die die Zugehörigkeit des Pixels zu einer der Farbgruppen charakterisiert. Das Markenbild wird JBIG-kodiert und dient dazu, die Farbe fehlerhaft dekodierter Pixel aus der korrekten Farbgruppe zu entnehmen. Po, Tan und Cham [PTC94, PT94] zerlegen das Pixelfeld in Blöcke mit lokalen Farbtabellen verschiedener Größe. Innerhalb eines Blockes werden die Indizes verlustfrei mit DPCM, Lauflängen- und Huffman-Kodierung verschlüsselt.

Von Chiang und Po [CP97] wird ein alternatives Verfahren vorgeschlagen. Ein Farbtabellenbild wird mit einem adaptiven LZW-Algorithmus kodiert, der verlustbehaftet arbeitet und dessen Ausgabedatenstrom von einem konventionellen GIF-Dekoder gelesen werden kann. Der Informationsverlust kommt dadurch zu Stande, dass zwei Pixelwerte als gleich angesehen werden, wenn der Abstand ihrer Farben bezüglich eines Abstandsmaßes im YUV-Farbraum einen adaptiven Schwellwert unterschreitet. Dieser Schwellwert berücksichtigt Eigenschaften des menschlichen visuellen Systems. Die Farbtabelle wird bei dieser Methode nicht verändert.

Keines der vorgestellten Verfahren sieht jedoch die Möglichkeit einer progressiven Verfeinerung der Farbinformation vor.

2.6.2 Progressive Verfeinerung der Farbinformation

Bell und Maeder schlagen ein Verfahren zur progressiven regionenbasierten Bildkompression vor, das sie in [MB94] um ein Schema zur Farbverfeinerung erweitern. Dieses Schema überträgt die Farbinformation in mehreren Stufen. Zunächst wird für jede Bildregion eine von 128 Startfarben kodiert. Diese Menge setzt sich aus den Farben reines Schwarz, reines Weiß und den sechs Primär- bzw. Sekundärfarben in 7 Helligkeits- und drei Sättigungsstufen zusammen. Im zweiten Schritt wird die durchschnittliche Farbe jeder Region kodiert, so dass so viele Farben notwendig sind wie Regionen existieren. Im dritten Schritt erfolgt für jede Region die Kodierung von Koeffizienten, die mithilfe interpolierender Polynome den Verlauf von Helligkeit und Sättigung über den Pixeln der Region beschreiben.

In einer Schülerarbeit zum Wettbewerb „Jugend forscht“ [Lau99] wird das verlustfreie, progressive Übertragungsformat PTI für Echtfarbbilder vorgestellt. Es zerlegt ein Bild in 8 Detaillierungsstufen, die jeweils eine Bitebene der drei *RGB*-Farbkomponenten enthalten. Jede Bitebene wird in gleich große Blöcke zerlegt, wobei das Produkt aus Breite und Höhe ein Vielfaches von 8 sein muss. Innerhalb eines Blockes werden jeweils die Bits von acht aufeinander folgenden Pixeln zu einem Eingabezeichen für die nachfolgende LZ77-Komprimierung zusammengefasst. Die Blockgröße hat Einfluss auf die Kompression und ist deshalb ein Parameter des Kodierverfahrens. An Hand von 7 Testbildern wurde PTI mit interlaced PNG verglichen. Die Größe der PTI-Dateien liegt zwischen 83% und 307% ihrer PNG-Pendants bei einem Durchschnitt von 134% und einem Median von 105%. Die PTI-Bilder sind jedoch bedeutend früher im Übertragungsverlauf erkennbar als die PNG-Bilder.

Das System PROTRAC [LPBD95] zerlegt ebenfalls Echtfarbbilder kanalweise in Bitebenen. Je Bitebene werden jeweils acht aufeinander folgende Bits zu einem Symbol zusammengefasst. Der Symbolstrom wird LZW-komprimiert.

Keines der bisher vorgestellten Schemata ist für Farbtabellenbilder geeignet. Die verlustfreie progressive Übertragung von Echtfarbbildern wird i.d.R. für Online-Applikationen eher von untergeordneter Bedeutung sein, da für diese Bildklasse mit JPEG bzw. waveletbasierten Verfahren Alternativen existieren, die für die Bildschirmanzeige von Bildern vergleichbare visuelle Qualität bei höherer Kompressionsrate bieten.

Malak und Baker [MJ91] schlagen ein System zur progressiven Verfeinerung von Echtfarbbildern in Auflösung und Farbtiefe vor, das bereits auf Seite 20 diskutiert wurde. Zum Zweck der Übertragung und Anzeige wird zunächst ein Farbtabellenbild mit 256 Farben erzeugt. Die Pixelwerte werden in drei Stufen zerlegt: eine Y-Komponente (3 Bits), eine I-Komponente (2 Bits) und eine Q-Komponente (3 Bits). Diese drei Komponenten können separat bitweise kodiert werden, wodurch drei Verfeinerungsstufen für die Farbtiefe entstehen. Das bedeutet, dass pro Pixel drei Bits (37% der Datenmenge) übertragen werden müssen, bevor die Darstellung einer ersten Approximation des Bildes möglich ist. Weiterhin erscheint dieses Schema für Bilder mit bedeutend weniger als 256 Farben ineffizient.

2.7 FishEye-Techniken zur Platz sparenden Bilddarstellung

2.7.1 Prinzip verzerrungsorientierter Displays

Im Bereich der Fotografie liefern Fischaugenobjektive einen Blickwinkel von 180 Grad. Sie bilden Objekte in der Mitte einer aufgenommenen Szene groß und mit wenigen Verzerrungen ab. Objekte in Randbereichen werden stark verkleinert und sphärisch verzerrt. Damit lässt sich erreichen, dass die interessierenden Bereiche (der *Fokus*) groß und detail-

liert abgebildet werden, wohingegen unwichtige Randbereiche wenig Platz in der Darstellung beanspruchen, jedoch als *Kontext* eine wichtige Orientierungshilfe darstellen können. Ein Beispiel aus der Fotografie findet man z.B. bei Feininger [Fei95, S. 35]. M. C. Escher setze 1945 in seiner Lithografie „Balkon“ [Ern94, S. 31] die Idee der FishEye-Darstellung ebenfalls um.

2.7.2 FishEye-Darstellungen als grafische Ausgabetechnik

Zur Darstellung großer Informationsmengen auf Computerdisplays bietet sich die Nutzung dieser Technik an, um dem Benutzer auch bei begrenzter Displaygröße die genaue Exploration eines interessierenden Ausschnitts kombiniert mit einem Grobüberblick über die gesamte Datenmenge bei geringem Platzbedarf zu ermöglichen. Abhängig von einem *Abstandsmaß* vom Fokus wird der Kontext verzerrt dargestellt, um dessen Platzbedarf zu verringern.

Im Bereich der Informatik wurden mit dem *Generalized Fish Eye View* Fokus-und-Kontext-Techniken erstmals 1982 von Furnas [Fur82] zur Darstellung strukturierter Daten eingesetzt. Die veröffentlichten Beispiele beschränken sich auf nicht-grafische Daten wie Programmquelltexte oder Dokumentstrukturen. Dabei wird z.B. der aktuell bearbeitete Textabschnitt vollständig dargestellt, weit von der Cursorposition entfernte Abschnitte jedoch nur noch als Überschriften bzw. Prozedurköpfe angezeigt. Interessant ist, dass Furnas den Begriff des *Level of Detail* für den Parameter benutzt, der den lokalen Detaillierungsgrad der Darstellung in Abhängigkeit vom Abstand zu einem Fokuspunkt steuert. LoD wird dabei als skalarer Wert aufgefasst. Daneben werden auch erste Ideen für eine FishEye-Darstellung euklidischer Räume diskutiert. Insbesondere weist Furnas auf die dabei unvermeidlichen Verzerrungen hin und führt aus, dass die Toleranz solcher Verzerrungen bei Nutzern, deren Aufgabe z.B. die Erkennung von Formen beinhaltet, niedriger sein wird als bei Benutzern, deren Aufgabe einen weiten Blickwinkel fordert.

2.7.3 Beispiele

Eine ausführliche Bibliographie zum Thema FishEye-Techniken (mit Online-Versionen vieler Veröffentlichungen) findet sich auf der *Nonlinear Magnification Homepage* [Kea98].

Vor allem auf dem Gebiet der Informationsvisualisierung wurden FishEye-Techniken erforscht, weil hier der Platzbedarf der darzustellenden großen Datenmengen oft höher ist als das Angebot an Darstellungsfläche auf einem Workstation-Monitor. Stellvertretend seien hier der *Hyperbolic Viewer* [LRP95] sowie die *Non-linear Magnification Fields* [KR96] genannt.

Die meisten dieser Techniken erzeugen die FishEye-Darstellung, indem für jedes Pixel ein anderer Skalierungsfaktor verwendet wird. Für die Nutzung von FishEye-Techniken zur Bildübertragung und -darstellung in mobilen Umgebungen impliziert dies einen großen Verwaltungsoverhead bzw. Ressourcenbedarf. In diesen Umgebungen ist es daher erforderlich, Techniken einzusetzen, bei denen für möglichst große, einfach beschreibbare Bildbereiche gleiche Skalierungsfaktoren verwendet werden. Zwei existierende Systeme benutzen bereits rechteckige Bereiche mit gleichen Skalierungsfaktoren.

CoMedi. Das Groupware-System CoMedi [C$^+$99, CBCC98] unterstützt virtuelle, geografisch verteilte Gruppen. Die Bilder der Gruppenmitglieder werden mit Videokameras aufgenommen und gemeinsam auf dem Bildschirm dargestellt. Ausgewählte Bilder können vergrößert werden, die verbleibenden Bilder werden verkleinert darum gruppiert angezeigt. Das System unterstützt jedoch keine verschieden skalierten Bereiche innerhalb eines Bildes, und die verkleinerte Darstellung wirkt sich nicht auf die Bildübertragung aus.

Rubber Sheets. Bei den Rubber Sheets [SSTR93] wird eine Vektorgrafik auf einem virtuellen „Gummituch“ dargestellt, bei dem jeder „Streifen“ mit einem anderen Faktor vergrößert werden kann. Durch die Aufteilung des Bildes in Streifen ergibt sich ein rechteckiges Gitter mit zwei Skalierungsfaktoren pro Teilrechteck: einem in X- und einem weiteren in Y-Richtung. Auch die Rubber Sheets sind eine reine Darstellungstechnik, die die Übertragung von Bildern nicht betrachtet.

2.8 Schlussfolgerungen und weiteres Vorgehen

In diesem Abschnitt sollen aus dem Stand der Forschung Schlussfolgerungen für das weitere Vorgehen gezogen werden.

Die bisher bereits mehrfach umgesetzte Adaption von Bildern an Leistungsparameter von Übertragungskanal und Endgerät unter Nutzung von Standard-Bildkodierungsverfahren bereitet Probleme bei der exakten Steuerung der Kompressionsrate (Fox und Brewer [FB96]). Zur Verfeinerung eines stark komprimierten Bildes ist bei diesen Verfahren außerdem das erneute Laden dieses Bildes mit geringerer Kompressionsrate erforderlich, d.h., redundante Daten müssen übertragen werden (Fox und Brewer [FB96] sowie Ortega et al. [WVOC97, OCAV97]). Diese Probleme können durch die Nutzung eines progressiven Bildkodierungsverfahrens gelöst werden. Weitere Bandbreiteneinsparungen sind durch die Integration von RoIs in ein solches Verfahren möglich, wobei die Detaillierungsstufe jeder Region möglichst exakt spezifizierbar sein sollte. Zur Umsetzung dieser Idee fehlt jedoch bisher ein konsistentes formales Modell für Detaillierungsstufen (Levels of Detail) von RoIs. Ein solches RoI/LoD-Modell soll daher in Kapitel 3 entwickelt und seine Umsetzbarkeit mit verschiedenen oben diskutierten progressiven Bildkodierungstechniken untersucht werden.

Eine große Anzahl von Arbeiten befasste sich mit der Integration von RoIs in Bildkodierungsverfahren. Wünschenswerte Merkmale eines RoI-Verfahrens sind in der letzten Zeile der Tabelle in Anhang A angegeben. Obwohl alle Teilaspekte bereits in einer oder mehreren Arbeiten vorgestellt wurden, fehlt ein System, das alle Merkmale vereint. Die Umsetzung eines solchen Systems auf der Basis des formalen Modells aus Kapitel 3 wird in Kapitel 4 gezeigt.

Eine weitere Einsparung von Bandbreite kann durch FishEye-Techniken erreicht werden, die bisher nur zur Reduktion des Platzbedarfes bei der Darstellung eingesetzt wurden. In Kapitel 5 wird auf der Basis des RoI/LoD-Modells eine solche kombinierte Übertragungs- und Darstellungstechnik entwickelt.

Farbtabellenbilder haben hinsichtlich der progressiven Verfeinerung eine Sonderstellung inne. Existierende Kodierungsmethoden erlauben nur eine Verfeinerung in der Auflösung. Ein häufiger Anwendungsfall dieser Bildklasse sind Navigationsbitmaps im WWW. Hierbei erweist sich jedoch die Auflösungsverfeinerung als suboptimal, da feine Details erst am Ende des Datenstromes kodiert werden. Dadurch ist das Bild erst spät im Übertragungsverlauf erkenn- und somit zur Navigation nutzbar. Eine Verfeinerung in der Farbtiefe wäre für diesen Anwendungsfall besser geeignet. In Kapitel 6 wird ein solches Verfeinerungsschema entworfen und mit den etablierten Methoden interlaced GIF und interlaced PNG verglichen.

Kapitel 3

Regions of Interest und Levels of Detail in der Bildübertragung

In diesem Kapitel soll ein allgemeines, abstraktes Spezifikationsmodell für Regions of Interest und Levels of Detail in Übertragungssystemen für Rasterbilder entwickelt (vgl. auch [Rau98a, Rau98b, Rau99b]) und dessen Anwendbarkeit für die Erweiterung verschiedener Bildkodierungsverfahren kurz diskutiert werden. Die drei folgenden Kapitel werden dann aufbauend auf dem Modell Teilaspekte detaillierter beleuchten und Anwendungen vorstellen.

3.1 Grundidee

Während der Bildübertragung wird davon ausgegangen, dass zu jedem Zeitpunkt das bereits übertragene Bild in einem gewissen *globalen Level of Detail* vorliegt. Dieser globale Level of Detail enthält eine oder mehrere überlappende *Regions of Interest*, deren Detailliertheit jeweils durch einen *lokalen Level of Detail* dargestellt wird. Ein solcher lokaler LoD kann als Kombination der Parameter *Auflösung*, *numerische Genauigkeit* der Koeffizienten bzw. Pixelwerte und *Farbe* beschrieben werden. Für jede RoI gibt es zwei lokale LoDs: den *Zustands-LoD* und den *Ziel-LoD*. Der Zustands-LoD repräsentiert die bereits für die RoI übertragenen Daten. Ist er gleich dem Ziel-LoD, kann die Datenübertragung für die RoI beendet werden.

Verfeinerung einer RoI ist realisierbar, indem nach Erhöhen des Ziel-LoDs weitere Daten übertragen werden, so dass der Zustands-LoD den neuen Ziel-LoD annähert. Eine Verfeinerung des gesamten Bildes (also des globalen LoD) kann durch den Nutzer auf zwei verschiedene Arten angefordert werden: Durch Definieren einer neuen RoI oder durch Vergrößerung des Ziel-LoD einer existierenden RoI. Um die redundante Übertragung von Daten zu vermeiden, werden als Reaktion auf eine *Verfeinerungsanforderung* nur differenzielle Daten übertragen. Daher dürfen die Daten von Bildbereichen, die von mehreren RoIs überlappt werden, nur einmal kodiert werden. Um das zu erreichen, müssen die RoIs in partitionierende Teilmengen gesplittet werden, die sich nicht überlappen.

Der folgende Abschnitt gibt eine formale Definition dieser Grundidee.

3.2 Formale Beschreibung

Ein *Level-of-Detail-Raum* $\mathbb{L}$ sei definiert als vierdimensionaler, diskreter euklidischer Raum mit den Dimensionen *X-Auflösung*, *Y-Auflösung*, *Genauigkeit* und *Farbe*. In diesem Raum kann ein *lokaler Level of Detail L* als eine Menge von Ortsvektoren

$$\overrightarrow{l_i} := [l_i^1, l_i^2, l_i^3, l_i^4]^T \text{ mit den Komponenten } l_i^j \in \{1, 2, ..., l_{max}^j\} \tag{3.1}$$

definiert werden, der die folgende Hülleneigenschaft aufweist:

$$\forall \overrightarrow{l_i} \in L \, \forall j \in \{1,2,3,4\} : (l_i^j = 1) \vee [..., l_i^j - 1, ...]^T \in L. \tag{3.2}$$

Diese konvexe Hülleneigenschaft ist aus zwei Gründen wichtig. Erstens vereinfacht sie die LoD-Spezifikation, da ein lokaler LoD einfach durch Aufzählung einer kleinen Menge von „Grenzpunkten“ in $\mathbb{L}$ definiert werden kann. In den meisten Fällen genügt es, für einen Ziel-LoD[1] lediglich einen Ortsvektor anzugeben, der die Maximalwerte für die vier Dimensionen definiert. Das System kann die fehlenden Elemente automatisch ergänzen, indem so lange neue Vektoren zum LoD hinzugefügt werden, bis Bedingung (3.2) erfüllt ist. Zweitens dient die Hülleneigenschaft der Konsistenzsicherung während der Übertragung bei Nutzung progressiver Übertragungsverfahren. Ist sie für den Zustands-LoD[1] zu jedem Übertragungszeitpunkt erfüllt, so ist das übertragene Bild auch zu jedem Zeitpunkt dekodierbar. Die Hülleneigenschaft kann durch Constraints gesichert werden (siehe Seite 46).

Eine *Region of Interest (RoI) r* kann nun wie folgt als Tupel definiert werden:

$$r := (F, T, Z). \tag{3.3}$$

Dabei gibt F den *Footprint* der RoI an, welcher eine Menge von zusammenhängenden Pixeln im Bild ist und z.B. durch ein Polygon beschrieben werden kann. *Ziel-LoD T* und *Zustands-LoD Z* sind lokale LoDs. Für jede RoI gilt $Z \subseteq T$. Am Beginn der Übertragung ist Z die leere Menge. Im Übertragungsverlauf werden sukzessive Elemente von T ausgewählt und zu Z hinzugefügt. Diese LoD-Traversierung wird von Constraints gesteuert.

Nachdem der Begriff „RoI“ eingeführt wurde, kann nun ein *globaler Level of Detail g* wie folgt definiert werden:

$$g := (R, \widehat{P}, \widehat{Z}, \widehat{D}, C), \tag{3.4}$$

Hierbei ist R eine Menge von RoIs, $\widehat{P}$ ist die Menge der *partitionierenden Teilmengen* der Footprints der RoIs, $\widehat{Z}$ ist die Menge der *Zustands-LoDs*, $\widehat{D}$ bezeichnet die Menge der *LoD-Deltas* und C steht für die Menge der *Traversierungsconstraints* im Raum $\mathbb{L}$. Die Mengen $\widehat{P}$, $\widehat{Z}$ und $\widehat{D}$ wurden eingeführt, um die redundanzfreie Verfeinerung von RoIs mit *überlappenden* Footprints zu unterstützen. Im Folgenden soll beschrieben werden, wie diese Mengen aus den Komponenten F, T und Z der einzelnen RoIs berechnet werden können.

Um redundante Übertragungen zu vermeiden, müssen solche Pixelmengen, die zu überlappenden Footprints gehören, in neue, nicht überlappende Pixelmengen unterteilt werden. Diese sollen als partitionierende Teilmengen bezeichnet werden. Jeder dieser Teilmengen wird ein eigener Zustands-LoD und ein eigenes LoD-Delta zugewiesen. Die Menge R definiert eine Menge $\widehat{F}$ von Footprints, aus der eine geordnete Menge $\widehat{P}$ von partitionierenden

[1] Ziel- und Zustands-LoD werden weiter unten in Gleichung (3.3) als Bestandteile einer Region of Interest (RoI) definiert.

Teilmengen P_i berechnet wird. Dazu partitioniert man die Menge aller Pixel, die zu mindestens einem Footprint in $\widehat{F}$ gehören, derart in Untermengen, dass die folgenden Bedingungen gelten:

$$\forall P_i \in \widehat{P}\ \forall P_j \in \widehat{P} : \begin{cases} P_i = P_j \text{ für } i = j \\ P_i \cap P_j = \emptyset \text{ sonst} \end{cases} \tag{3.5}$$

$$\bigcup_{P_i \in \widehat{P}} P_i = \bigcup_{F_j \in \widehat{F}} F_j \tag{3.6}$$

$$\forall P_i \in \widehat{P}\ \forall F_j \in \widehat{F} : P_i \subseteq F_j \vee P_i \cap F_j = \emptyset \tag{3.7}$$

Für jedes Element P_i von $\widehat{P}$ wird ein eigener Zustands-LoD Z_i eingeführt. Dieser repräsentiert die Vereinigung der Zustands-LoDs[2] $r_j.Z$ aller RoIs r_j, zu deren Footprints $r_j.F$ das entsprechende P_i beiträgt. Die *Zustands-LoD-Menge* $\widehat{Z}$ eines globalen LoD wird als geordnete Menge der Z_i definiert. Jedes Z_i wird also wie folgt berechnet:

$$Z_i = \bigcup_j r_j.Z \tag{3.8}$$

Dabei ist j der Laufindex über alle die RoIs, deren Footprint $r_j.F$ das entsprechende P_i enthält, d.h.

$$\forall j\ \exists r_j \in g.R : P_i \subseteq r_j.F \tag{3.9}$$

Ein *LoD-Delta* D_i ist eine geordnete Menge von Ortsvektoren in $\mathbb{L}$, die die zu übertragenden differenziellen Daten repräsentiert. Ein solches Delta wird als Mengendifferenz zwischen einem Ziel-LoD und dem zugehörigen Zustands-LoD berechnet. Das heißt, dass ein LoD-Delta im Unterschied zu Zustands- und Ziel-LoD die Hülleneigenschaft (vgl. Gleichung (3.2)) nicht erfüllt. Die Ordnung des LoD-Deltas steuert die Übertragungsreihenfolge der Daten, die den Ortsvektoren in $\mathbb{L}$ entsprechen, und wird wiederum von *Constraints* kontrolliert, die detailliert weiter unten beschrieben werden.

Eine *LoD-Delta-Menge* $\widehat{D}$ eines globalen LoD ist eine geordnete Menge von LoD-Deltas D_i. Jedes D_i wird aus Zustands- und Ziel-LoD all der RoIs r_j berechnet, deren Footprints die entsprechende partitionierende Teilmenge P_i überlappen:

$$D_i = \bigcup_j r_j.T \backslash r_j.Z \tag{3.10}$$

wobei j wie in Gleichung (3.9) definiert ist.

Die Mengen $\widehat{P}$, $\widehat{Z}$ und $\widehat{D}$ sind geordnet, um eine eineindeutige Zuordnung der entsprechenden Z_i, D_i und P_i zueinander zu ermöglichen. Daher ist das hier verwendete Ordnungskriterium beliebig, so lange es diese Zuordnung ermöglicht – im Gegensatz zum Ordnungskriterium für die *Elemente* der Mengen D_i, das weiter unten in Form von Constraints beschriebenen wird.

Verfeinerung steht für die Übertragung differenzieller Daten zur Steigerung des Detaillierungsgrades eines Bildes. Für eine Pixelmenge und die korrespondierenden Zustands-LoDs und LoD-Deltas bedeutet Verfeinerung, dass die Daten übertragen werden, die dem ersten Element[3] $D[0]$ des LoD-Deltas D entsprechen. Im Anschluss wird dieses Element aus dem

[2] Zur besseren Lesbarkeit der folgenden Formeln wird die Notation „$a.B$“ für die Komponente B eines Tupels a benutzt.

[3] Das erste Element einer geordneten Menge M wird hier mit $M[0]$ bezeichnet.

LoD-Delta entfernt und zum Zustands-LoD Z hinzugefügt. Ein solcher *lokaler Verfeinerungsschritt* V_L kann wie folgt als Funktion geschrieben werden:

$$\begin{aligned} \overrightarrow{l_0} &= D[0] \\ V_L(Z,D) &= (Z \cup \overrightarrow{l_0}, D \setminus \overrightarrow{l_0}). \end{aligned} \tag{3.11}$$

Ein globaler LoD g kann verfeinert werden, indem eine Folge *globaler Verfeinerungsschritte* auf ihn angewendet wird. Während jedes dieser Schritte wird durch Anwendung eines prioritätenbasierten Selektionsverfahrens eine Menge von RoIs $R' \subseteq g.R$ mit nicht leeren LoD-Deltas zur Verfeinerung ausgewählt. Die LoD-Deltas müssen alle dasselbe erste Element $\overrightarrow{l_0}$ aufweisen. Für alle die partitionierenden Pixelmengen P_i in $g.\widehat{P}$, die Teilmengen des Footprints mindestens einer RoI aus R' sind und deren entsprechendes LoD-Delta D_i als erstes Element $\overrightarrow{l_0}$ aufweist, wird ein lokaler Verfeinerungsschritt $V_{L_0}(Z_i, D_i)$ ausgeführt. Jeweils vor einem globalen Verfeinerungsschritt ist eine Verfeinerungsanforderung möglich, die entweder neue Elemente zum Ziel-LoD einer existierenden RoI oder eine neue RoI zum globalen LoD hinzufügt. Die Folge globaler Verfeinerungsschritte ist beendet, wenn die LoD-Deltas aller partitionierenden Teilmengen leer sind.

Constraints. Die Ordnung der Elemente der LoD-Deltas wird von einer Menge C von *Constraints* gesteuert, so dass die Hülleneigenschaft der Zustands-LoDs zu jedem Zeitpunkt während der Übertragung gesichert ist. Das heißt, die Constraints können durch jede mögliche Traversierungsfolge in $\mathbb{L}$ erfüllt werden, die in keinem Schritt die Hülleneigenschaft verletzt; sie dürfen jedoch keine Traversierungsfolge erlauben, die dies tut.

Ein Constraint wird als Paar (Δ_1, Δ_2) von disjunkten Mengen definiert, wobei $\Delta_1 \subset \mathbb{L}$ und $\Delta_2 \subset \mathbb{L}$ gilt. Bei Δ_1 und Δ_2 handelt es sich nicht um lokale LoDs, da sie die Hülleneigenschaft im Allgemeinen nicht erfüllen. Ein Constraint wird wie folgt interpretiert: Sind alle Ortsvektoren in Δ_1 Elemente im Zustands-LoD, jedoch keiner der Ortsvektoren in Δ_2, dann kann jeder der Ortsvektoren in Δ_2 das erste Element des LoD-Deltas für den nächsten lokalen Verfeinerungsschritt sein.

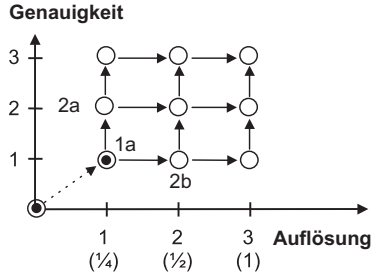


Abbildung 3.1: Grafische Notation der Constraints.

Eine sehr flexible grafische Beschreibungsmöglichkeit für Constraints und die Mengen Δ_1 sowie Δ_2 ist in Abbildung 3.1 dargestellt. Jeder Knoten des gerichteten Graphen entspricht einem Ortsvektor $\overrightarrow{l_i}$ in $\mathbb{L}$. Ein Knoten wird *markiert*, wenn ein *Übertragungsschritt* durchlaufen wird, der $\overrightarrow{l_i}$ zum Zustands-LoD hinzufügt. Ein Knoten η_i kann nur dann markiert (d.h., der korrespondierende Übertragungsschritt ausgeführt) werden, wenn alle Vorgängerknoten von η_i, die durch auf η_i weisende Kanten mit diesem verbunden sind, bereits eine Markierung aufweisen. Nach jedem Übertragungsschritt besteht somit die Menge Δ_1 des anwendbaren Constraints aus allen markierten Knoten, die mindestens einen nicht markierten Nachfolger besitzen. Das entsprechende Δ_2 enthält diejenigen nicht markierten Knoten, deren Vorgänger alle markiert sind. In Abbildung 3.1 gilt $\Delta_1 = \{1a\}$ und $\Delta_2 = \{2a, 2b\}$. Interpretiert man alle noch nicht markierten Knoten als Elemente eines LoD-Deltas (das ja

als geordnete Menge eingeführt wurde), so kann die Pfadlänge im Graphen vom Ursprung zu einem Knoten η_i herangezogen werden, um das gewünschte Ordnungskriterium auf den Elementen dieser Menge zu definieren. Die Pfadlänge definiert jedoch nur eine Halbordnung, so dass für eine Vollordnung zusätzliche Bedingungen hinzugefügt werden müssen (siehe unten).

		Auflösung $w_2 = 1$			**Auflösung** $w_2 = 2$		
		1	**2**	**3**	**1**	**2**	**3**
Genauigkeit $w_1 = 1$	**1**	2.0	3.0	4.0	1.5	2.0	2.5
	2	3.0	4.0	5.0	2.5	3.0	3.5
	3	4.0	5.0	6.0	3.5	4.0	4.5

Tabelle 3.1: Koordinatenmodell der Constraints.

Eine kompaktere Notation ist das *Koordinatenmodell*, das in Tabelle 3.1 veranschaulicht wird. Hier wird das Halbordnungskriterium durch die Summe der Koordinaten der Ortsvektoren $\overrightarrow{l_i}$ in $\mathbb{L}$ vorgegeben. Ein Vorteil dieser Notation besteht in der Möglichkeit, den Achsen von $\mathbb{L}$ Prioritätsfaktoren $w_j \geq 1$ zuzuweisen, durch die die Koordinaten vor Berechnung der Summe dividiert werden. Je höher w_j, desto mehr wird die Achse j bei der Traversierung bevorzugt. Eine mögliche Anwendung der Prioritätsfaktoren sind Levels of Detail für Echtfarbbilder, die für die Bildkodierung häufig im YC_bC_r-System abgelegt werden. Wegen der geringeren Empfindlichkeit des menschlichen Auges für die Farbkomponenten im Vergleich zur Helligkeitskomponente werden erstere häufig nur mit reduzierter Auflösung kodiert. Dies kann im vorgeschlagenen System durch eine niedrigere Priorisierung der Farbachse im Vergleich zu den Auflösungsachsen erreicht werden.

Eine Constraint-Menge, die für einen n-dimensionalen LoD-Raum $\mathbb{L}$ nur die Hülleneigenschaft sichert, ist unterbestimmt und definiert nur eine Halbordnungsrelation auf den Elementen des LoD-Deltas. Zum Beispiel wird die Relation zwischen den Knoten 2a und 2b in Abbildung 3.1 nicht durch die Constraints bestimmt. Zur Steuerung eines Enkoders und zu dessen Synchronisation mit einem Dekoder muss das Constraint-Modell komplett deterministisch gestaltet werden, so dass es eine Vollordnung definiert. Dazu ist es notwendig, eine Regel zur Festlegung der Reihenfolge für solche Elemente von $\mathbb{L}$ einzuführen, die im Constraint-Modell die gleiche Pfadlänge bzw. die gleiche gewichtete Koordinatensumme besitzen. Das kann im einfachsten Fall ebenfalls über die Priorisierung von Achsen geschehen.

3.3 Integrierbarkeit von RoIs und LoDs in verschiedene Bildkodierungsverfahren

Methode	RoI	Auflösung (Subsampling) X	Auflösung (Subsampling) Y	Genauigkeit Selektion	Genauigkeit Quantisierung	Farbe
GIF	–	–	8,4,2,1	–	–	–
PNG	–	8,4,2,1	8,4,2,1 mit $X \geq Y$	–	–	-
Bit-Quadtree	ja	$(2^n,...,1)$, $X = Y$	$(2^n,...,1)$, $X = Y$	–	Bitebenen	Y, C_b, C_r getrennt
BCQ	ja	$(2^n,...,1)$, $X = Y$	$(2^n,...,1)$, $X = Y$	–	Kombination von Bitebenen (nicht uniform)	Y, C_b, C_r getrennt
progressive JPEG	(ja)[4]	(teilweise)[5]	(teilweise)[5]	Bänder von DCT-Koeffizienten	Bitebenen von DCT-Koeffizienten	Y, C_b, C_r getrennt
Wavelet	ja	$(2^n,...,1)$[6]	$(2^n,...,1)$[6]	–	Bitebenen von Wavelet-Koeffizienten	Y, C_b, C_r getrennt

Tabelle 3.2: Integrierbarkeit von RoIs und LoDs in verschiedene Bildformate.

Im Folgenden soll untersucht werden, wie existierende Bildkompressionstechniken in das entwickelte Modell integriert werden können. Tabelle 3.2 gibt für einige gebräuchliche Bildkodierungsmethoden eine Zusammenfassung der Wertebereiche der LoD-Merkmale an. Weiterhin ist dargestellt, ob RoIs integrierbar sind. Auf die einzelnen Verfahren wird in den folgenden Abschnitten genauer eingegangen.

3.3.1 Verfahren für Farbtabellenbilder

Die Verfahren interlaced GIF und interlaced PNG erlauben keine Integration von RoIs. Im interlaced Mode bieten sie aber rudimentäre LoD-Unterstützung in Gestalt von 4 bzw. 7 Auflösungsstufen. Sie werden vor allem zur Übertragung von Farbtabellenbildern eingesetzt. Da die Gesamtdatenmenge bei dieser Bildklasse geringer ist als bei Echtfarbbildern, erscheint die Integration von RoIs zur Bandbreiteneinsparung für diese Bildklasse nicht so kritisch wie für Bilder mit vielen Farben bzw. Graustufen.

Farbtabellenbilder werden jedoch häufig eingesetzt, um in Online-Diensten Navigationselemente bereitzustellen. Die sich dabei aus der ausschließlichen Unterstützung der LoD-Dimensionen X- und Y-Auflösung ergebenden Probleme für den Zugriff über langsame Netze auf diese Dienste wurden bereits in 2.6 diskutiert.

Dieses Problem kann durch die Entwicklung eines neuen Dateiformates für Farbtabellenbilder gelöst werden, das die LoD-Dimension „Genauigkeit“ integriert. Dazu muss eine Kodierung gefunden werden, die trotz Indizierung der Pixelwerte eine frühe Unterscheidbarkeit feiner Details mit starkem Kontrast bei hoher Kompressionsrate ermöglicht. Ein solches im Rahmen vorliegender Arbeit entwickeltes Format wird in Kapitel 6 vorgestellt.

[4] Nur im optionalen Teil III des Standards in Form von Verfeinerungsrechtecken vorgesehen. Mit Standard-Datenstromsyntax nur durch Tricks möglich.

[5] Funktioniert mit speziellen Algorithmen in der DCT-Domain oder durch Nutzung des hierarchischen JPEG-Modus. Es gibt bisher keine allgemein zugängliche Implementation dieses Modus.

[6] Die Subsampling-Faktoren in X- und Y-Richtung dürfen maximal um den Faktor 2 differieren. In 4.4 wird ein neues Wavelet-Dekompositionsschema entwickelt, das diese Einschränkung nicht aufweist.

3.3.2 Quadtree-Verfahren

Die Quadtree-Verfahren *Bit-Quadtree* und *Bitwise Condensed Quadtree (BCQ)* zur Bildkodierung wurden in 2.4.5.4 beschrieben. Mit jeder Ebene im Quadtree ist eine Verdopplung der *Auflösung* sowohl in X- als auch in Y-Richtung verbunden. Daher erscheinen Quadtrees als ein geeigneter Kandidat zur Umsetzung von LoDs. Der Bit-Quadtree gestattet durch die Separierung der Bitebenen eine einfache Integration der Dimension *Genauigkeit*, lässt aber eine geringere Kompressionsrate als BCQ erwarten, da er Kohärenzen zwischen Bitebenen ignoriert.

		Dateigröße (Bytes)			
Testbild	**Graustufen**	**Bit-Quadtree**	**BCQ**	**GIF**	**Wavelets**
logos (Abbildung B.2(a))	19	23884	20511	7854	29357
lena (Abbildung B.2(b))	215	249618	191415	264434	143860
boat (Abbildung B.2(d))	224	248939	190890	236610	147138

Tabelle 3.3: Vergleich der komprimierten Dateigrößen bei der verlustfreien Kodierung mit Bit-Quadtree, BCQ, GIF und Wavelets[8]. Testbilder siehe Abbildung B.2.

Ein Vergleich der Quadtree-Verfahren mit dem etablierten Standardverfahren GIF sowie dem in dieser Arbeit entwickelten waveletbasierten Kompressionsverfahren MOVIWAVE (siehe 4.3.2) ergab die folgenden Ergebnisse:

- Im Vergleich zu Bit-Quadtrees erreicht das BCQ-Verfahren durch die Zusammenfassung von Bitebenen eine substanziell bessere Kompression.
- Vergleicht man BCQ mit GIF und Waveletkodierung, so wird deutlich, dass für Bilder mit wenigen Graustufen das GIF-Format eine bessere Kompressionsrate liefert als BCQ; für Bilder mit vielen Graustufen ist der Wavelet-Kodierer dem BCQ-Format überlegen.
- Betrachtet man den Verlauf einer progressiven Verfeinerung (siehe Abbildung B.8), so ist das Wavelet-Verfahren auch hinsichtlich der subjektiven Bildqualität dem BCQ überlegen.

Tabelle 3.3 verdeutlicht die Kompressionsraten exemplarisch anhand der komprimierten Dateigrößen von drei Testbildern.

Obwohl Quadtrees die Realisierung des RoI/LoD-Modells ermöglichen, existieren also hinsichtlich der Kompressionsrate für jede Bildklasse bessere Methoden. Auf Verfahren für Farbtabellenbilder wurde bereits eingegangen. Im Folgenden werden deshalb Verfahren mit Transformationskodierung diskutiert und eine Wahl zwischen JPEG und Wavelets als Basisverfahren für die RoI/LoD-Kodierung getroffen.

3.3.3 Verfahren mit Transformationskodierung

Als transformationsbasierte Verfahren wurden in Tabelle 3.2 progressive JPEG und waveletbasierte Methoden betrachtet. Das progressive JPEG-Verfahren nimmt hinsichtlich der Genauigkeit eine Sonderstellung ein, da die Genauigkeit (Qualität) des dekodierten Bildes sowohl durch „spektrale Selektion“ von Spektralbändern als auch durch „sukzessive Approximation“ beeinflusst werden kann (vgl. 2.4.5.5).

Neben dem etablierten JPEG kommt waveletbasierten Methoden eine wachsende Bedeutung zu. Eine Standardisierung waveletbasierter Bildkodierung (JPEG2000, als ISO-Stan-

[8] Die Dateigrößen für die Waveletkodierung wurden mit dem dieser Arbeit entwickelten MOVIWAVECODEC (siehe Abschnitte 4.3.1 und 4.3.2) ermittelt.

dard 15444-1 und als ITU-Standard ITU-T.800) ist für das Jahr 2001 geplant [JPG99a, JPG99b, SC99]. Einige Vorteile von Wavelet-Verfahren gegenüber JPEG sind unmittelbar aus Tabelle 3.2 ersichtlich:

- Die Wavelet-Repräsentation unterstützt direkt verschiedene Auflösungsstufen. Hierbei können die Auflösungen in X- und Y-Richtung um den Faktor 2 voneinander abweichen, wodurch Applikationen größere Flexibilität erhalten als beispielsweise bei GIF oder Quadtrees. In 4.4.2 wird ein Wavelet-Dekompositionsschema entwickelt, das noch größere Flexibilität bietet. Im Gegensatz dazu ist bei JPEG eine Progression in der Auflösung nur möglich, wenn ein besonderer hierarchischer Modus verwendet wird. In der Literatur wurden in der DCT-Domain anwendbare Skalierungsverfahren für die Auflösung vorgeschlagen [Cha94, CM95, NB95, SS95]. Diese sind jedoch zur Bearbeitung kodierter Bilder gedacht und nicht auf eine progressive Auflösungserhöhung hin ausgelegt.
- Der JPEG-Kernstandard sieht keine Kodierung von RoIs vor. Mit einem Standard-JPEG-Datenstrom ist daher nur eine eingeschränkte Realisierung von RoIs möglich, da keine adaptive Quantisierung unterstützt wird. Es besteht jedoch die Möglichkeit, außerhalb der RoIs viele DCT-Koeffizienten zu Null zu quantisieren. Allerdings ist dann keine Verfeinerung der Gebiete außerhalb der RoI durch Definition neuer RoIs möglich. Damit würde eine vollständige Umsetzung des vorgeschlagenen Modells nicht-standardisierte Erweiterungen der Datenstromsyntax erfordern. Die im Teil III des Standards als Erweiterung definierten Verfeinerungsrechtecke (vgl. 2.5.2.2) fanden keine weite Verbreitung und sind nicht sehr flexibel.

Ein weiterer Vorteil von Wavelets gegenüber JPEG besteht darin, dass mit eingebetten Kodierverfahren eine exakte Steuerung der Größe der kodierten Bilddatei sehr einfach möglich ist. Für JPEG sind dazu umfangreiche Modifikationen des Enkoders notwendig (siehe [ISK98]), und der resultierende Datenstrom ist nicht mehr weiter durch Übertragung differenzieller Daten verfeinerbar. Nutzt man einen Standard-Enkoder, so ermöglicht dieser meist keine Spezifikation einer Zieldateigröße, sondern nur die Angabe eines Skalierungsfaktors für die Quantisierungstabellen. Eine in Vorarbeiten [Rau97] zu dieser Arbeit realisierte iterative Schätzung dieses Parameters zur Steuerung der Dateigröße erwies sich als relativ ungenau.

Bei niedrigen Bitraten ist die mit Waveletkodierung erzielbare Bildqualität bedeutend besser als mit JPEG (siehe Abbildungen B.4 und B.5 im Anhang B).

Nachteilig an Waveletverfahren ist der im Vergleich zu JPEG höhere Rechenaufwand. Schnelle Transformationsverfahren wie das Lifting-Schema [Swe95, Swe96a, Swe96b], die Entwicklung immer schnellerer Hardware und das der Wavelet-Software im Vergleich zu den ausgereiften JPEG-Codecs noch innewohnende Optimierungspotenzial werden diesen Nachteil jedoch weiter relativieren.

Auf Grund der offensichtlichen Vorteile werden waveletbasierte Verfahren als Grundlage für die im Kapitel 4 beschriebene Umsetzung des RoI/LoD-Modells zum Einsatz kommen.

Kapitel 4

Regions of Interest und Levels of Detail mit Wavelets

4.1 Überblick

Dieses Kapitel beschreibt die Umsetzung des in Kapitel 3 entwickelten RoI/LoD-Modells durch Erweiterung eines waveletbasierten Bildkodierungssystems (siehe auch [Rau98a, Rau98b, Rau99b]).

Zunächst wird in 4.2 dargelegt, wie der LoD-Raum auf die Struktur eines Wavelet-Koeffizientenfeldes abgebildet werden kann. Als Grundlage wird danach das auf EZW aufbauende Basiskodierverfahren MOVIWAVE beschrieben, welches im Rahmen dieser Arbeit entwickelt wurde. Da das etablierte dyadische Wavelet-Dekompositionsschema (vgl. 2.4.3.1) nicht flexibel genug ist, um den vollen LoD-Raum abzubilden, wird in 4.4 ein neues Dekompositionsschema entwickelt, das die dyadische Zerlegung als Sonderfall enthält, und dieses in MOVIWAVE integriert.

Danach wird das MOVIROILOD-Verfahren vorgestellt, das RoIs sowie LoDs auf der Basis von MOVIWAVE realisiert. MOVIROILOD unterstützt insbesondere die interaktive Definition multipler, möglicherweise überlappender RoIs mit wählbaren LoDs sowohl vor als auch während der Übertragung. Bei überlappenden RoIs wird Wert darauf gelegt, durch Übertragung differenzieller Daten Redundanzfreiheit zu erreichen. Die Footprints der RoIs werden sowohl bei der Signalisierung als auch bei der Speicherung des Übertragungszustandes geometrisch als Punktliste repräsentiert, was gegenüber der häufig verwendeten Masken-Repräsentation Bandbreite und Speicherplatz spart.

Am Schluss dieses Kapitels wird die softwaretechnische Umsetzung LIBROILOD vorgestellt.

4.2 Umsetzung des LoD-Raumes mit Wavelets

Die Wavelet-Transformation (siehe 2.4.3.1) liefert eine Mehrfachauflösungsrepräsentation von Bilddaten. Damit können die Dimensionen *X-Auflösung* und *Y-Auflösung* des LoD-Raumes durch Selektion der entsprechenden Oktavbänder realisiert werden. Die Dimension *Genauigkeit* ist durch Auswahl der gewünschten Anzahl von Bitebenen umsetzbar. Hier sind jedoch auch komplexere Maße möglich, wie z.B. kodierte Bits pro Pixel. Bei Farbbildern erweist sich vor der Transformation eine Konvertierung in den YC_bC_r-Farbraum

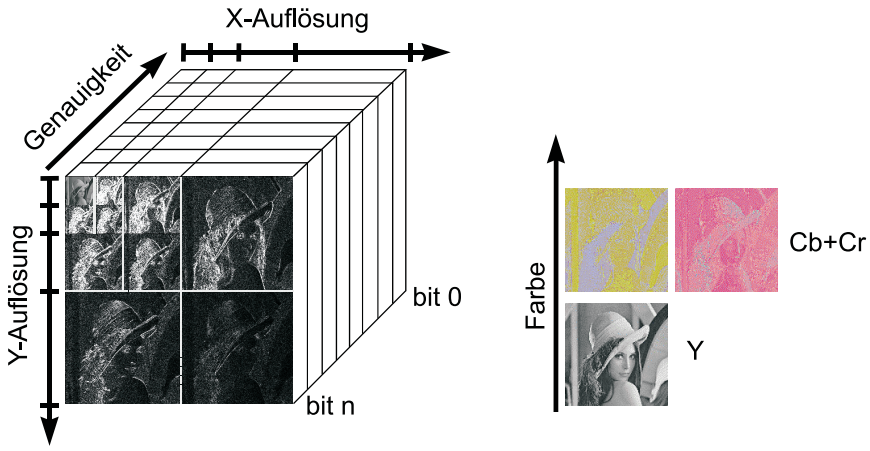


Abbildung 4.1: Repräsentation des LoD-Raumes mit Wavelets.

als günstig, um die Farbkomponenten zu dekorrelieren; nachfolgend wird jeder Farbkanal separat transformiert. Damit kann die *Farbdimension* des LoD-Raumes abgebildet werden, indem entweder nur der Y-Kanal oder aber alle drei Kanäle betrachtet werden. Abbildung 4.1 veranschaulicht dieses Realisierungsprinzip.

4.3 Das Basis-Kodierverfahren

4.3.1 Auswahl und Umsetzung

Die Umsetzung von RoIs und LoDs soll durch Modifikation eines bewährten Wavelet-Kodierungsverfahrens erfolgen. Dieses Verfahren muss die folgenden Anforderungen erfüllen:

1. Erzeugung eines eingebetteten Datenstroms zur Unterstützung von LoDs und progressiver Verfeinerung,
2. Hohe Kompressionsrate durch Ausnutzung von Interbandkohärenzen,
3. Einfache Abbildbarkeit von Positionen im Bild auf Positionen innerhalb der Traversierungsreihenfolge des Koeffizientenfeldes zur Integration von RoIs,
4. Verfügbarkeit des Quellkodes zur einfachen Umsetzung der Erweiterungen.

Die ersten beiden Kriterien werden sowohl von SPIHT (2.4.5.7) als auch vom Zerotree-Verfahren (2.4.5.6) erfüllt. Forderung 3 wird von EZW besser als von SPIHT erfüllt, da bei EZW das Koeffizientenfeld direkt traversiert werden kann, statt wie bei SPIHT mit Listen von Koordinaten zu arbeiten. Die letzte Forderung ist bei beiden Methoden nur durch eine eigene Implementierung zu erfüllen. Die Wahl fiel auf Grund dieser Kriterien auf das Zerotree-Verfahren. EBCOT 2.4.5.8 wurde nicht berücksichtigt, da es zum Zeitpunkt der Auswahlentscheidung noch nicht publiziert war.

Es wurde das Programm MOVIWAVECODEC implementiert, das den EZW-Algorithmus umsetzt und als Grundlage der in Abschnitt 4.5 beschriebenen Integration von RoIs und LoDs dient. Um die Positionsabbildung (Kriterium 3) weiter zu vereinfachen und Speicherplatz zu sparen, wurde statt der Nutzung von Listen (dominant und subordinate list, siehe Seite 28) einer *listenlosen* Umsetzung der Vorzug gegeben, die die Zuordnung der Koeffizienten zu den beiden Pässen durch Vergleich des Koeffizientenwertes mit einem

Schwellwert realisiert.

Im Folgenden wird die Leistungsfähigkeit des umgesetzten Basisverfahrens evaluiert.

4.3.2 Performance von MOVIWAVECODEC

Um die Performance des implementierten Basisverfahrens MOVIWAVECODEC einzuschätzen, wurde dieses mit der JPEG-Implementation der Independent JPEG Group [L+], dem kommerziellen Wavelet-Codec LuRaWave [LWF] und dem SPIHT-Codec [SP96b] hinsichtlich PSNR, Kodierzeit und Dekodierzeit verglichen.

Als Testbild diente das häufig verwendete Graustufenbild „lena“ (vgl. Abbildung B.2(b)) mit einer Größe von 512x512 Pixeln. Die Zeiten wurden auf einer SUN SPARC ULTRA-1 mit 166MHz gemessen. JPEG wurde dabei auf den progressiven Modus und optimierte Huffman-Tabellen eingestellt. Bei SPIHT wurde die Variante mit arithmetischer Kodierung verwendet, im MOVIWAVECODEC das 5/3-Wavelet als Filter eingestellt.

Da JPEG keine direkte Einstellung einer Zieldateigröße ermöglicht, wurde das Bild mit verschiedenen Qualitätsparametereinstellungen (darunter den kleinstmöglichen) kodiert. Auf die damit erreichten komprimierten Dateigrößen wurden nachfolgend die drei eingebetteten Wavelet-Verfahren parametrisiert. Zusätzlich wurden die Dateigrößen 500 und 1000 Bytes in den Test aufgenommen, die von JPEG nicht erreicht werden konnten.

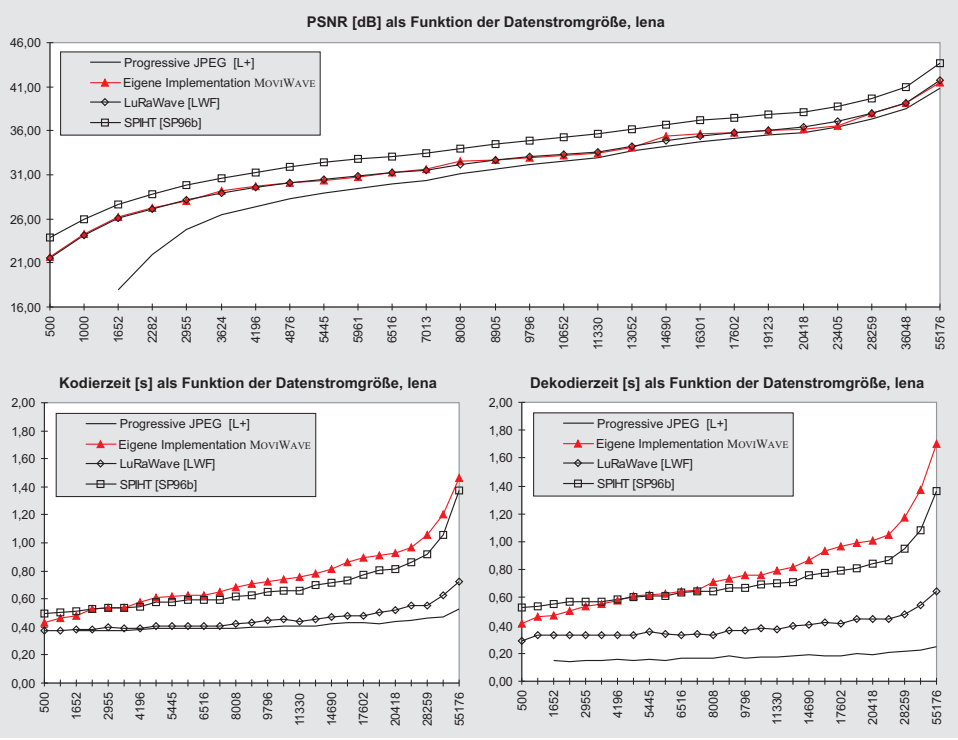


Abbildung 4.2: Performancevergleich existierender Bildkompressoren mit dem implementierten Basisverfahren MOVIWAVECODEC.

Abbildung 4.2 zeigt die Ergebnisse:

- Mit Wavelets können Bilder stärker komprimiert werden als mit JPEG.
- Vor allem bei niedrigen Bitraten ist der durch Wavelet-Kompression hervorgerufe-

ne Bildfehler geringer als der durch JPEG verursachte. Das gilt nicht nur für das Verzerrungsmaß PSNR, das bei Wavelets höhere Werte aufweist, sondern kann auch an Hand der Abbildungen B.4 und B.5 für die visuelle Bildqualität nachvollzogen werden.

- Alle Verfahren benötigen zum Komprimieren mehr Zeit als zum Dekomprimieren.
- JPEG ist schneller als die Waveletverfahren.
- Der MOVIWAVECODEC ist hinsichtlich der Leistungsfähigkeit mit existierenden kommerziellen Systemen und Forschungsprototypen vergleichbar und bildet somit eine gute Grundlage für die Erweiterung um RoIs und LoDs zum MOVIROILOD-Algorithmus.

4.4 Ein neues Wavelet-Dekompositionsschema

4.4.1 Problem

Die klassische dyadische Wavelet-Transformation (vgl. 2.4.3.1) bietet gute Voraussetzungen für die Integration von Levels of Detail, wenn die Auflösungen in X- und Y-Richtung identisch sind oder sich maximal um den Faktor 2 unterscheiden. In diesem Fall liefert die dyadische Wavelet-Zerlegung direkt die gewünschte Auflösung durch Auswahl der entsprechenden Oktavbänder. Abbildung 4.3 (links) zeigt, dass damit jedoch nicht alle Ortsvektoren in der XY-Ebene des LoD-Raumes abgedeckt sind. Für einige Anwendungen (wie den RECHTECKIGEN FISHEYE-VIEW, vgl. 5.4) können sich die Auflösungen in X- und Y-Richtung jedoch um beliebige Zweierpotenzen unterscheiden, so dass alle Ortsvektoren in der XY-Ebene erreichbar sein müssen. Die Nutzung der dyadischen Dekomposition hätte bei diesen Applikationen zur Folge, dass Daten in einer höheren Auflösung als benötigt übertragen werden müssten, und dass diese Daten vor der Anzeige herunterzuskalieren wären. Beispielsweise müsste eine Region of Interest, die in voller X-Auflösung und $1/4$ Y-Auflösung benötigt wird, zunächst in voller X-Auflösung und $1/2$ Y-Auflösung übertragen werden. Danach wäre eine Verkleinerung der Auflösung in Y-Richtung auf den geforderten Faktor von $1/4$ notwendig.

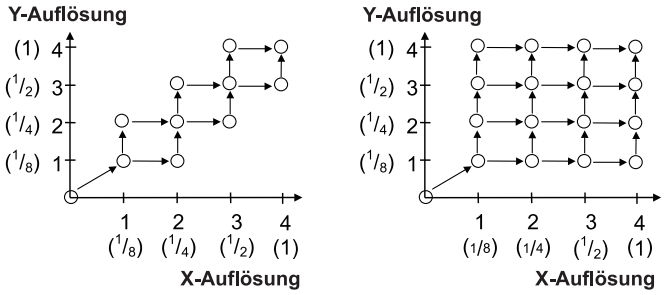


Abbildung 4.3: XY-Ebene des LoD-Raumes (links: dyadisches, rechts: neues Dekompositionsschema).

Um diese unnötigen Datenübertragungen und Verarbeitungsschritte zu vermeiden, soll im Folgenden ein neues Wavelet-Dekompositionsschema vorgeschlagen werden, das die Abbildung aller Ortsvektoren in der XY-Ebene des LoD-Raumes ermöglicht.

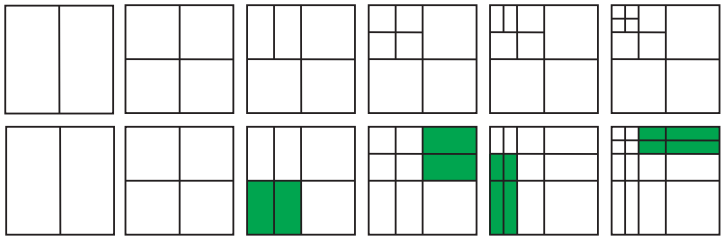


Abbildung 4.4: Neues Zerlegungsschema für größere Unabhängigkeit der Skalierungsfaktoren in X- und Y-Richtung. Oben: Dyadische Zerlegung. Unten: Grundidee des neuen Schemas.

4.4.2 Lösungsprinzip

Ein Dekompositionsschema, das die oben geforderten Eigenschaften aufweist, wird wie folgt konstruiert: Die ersten beiden Zerlegungen zur Erzeugung des LL-Bandes laufen wie bei der dyadischen Wavelet-Transformation ab. Jeder der folgenden Zerlegungsschritte wird nicht nur auf dem aktuellen LL-Band, sondern auch auf den LH- bzw. HL-Bändern ausgeführt. Dabei zerlegen die vertikalen Dekompositionsschritte alle HL-Bänder weiter, die horizontalen Schritte tun dasselbe mit den LH-Bändern. Die Umsetzung der Schritte besteht somit darin, den Hoch- und den Tiefpassfilter nicht nur für das LL-Band zu benutzen, sondern sie stets über die gesamte Breite des Koeffizientenfeldes anzuwenden. Abbildung 4.4 stellt die dyadische und die neue Zerlegungstechnik, die als *XY-unabhängige Zerlegung* bezeichnet werden soll, gegenüber. Die bei der dyadischen Dekomposition entstehenden *Bänder* werden dabei weiter in *Unterbänder* zerlegt. Letztere sind in Abbildung 4.4 schattiert dargestellt.

Die zusätzlichen Zerlegungsschritte bei der neuen Technik sollen im Gegensatz zu den im LL-Band verwendeten dyadischen bzw. Zwei-Richtungs-Zerlegungen als *Ein-Richtungs-Zerlegungen* bezeichnet werden, da sie keinen Gegenpart in der jeweils anderen Richtung besitzen. Mit diesem Zerlegungsschema sind die Auflösungen in X- und Y-Richtung um einen Faktor von bis zu 2^l voneinander unabhängig, wobei l die Anzahl von Zerlegungsstufen[1] ist.

4.4.3 Ungenauigkeiten und Kompressionsartefakte

Wenn das verwendete Wavelet-Filter perfekte Rekonstruktion unterstützt, so bleibt diese Eigenschaft auch bei Verwendung der XY-unabhängigen Dekomposition erhalten. Werden jedoch Integer-Implementationen reellwertiger Filterkoeffizienten eingesetzt, so vergrößert die Nutzung der neuen Methode geringfügig den Rundungsfehler, da mehr Filterungsschritte und damit mehr Rundungsoperationen erforderlich sind.

Ein ernsteres Problem besteht darin, dass das neue Zerlegungsschema neue Artefakte verursachen kann. Abbildung 4.5 zeigt ein Beispiel. Bei niedrigen Bitraten werden horizontale und vertikale Linien verlängert dargestellt. Sie können dann homogene Bereiche überlagern, wie z.B. die Kinnregion in Abbildung 4.5 B. Die Ursache für diese Artefakte liegt darin, dass Strukturen mit hohen Frequenzen in einer Richtung und niedrigen Frequenzen in der anderen Richtung durch die Durchschnittsbildung im Tiefpassteil der Ein-Richtungs-Dekompositionen verlängert und verstärkt werden. Sie treten auf Grund ihrer hohen Koeffizientenwerte bereits früh im progressiv kodierten Datenstrom auf und werden nach der

[1] Zerlegungsstufen werden hier zweidimensional interpretiert, eine Zerlegungsstufe besteht - wie bei separierbaren Transformationen üblich – immer aus einem vertikalen und einem horizontalen Zerlegungsschritt.

A: dyadisch, 5/3, *l=4* **B:** neu, 5/3, *l=4* **C:** neu, 5/3, *l=3* **D:** neu, 5/3+Haar, *l=4*

Abbildung 4.5: Kodierungsartefakte mit neuem Zerlegungsschema und deren Reduktion; Teil des „Barbara"-Testbildes, Kodierung mit 0.25bpp. Der Parameter l bezeichnet die Anzahl der Zerlegungsstufen.

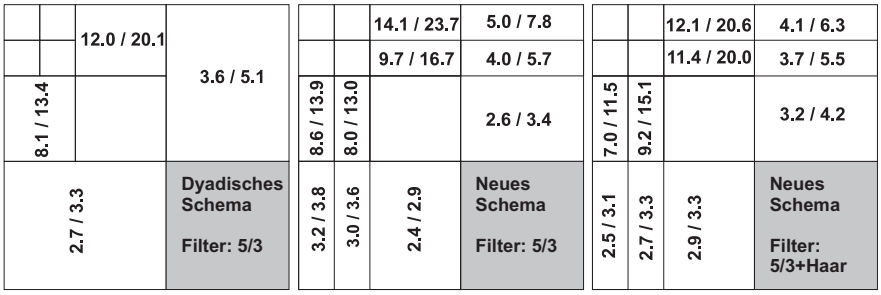


Abbildung 4.6: Vergleich der Standardabweichungen σ und Erwartungswerte μ der Koeffizientenbeträge der einzelnen Bänder für verschiedene Filteralternativen. Notation: σ/μ. Testbild: barbara (Abbildung B.2(c)).

inversen Transformation als lange dünne horizontale oder vertikale Strukturen sichtbar. Zur Abschwächung der Artefakte gibt es zwei mögliche Ansätze:

1. Je höher die Anzahl der Zerlegungsstufen, desto höher ist auch der Anteil der unerwünschten Koeffizienten, die lange dünne vertikale oder horizontale Strukturen repräsentieren (vgl. Abbildungen 4.5 B (4 Zerlegungsstufen) und 4.5 C (3 Zerlegungsstufen)). Daher sollten nur so viele Stufen benutzt werden, wie für den gewünschten Quotienten zwischen X- und Y-Auflösung benötigt werden. Zu wenige Dekompositionsstufen (weniger als 3) wirken sich allerdings negativ auf die Kompressionsrate aus.

2. Auch die Nutzung eines Wavelet-Filters mit schlechteren Dekorrelationseigenschaften und wenigen Filterkoeffizienten für die Ein-Richtungs-Dekompositionen resultiert in einer Artefaktreduktion, da dann die unerwünschten Koeffizienten einen geringeren Betrag besitzen und durch die kürzeren Filterkerne eine geringere „Verlängerung" der Strukturen auftritt. Abbildung 4.6 unterlegt die Aussage über die geringere Magnitude der Koeffizienten bei Nutzung des Haar-Filters durch Darstellung von Erwartungswert und Standardabweichung der Koeffizienten bei verschiedenen Filterkonfigurationen, Abbildung 4.5 D zeigt den visuellen Effekt der Verwendung des Haar-Wavelets statt des 5/3-Wavelets bei 4 Zerlegungsstufen.

4.4.4 Das resultierende Zerlegungsschema

Die wirkungsvollste Artefaktreduktion kann durch eine Kombination dieser beiden Ansätze erreicht werden. Da sich eine zu geringe Anzahl von Dekompositionsstufen negativ

auf die Kompressionsleistung auswirkt, wird aus dem dyadischen und dem ursprünglichen XY-unabhängigen Zerlegungsschema ein hybrides Schema gebildet, das dyadische und Ein-Richtungs-Dekompositionsschritte getrennt behandelt. Zwei Parameter beschreiben das Schema:

- die Anzahl der dyadischen Zerlegungsstufen l und
- die Anzahl der Ein-Richtungs-Zerlegungsstufen il.

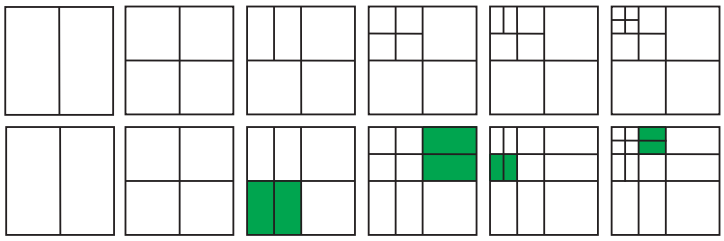


Abbildung 4.7: Resultierendes hybrides Zerlegungsschema zur Artefaktreduktion. Oben: Dyadische Zerlegung. Unten: Beispiel für hybrides Schema mit $l = 3$ und $il = 2$.

Abbildung 4.7 (unten) illustriert das Schema. Der Parameter l steuert die dyadische Zerlegung und die Anzahl der resultierenden Bänder. In wie viele eindirektionale *Unterbänder* die LH- bzw. HL-Bänder zerlegt werden, kann mit dem Parameter il spezifiziert werden. Dabei gilt $1 \leq il \leq l$. Sonderfälle dieses Schemas sind die dyadische Zerlegung ($il = 1$) und die oben eingeführte Grundidee der XY-unabhängigen Zerlegung ($il = l$). Die dyadische Struktur legt Constraints für die in einem bestimmten HL- bzw. LH-Band mögliche Anzahl von Unterbändern fest. Sei j ($1 \leq j \leq l$) die Nummer der Zerlegungsstufe eines HL- bzw. LH-Bandes, wobei die Zählung bei den unmittelbar dem LL-Band benachbarten Band mit 1 beginnt. Dann ergibt sich die Anzahl $il_{max}(j)$ der eindirektionalen Unterbänder, in die ein Band der Zerlegungsstufe j zerlegt wird, wie folgt:

$$il_{max}(j) = min(j, il) \tag{4.1}$$

Diese Dekompositionsstruktur gestattet es, dass X- und Y-Auflösung in der j-ten Zerlegungsstufe um den Faktor $2^{il_{max}(j)}$ differieren können. Die Repräsentation ist somit im Rahmen des obigen Constraints *zoombar*, das heißt, bei schrittweiser Verkleinerung der Auflösung um eine Zweierpotenz bleibt über einen gewissen Bereich die Flexibilität unabhängiger Auflösungen in X- und Y-Richtung gewahrt.

Zusätzlich werden verschiedene Filter eingesetzt: ein biorthogonales Filter für die dyadischen Dekompositionsstufen und das Haar-Filter für die Ein-Richtungs-Dekompositionen. In einem System zur progressiven Bildübertragung reduziert der Einsatz des hybriden Schemas die Artefakte zufrieden stellend. Eventuell noch verbleibende Artefakte in frühen Übertragungsphasen verschwinden relativ schnell als Ergebnis der progressiven Verfeinerung.

4.4.5 Anpassung der Zerotree-Kodierung an das neue xy-unabhängige Zerlegungsschema

Das XY-unabhängige Wavelet-Zerlegungsschema erfordert eine Anpassung des im MoviWaveCodec verwendeten Zerotree-Algorithmus. Abbildung 2.7 auf Seite 27 zeigt die originale Traversierungsreihenfolge und Zerotree-Struktur.

Die Traversierungsfolge muss für das neue Dekompositionsschema derart verändert wer-

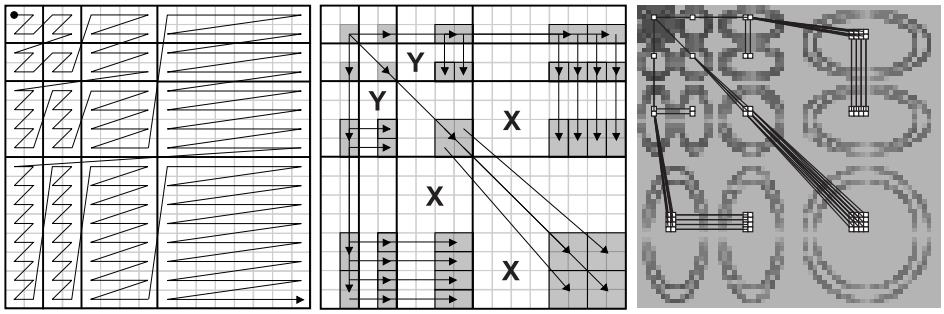


Abbildung 4.8: Modifizierte Zerotree-Kodierung für neues Zerlegungsschema. Links: Traversierungsreihenfolge. Mitte: Generische Zerotree-Struktur. Rechts: Umgesetzte Zerotree-Struktur für hybrides Dekompositionsschema am Beispiel von $l = 3$ und $il = 2$.

den, dass innerhalb eines Bandes die Unterbänder in umgekehrter Reihenfolge ihrer Erzeugung durchwandert werden (siehe Abbildung 4.8 links).

Auch die Zerotree-Struktur muss modifiziert werden (siehe Abbildung 4.8 Mitte). Die Struktur der HH-Bänder bleibt dabei unverändert. Für die HL- und LH-Bänder müssen zusätzlich zu den Beziehungen zwischen den Zerlegungsstufen noch die Beziehungen zwischen Unterbändern repräsentiert werden. Abbildung 4.8 rechts zeigt schließlich die modifizierte Zerotree-Struktur für das hybride Zerlegungsschema mit $l = 3$ und $il = 2$.

Die veränderte Zerotree-Struktur bedingt Änderungen der Zerotree-Map. Wie bereits in 2.4.5.6 beschrieben, ist die Speicherung von Strukturinformation in der Map nur für die Koeffizienten notwendig, die keine Blattknoten im Baum sind. Daher müssen die Unterbänder, die in Abbildung 4.8 (Mitte) mit „X“ oder „Y“ gekennzeichnet sind, nicht in der Zerotree-Map enthalten sein. Zur Vereinfachung der Implementierung reicht es aus, nur die mit „X“ gekennzeichneten Bereiche wegzulassen, da hier die größte Speicherplatzeinsparung erzielt wird.

4.4.6 Performance des neuen Schemas

4.4.6.1 Rechenzeit

Das neue Schema erfordert mehr Rechenzeit als die dyadische Dekomposition, da zusätzliche Filterungsschritte auszuführen sind. Seien m und n die Dimensionen des Bildes und l die Anzahl der Zerlegungsstufen. Eine Filteroperation sei definiert als die Berechnung eines Wavelet-Koeffizienten[2]. Da die Zerlegung separierbar ist, braucht man nur alle horizontalen Zerlegungsschritte zu betrachten und das Ergebnis zu verdoppeln. Es gilt für die Anzahl der Filteroperationen F_{dyad} bei der dyadischen Dekomposition:

$$F_{dyad}(l) = 2mn \cdot \sum_{i=0}^{l-1} \frac{1}{2^{2i}} \tag{4.2}$$

Die Anzahl der für das neue Zerlegungsschema zusätzlich notwendigen Filteroperationen

[2]Eigentlich müsste man die Berechnung von Tiefpass- und Hochpasskoeffizienten getrennt betrachten. Da aber immer die Hälfte aller Koeffizienten in eine dieser Klassen fällt, ist obige Vereinfachung zulässig.

sei exemplarisch zunächst für die ersten fünf Zerlegungsstufen separat angegeben:

$$F_0 = 0 \tag{4.3}$$

$$F_1 = 2mn \cdot \left(\frac{1}{4}\right) \tag{4.4}$$

$$F_2 = 2mn \cdot \left(\frac{1}{4} \cdot \frac{1}{2} + \frac{1}{16}\right) \tag{4.5}$$

$$F_3 = 2mn \cdot \left(\frac{1}{4} \cdot \frac{1}{4} + \frac{1}{16} \cdot \frac{1}{2} + \frac{1}{64}\right) \tag{4.6}$$

$$F_4 = 2mn \cdot \left(\frac{1}{4} \cdot \frac{1}{8} + \frac{1}{16} \cdot \frac{1}{4} + \frac{1}{64} \cdot \frac{1}{2} + \frac{1}{256}\right) \tag{4.7}$$

$\ldots$

Durch Zusammenfassen und Verallgemeinern der Einzelschritte in den Gleichungen (4.3) bis (4.7) erhält man die Anzahl der für das neue Zerlegungsschema maximal zusätzlichen notwendigen Filteroperationen F_{xy+}:

$$\begin{aligned} F_{xy^+}(l) &= 2mn \cdot \sum_{j=1}^{l-1} \sum_{i=0}^{l-j} \frac{1}{2^i \cdot 2^{2j}} \\ &= 2mn \cdot \sum_{j=1}^{l-1} \sum_{i=0}^{l-j} \frac{1}{2^{2j+i}} \end{aligned} \tag{4.8}$$

Betrachtet man das Konvergenzverhalten der Gleichungen (4.2) und (4.8), so ergibt sich:

$$\lim_{l \to \infty} F_{dyad}(l) = 2mn \cdot \frac{4}{3} \tag{4.9}$$

$$\lim_{l \to \infty} F_{xy^+}(l) = 2mn \cdot \frac{2}{3} \tag{4.10}$$

Der Anteil der für die XY-unabhängige Dekomposition zusätzlich erforderlichen Filteroperationen konvergiert damit gegen 50% der für die dyadische Dekomposition notwendigen Filteroperationen. Ein praktisches Beispiel: Für die vierstufige dyadische Zerlegung eines Graustufenbildes mit 512x512 Pixeln ergeben sich nach Gleichung (4.2) für $F_{dyad} = 2 \cdot 512^2 \cdot 85/64 = 696320$ Operationen. Nach Gleichung (4.8) erfordert die XY-unabhängige Zerlegung zusätzlich dazu $F_{xy^+} = 2 \cdot 512^2 \cdot 35/64 = 286720$ (41.2%) weitere Operationen. In der praktischen Implementierung MOVIWAVECODEC steigt die Zeit[3] zur Berechnung der Wavelet-Transformation obigen Bildes jedoch nur um 31% von 0.34 s für die dyadische Zerlegung auf 0.45 s für das neue Schema. Die Rechenzeit für die inverse Transformation wächst um 44% von 0.31 s auf 0.44 s. Eine mögliche Erklärung für die Diskrepanz kann in der teilweise parallelen Ausführung von Befehlen und in den Caching-Mechanismen moderner Prozessoren gesucht werden.

Die Nutzung von mehr Ebenen in den Teilbäumen durch die zusätzlichen Dekompositionsschritte beim neuen Zerlegungsschema führt zu einer geringfügig schnelleren Zerotree-Kodierung im Vergleich zur dyadischen Zerlegung. Tabelle 4.1 zeigt die Rechenzeiten für verschiedene Bitraten. Der geringere Kodieraufwand gleicht somit teilweise die etwas höhere Transformationszeit wieder aus. In praktischen Anwendungen sind die resultierenden Geschwindigkeitsunterschiede so gering, dass der Nutzer sie nicht bemerkt.

[3] Alle Zeiten wurden auf einem Intel PENTIUM P133 gemessen.

bpp	vermessene Pipelineschritte	Kodierzeit [s] $il=1$	Kodierzeit [s] $il=l$	Kodierzeit [s] %	Dekodierzeit [s] $il=1$	Dekodierzeit [s] $il=l$	Dekodierzeit [s] %
0.00	T	0.34	0.45	131.0	0.31	0.44	144.2
0.25	Q^+V^+AE	0.97	0.91	94.4	0.85	0.80	93.7
1.00	Q^+V^+AE	1.63	1.61	98.4	1.59	1.52	95.6
lossless	Q^+V^+AE	4.20	4.00	95.3	4.60	4.34	94.5
0.25	TQ^+V^+AE	1.31	1.36	103.9	1.16	1.24	107.1
1.00	TQ^+V^+AE	1.98	2.06	104.0	1.90	1.97	103.5
lossless	TQ^+V^+AE	4.54	4.45	98.0	4.90	4.79	102.5

Tabelle 4.1: MOVIWAVE-Kodierungs- und -Dekodierungszeiten für dyadisches ($il = 1$) und neues Dekompositionsschema ($il = l$) mit $l = 4$ bei verschiedenen Bitraten[5].

4.4.6.2 Speicherplatz

Für die Speicherung der Wavelet-Koeffizienten belegen sowohl dyadisches als auch XY-unabhängiges Schema denselben Speicherplatz. Das gilt jedoch nicht für die Datenstrukturen zur Unterstützung der Aggregationskodierung. Hier benötigt die vorgeschlagene Erweiterung für die Hälfte aller Koeffizienten einen Eintrag in der Zerotree-Map. Sie belegt damit den doppelten Speicher im Vergleich zur dyadischen Zerlegung, die nur für ein Viertel der Koeffizienten Zerotree-Map-Einträge erfordert. In der Realisierung MOVIWAVE-CODEC steigt dadurch der Gesamtspeicherbedarf für Koeffizientenfeld und Zerotree-Map von 2.5 auf 3 Bytes pro Koeffizient und Farbkanal.

4.4.6.3 Diskussion

Das neue Dekompositionsschema wurde entworfen, um in der Wahl der Auflösungskombinationen bei der LoD-Definition volle Flexibilität zu gewährleisten. Wie gezeigt wurde, bedingt diese Flexibilität zusätzliche Artefakte, die jedoch für die progressive Übertragung beherrschbar sind, sowie einen gewissen Mehraufwand an Rechenleistung und Speicherplatz.

Es existieren Anwendungen, die trotz erhöhten Aufwandes für die Wavelet-Transformation durch Ausnutzung der Flexibilität des neuen Schemas insgesamt einen Performancegewinn erfahren. Das ist insbesondere dann der Fall, wenn eine Applikation bei Nutzung des dyadischen Schemas Bildteile mit einer Kombination von X- und Y-Auflösung benötigt, die mit dem dyadischen Schema nur durch Übertragung der Daten in höherer Auflösung, gefolgt von einer Auflösungsverringerung auf Clientseite, realisierbar wäre.

Zum Beispiel muss der RECHTECKIGE FISHEYE-VIEW (siehe 5.4) mit dem neuen Schema bei einer typischen Parametrisierung (siehe Tabelle 5.3 auf Seite 99) nur 66% der Datenmenge übertragen, die bei Nutzung der dyadischen Dekomposition benötigt werden würde. Da bei dieser Applikation die Auflösungen in X- und Y-Richtung um einen Faktor größer als 2 differieren können, wird so Übertragungsbandbreite und – aufgrund der nicht notwendigen Verkleinerung solcher Bildteile auf der Dekoderseite – auch Rechenzeit gespart, was den zusätzlichen Aufwand rechtfertigt.

Andererseits gibt es auch Applikationen, in denen die höhere Flexibilität des neuen Schemas nicht benötigt wird und deshalb auch dessen Overhead nicht in Kauf genommen

[5] Kodierung und Dekodierung von sechs 512x512 Pixel großen Testbildern (lena, barbara, boat, mandrill, goldhill, peppers; siehe Abbildung B.2) auf einem INTEL PENTIUM 133. Zur verlustfreien Kodierung wurde nur das Testbild lena verwendet, da verschiedene Bilder verschiedene Bitraten ergeben und die Rechenzeit abhängig von der Bitrate ist. Die Tabelle zeigt die Zeiten nur für die Ausführung der Transformation (T), nur für die Zerotree-(De)Kodierung (Q^+V^+AE) sowie für Transformation und (De)Kodierung zusammen (TQ^+V^+AE).

werden muss. Daher sollte die Entscheidung über das verwendete Dekompositionsschema beim Anwendungsentwickler liegen. Ist die zusätzliche Flexibilität nicht erforderlich, deaktiviert die Parametrisierung mit $il = 1$ die zusätzlichen Zerlegungsstufen, und das dyadische Schema kommt zum Einsatz.

4.5 Das MOVIROILOD-Verfahren: Waveletbasierte Umsetzung von RoIs und LoDs

Nachdem eine Umsetzung der neuen Wavelet-Dekomposition auf der Basis des MOVIWAVE-Algorithmus beschrieben wurde, werden nun in diesem Abschnitt die notwendigen Erweiterungen vorgestellt, um diesen Algorithmus LoD- und RoI-fähig zu machen. Das resultierende Verfahren soll als MOVIROILOD-Verfahren bezeichnet werden.

4.5.1 Anpassung von Quantisierung und Traversierung für RoIs und LoDs

4.5.1.1 Umsetzung der Dimensionen des LoD-Raumes

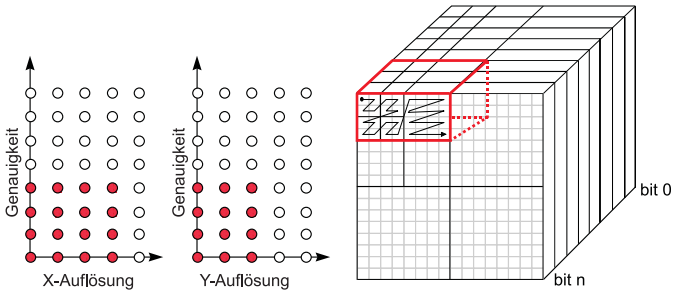


Abbildung 4.9: Anpassung der sukzessiven Approximation und der Traversierung des Wavelet-Koeffizientenfeldes für lokale LoDs.

In 3.2 wurde ein lokaler LoD als Menge von Ortsvektoren in einem LoD-Raum $\mathbb{L}$ mit den Komponenten X-Auflösung, Y-Auflösung, Genauigkeit und Farbe spezifiziert. Nach Durchführung der Wavelet-Transformation liegt jeder Farbkanal als eigenes Koeffizientenfeld vor. Die Bänder bzw. Unterbänder repräsentieren Kombinationen von *Auflösungen* in X- und Y-Richtung, also die Ortsvektoren in der XY-Ebene des LoD-Raumes. Die Dimension *Genauigkeit* kann durch die Bitebenen der Koeffizienten dargestellt werden.

Ein Ziel-LoD wird auf der Basis von MOVIWAVE realisiert, indem die Traversierung jedes Koeffizientenfeldes auf diejenigen Unterbänder (Auflösung) und die sukzessive Approximation auf diejenigen Bitebenen (Genauigkeit) beschränkt wird, die den Elementen des lokalen LoDs entsprechen (siehe Abbildung 4.9). Die Dimension *Farbe* kann durch abwechselnde Traversierung der Koeffizientenfelder, die den Farbkanälen entsprechen, realisiert werden. Farbtabellenbilder werden hierbei wegen ihrer stark abweichenden Charakteristik nicht unterstützt; eine effiziente Kodierung für diese Bildklasse wird in Kapitel 6 beschrieben.

4.5.1.2 Umsetzung von RoIs

Grundidee. Zur Umsetzung von RoIs muss die gemäß Ziel-LoD bereits auf die entsprechenden Unterbänder eingeschränkte Traversierung derart angepasst werden, dass nur solche Koeffizienten besucht werden, die zum Footprint der RoI beitragen. Dazu ist zunächst eine Mehrfachauflösungsrepräsentation des Footprints (die als *Footprint-Maske* bezeichnet werden soll) im Wavelet-Raum erforderlich. Abbildung 4.10 illustriert das für eine rechteckige RoI. Bei der Berechnung der Mehrfachauflösungsrepräsentation muss die Konsistenz der Zerotree-Struktur erhalten bleiben. Das bedeutet, wenn ein Koeffizient innerhalb der Footprint-Maske für ein bestimmtes Band bzw. Unterband liegt, dann müssen auch alle seine „Vorfahren“ in der Zerotree-Struktur innerhalb der Footprint-Maske für die entsprechenden Vorfahren-Bänder bzw. -Unterbänder liegen.

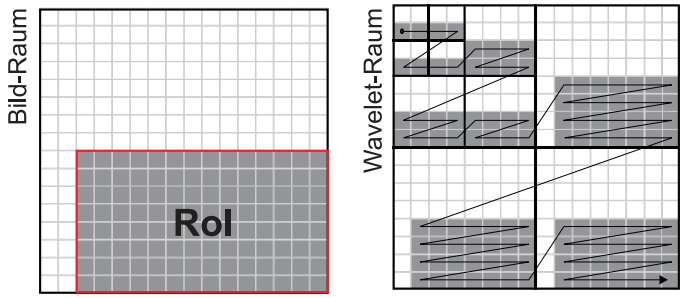


Abbildung 4.10: Anpassung der Traversierung des Wavelet-Koeffizientenfeldes für RoIs.

Die obige Randbedingung stellt die Konsistenz der Zerotree-Strukturen bei der Kodierung von RoIs mit einer RoI-Maske sicher. Wenn man die inverse Wavelet-Transformation betrachtet, spielt aber noch ein weiterer Aspekt eine Rolle: Die Wavelet-Filterkerne besitzen eine gewisse Länge, so dass auch Koeffizienten, die in einer kleinen Umgebung um die Footprint-Maske liegen, zur dekodierten RoI beitragen. Insbesondere für Archivierungsanwendungen in der Medizin, wo eine RoI in der Regel verlustfrei gespeichert werden muss, ist daher eine Erweiterung der Maske erforderlich (vgl. 2.5.2.2 sowie [LKB98, NC98]).

Repräsentation der RoI-Maske. Nachteilig an obigem Verfahren ist, dass die RoI-Maske als Bitmap gespeichert werden muss. Eine Speicherung in Form von Polygonen hat demgegenüber den Vorteil eines geringeren Speicherplatzbedarfes. Die Bitmap-Repräsentation ist besonders dann unvorteilhaft, wenn zusätzlich zu existierenden RoIs im Rahmen eines interaktiven Verfahrens während der Übertragung neue RoIs spezifiziert werden, die die bereits vorhandenen überlappen. In diesem Fall ist bei jeder neuen RoI-Definition eine neue Bitmap erforderlich, wenn eine differenzielle Übertragung unterstützt werden soll. Deshalb soll in dieser Arbeit die *spannenbasierte Traversierung einer polygonbasierten Repräsentation* verwendet werden (siehe 4.5.1.3), die gegenüber der Bitmap-Repräsentation bedeutend Speicherplatz sparender arbeitet. Dazu muss jedoch die Länge des Wavelet-Filterkerns vernachlässigt werden. Praktische Realisierungen haben gezeigt, dass für den Einsatzbereich der bedarfsgesteuerten Bildübertragung die dadurch entstehenden Ungenauigkeiten am Rand der RoI nicht störend in Erscheinung treten.

Traversierung. Bei der Traversierung ist eine Bitebene eines Unterbandes die kleinste zusammenhängende Einheit innerhalb eines Koeffizientenfeldes. Das bedeutet, dass die Kodierung nur an definierten Punkten unterbrochen werden kann, nämlich immer dann, wenn in einem Koeffizientenfeld eine Bitebene aller Koeffizienten innerhalb eines Unterbandes, die zu einer RoI beitragen, kodiert wurde. Da hier die Kodierung nicht an jeder beliebigen Position abgebrochen werden kann, hat diese Einschränkung zur Folge, dass die exakte Steuerbarkeit der Größe des Ausgabedatenstromes verloren geht; stattdessen kann diese Größe nur ungefähr eingehalten werden. Es hat sich gezeigt, dass diese Abweichung

in vielen Fällen vernachlässigbar ist. Abbildung 4.11 belegt das anhand der Kodierung des 512x512 Pixel großen Testbildes „lena“. Hier tritt jeweils dann ein Sprung in der erzielten Bitrate des kodierten Bildes auf, wenn die geforderte Größe der kodierten Datei um ein Byte größer ist als die Dateigröße nach der vollständigen Kodierung einer Bitebene in einem Unterband.

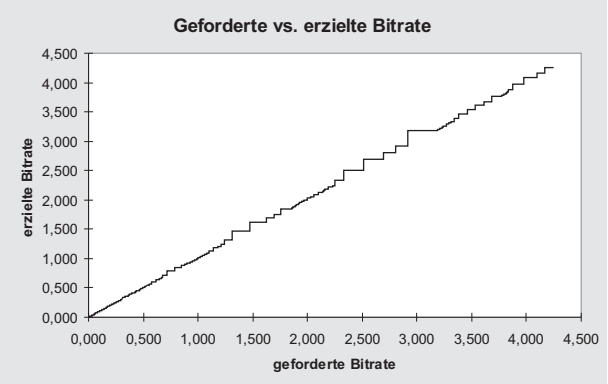


Abbildung 4.11: Abweichung der kodierten Bitrate von der Zielbitrate bei der Kodierung des Testbildes „lena“ (Abbildung B.2(b)) mit einer RoI, die das gesamte Bild umfasst ($l = 4$, $il = 1$).

Bei großen Bildern können stärkere Abweichungen auftreten. In diesen Fällen erscheint eine Unterbrechbarkeit an beliebigen Positionen wünschenswert. Da jedoch eine redundanzfreie Verfeinerbarkeit gefordert ist, müsste dann die Berechnung der partitionierenden Teilmengen je RoI und je Band bzw. Unterband zwei Mengen von Koeffizienten berücksichtigen: Die Menge der bereits kodierten und die Menge der noch nicht kodierten Koeffizienten. Dies würde eine höhere Genauigkeit bei der Steuerung der Bitrate ermöglichen, jedoch eine erhöhte Komplexität in der Umsetzung bedingen. In Abschnitt 5.1.4 wird evaluiert, welche Auswirkungen die Unterbrechbarkeit an festen Positionen auf das Antwortverhalten des auf der Grundlage von MOVIROILOD umgesetzten interaktiven Bildübertragungssystems hat.

4.5.1.3 Umsetzung der partitionierenden Teilmengen für überlappende Regions of Interest

Bisher wurde beschrieben, wie die Traversierung des Koeffizientenfeldes für eine einzelne RoI und den zugeordneten Ziel-LoD erfolgt. Dieses Schema kann leicht auch auf mehrere RoIs, die sich nicht überlappen, angewendet werden. Probleme treten bei der Überlappung von RoIs auf, da hier solche Koeffizienten, die zu mehreren RoIs beitragen, mehrfach übertragen würden. Zur Sicherstellung der redundanzfreien Übertragung wurde bereits in 3.2 das Konzept der partitionierenden Teilmengen vorgeschlagen. Hier soll die Umsetzung dieses Konzeptes im Wavelet-Raum auf der Basis einer Mehrfachauflösungsrepräsentation beschrieben werden.

Konvertierung der partitionierenden Teilmengen in den Wavelet-Raum. Jede der partitionierenden Teilmengen in $\widehat{P}$ beschreibt eine Menge von Pixeln, die in den Footprints einer bestimmten Menge von RoIs enthalten sind. Die Konvertierung der Footprints in den Wavelet-Raum (siehe 4.5.1.2) impliziert auch eine Konvertierung der partitionierenden Teilmengen, wodurch partitionierende Teilmengen von Wavelet-Koeffizienten entstehen.

Dazu wird jedes Band bzw. Unterband[6] der Wavelet-Repräsentation als verkleinerte Version des Originalbildes betrachtet und eine Kopie der Footprints aller RoIs entsprechend skaliert sowie relativ zur linken oberen Ecke des entsprechenden (Unter)bandes transliert. Die Skalierungsfaktoren sind generell Zweierpotenzen. Bei Verwendung der dyadischen Dekomposition sind sie in X- und Y-Richtung gleich, für die XY-unabhängige Zerlegung können sie voneinander abweichen. Als Ergebnis dieses Schrittes existiert je (Unter)band im Wavelet-Raum für jeden Footprint eine entsprechend verkleinerte Kopie. Die in den Wavelet-Raum skalierten Footprints werden nun analog zu den partitionierenden Teilmengen P_i von Pixeln für jedes Band bzw. Unterband j separat in partitionierende Teilmengen $\overline{P}_i^j$ von *Wavelet-Koeffizienten* unterteilt (vgl. Gleichung (3.5) und folgende auf Seite 45). Jedes $\overline{P}_i^j$ bildet somit eine verkleinerte und translierte Repräsentation der entsprechenden partitionierenden Teilmenge P_i von Pixeln im Bildraum.

Notwendigkeit einer Mehrfachauflösungsrepräsentation. Um die Konsistenz der Zerotree-Kodierung einhalten zu können, gilt für jede partitionierende Teilmenge im Wavelet-Raum die Bedingung, dass vor der Traversierung eines Koeffizienten dieser Menge alle Vorfahren-Koeffizienten in der Baumstruktur bereits traversiert worden sein müssen (vgl. 4.5.1.2). Durch die aus Bändern bzw. Unterbändern bestehende Dekompositionsstruktur wird eine Hierarchie auf den partitionierenden Teilmengen $\overline{P}_i^j$ der Wavelet-Koeffizienten definiert. Diese Hierarchie entspricht wegen der abgestuften Skalierungsfaktoren einer Mehrfachauflösungsrepräsentation der partitionierenden Teilmengen P_i von Pixeln.

Betrachtet man alle Koeffizienten C_{xy} einer partitionierenden Teilmenge $\overline{P}_i^j$, so kann die Mehrfachauflösungseigenschaft durch Definition der „Elternmenge“ $\overline{P}_i^{j'}$ von Koeffizienten $C_{x'y'}$ gemäß einer der folgenden beiden Bedingungen realisiert werden:

$$C_{xy} \in \overline{P}_i^j \quad \Leftrightarrow \quad C_{x'y'} \in \overline{P}_i^{j'} \tag{4.11}$$

oder

$$C_{xy} \in \overline{P}_i^j \quad \Rightarrow \quad C_{x'y'} \in \overline{P}_i^{j'}\,. \tag{4.12}$$

Während die strengere Bedingung (4.11) einfacher umzusetzen ist, lässt die Bedingung (4.12) eine höhere Kompressionsrate erwarten, da in höher aufgelösten Teilbändern weniger Koeffizienten als zu RoIs zugehörig klassifiziert werden.

Implizite Realisierung der partitionierenden Teilmengen. Eine explizite Speicherung der partitionierenden Teilmengen für jedes Teilband ist sehr speicheraufwändig, weshalb nach einer effizienteren Umsetzung gesucht wurde. Durch Diskretisierung der Footprints auf einem Mehrfachauflösungsgitter unter Beachtung der Bedingung (4.12) können die partitionierenden Teilmengen von Wavelet-Koeffizienten *implizit* realisiert werden. Hierbei wird Speicherplatz gespart, da nur die Footprint-Geometrie im Bildraum explizit in Form von Punktlisten gespeichert werden muss. Je RoI werden zusätzlich zu dieser Punktliste der Ziel-LoD, der Zustands-LoD und das LoD-Delta verwaltet.

Zur Diskretisierung wurde eine Methode entwickelt, die auf dem Scanline-Algorithmus zum Füllen von 2D-Polygonen (siehe [FvDFH90, S. 92ff]) basiert. Hierbei wird die Menge der Koeffizienten eines Bandes bzw. Unterbandes in so genannte *Spannen* aufgeteilt. Eine solche Spanne kennzeichnet eine Folge von Koeffizienten auf einer Scanline, die zur gleichen Menge von überlappenden RoIs gehören, also in derselben partitionierenden Teilmenge liegen.

Die Grundidee dieses Verfahrens ist, dass sich die Zugehörigkeitsbeziehungen von Wavelet-Koeffizienten zu RoIs nur an definierten Punkten auf einer Scanline ändern können. Diese

[6] Bei der dyadischen Dekomposition sind Bänder, bei der XY-unabhängigen Zerlegung Bänder und Unterbänder das Ergebnis der Wavelet-Transformation.

Punkte sind durch die Kanten der Footprints aller RoIs bestimmt; hierbei wird die skalierte Repräsentation der Kanten im aktuellen (Unter)band benutzt. Eine aktive Kantenliste (*active edge table*) enthält die Kanten aller Footprints, die einen Schnittpunkt mit der aktuellen Scanline besitzen. Diese Liste ist nach der X-Koordinate des Schnittpunktes der jeweiligen Kante mit der Scanline sortiert. Spannen von Koeffizienten werden in dieser sortierten Liste zwischen je zwei benachbarten Kanten ermittelt, und für jede Spanne wird die Menge R_0 von RoIs bestimmt, zu deren Footprints diese Spanne beiträgt.

Bei geeigneter Steuerung der Spannenbestimmung (siehe 4.5.2.2) enthält die Menge R_0 mindestens eine RoI r_0, für die aktuell Daten übertragen werden sollen. Diese Daten werden durch den Ortsvektor $\overrightarrow{l_0}$ im LoD-Raum bezeichnet, der das erste Element des LoD-Deltas der RoI r_0 bildet. Die Komponenten von $\overrightarrow{l_0}$ sind: der Index des aktuellen Farbkanals, die Nummer der aktuellen Bitebene des Koeffizientenfeldes sowie die Nummer des aktuellen Unterbandes. Eine Zerotree-Kodierung der Koeffizienten einer Spanne und damit eine Übertragung von Daten erfolgt genau dann, wenn für diese Spanne noch keine $\overrightarrow{l_0}$ entsprechenden Daten kodiert wurden. Das ist der Fall, wenn $\overrightarrow{l_0}$ bei keiner der RoIs aus R_0 im Zustands-LoD enthalten ist. Damit ist eine redundanzfreie Übertragung möglich.

Diskretisierung bei Beachtung der Mehrfachauflösungseigenschaft. Während der zeilenweisen Traversierung des Koeffizientenfeldes muss für jede Zeile die *active edge table* aktualisiert werden, da Kanten hinzukommen können oder entfernt werden müssen. Neu hinzugekommene Kanten müssen in der entsprechenden Auflösungsstufe der Mehrfachauflösungsrepräsentation aus den Koordinaten der Footprint-Polygone berechnet werden, so dass Bedingung (4.12) erfüllt ist. Dafür soll im folgenden ein Verfahren vorgeschlagen werden.

Die Mehrfachauflösung wird durch Abbildung der Wavelet-Koeffizienten auf *Superpixel* verschiedener Größe (g_x, g_y) im Bild abhängig von der Skalierung des aktuellen (Unter)bandes realisiert. Ein solches Superpixel repräsentiert einen Bereich mit einer Breite g_x und einer Höhe g_y im Bildraum, der einem Koeffizienten in einem bestimmten (Unter)band der Wavelet-Repräsentation entspricht. Sowohl g_x als auch g_y sind stets Zweierpotenzen. Abbildung 4.12 illustriert die Grundidee. Die verschieden dunkel schattierten Bereiche stellen Superpixel verschiedener Größe dar, die zur Sicherung von Bedingung (4.12) Anteil an dem eingezeichneten Dreieck haben müssen.

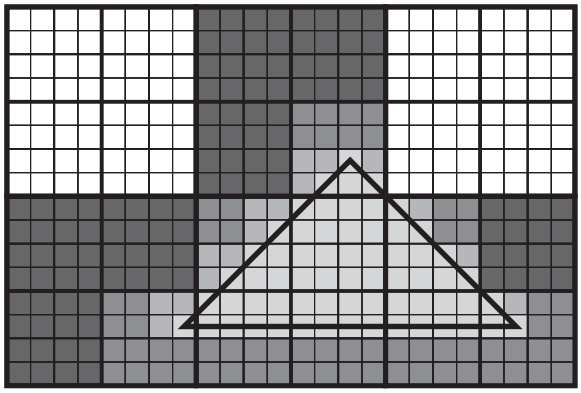


Abbildung 4.12: Mehrfachauflösungsrepräsentation eines Dreiecks.

Wichtig ist es weiterhin, zwischen *Superpixelkoordinaten* (die ein Superpixel, also eine rechteckige Teilfläche des Bildes mit den Kantenlängen g_x und g_y adressieren) und *Punktkoordinaten* (die einen Punkt in der Ebene im Bildraum darstellen) zu unterscheiden. Die Eckpunkte der Footprints werden mithilfe von Punktkoordinaten, die Kanten in der *active edge table* mithilfe von Superpixelkoordinaten definiert. Superpixelkoordinaten sind ganz-

zahlig. Die Berechnung der Koordinaten des Superpixels $P' := [x', y']^T$, in dem ein Punkt $P := [x, y]^T$ liegt, geschieht wie folgt:

$$x' = \lfloor x/g_x \rfloor \qquad y' = \lfloor y/g_y \rfloor \tag{4.13}$$

Zugehörigkeitsbeziehungen zu partitionierenden Teilmengen von Wavelet-Koeffizienten können sich immer dann ändern, wenn das aktuelle Superpixel Element der *Begrenzung* des Footprints ist. Dies ist der Fall, wenn eine der horizontalen Begrenzungslinien[7] des Superpixels an der X-Position x_S von einer Kante des Footprints geschnitten wird.

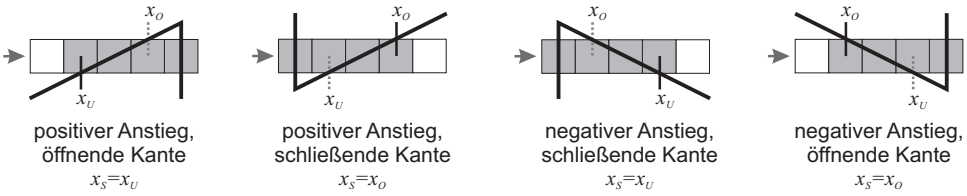


Abbildung 4.13: Mögliche Fälle bei der Berechnung der X-Koordinate x_S des Schnittpunktes einer Spanne mit einer nicht-senkrechten Kante.

Berücksichtigt man die Richtung der Traversierung einer Scanline und die Topologie der Footprints, so können für nicht-senkrechte Kanten die in Abbildung 4.13 skizzierten vier Fälle auftreten. Abhängig von der Klassifizierung der Kante und vom Vorzeichen ihres Anstiegs wird zur Berechnung von x_S entweder die untere oder die obere Superpixelbegrenzung verwendet. Hierbei wird eine Kante als *öffnend* klassifiziert, wenn die aktuelle Scanline vor Erreichen dieser Kante bereits eine gerade Anzahl von Kanten desselben Footprints geschnitten hat; bei einer *schließenden* Kante ist diese Anzahl ungerade.

Die Schnittpunktberechnung muss je Kante nur für die erste Scanline durchgeführt werden, die diese Kante schneidet. Sie erfolgt in Punktkoordinaten, und das Ergebnis wird in Superpixelkoordinaten umgerechnet. Für die darauf folgenden Scanlines wird ein inkrementeller Algorithmus genutzt. Der bei jeder Berechnung auftretende Rundungsfehler wird gespeichert und bei der jeweils folgenden Scanline berücksichtigt.

Einschränkungen und Optimierungen. Das beschriebene Verfahren funktioniert sowohl für konvexe als auch für konkave polygonale Footprints. Probleme verursachen jedoch Polygone, die sich selbst berühren oder durchdringen, da sich hierbei die Berechnungsvorschrift von x_S während des Verlaufes einer Kante ändern kann. Ein solcher Fall kann jedoch erkannt und das Polygon durch seine konvexe Hülle ersetzt werden. Hierzu steht eine Anzahl veröffentlichter Algorithmen, wie Jarvis March [Jar73, CLR91], Graham Scan [Gra72] oder Quickhull [BDH96], zur Verfügung.

In vielen Applikationen werden aufgrund ihrer einfachen interaktiven Definierbarkeit ausschließlich rechteckige RoIs verwendet. Hierfür kann zusätzlich zur Spannenkohärenz, die der Scanline-Füllalgorithmus ausnutzt, auch noch die Kohärenz zwischen den Rechtecken in Y-Richtung genutzt werden. Das heißt, die Spannenaufteilung kann in einer Reihe von aufeinander folgenden Scanlines genau dann als gleich angenommen werden, wenn keine Änderungen an der aktiven Kantenliste erfolgen. In diesen Fällen kann der Aufwand für die Neuberechnung der Spannen gespart werden. Die Shape Algebra von Steinhart [Ste91] generalisiert diese Idee für Polygone.

[7] Da im Scanline-Füllalgorithmus waagerechte Kanten nicht berücksichtigt werden, genügt die Betrachtung der Schnittpunkte mit den horizontalen Begrenzungen der Superpixel.

4.5.1.4 Performance-Implikationen

Im Vergleich zu einem monolithischen Enkoder, der die Traversierungsfolge in Abbildung 4.8 (links) auf Seite 58 umsetzt, verlangsamt die Spannentraversierung die Berechnung. Tabelle 4.2 zeigt, dass – besonders bei niedrigen Bitraten – die Rechenzeiten bis zu einem Drittel steigen können, dass jedoch die absolute Differenz für die verwendeten Bildgrößen eher gering ist. Bei höheren Bitraten ist der Anteil der Spannentraversierung an der Gesamtrechenzeit geringer; verlustfreie Kodierung erfordert nur noch einen um 12% höheren Aufwand.

Methode	bpp	Zeit, MOVIWAVE	Zeit, MOVIROILOD	Verlangsamung
verlustfreie Kodierung	4.95	3.8 s	4.3 s	12.9%
verlustfreie Dekodierung	4.95	4.2 s	4.7 s	11.6%
Kodierung	0.25	0.7 s	0.9 s	28.5%
Dekodierung	0.25	0.6 s	0.8 s	33.3%

Tabelle 4.2: Vergleich der durchschnittlichen Kodierungs- und Dekodierungszeiten des monolithischen MOVIWAVE-Codecs und des spannenbasierten MOVIROILOD-Codecs[9].

Da in mobilen Umgebungen häufig die Übertragungsbandbreite der Flaschenhals ist, wird dieses Problem durch die eingesparte Übertragungszeit auf Grund von Redundanzvermeidung und Einschränkung der Übertragung auf die Regions of Interest mehr als kompensiert. Es sei hier daran erinnert, dass z.B. bei Nutzung einer Gauß-Pyramide wie im FlashPix-Format (siehe 2.4.5.11) bereits 33% Redundanz anfallen würden.

Das vorgestellte Schema wurde entwickelt, um eine niedrige Bandbreite durch den Einsatz von Rechenleistung zu kompensieren. Daher soll im Folgenden versucht werden, die Auswirkungen des Kompromisses zwischen Datenrate und Rechengeschwindigkeit an Hand eines praktischen Client-Server-basierten Anwendungsszenarios zur Bildübertragung exemplarisch zu quantifizieren. Dazu erfolgt eine Extrapolation des zu erwartenden Durchsatzes[10] auf Basis der Daten in Tabelle 4.2.

Bei niedrigen Bitraten bis 0.25 bpp beträgt der durchschnittliche Durchsatz bei der Kodierung 72818 bps, bei der Dekodierung 81920 bps. Für die verlustfreie Kodierung lauten die entsprechenden Werte 301770 bps für die Kodierung und 276088 bps für die Dekodierung. Das bedeutet, dass bei niedrigen Bitraten (am Anfang des eingebetteten Datenstromes) der Rechenaufwand für die Ausgabe eines Bits höher ist als bei hohen Bitraten, ein Fakt, der sich durch die listenlose Implementierung des Zerotree-Algorithmus (siehe 4.3.1) begründen lässt.

Auf Clientseite ist die Dekodiergeschwindigkeit zur Bildübertragung über langsame Kanäle vollkommen ausreichend – hier stellt der Übertragungskanal selbst bei einer ISDN-Anbindung mit 64kbps den Flaschenhals dar. Ein interaktiver Bildserver, der mit einem PENTIUM 133 als Prozessor ausgerüstet ist, könnte bei der Kodierung mit niedrigen Bitraten theoretisch bis zu 7 Clientrechner bedienen, die über GSM mit 9600 bps angebunden sind. Bei der verlustfreien Kodierung wären es bis zu 31 Clients. Da man in der Regel leistungsfähigere Maschinen als Server benutzt, liegt in der Praxis die zu erwartende Anzahl bedienbarer Clients höher.

[9] Durchschnitt der Zeiten zur Kodierung und Dekodierung von sechs 512x512 Pixel großen Graustufenbildern (lena, barbara, boat, mandrill, goldhill, peppers) auf einem PENTIUM 133. Der monolithische Codec unterstützt keine RoIs. Zur Sicherstellung der Vergleichbarkeit verwendete daher der spannenbasierte Codec eine einzelne RoI, die sich über das gesamte Bild erstreckte.

[10] Der Durchsatz ergibt sich als Produkt der Anzahl der Pixel im Bild und der Bitrate dividiert durch die Rechenzeit.

4.5.2 Steuerung der Übertragung

Bisher wurde das Kodierungs- bzw. Dekodierungsverfahren beschrieben und diskutiert, wie Regions of Interest und Levels of Detail integriert werden können. Ziel dieses Abschnittes ist es, eine flexible Methode zur Steuerung der Übertragung zu entwickeln, die in einem interaktiven Bildübertragungssystem RoIs und LoDs realisieren kann und die auch für die nicht-interaktive Übertragung nutzbar ist. Eine solche Steuerung muss folgende Funktionalität bereitstellen:

- Definition neuer RoIs vor und während der Übertragung,
- Priorisierung von RoIs,
- Steuerung von Kodierung bzw. Dekodierung der zu den RoIs beitragenden Wavelet-Koeffizienten entsprechend ihres Ziel-LoDs und ihrer Priorität,
- Modifikation des Ziel-LoDs und der Priorität existierender RoIs vor und während der Übertragung,
- Vorzeitiges Beenden der Übertragung „unerwünschter“ RoIs.

4.5.2.1 Priorisierung von RoIs

Durch die Spezifikation des Ziel-LoDs kann für jede RoI bestimmt werden, wie detailliert sie nach Abschluss der Übertragung vorliegen soll. Dies wirkt sich jedoch nicht auf den Übertragungsverlauf aus. Da nicht alle RoIs dieselbe Wichtigkeit besitzen, erscheint es wünschenswert, dass wichtigere Regionen ihren Ziel-LoD schneller im Verlauf der Übertragung erreichen als unwichtige. Um das zu realisieren, ist es sinnvoll, *Prioritäten* für die RoIs zu vergeben, um wichtigen RoIs einen „Vorsprung“ in der Übertragung einräumen. Der Prioritätswert kann verschiedene Bedeutungen besitzen:

Bitebenen-Offset. Hierbei wird als Priorität eine ganze Zahl angegeben, die die Verschiebung der Wertigkeit der Bitebenen bei der sukzessiven Approximation angibt. So können beispielsweise bei einem Offset von 2 die ersten beiden Bitebenen der Koeffizienten einer wichtigen RoI übertragen werden, bevor die Übertragung der Koeffizienten einer unwichtigen RoI beginnt. Im weiteren Übertragungsverlauf wird für die wichtigere RoI bis zum Erreichen ihres Ziel-LoDs immer die $n+2$te Bitebene traversiert, wenn für die unwichtigere RoI die $n$te Bitebene durchlaufen wird.

Sub-Bitebenen-Offset Diese Priorität funktioniert analog zum Bitebenen-Offset, jedoch können „Teile“ von Bitebenen als Offset angegeben werden. Die Anzahl der möglichen Teilbitebenen ergibt sich aus der Anzahl von Kodierungsdurchläufen. Beim Zerotree-Verfahren sind zwei Teilbitebenen möglich (dominanter und nachgeordneter Pass), bei EBCOT [Tau99a, Tau99b] vier.

Bitraten-Verhältnis. Das Bitraten-Verhältnis erlaubt potenziell eine feinere Steuerung als der Bitebenen-Offset. Hierbei wird für jede RoI zusätzlich die für die Kodierung benötigte Bitrate gespeichert und dafür gesorgt, dass die Bitraten einzelner RoIs im angegebenen Verhältnis stehen. Hier einen Offset zu verwenden scheint unrealistisch, da die Bitrate in späteren Bitebenen exponentiell wächst und somit bei Nutzung eines Offsets die Qualitätsunterschiede zwischen „wichtigen“ und „unwichtigen“ RoIs in späten Übertragungsstufen schnell geringer werden.

Das Bitraten-Verhältnis kann bei überlappenden RoIs problematisch sein, da durch die differenzielle Übertragung möglicherweise nicht alle Koeffizienten im Footprint der RoI zur Übertragung beitragen, so dass das Maß „kodierte Bitrate pro RoI-Pixel“ verfälscht wird. Für praktische Anwendungen ist der Bitebenen-Offset ausreichend. Abhängig vom Bild-

inhalt, dem verwendeten Wavelet-Filter und der Anzahl der Zerlegungsstufen belegen die Wavelet-Koeffizienten 10 bis 15 Bits, was mindestens 10 Abstufungen in der Priorität ermöglicht. Damit sind die meisten Nutzer bereits überfordert, weshalb für die Priorität eine drei- bis fünfstufige Skala von „unwichtig“ bis „wichtig“ als ausreichend erscheint. Daher soll im Folgenden nur noch der Bitebenen-Offset betrachtet werden.

4.5.2.2 RoI-Scheduling

Zur Steuerung der Kodierungs- und Dekodierungstraversierung wurde das Konzept der *RoI-Scheduler* entwickelt. Diese Namensgebung erfolgte in Analogie zum Scheduler eines Betriebssystems, da die Funktionalität ähnlich ist. Ein RoI-Scheduler arbeitet eine Schleife ab, wobei in jedem Durchlauf eine Menge von RoIs *aktiviert* und für die Koeffizienten in deren Footprints eine Anzahl von Kodierungs- bzw. Dekodierungsoperationen ausgeführt wird. RoIs können analog zu Prozessen eine Priorität besitzen (siehe oben) und einen der vier Zustände `WAITING`, `STOPPED`, `ACTIVE` und `FINISHED` annehmen. Dazu müssen die RoI-Scheduler auf der Enkoder- und der Dekoderseite synchron arbeiten und die Folge der globalen LoDs kennen.

Da die partitionierenden Teilmengen innerhalb des globalen LoDs implizit durch Scantraversierung repräsentiert werden, ist pro RoI neben dem Ziel-LoD, dem Zustands-LoD und dem Prioritätswert zusätzlich die Speicherung des Delta-LoDs erforderlich. Dieser ist eine sortierte Liste von Tupeln, die die als Nächstes zu besuchenden Ortsvektoren im LoD-Raum beinhaltet.

Die Selektion der als Nächstes zu aktivierenden RoIs durch den Scheduler erfolgt an Hand des LoD-Deltas und der Priorität der einzelnen RoIs. Das erste Element des LoD-Deltas $d[0] := [c,b,q,x,y]^T$ beschreibt jeweils den Ortsvektor[11] im LoD-Raum, der für eine RoI als Nächstes zu besuchen ist. Das bedeutet, dass die im nächsten Schritt zu (de)kodierende Kombination aus Farbkanal c, Bitebene[12] b, Pass[13] q, X-Auflösung x sowie Y-Auflösung y durch dieses Element $d[0]$ bestimmt ist. Zur Steuerung der Übertragung wird eine Kombination von Zerlegungsstufe l, Band d und Unterband t berechnet, die der Auflösungskombination (x,y) entspricht. Von der Bitebene wird die Priorität p ($p \in \{\ldots,-1,0,1,\ldots\}$) subtrahiert. Zwei RoIs r_1 und r_2 lassen sich anhand des ersten Elements ihres LoD-Deltas unter Einbeziehung der Priorität wie folgt ordnen:

$$\begin{aligned}
\delta_1 &:= ([c_1,b_1,q_1,x_1,y_1]^T,p_1)\\
\delta_2 &:= ([c_2,b_2,q_2,x_2,y_2]^T,p_2)\\
\delta_1 < \delta_2 &:\Leftrightarrow c_1 < c_2\\
&\quad \vee (c_1 = c_2 \wedge b_1 - p_1 < b_2 - p_2\\
&\quad \vee (b_1 - p_1 = b_2 - p_2 \wedge q_1 < q_2\\
&\quad \vee (q_1 = q_2 \wedge x_1 < x_2\\
&\quad \vee (x_1 = x_2 \wedge y_1 < y_2))))
\end{aligned} \tag{4.14}$$

Durch diese Ordnung kann immer eine Teilmenge von RoIs mit jeweils minimalen δ-Werten gefunden und für die Übertragung aktiviert werden. Ein auf die Aktivierung folgender unteilbarer *Übertragungsschritt* traversiert mithilfe des Scanline-Algorithmus aus

[11] Hierbei werden die Koordinaten eines LoD-Ortsvektors anstelle der abstrakten l^j in Gleichung (3.1) mit Bezeichnungen versehen, die die Achsensemantik besser widerspiegeln.

[12] Für die Bitebene gilt $b \in \{1,2,\ldots\}$, wobei $b = 1$ dem *most significant bit* entspricht.

[13] Für die Realisierung mit dem Zerotree-Algorithmus ist es notwendig, zusätzlich die Dimension *Pass* einzuführen, die jeweils zwischen dominantem und nachgeordnetem Pass auswählt. Konzeptionell kann diese Dimension auch als Sub-Bitebene betrachtet und mit der Dimension *Bitebene* gefaltet werden.

4.5.1.3 die Bitebene b der Koeffizienten des Farbkanals c, die zur Mehrfachauflösungsrepräsentation der Footprints der aktivierten RoIs im Unterband t des Bandes d der Zerlegungsstufe l beitragen. Durch die Übertragung ändern sich die Zustands-LoDs und LoD-Deltas und damit die δ-Werte der aktiven RoIs. Im nächsten Schritt können so die δ-Werte anderer RoIs minimal sein, wodurch eine abwechselnde Ausführung von Übertragungsschritten für alle RoIs sichergestellt wird.

Der Begriff „Übertragungsschritt“ wird im folgenden Abschnitt noch eine wichtige Rolle spielen.

4.5.2.3 Steuerung globaler LoDs

Nutzer von Bildübertragungssystemen können eine von zwei Rollen annehmen. Während der *Bildautor* das Bild bereitstellt, ist der *Bildbetrachter* derjenige, der die Bildübertragung initiiert, um aus dem Bild Informationen zu entnehmen. Das Ziel bedarfsgesteuerter Bildübertragung liegt in der Unterstützung beider Nutzerrollen bei der Spezifizierung des antizipierten bzw. tatsächlichen Bedarfs.

Bildautoren können Standard-RoIs definieren und Bildteile, die sie als wichtig erachten, priorisieren. Die Rolle des Bildautors kann auch ganz wegfallen; in diesem Fall wird dem Betrachter das Bild zunächst in niedriger Detailliertheit präsentiert und ihm die Auswahl interessierender Bereiche überlassen. Diese Funktionalität unterstützt sowohl nichtinteraktive als auch interaktive Bildübertragungssysteme.

Der Betrachter eines übertragenen Bildes hat unter Umständen einen ganz anderen Bedarf als der vom Autor antizipierte. Daher sollte er in einem interaktiven Bildübertragungssystem die Kontrolle über den laufenden Übertragungsprozess übernehmen können, um die Standardvorgaben zu ignorieren und beispielsweise eigene RoIs zu definieren.

Kommando	**Erklärung**
DEFRECT(foot, lod, prio)	Erzeugen einer neuen rechteckigen RoI mit rechteckigem Footprint *foot*, Ziel-LoD *lod* und Priorität *prio*
DEFELIPS(foot, lod, prio)	Erzeugen einer neuen elliptischen RoI mit rechteckiger Footprint-Boundingbox *foot*, Ziel-LoD *lod* und Priorität *prio*
DEFPOLY(n, pts, lod, prio)	Erzeugen einer neuen polygonalen RoI mit Anzahl der Eckpunkte *n*, Eckpunktliste *pts*, Ziel-LoD *lod* und Priorität *prio*
STOPROI(RoI_id)	Stoppen der Übertragung einer RoI
CONTROI(RoI_id)	Fortsetzen der Übertragung einer RoI
PRECPRIOINCDEC(RoI_id, offset)	Addieren eines positiven oder negativen *offset* zur Priorität einer RoI
PRECINC(RoI_id, increment)	Erhöhen der Komponente *Genauigkeit* des Ziel-LoDs einer RoI
XRESINC(RoI_id, increment)	Erhöhen der Komponente *X-Auflösung* des Ziel-LoDs einer RoI
YRESINC(RoI_id, increment)	Erhöhen der Komponente *Y-Auflösung* des Ziel-LoDs einer RoI
COLRINC(RoI_id)	Setzen der Komponente *Farbe* des Ziel-LoDs einer RoI auf den Wert *Farbe*
PRECPRIOMOD(RoI_id, prio)	Setzen der Priorität einer RoI auf den Wert *prio*
LLODMOD(RoI_id, lod)	Setzen des Ziel-LoD einer RoI auf den Wert *lod*
EOI()	Bildende
ID_ALL_ROIS	Wird der reservierte Wert ID_ALL_ROIS für RoI_id angegeben, so bezieht sich das Kommando auf alle bisher definierten RoIs.

Tabelle 4.3: Beispiele für Steuerkommandos.

Die Übertragung wird sowohl auf der Server- als auch auf der Clientseite von synchron laufenden RoI-Schedulern gesteuert. Ein interaktives Eingreifen des Betrachters in den laufenden Übertragungsprozess erfordert einen Satz von *Steuerkommandos*, die eine Modifikation der globalen LoDs der RoI-Scheduler zur Übertragungszeit gestatten. Tabelle 4.3 zeigt eine Auswahl der Kommandos. Durch Angabe eines speziellen Parameterwertes kann ein Kommando – statt auf eine bestimmte RoI – auch auf alle definierten RoIs angewendet werden.

Kommandos werden von einer Clientkomponente an den serverseitigen Scheduler gesendet, der daraufhin seinen globalen LoD ändert und die Kodierung anpasst. Da eine Synchronisation des clientseitigen Schedulers mit dem neuen globalen LoD erforderlich ist, bevor die damit generierten Teile des Bitstromes dekodiert werden, müssen die Kommandos zur richtigen Zeit auch vom clientseitigen Scheduler ausgeführt werden. Zur Übertragung der Steuerinformation gibt es zwei Alternativen:

- Übertragung als Seiteninformation über einen separaten Steuerkanal oder
- Einbettung der Kommandos in den kodierten Datenstrom.

Die zweite Alternative erlaubt eine einfachere Synchronisation und bietet den Vorteil, dass der Datenstrom wiederverwendet werden kann, selbst wenn keine Verbindung zum Enkoder besteht (nicht-interaktive Bildübertragung). Zur Einbettung ist eine Modifikation der Entropiekodierung notwendig, die im Folgenden beschrieben werden soll.

Eine Änderung des globalen LoDs kann nur an definierten Punkten während der Traversierung erfolgen, nämlich jeweils zwischen zwei Übertragungsschritten (siehe 4.5.2.2). Wegen möglicher Überlappungen von RoIs und wegen der Vater-Kind-Beziehungen im Zerotree können dabei leere Schritte auftreten[14]. Um ein Kommando jedoch eindeutig mit einem bestimmten Schritt zu assoziieren, muss es unmittelbar vor den kodierten Symbolen dieses Schrittes in den Datenstrom eingebettet werden. Leere Übertragungsschritte führen hier zu Mehrdeutigkeiten und müssen daher übersprungen werden. Um das sicherzustellen, werden die Kommandos in eine Warteschlange eingereiht. Unmittelbar bevor das erste Symbol eines nicht leeren Übertragungsschrittes ausgegeben wird, wertet der Entropiekodierer diese Warteschlange aus. Ist sie nicht leer, so wird ein spezielles Markersymbol[15] ausgegeben, der Entropiekodierer geschlossen, das Kommando in den Datenstrom geschrieben und in der Warteschlange als ausführbar markiert. Dann wird der Übertragungsschritt wie geplant fortgesetzt, und nach dessen Abschluss werden die markierten Kommandos ausgeführt und aus der Warteschlange entfernt. Die Ausführung eines Kommandos erfolgt also immer unmittelbar im Anschluss an den nicht leeren Übertragungsschritt, an dessen Anfang es übertragen wurde. Auf der Dekoderseite wird beim Dekodieren des speziellen Markersymbols das Kommando aus dem Datenstrom extrahiert und wiederum in eine Warteschlange gestellt. Nach Abschluss eines Übertragungsschrittes werden dann wie auf der Enkoderseite die Kommandos in der Warteschlange ausgeführt und gelöscht.

Der folgende Pseudokode gibt die vom Enkoder zusätzlich auszuführenden Schritte noch einmal wieder. Während eines Übertragungsschrittes führt der RoI-Scheduler die in 4.5.1.3 vorgestellte Spannentraversierung durch. Wenn das erste Symbol s_0 für den aktuellen Übertragungsschritt ausgegeben werden soll, durchläuft der arithmetische Kodierer die folgenden zusätzlichen Schritte:

```
IF Kommando-Queue nicht leer
THEN
  eof-Symbol kodieren
  Ausgabepuffer des arithmetischen Kodierers leeren
  start-Marker in binären Datenstrom schreiben
```

[14] Leere Übertragungsschritte sind Schritte, während denen kein Symbol ausgegeben wird.

[15] Bei Verwendung der arithmetischen Kodierung nach [WNC87] kann dafür das EOF-Symbol genutzt werden.

```
  FOREACH cmd in Kommando-Queue
  BEGIN
    cmd in binären Datenstrom schreiben
    cmd als ausführbar markieren
  END
  nul-Kommando in binären Datenstrom schreiben
  Arithmetischen Enkoder neu starten
ENDIF
Symbol s_0 kodieren
```

Das Dekodieren erfolgt analog. Wenn während eines Übertragungsschrittes das erste Symbol s_0 eingelesen wird, so führt der arithmetische Dekoder die folgenden Schritte aus:

```
Symbol s_0 dekodieren
IF s_0 = eof
THEN
  start-Marker im binären Datenstrom suchen
  Leseposition auf das Byte nach dem start-Marker setzen
  REPEAT
    Kommando cmd aus dem binären Datenstrom einlesen
    cmd in die Warteschlange einreihen und als ausführbar markieren
  UNTIL cmd = nul-Kommando
  Dekoder neu starten
  Symbol s_0 dekodieren
ENDIF
```

4.5.2.4 Zustandsübergänge von RoIs

Durch Steuerkommandos sowie durch die Übertragungssteuerung mittels des RoI-Schedulers ergeben sich für jede RoI Zustandsänderungen, die den Übertragungsverlauf beschreiben. Abbildung 4.14 zeigt das Zustandsübergangsdiagramm einer RoI.

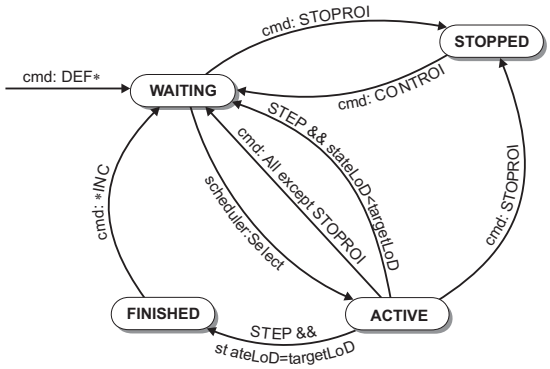


Abbildung 4.14: Mögliche Zustände und Zustandsübergänge einer RoI während der Übertragung.

Eine mit einem DEF$\star$-Kommando[16] neu erzeugte RoI befindet sich zunächst im Zustand WAITING. Der RoI-Scheduler kann in seinem prioritätenbasierten Selektionsschritt (vgl. 4.5.2.2) die RoI für die Übertragung aktivieren, wodurch sie in den Zustand ACTIVE ge-

[16] Dies umfasst die Kommandos DEFRECT, DEFPOLY und DEFELIPS.

langt. Aus diesem Zustand fällt sie nach einem Übertragungsschritt `STEP` entweder zur erneuten Selektion in den Wartezustand zurück, wenn der Ziel-LoD mächtiger als der Zustands-LoD ist (das heißt, wenn noch Daten zu übertragen sind), oder sie gelangt bei abgeschlossener Übertragung in den Zustand `FINISHED`. Aus diesem Zustand kann sie wieder in den Wartezustand gelangen, wenn ein `*INC`-Kommando[17] ausgeführt wird, das eine Komponente des Ziel-LoDs erhöht. Durch ein `STOPROI`-Kommando gehen RoIs in den Zustand `STOPPED` über, aus dem sie mit dem Kommando `CONTROI` wieder in den Wartezustand überführt werden können.

Nach Anwendung eines Kommando außer `STOPROI` auf eine beliebige RoI fallen alle aktiven RoI in den Wartezustand zurück, da sich durch das Kommando relevante Scheduling-Parameter geändert haben können. Der Scheduler führt dann erneut einen Selektionsschritt aus, um die RoIs zu aktivieren, für die nach Gleichung (4.14) der nächste Übertragungsschritt ausgeführt werden müsste.

4.6 Partielle Dekodierung

Bisher wurde die Unterstützung von RoIs und LoDs zur Einsparung von Übertragungsbandbreite beschrieben. In manchen Applikationen (wie dem RECHTECKIGEN FISHEYE-VIEW, 5.4), können sie auch zur Einsparung von Bildschirmplatz bei der Anzeige und Rechenzeit bei der inversen Wavelet-Transformation eingesetzt werden. Dazu werden nur Teilbereiche des Bildes rekonstruiert, und die Rekonstruktion erfolgt möglicherweise in reduzierter Auflösung.

Bei dieser *partiellen Dekodierung* einer RoI wird zunächst deren Bounding Box berechnet und daraus in mehreren Schritten eine Mehrfachauflösungsrepräsentation analog zu Abbildung 4.10 auf Seite 62 erzeugt. Für das dyadische Dekompositionsschema beispielsweise werden je Schritt drei Rechtecke in den Bändern LH, HL und HH generiert, indem die Eckpunktkoordinaten ganzzahlig durch 2 dividiert und um den Offset des entsprechenden Bandes verschoben werden. Ein letzter Schritt erzeugt die Repräsentation des Rechtecks im LL-Band. Die Generierung für die XY-unabhängige Zerlegung erfolgt analog.

Wenn zwei benachbarte RoIs so definiert sind, dass ihre Bounding Boxen eine gemeinsame Grenzlinie besitzen, sollte an dieser Stelle im rekonstruierten Bild kein Sprung erkennbar sein. Für einen solchen glatten Übergang muss durch eine Randbehandlung sichergestellt werden, dass alle Koeffizienten, die zu den Pixeln im interessierenden Rechteck beitragen, von den Rekonstruktionsfiltern erfasst werden. Dazu werden alle Kanten jedes Rechtecks um k Koeffizienten nach außen verschoben (k errechnet sich aus der Länge n des längsten Filters als $k = n/2 + 1$; beim 9/7-Wavelet gilt also $k = 5$). Dabei dürfen die Kanten die Teilbandgrenze nicht überschreiten.

Soll eine Region nicht in voller Auflösung rekonstruiert werden, so ist zusätzlich die Skalierung der Filter mit $\sqrt{2}$ rückgängig zu machen (vgl. 2.4.3.1). Das erfolgt durch Division aller rekonstruierten Pixelwerte durch $\sqrt{2}^k$ im Anschluss an die inverse Wavelet-Transformation, wobei k für die Anzahl der nicht ausgeführten Rekonstruktionsschritte steht. Als ein Rekonstruktionsschritt wird dabei die Anwendung der Synthesefilter auf die Zeilen *oder* die Spalten des Koeffizientenfeldes bezeichnet. Eine Rekonstruktion in halber X- und viertel Y-Auflösung liefert also z.B. $k = 3$, da in Zeilenrichtung ein Schritt und in Spaltenrichtung zwei Schritte nicht ausgeführt werden.

[17] Dies umfasst die Kommandos `PRECINC`, `XRESINC`, `YRESINC` und `COLRINC`.

4.7 LIBROILOD: Ein System zur bedarfsgesteuerten Bildübertragung

Zur Umsetzung und Evaluierung der skizzierten Konzepte wurde die Programmbibliothek LIBROILOD entwickelt, die die Grundlage sowohl für Bildkompressionslösungen als auch für Client-Server-Bildübertragungsapplikationen bilden kann. Wichtige Entwurfsziele waren eine hohe Rechengeschwindigkeit, Portabilität sowie eine einfach zu benutzende Programmierschnittstelle.

Funktionalität. Die Bibliothek wurde in C++ programmiert und kann unter 32-Bit-Windows, LINUX, Solaris sowie IRIX eingesetzt werden. Sie bietet die folgende Funktionalität:

- Realisierung der Basisfunktionalität von MOVIWAVE als Wavelet-Codec-Klasse (Performance siehe Abbildung 4.2),
- Realisierung der RoI/LoD-Funktionalität von MOVIROILOD als RoI/LoD-Codec-Klasse,
- Realisierung einer Kommandoschnittstelle zur Steuerung des RoI/LoD-Codecs mit Kommandos aus einem Programm heraus,
- Realisierung eines Importformates (Anhang D) zur Definition von RoIs und Steuerkommandos in einem Autorenwerkzeug oder mithilfe eines Texteditors,
- Realisierung umfangreicher Debugging-Funktionalität.

Architektur. Abbildung 4.15 zeigt die Architektur des RoI/LoD-Codecs, dessen Kernkomponente ein RoI-Scheduler ist. Dieser Scheduler verwaltet eine RoI-Liste und die Zerotree-Map. Er besteht aus einem scanlinebasierten Zerotree-Codec, einer Komponente zur arithmetischen Kodierung/Dekodierung, einem Kommandoprozessor und einer Steuerkomponente. Mithilfe der Steuerkomponente sowie über das Kommandointerface kann eine Applikation Einfluss auf die Kodierung nehmen.

Zusätzlich zum Scheduler enthält der RoI/LoD-Codec weitere Komponenten. Import/Export-Routinen lesen und schreiben Bilddaten sowie das in Anhang D beschriebene Importfor-

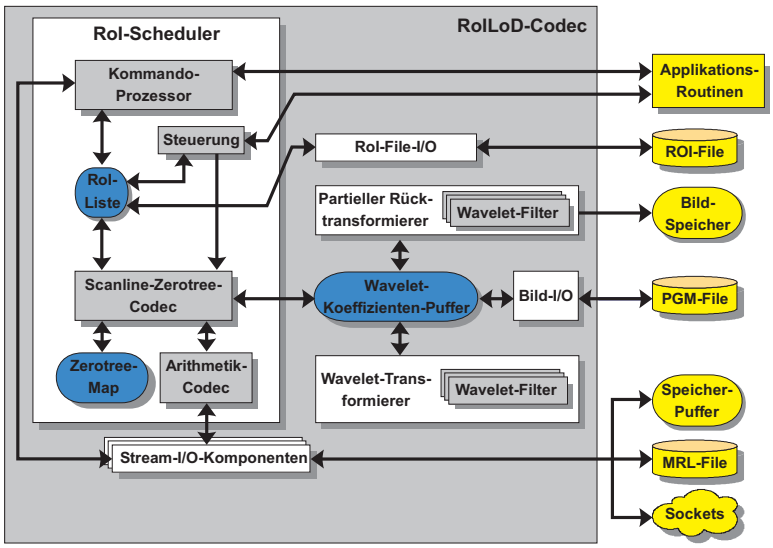


Abbildung 4.15: Architektur von LIBROILOD.

mat, mit dem die RoIs für ein bestimmtes Bild vor Beginn der Übertragung spezifiziert werden können. Der Wavelet-Koeffizientenpuffer speichert die Wavelet-Koeffizienten, die von einem Wavelet-Transformierer aus den Bilddaten berechnet werden. Dieser Wavelet-Transformierer realisiert das in 4.4.4 beschriebene Dekompositionsschema und ermöglicht es, austauschbare Wavelet-Filter zu verwenden. Mithilfe des partiellen Rücktransformierers kann eine Bildbetrachtungsapplikation Teilbereiche des Bildes in wahlweise verringerter Auflösung anzeigen (vgl. 4.6). Die Stream-I/O-Komponenten sind ebenfalls austauschbare Objekte, so dass eine Applikation den Codec an das verwendete Übertragungsmedium anpassen kann.

Debugging-Unterstützung. Bei der Entwicklung einer Bildübertragungsbibliothek müssen Fehler, die in *real-world*-Anwendungen auftreten, in einer definierten Testumgebung reproduzierbar sein. Da Bildkompressionsanwendungen schwierig zu debuggen sind, war dafür die Entwicklung einer Testmethodik erforderlich.

Problematisch ist, dass von einem Artefakt im dekodierten Bild nicht visuell auf die Fehlerursache geschlossen werden kann. Durch die Komplexität des Datenstromes kann bereits ein einzelnes falsches Bit die verschiedensten Fehlerauswirkungen haben. Existierende Debugger geben aber keine Unterstützung bei der Fehlersuche auf Bit-Ebene. Daher wurde eine Testumgebung selbst entwickelt, die es ermöglicht, die einzelnen Kodierungs- bzw. Dekodierungsschritte textlich bzw. visuell darzustellen.

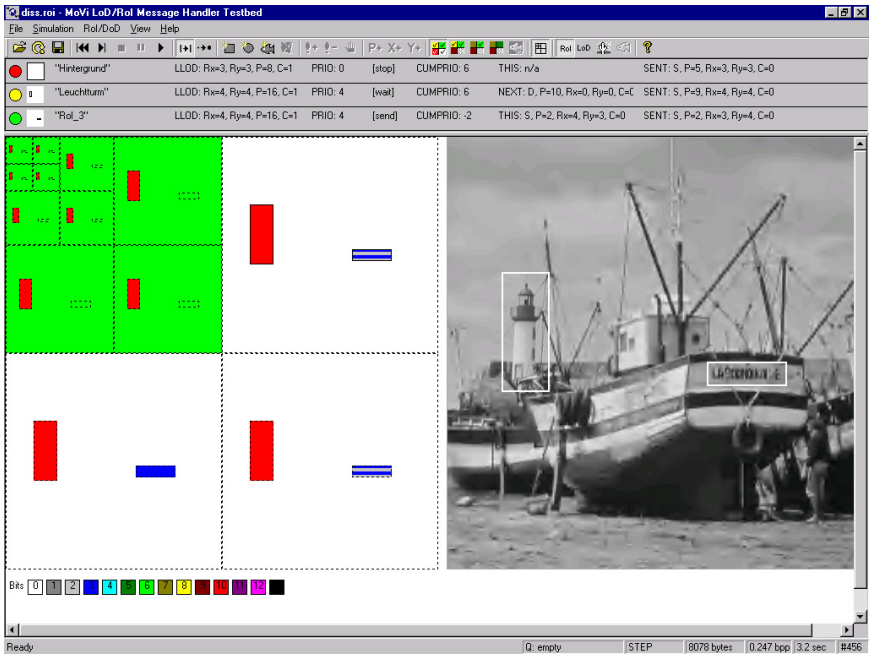
Erfahrungen in der Entwicklung zeigten außerdem, dass viele Fehler auf mangelnde Synchronisation zwischen Enkoder und Dekoder zurückzuführen waren. Solche Fehler sind vor allem dann schwer einzugrenzen, wenn die Bibliothek in „echten“ Client-Server-Anwendungen zur Bildübertragung zum Einsatz kommt. Die Reproduzierbarkeit von Fehlern in solchen Anwendungen wird dadurch realisiert, dass die Serverkomponente die geladene Bilddatei, die definierten RoIs sowie alle ausgeführten Steuerkommandos in einer RoI-Definitionsdatei protokolliert, die das in Anhang D beschriebene Format hat.

Es wurde eine Testumgebung entwickelt, die eine solche Protokolldatei laden und ausführen kann (siehe Abbildung 4.16). Diese Umgebung bindet sowohl Enkoder als auch Dekoder aus der Bildübertragungsbibliothek ein und erlaubt die Simulation der Übertragung mit wählbarer Bandbreite bis hin zu einzelnen Übertragungsschritten. Über Kommandobuttons können zu jeder Zeit während der Übertragung Steuerkommandos an den Enkoder gesendet werden. Zur Anzeige des Ergebnisses jedes Simulationsschrittes steht umfangreiche Visualisierungsfunktionalität zur Verfügung:

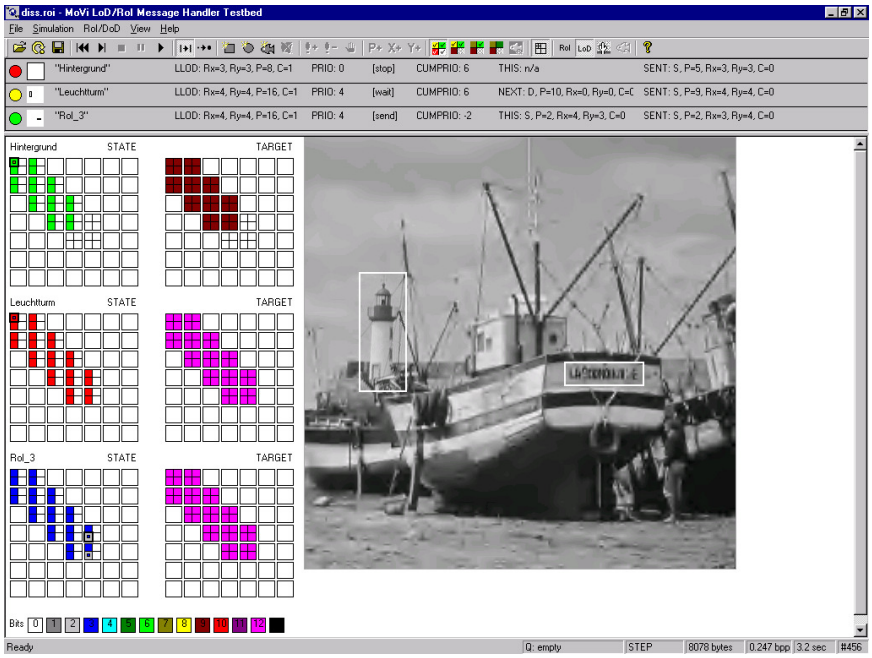
- Anzeige des dekodierten Bildes,
- Visualisierung der bereits kodierten Bits im Wavelet-Koeffizientenfeld,
- Visualisierung der lokalen LoDs der RoIs,
- Darstellung der Zustände der RoIs.

Für die Fehlersuche im Bitstrom wurde zusätzliche Unterstützungsfunktionalität implementiert, die die Struktur des Symbolstromes lesbar in getrennten Log-Dateien für Enkoder und Dekoder protokolliert. Durch Vergleich der Dateien können Synchronisationsprobleme zwischen Enkoder und Dekoder aufgedeckt werden.

Einschränkungen der Realisierung. Das umgesetzte System unterstützt bisher nur Graustufenbilder. RoIs müssen rechteckige Footprints besitzen. Im Rahmen weiter führender Arbeiten ist die Unterstützung von Echtfarbbildern ebenso geplant wie die Realisierung polygonaler Footprints.



(a) Zustandsanzeige der RoIs (oben), Übertragungssimulation (rechts), Visualisierung der Mehrfachauflösungsrepräsentation der RoIs und Farbkodierung der letzten übertragenen Bitebene (links)



(b) Zustandsanzeige der RoIs (oben), Übertragungssimulation (rechts), Visualisierung von Zustands- und Ziel-LoD der RoIs (links)

Abbildung 4.16: Testumgebung für die RoI/LoD-Bibliothek.

Kapitel 5

Anwendungen

5.1 Client-Server-Architektur der Anwendungen

In diesem Kapitel soll auf drei Anwendungen des in Kapitel 4 vorgestellten Übertragungsmechanismus eingegangen werden. Dazu wird zunächst die Basisstruktur beschrieben, die allen Applikationen gemeinsam ist.

Grundfunktion aller Anwendungen ist es, Bilder, die auf einem Server gespeichert sind, zu einem Clientrechner zu übertragen und sie bereits während des Transfers online anzuzeigen. Dabei wird das RoI/LoD-Schema verwendet, um eine flexible Steuerung der Übertragung zu realisieren. Die Übertragungsfunktionalität erfordert die Trennung der Applikationen in eine Client- und eine Serverkomponente. Während die Clientkomponente die konkrete Applikationsfunktionalität realisiert, sollte die Serverkomponente möglichst generisch sein. Das ermöglicht es, mehrere Applikationen mit derselben Serversoftware zu realisieren. Sowohl Client als auch Server enthalten die in 4.7 beschriebene Bibliothek.

Zur Kommunikation zwischen Client und Server sind zwei Kanäle erforderlich: ein *Downloadkanal*, der Bilddaten und eingebettete Steuerkommandos vom Server zum Client transportiert, und ein *Steuerkanal*, über den der Client Anforderungen an den Server senden kann, um eine neue Übertragung zu initiieren oder eine laufende zu reparametrisieren.

5.1.1 Die Clientkomponenten

Die Clientsoftware stellt die Anwendungsfunktionalität zur Verfügung und ist für die grafische Ein- und Ausgabe zuständig. Benutzereingaben zur Steuerung der Bildübertragung werden in Steuerkommandos umgesetzt (vgl. 4.3) und über den Steuerkanal an den Server übermittelt. Der vom Server empfangene Datenstrom wird von der Applikation übernommen, der integrierten RoI/LoD-Bibliothek als Eingabe übergeben und von dieser dekodiert. Flexible Funktionen zur Ausführung der inversen Wavelet-Transformation (vgl. 4.6) ermöglichen es, die bereits dekodierten Bilddaten in variablen Auflösungen zu rekonstruieren, so dass Darstellung und Bildübertragung asynchron arbeiten können.

Die Clientkomponenten wurden als Netscape-Plugins realisiert, um eine Einbettung in die existierende Internet-Infrastruktur zu erlauben.

5.1.2 Die RoI/LoD-Serverkomponente

Der Bildserver stellt die Funktionalität der RoI/LoD-Bibliothek so zur Verfügung, dass Applikationen über das Netz darauf zugreifen können. Kernkomponente ist der RoI/LoD-Codec. Der Server muss folgende Funktionalität unterstützen:

- Auf- und Abbau der Verbindung zur Übertragung eines angeforderten Bildes,
- Einlesen des Bildes und eventuell vordefinierter RoIs für dieses Bild,
- Kodierung des angeforderten Bildes im RoI/LoD-Codec, Ausgabe des kodierten Datenstromes im Download-Kanal,
- Einlesen von Steuerkommandos aus dem Steuerkanal während laufender Kodierung; Übergabe dieser Parameter an den RoI/LoD-Codec zur Beeinflussung der Übertragung.

Da die Clientkomponenten als Plugins für den Webbrowser Netscape umgesetzt wurden, erfolgte auch die Realisierung des Servers mit Webtechnologie auf der Basis von `CGI`[1]. Fordert ein Client ein neues Bild an, so wird ein `CGI`-Programm gestartet, das einen neuen Downloadkanal öffnet und einen RoI/LoD-Codec kreiert. Dieser Codec liest ein Bild als PGM-Datei [P$^+$] und zusätzlich ein eventuell existierendes RoI-File mit vordefinierten RoIs für dieses Bild ein (siehe 4.3.2 sowie Anhang D) und beginnt in einer Hauptschleife, das Bild mit diesen Einstellungen zu kodieren und den resultierenden Datenstrom in den Downloadkanal auszugeben. Parallel dazu lauscht das Programm am Steuerkanal und gibt dort ankommende Kommandos an den Scheduler weiter. Dieser führt sie aus und ändert dadurch seinen globalen LoD. Damit die Änderungen an den Client weitergegeben werden, verschachtelt er die Kommandos außerdem mit dem ausgegebenen Datenstrom.

5.1.3 Client-Server-Kommunikation

Für die Realisierung des Steuerkanals gibt es verschiedene Möglichkeiten, die sich hinsichtlich der Latenzzeit und der Anforderungen an die Serverressourcen unterscheiden. Als Latenzzeit wird hier die Zeit bezeichnet, die ein Steuerkommando von der Aussendung durch den Client im Steuerkanal bis zu seiner Ankunft beim Client im Downloadkanal benötigt. Folgende Alternativen bieten sich an:

HTTP+CGI. Hierbei geschieht das Senden von Steuerinformationen durch Aufbau einer neuen HTTP-Verbindung zum Server, der eine neue Instanz des `CGI`-Programms mit den Kommandos als Parameter aufruft. Diese Instanz schreibt die Kommandos in eine Wartedatei und beendet sich wieder. Die parallel dazu laufende kodierende `CGI`-Instanz wertet nach jedem Übertragungsschritt den Inhalt der Wartedatei aus.

Diese Alternative ist relativ langsam, da der `HTTP`-Verbindungsaufbau zum Server und insbesondere der Programmstart zeitintensive Operationen darstellen. Weiterhin müssen die bei Ankunft der Befehle im Ausgabepuffer des WWW-Servers vorhandenen Daten noch an den Client gesendet werden, was die Latenzzeit bei langsamen Verbindungen zusätzlich erhöht.

HTTP+FastCGI. FastCGI [FCG] ist eine Erweiterung von `CGI`, die die Zeit für das Starten und Initialisieren eines `CGI`-Programms dadurch einspart, dass mindestens eine Reserve-Instanz des Serverprogramms ständig aktiv ist. Bei Anforderung der Aus-

[1] `CGI`: Common Gateway Interface [CGI]. Ein `CGI`-Programm ist ein Programm auf einem WWW-Server, das als Ausgabe einen `HTTP`-Datenstrom erzeugt. Der Aufruf eines solchen Programms erfolgt durch Angabe einer URL, die den Namen des Programms enthält. Im Gegensatz zur Anforderung einer „normalen“ URL, die der WWW-Server durch Übertragen der adressierten Datei beantwortet, wird eine `CGI`-Anforderung durch Ausführen der adressierten Datei und Übertragung von dessen Standardausgabe bedient.

gabe eines CGI-Programms wird, statt ein neues Programm zu starten, die Anforderung in eine Warteschlange gestellt und einer der aktiven Instanzen übergeben. Dadurch sinkt die Latenzzeit; pro Kontaktierung des Servers ist lediglich ein HTTP-Verbindungsaufbau erforderlich.

Sockets. Durch eine Realisierung auf der Basis von Sockets kann auch die Zeit für den Verbindungsaufbau eingespart werden. Sockets gestatten im Gegensatz zu HTTP das Offenhalten einer bidirektionalen Verbindung über die gesamte Laufzeit von Client- und Serverprogramm; der Verbindungsaufbau erfolgt einmalig beim Programmstart. Somit können Client und Server mit minimaler Zeitverzögerung miteinander kommunizieren. Dieser Vorteil wird durch die ständige Belegung von Kommunikationsressourcen erkauft. Bei Verwendung von Firewalls erfordert die Socket-Methode zusätzlichen administrativen Aufwand.

5.1.4 Latenzzeit

Für den Nutzer stellt die Latenzzeit die Wartezeit vom Auslösen eines Kommandos bis zu dessen sichtbarer Ausführung dar. Sie muss daher so kurz wie möglich sein.

5.1.4.1 Einflussfaktoren

Die Latenzzeit eines Kommandos wird von drei Faktoren beeinflusst:

1. Latenzzeit der Netzverbindung und Effizienz der Behandlung von Übertragungsfehlern im Netzwerkprotokoll,
2. Latenzzeit der Parameterübergabe auf dem Server,
3. Zeit, die der Enkoder für das Abschließen des Übertragungsschrittes benötigt, in dessen Verlauf das Kommando eintrifft.

Die verschiedenen o.a. Kommunikationsalternativen unterscheiden sich anwendungsabhängig hinsichtlich der ersten beiden Faktoren. Der dritte Faktor wird maßgeblich von der Anzahl der in einem Übertragungsschritt zu traversierenden Koeffizienten bestimmt. Bei der Traversierung von Teilbändern mit vielen Koeffizienten könnte somit die Latenzzeit höher sein als in „kleinen“ Teilbändern. Dies ist durch die Unteilbarkeit von Übertragungsschritten (vgl. 4.5.1.2 und 4.5.2.3) bedingt.

Zur Messung der Latenzzeit kam als Server ein Rechner mit PENTIUM II, 450 MHz und 256 MB RAM sowie als Client ein Notebook mit Mobile PENTIUM II, 266 MHz und 160 MB RAM zum Einsatz. Dazu wurde über TCP/IP eine Verbindung zwischen Client und Server aufgebaut – wahlweise über GSM oder über ein schnelles LAN. Vom Server wurde ein Bild der Größe 1024x1024 Pixel mit einer RoI, die das gesamte Bild umfasst, drei Minuten lang übertragen. Im Abstand von jeweils drei Sekunden sendete der Client Messkommandos an den Server, die dieser unverändert in den kodierten Datenstrom schrieb. Je Kombination aus Kommunikationsalternative[2] und Netztyp erfolgten sechs Messdurchläufe.

5.1.4.2 Performance der Kommunikationsalternativen

In einem ersten Versuch wurde die Performance der Alternativen *HTTP+CGI* und *Sockets* ermittelt. Der Netzzugriff erfolgte dabei über GSM.

[2] Es konnten nur *HTTP+CGI* und *Sockets* getestet werden, da für FastCGI auf der benutzten Serverplattform noch keine Implementierung verfügbar war.

Netz	Kommunikation	RoI-Größe	Erwartungswert	Standardabweichung	Min	Max
GSM	HTTP+CGI	1024x1024	33.3 s	7.9 s	12.1 s	64.7 s
GSM	Sockets	1024x1024	15.6 s	9.7 s	3.0 s	43.2 s

Tabelle 5.1: Erwartungswert, Standardabweichung, Minimum und Maximum der Latenzzeit abhängig von der benutzten Kommunikationsmethode im GSM-Netz.

Für die `HTTP+CGI`-Variante ergibt sich danach eine durchschnittliche Latenzzeit von 33.3 Sekunden mit einer Standardabweichung von 7.9 Sekunden. Diese Zeit liegt für die Socket-Variante bei 15.6 Sekunden mit einer Standardabweichung von 9.7 Sekunden (siehe Tabelle 5.1). Abbildung B.10 veranschaulicht in den Kurven „HTTP (GSM), Mittel“ und „Sockets (GSM), Mittel“ die Latenzzeiten für *Sockets* und *HTTP+CGI* im Verlauf der Übertragung.

Bei `HTTP+CGI` ist die resultierende durchschnittliche Antwortzeit damit mehr als doppelt so lang wie bei Sockets. Da für jedes Kommando ein neuer Serverprozess gestartet werden muss, ist hier auch die CPU-Belastung des Servers größer (vgl. Abbildung B.9). Als problematisch bei `HTTP+CGI` erweist sich darüber hinaus, dass Kommandos in der falschen Reihenfolge oder auch gar nicht ankommen können. Tabelle C.1 illustriert das anhand der Ankunftsreihenfolge der Kommandos bei einer der Testübertragungen. Hierbei trafen von 41 Kommandos 13 verspätet und eines überhaupt nicht wieder beim Client ein. Dieses Problem liegt darin begründet, dass auf Grund der Zustandslosigkeit von `HTTP` für jedes Kommando eine neue Verbindung aufgebaut wird, unabhängig davon, ob das vorherige Kommando bereits erfolgreich übertragen wurde oder nicht.

Damit ist `HTTP+CGI` ohne zusätzliche übertragungssichernde Maßnahmen nicht zur Signalisierung einsetzbar, eine Realisierung auf der Basis von Sockets also vorzuziehen.

5.1.4.3 Einfluss der Unteilbarkeit der Übertragungsschritte

Zusätzlich wurde geprüft, wie sich die Unteilbarkeit der Übertragungsschritte auf die Latenzzeit auswirkt. Der obige Versuch mit der Socket-Kommunikation wurde dazu über eine unbelastete Highspeed-Ethernetverbindung (100 MBit/s) wiederholt, um den Einfluss der Übertragungsstrecke auf die Latenzzeit zu minimieren. Durch softwaremäßige Beschränkung der Ausgabebandbreite des Serverprogramms erfolgte die Simulation der Bandbreite eines GSM-Netzes.

Netz	Kommunikation	RoI-Größe	Erwartungswert	Standardabweichung	Min	Max
LAN	Sockets	512x512	2.8 s	2.2 s	0.5 s	7.6 s
LAN	Sockets	1024x1024	6.8 s	4.0 s	1.1 s	18.3 s

Tabelle 5.2: Erwartungswert, Standardabweichung, Minimum und Maximum der Latenzzeit abhängig von der RoI-Größe im Highspeed-LAN.

Die Kurve „Sockets (LAN), Mittel“ in Abbildung B.10 zeigt den dabei gemessenen Verlauf der Latenzzeiten. Die Peaks in der Kurve werden von Übertragungsschritten in höherfrequenten Teilbändern verursacht, in denen viele Koeffizienten traversiert werden müssen. In diesen ist die Latenzzeit bis zu dreimal so hoch wie im Durchschnitt (vgl. Tabelle 5.2). Da die maximale Anzahl der Wavelet-Koeffizienten in einem Teilband abhängig von der RoI-Größe ist, treten bei größeren RoIs auch längere Latenzzeiten auf.

Für große Bilder und große RoIs hat somit die Design-Entscheidung, unteilbare Übertragungsschritte zu verwenden, einen merklichen negativen Einfluss auf die Latenzzeit. Die

in Abschnitt 4.5.1.2 erwähnte Möglichkeit abbrechbarer Übertragungsschritte offeriert hier Verbesserungspotenzial. Eine Abbrechbarkeit jeweils am Ende einer Scanline vereinfacht die Realisierung und sollte zur Reduktion der Latenzzeit ausreichend sein.

5.2 RoI-Unterstützung für Bildautor und Bildbetrachter

5.2.1 Problemstellung

An der visuellen Kommunikation über Online-Dienste sind zwei Nutzergruppen beteiligt: die *Bildautoren*, die Bilder erstellen bzw. aufbereiten und sie im Netz verfügbar machen, und die *Bildbetrachter*, die diese Bilder herunterladen und auf dem Display ihres Rechners betrachten (vgl. 4.5.2.3). Eine nahe liegende Idee ist es, die Funktionalität des RoI/LoD-basierten Übertragungssystems mit einer nutzerfreundlichen Schnittstelle zu versehen, um diese Nutzergruppen zu unterstützen.

Beide Nutzergruppen haben dabei unterschiedliche Anforderungen. Die Arbeitsaufgabe des Bildautors kann die Erzeugung von RoIs beliebiger Form im Bild, die Definition von deren lokalen LoDs, die Priorisierung von RoIs und die Kontrolle der Auswirkungen der gewählten Parametrierung umfassen. Der Betrachter des Bildes möchte dagegen schnell die ihn interessierenden Teile des Bildes in akzeptabler Qualität angezeigt bekommen. Während also der Bildautor große Flexibilität und reichhaltige Einstellmöglichkeiten benötigt, muss das User Interface für den Bildbetrachter möglichst einfach aufgebaut und leicht zu bedienen sein. Da Bildautor und Betrachter dieselbe Region als unterschiedlich wichtig empfinden können, muss der Betrachter in die Lage versetzt werden, neue RoIs zu definieren oder existierende anders zu priorisieren. Die Definition neuer RoIs erfolgt dabei während laufender Übertragung, deshalb muss sie sehr schnell möglich sein.

5.2.2 Realisierung

5.2.2.1 Autorenwerkzeug

Die oben beschriebenen Anforderungen des Bildautors wurden in einem Autorenwerkzeug realisiert. Das Werkzeug unterstützt die folgenden Funktionen:

- Laden eines Bildes,
- Definition rechteckiger, elliptischer und polygonaler RoIs auf dem Bild per Mausklick,
- Spezifikation des lokalen LoDs und der Priorität jeder RoI per Dialogbox,
- Speichern der Einstellungen in einer RoI-Definitionsdatei,
- Simulieren einer Übertragung des Bildes über einen Kanal mit wählbarer Bandbreite zur Überprüfung der Einstellungen.

Falls der Footprint einer definierten RoI ein sich selbst durchdringendes Polygon ist, kann es Probleme mit dem Scanline-Algorithmus (vgl. 4.5.1.3) geben. Daher wird ein solcher Footprint durch seine konvexe Hülle ersetzt, die mit dem Jarvis-March-Algorithmus [Jar73] berechnet wird. Durch die gespeicherte RoI-Definitionsdatei wird der RoI/LoD-Codec auf der Enkoderseite (Bildserver) gesteuert. Die Syntax dieser Datei ist in Anhang D angegeben. Per Texteditor kann an diese Datei ein Kommandoscript mit vordefinierten Kommandos angehängt werden, das bei der Übertragung ausgeführt wird.

5.2.2.2 Betrachtungswerkzeug

Zur Steuerung der Bildübertragung durch den Betrachter wurde ein Betrachtungswerkzeug realisiert, dessen minimalistisch gehaltene Nutzerschnittstelle lediglich drei Funktionen anbietet:

- Definition einer neuen rechteckigen bzw. elliptischen RoI,
- Beschränken der gesamten verfügbaren Übertragungsbandbreite auf eine bestimmte RoI,
- Beendigung der Übertragung.

Bei der RoI-Definition wurden polygonale RoIs ausgeschlossen, da hier die grafische Eingabe komplizierter ist und mehr Zeit kostet als das einfache „Aufziehen“ eines Rechtecks oder einer Ellipse mithilfe der Maus. Wird eine neue RoI definiert, so werden die Komponenten ihres lokalen LoD sowie die Priorität auf die maximal möglichen Werte gesetzt und das Kommando `DEFRECT` bzw. `DEFELIPS` zum Server gesendet. Vorher wird die Priorität aller anderen RoIs durch Senden des Kommandos `PRECPRIOMOD(ID_ALL_ROIS)` auf Null gesetzt. Nun steht zunächst die volle Übertragungsbandbreite für die zuletzt spezifizierte RoI zur Verfügung, die daher sehr schnell verfeinert wird. Ist die Übertragung von Daten für die neue RoI beendet, so wird die Verfeinerung der verbleibenden RoIs fortgesetzt. Die Prioritäten von durch den Bildautor vordefinierten RoIs werden so durch andere Werte ersetzt, die den Bedarf des Bildbetrachters besser widerspiegeln.

Die zweite Funktion, das Beschränken der verfügbaren Bandbreite auf eine bestimmte RoI, dient dazu, schnell die vom Autor vorgegeben Einstellungen für den lokalen LoD und die Priorität einer bereits existierenden RoI zu erhöhen, und alle anderen RoIs zunächst von der Übertragung auszuschließen. Diese Funktion erfordert lediglich das Bewegen des Mauszeigers in das Innere des Footprints der gewünschten RoI. Das Kommando `PRECPRIOMOD` setzt die Priorität dieser RoI auf einen hohen Wert, und `LLODMOD` wird benutzt, um die Komponenten des lokalen LoDs der RoI auf Maximalwerte zu setzen. Außerdem wird wiederum die Priorität aller anderen RoIs mit dem Kommando `PRECPRIOMOD(ID_ALL_ROIS)` auf Null gesetzt. Das bewirkt zunächst eine maximale Verfeinerung der ausgewählten RoI, bevor die Übertragung von Daten für die verbleibenden RoIs fortgesetzt wird.

Die dritte Funktion, das Stoppen der Übertragung, beendet die Übertragung des Bildes, um die knappe Bandbreite für andere Bilder nutzen zu können. Das wird durch Senden des Kommandos `STOPROI(ID_ALL_ROIS)` realisiert.

5.2.3 Übertragungsbeispiel

Abbildung 5.1 (links) zeigt ein Beispiel für die Unterstützung des Autors. Dieser hat eine Hintergrund-RoI mit dem gesamten Bild als Footprint und $1/8$ der Bildauflösung in beiden Richtungen sowie 8 Bit Genauigkeit als lokalen LoD definiert. Weiterhin hält er den Leuchtturm für das bildwichtigste Detail und definiert daher dort eine weitere RoI mit voller Auflösung und Genauigkeit. Außerdem wird dieser RoI eine höhere Priorität zugewiesen.

Der Betrachter interessiert sich jedoch für den Namen des Bootes. Er definiert daher während der Übertragung eine neue RoI (siehe Abbildung 5.1 (rechts)). Da nach dem Absenden der oben beschriebenen Befehlssequenz an den Server und deren Einbettung in den kodierten Datenstrom die gesamte Bandbreite für dieses kleine Gebiet zur Verfügung steht, erreicht die neue RoI fast augenblicklich die maximal mögliche Detaillierung, bevor die Hintergrund-RoI weiter verfeinert wird.

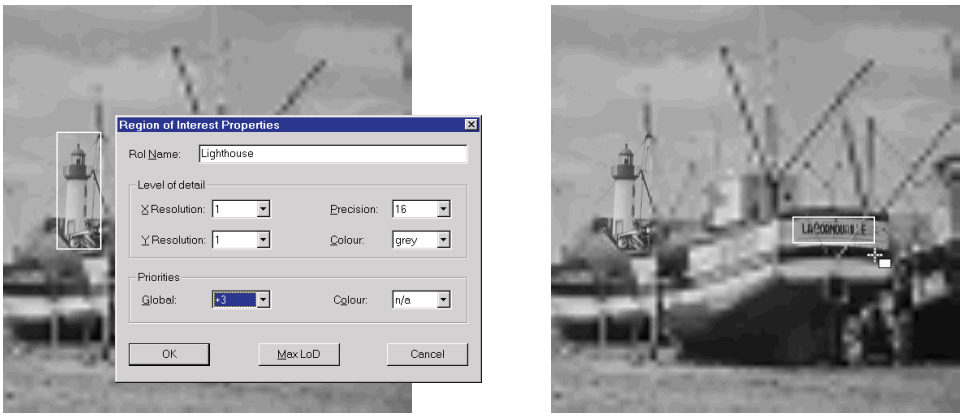


Abbildung 5.1: Beispiel flexibler RoI-Unterstützung für Bildautor (links) und Bildbetrachter (rechts).

5.2.4 Diskussion

Durch die Definition von mehreren RoIs im Bild kann der Bildautor einen Kompromiss zwischen der Detailliertheit und dem Bandbreitenbedarf festlegen. Durch Kodierung „unwichtiger“ Bildteile mit niedrigem LoD wird Bandbreite eingespart, ohne dass auf hohe Detailtreue in wichtigen Bildbereichen verzichtet werden muss. Das Einsparpotenzial ist sehr unterschiedlich und hängt von der konkreten Größe und Parametrisierung der RoIs ab; es ist schwierig, hier ein „typisches“ Szenario anzugeben.

Hervorzuheben ist die Steuerbarkeit durch den Bildbetrachter zur Übertragungszeit; bei kleinen interessierenden Bereichen (wie in Abbildung 5.1) ist die RoI fast sofort in voller Detaillierung verfügbar, eine kurze Latenzzeit (siehe 5.1.4) vorausgesetzt.

5.3 Übertragung und Anzeige großer Bilder

Bisher wurde eine Anwendung betrachtet, bei der die übertragenen Bilder vollständig auf der verfügbaren Bildschirmfläche dargestellt werden können, und der RoI/LoD-Mechanismus dazu verwendet, die Detailliertheit und damit die Bandbreitenanforderungen ausgewählter Bildbereiche anzupassen.

Sind die zu übertragenden Bilder größer als die verfügbare Bildschirmfläche, so bietet es sich an, nur die Bildteile zu übertragen, die auch auf dem Bildschirm sichtbar sind, um Bandbreite zu sparen.

Eine nahe liegende Anwendung dieses Konzeptes ist ein Bildbetrachter, der mit dem konventionellen Zoom- und Pan-Ansatz große Bilder anzeigt, die auf einem Server gespeichert sind und online betrachtet werden sollen. Dabei gibt es zu jedem Zeitpunkt genau eine aktive RoI im Bild – den sichtbaren Bildausschnitt. Hineinzoomen verdoppelt jeweils dessen Auflösung, Herauszoomen halbiert sie. Durch *Panning* oder *Scrolling* kann der sichtbare Ausschnitt auf dem Originalbild verschoben werden. Bei jeder Änderung von Auflösung oder Position des Ausschnitts wird zunächst mit einem `STOPROI(ID_ALL_ROIS)`-Kommando die Übertragung gestoppt. Danach wird mit `DEFRECT` eine neue rechteckige RoI definiert, deren Footprint den sichtbaren Bildausschnitt beschreibt; die im lokalen LoD angegebene Auflösung wird vom verwendeten Zoom-Faktor bestimmt. Falls die gewünschte Auflösung größer als die Auflösung des Originalbildes ist, so wird die RoI in voller Auflösung übertragen. Die „fehlende“ Vergrößerung kann auf Client-Seite durch Anwenden

verschiedener Skalierungsverfahren berechnet werden, beispielsweise indem die interpolierenden Eigenschaften der inversen Wavelet-Transformation ausgenutzt werden.

Obwohl bei großen Bildern meist nur ein Ausschnitt in voller Detailliertheit benötigt wird, ist es wünschenswert, diesen im Kontext des Gesamtbildes betrachten zu können. Im nächsten Abschnitt geht es um eine solche Fokus-und-Kontext-Technik für große Bilder, die verschiedene Bildteile in unterschiedlicher Auflösung anzeigt und für die Übertragung ebenfalls das Konzept nutzt, nur die zur Anzeige benötigten Bilddaten zu transferieren.

5.4 Die Fokus-und-Kontext-Technik „RECHTECKIGER FISHEYE-VIEW“

5.4.1 Problemstellung

Der Platzbedarf großer Digitalbilder stößt schnell an die Grenzen der verfügbaren Bildschirmfläche. Daher ist eine Entscheidung zu treffen, welche Teile des Bildes sichtbar sein sollen. Der Nutzer hat bei einer solchen Präsentation zwei einander widersprechende Ziele: Ein hoher *Detaillierungsgrad* ist ebenso gefordert wie ein *Überblick* über die gesamte Darstellung. Die hohe Detailliertheit ist jedoch häufig bei großen Bildern zu jedem Zeitpunkt nur für ein Teilgebiet *(Fokus)* notwendig, für das der Betrachter gerade Interesse hat. Diese Annahme kann genutzt werden, um den Konflikt zwischen Detail- und Übersichtsproblem zu lösen. Für die Visualisierung großer Datenmengen auf Desktop-Workstations wurden dazu *FishEye-Techniken* entwickelt, von denen einige bereits in 2.7 vorgestellt wurden.

Bei der Nutzung mobiler Rechner zur Bildübertragung und -darstellung sind sowohl Rechenleistung als auch Bildschirmplatz noch eingeschränkter als im skizzierten Workstation-Szenario, und die benutzten drahtlosen Netze sind langsam. Eine für diese Umgebungen angepasste Präsentationstechnik muss in der Lage sein, große Rasterbilder Bildschirmplatz sparend anzuzeigen, wobei ein sinnvoller Kompromiss zwischen den beiden Forderungen „Detail“ und „Überblick“ gefunden werden muss. Diese Anforderung lässt sich gut durch eine Fokus-und-Kontext-Technik erfüllen, die sich jedoch durch geringe Anforderungen an die Rechenleistung auszeichnen muss, um auf mobiler Hardware einsetzbar zu sein. Um die Übertragungsbandbreite effektiv zu nutzen, sollte sie außerdem mit der in Kapitel 4 vorgeschlagenen Übertragungstechnik kombinierbar sein, damit nur die Daten übertragen werden müssen, die auch zur Darstellung benötigt werden.

Neben diesen grundlegenden Anforderungen ist eine flexible Konfigurierbarkeit von Fokusregion und Kontextbereichen sehr wünschenswert. Insbesondere sollte der Nutzer dabei unterstützt werden, Größe, Position und Detailliertheit des Fokusbereiches zu spezifizieren. Da nicht alle Teilbereiche der Darstellung denselben Detaillierungsgrad bieten, müssen direktmanipulative Interaktionstechniken mit kurzer Antwortzeit angeboten werden, die es durch Repositionierung des Fokus gestatten, in Kontextbereichen „verborgene“ Details sichtbar zu machen. Im Fokusbereich müssen grundlegende geometrische Eigenschaften des Bildes (wie Winkel, Proportionalität bzw. Parallelität) originalgetreu wiedergegeben werden. Daher ist hier keine Verzerrung akzeptabel. Im Kontextbereich ist eine durch Verkleinerung entstehende Verzerrung die notwendige Voraussetzung für die Einsparung von Bildschirmplatz und kann daher nicht vermieden werden. Interaktive Kontrolle über die Stärke der Verzerrung muss es dem Nutzer erlauben, den Kompromiss zwischen Verzerrung und Platzbedarf an die aktuelle Arbeitsaufgabe anzupassen.

Im folgenden Abschnitt soll der RECHTECKIGE FISHEYE-VIEW als eine neue Fokus-und-Kontext-Technik vorgeschlagen werden, die die obigen Anforderungen erfüllt (vgl. auch [Rau98a, Rau00c]).

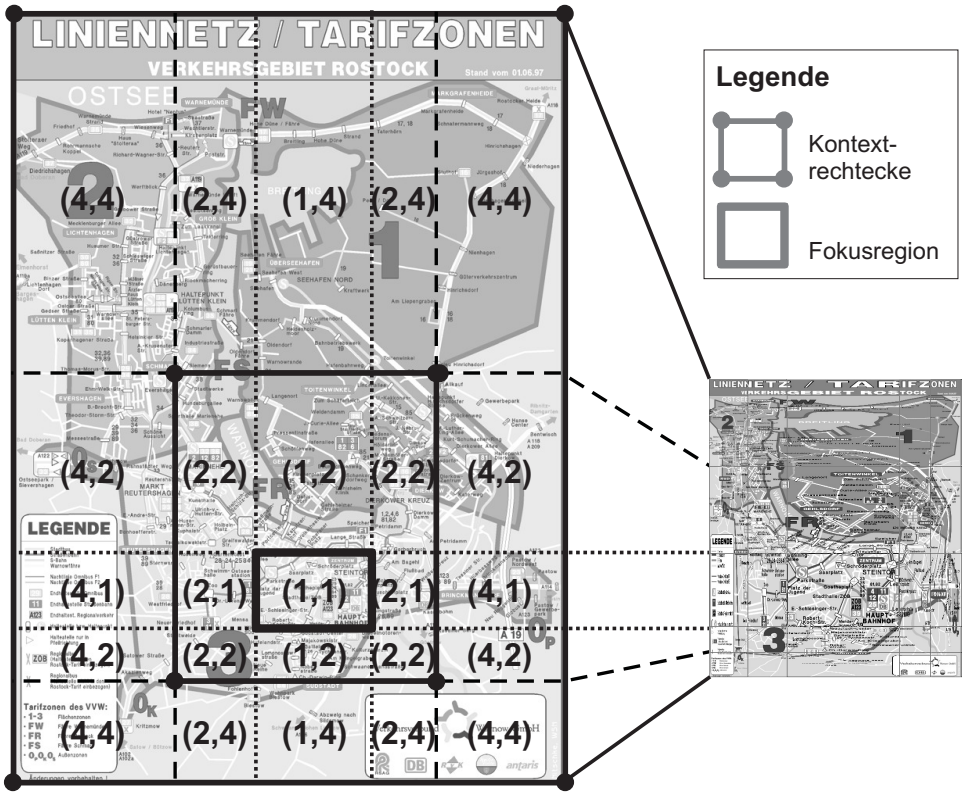


Abbildung 5.2: Originalbild mit überlagertem Skalierungsgitter (links, 1252x1760 Pixel) und resultierender RECHTECKIGER FISHEYE-VIEW (rechts, 575x680 Pixel).

5.4.2 Funktionsweise des RECHTECKIGEN FISHEYE-VIEWS

5.4.2.1 Grundprinzip

Um die Anforderung nach geringer Rechenzeit zu erfüllen und die Integration mit der RoI/LoD-basierten Übertragungstechnik zu ermöglichen, wird das Bild in nicht überlappende rechteckige Teile unterteilt, in denen die Verzerrung jeweils für alle Pixel gleich ist. Als Verzerrungsparameter werden ausschließlich Verkleinerungsfaktoren benutzt, die Zweierpotenzen sind. Das dient der Verringerung des Rechenaufwandes und ist für die Integration mit der Übertragungsmethode zwingend notwendig.

Die Abbildungen 5.2 und 5.3 illustrieren die Idee. Eine *Fokusregion* R_0 in der Bildmitte zeigt einen Teil des Bildes in Originalgröße an. Sie ist von *Kontextringen* umgeben, die die verbleibenden Bildteile verkleinert darstellen, um Bildschirmplatz zu sparen. Jeder Kontextring C_i $(i > 0)$ ist auf der Basis von *Kontextrechtecken* R_i (siehe Abbildung 5.2) wie folgt als Pixelmengendifferenz definiert:

$$C_i = R_i \setminus R_{i-1}. \tag{5.1}$$

Dem i-ten Ring wird ein *Skalierungsfaktor* $s_i := 2^i$ zugewiesen, der die Verzerrung des Ringes und damit die Platzersparnis auf dem Bildschirm steuert. Da das Interesse des Betrachters an Bilddetails mit wachsender Entfernung vom Fokus sinkt, ist der Verkleinerungsfaktor eines Ringes umso größer, je weiter der Ring von Fokus entfernt ist. Jeder Kontextring soll an den benachbarten Ring ohne Diskontinuitäten anschließen. Daher wird ein Ring C_i in $8 \cdot i$ *Teilrechtecke* unterteilt. Jedes dieser Rechtecke besitzt einen Skalierungsfaktor

für die X- und einen weiteren für die Y-Richtung. Einige dieser Faktoren werden durch die Kontextrechtecke innerhalb des Ringes C_i bestimmt, die verbleibenden werden auf 2^i gesetzt. Das Gitter der hierbei entstehenden Teilrechtecke soll im Folgenden als *Skalierungsgitter* bezeichnet werden.

Abbildung 5.2 (links) zeigt ein solches Gitter mit zwei Kontextringen, wobei für jedes Teilrechteck ein Paar von Verkleinerungsfaktoren als *Skalierungspaar* (s_X, s_Y) angegeben ist. Abbildung 5.2 (rechts) zeigt das Ergebnis der Skalierung. Damit entsteht eine den Rubber Sheets (vgl. 2.7.3) ähnliche Darstellung. Im Unterschied zu dieser Arbeit funktioniert diese Technik jedoch für Rasterbilder, integriert ein Übertragungsschema und weist mehrere Kontextringe auf.

Hinsichtlich der Konfigurierbarkeit des Fokus wurde gefordert, dass neben seiner Größe und Position auch der gewünschte Detaillierungsgrad definiert werden kann. Das erfolgt durch *Zooming*, wodurch ein *Zoomfaktor* f_{zoom} für den Fokus spezifiziert wird. Der Konsistenz mit obigem Schema wegen können die Zoomfaktoren ebenfalls nur Werte annehmen, die Zweierpotenzen sind. Alle Skalierungsfaktoren werden mit dem Zoomfaktor[3] multipliziert, damit die Eigenschaft erhalten bleibt, dass der Fokus die höchste Detaillierung bietet.

Die Kontextringe können in verschiedenen Breitenverhältnissen zueinander dargestellt werden, was durch die Spezifikation von *Verhältniswerten* gesteuert wird. Das heißt, dass z.B. der erste Ring mit $s_1 = 2$ durch einen Verhältniswert von 2 so konfiguriert werden kann, dass er in der Darstellung zwei Mal so breit ist wie der zweite Ring mit $s_2 = 4$ und einem Verhältniswert von 1. Die Verhältniswerte erlauben es somit dem Nutzer, die Kontextringe hinsichtlich Verzerrung und Platzbedarf an seine Bedürfnisse anzupassen. Zusätzliche Flexibilität wird dadurch erreicht, dass der Benutzer mit einem *Ring-Sichtbarkeitsflag* pro Ring bestimmen kann, welche Ringe angezeigt werden.

5.4.2.2 Berechnung der Darstellungsgeometrie

Nachdem die Grundstruktur des RECHTECKIGEN FISHEYE-VIEWS beschrieben wurde, gibt dieser Abschnitt konkrete Formeln zur Berechnung der Darstellungsgeometrie an. Zunächst soll die Berechnung der FISHEYE-Geometrie auf der Basis der im vorigen Abschnitt vorgestellten Steuerungsparameter beschrieben werden. Im Anschluss wird ein Ansatz zur automatischen Berechnung der Kontextierungsparameter *Ring-Sichtbarkeit* und *Verhältniswerte* vorgestellt.

Gesucht sind die Eckpunktkoordinaten $Pos_{r,i}$ der Kontextrechtecke im Originalbild für $i = 1, \ldots, nBelt - 1$. Sie werden aus den Größen *Bildgröße*, *Position und Größe der Fokusregion* sowie den *Verhältniswerten* und der *Sichtbarkeitsinformation* der Kontextringe berechnet. Durch Änderung dieser Parameter mithilfe direkter Manipulation (siehe 5.4.2.4) kann der RECHTECKIGE FISHEYE-VIEW durch den Nutzer konfiguriert werden. Die folgenden Eingabeparameter werden benötigt:

$PicWid$	$\Rightarrow$	Breite des Originalbildes
$PicHgt$	$\Rightarrow$	Höhe des Originalbildes
$nBelt$	$\Rightarrow$	Anzahl der Ringe plus Fokus
Scl	$\Rightarrow$	Feld von Skalierungsfaktoren Scl_i
$Ratio$	$\Rightarrow$	Feld von Verhältniswerten $Ratio_i$
		jeweils für $nBelt$ Ringe, wobei $i = 0, \ldots, nBelt - 1$

[3] Der Zoomfaktor wird ebenfalls als Verkleinerungsfaktor interpretiert – je höher der Faktor ist, desto kleiner ist das dargestellte Bild.

$$\begin{aligned} Pos_{left,0} & \\ Pos_{top,0} & \\ Pos_{right,0} & \\ Pos_{bot,0} & \Rightarrow \text{ Koordinaten des Fokus} \end{aligned}$$

Das Feld *Scl* enthält für den Fokus den Zoomfaktor f_{zoom} und für jeden sichtbaren Ring das Produkt des entsprechenden Skalierungsfaktors s_i und f_{zoom}. Analog enthält $Ratio_i$ einen undefinierten Wert für den Fokus und den korrespondierenden Verhältniswert für jeden sichtbaren Ring. Die Ring-Sichtbarkeitsflags werden also ausgewertet, indem bei der Berechnung von $nBelt$, Scl_i und $Ratio_i$ „unsichtbar“ geschaltete Ringe ignoriert werden.

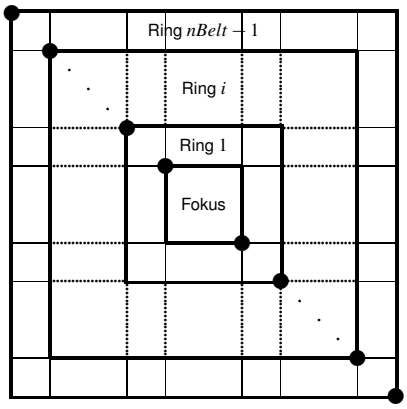


Abbildung 5.3: Geometrische Struktur des RECHTECKIGEN FISHEYE-VIEW-Gitters.

Um die geometrische Struktur des RECHTECKIGEN FISHEYE-VIEWS zu beschreiben, werden *Ringrechtecke* zur konsistenten Behandlung von Fokus und Kontext eingeführt und deren Eckpunktkoordinaten $Pos_{r,i}$ berechnet (vgl. Abbildung 5.3). Dabei bildet der Fokus das Ringrechteck mit dem Index 0, die Ringrechtecke mit Indizes größer als 0 entsprechen den Kontextrechtecken. Somit müssen folgende Koordinaten berechnet werden:

$$\begin{aligned} Pos_{r,i} \quad \Rightarrow \quad & \text{Koordinate des Ringrechtecks } i \text{ in Richtung r} \\ & \text{im Originalbild} \\ & \text{wobei } r \in \{left, top, right, bot\} \end{aligned}$$

Die Koordinaten des Fokus sind gegeben, und die Koordinaten $Pos_{r,nBelt-1}$ werden implizit durch den Punkt $(0,0)$ und die Größe des Originalbildes $(PicWid, PicHgt)$ definiert. Damit müssen nur noch die Eckpunktkoordinaten der Ringrechtecke $i = 1, \ldots, nBelt-2$ berechnet werden. Der Raum zwischen Fokus und Bildrand wird zwischen diesen Ringen in einem Verhältnis aufgeteilt, das durch die Verhältniswerte in $Ratio_i$ definiert ist. Zur besseren Handhabbarkeit beschreiben die Verhältniswerte das Größenverhältnis der Ringe im resultierenden RECHTECKIGEN FISHEYE-VIEW. Daher müssen sie zunächst in das Koordinatensystem des Originalbildes umgerechnet werden:

$$sclRatio_i \quad = \quad Ratio_i \cdot Scl_i \tag{5.2}$$

Die Breite des Ringes i in Richtung r wird nun wie folgt berechnet:

$$Wid_{r,i} \quad = \quad \left| Pos_{r,nBelt-1} - Pos_{r,0} \right| \cdot \frac{sclRatio_i}{\sum_{k=1}^{nBelt-1} sclRatio_k} \tag{5.3}$$

Danach kann die Breite von Ring i in den vier Richtungen benutzt werden, um die Koordinaten von dessen Ringrechteck auf der Basis der (bereits ermittelten) Koordinaten des

nächstinneren Ringrechtecks $i-1$ wie folgt zu berechnen:

$$Pos_{r,i} = \begin{cases} Pos_{r,i-1} - Wid_{r,i} & r \in \{left, top\} \\ Pos_{r,i-1} + Wid_{r,i} & r \in \{right, bot\} \end{cases} \tag{5.4}$$

Wie bereits erwähnt, werden die Teilrechtecke im RECHTECKIGEN FISHEYE-VIEW aus rechteckigen Regionen im Originalbild erzeugt, indem letztere in X- und Y-Richtung mit (möglicherweise verschiedenen) Skalierungsfaktoren verkleinert werden. Nach der Berechnung von Gleichung (5.4) kann das Problem auftreten, dass Breite und Höhe der resultierenden Rechtecke nicht ganzzahlig durch die entsprechenden Skalierungsfaktoren teilbar sind. Die Konsequenz wären Diskontinuitäten an den Übergängen zwischen zwei Ringen auf Grund weggelassener Pixel. Um dieses Problem zu lösen, wird eine Randbedingung für Gleichung (5.3) eingeführt, die $Wid_{r,i}$ für alle Ringrechtecke außer dem Fokus so variiert, dass eine Division ohne Rest durch das entsprechende Scl_i möglich ist:

$$Wid_{r,i} \equiv 0 \pmod{Scl_i} \tag{5.5a}$$

$$(Dist_{r,i} - Wid_{r,i}) \equiv 0 \pmod{Scl_{i+1}} \tag{5.5b}$$

$$Dist_{r,i} \geq Wid_{r,i} \geq 0 \tag{5.5c}$$

wobei

$$\begin{aligned} i &= 1, \ldots, nBelt - 1 \\ Dist_{r,i} &= |Pos_{r,nBelt-1} - Pos_{r,i-1}| \end{aligned}$$

Diese Nebenbedingung setzt voraus, dass Breite und Höhe der Fokusregion Vielfache des Zoomfaktors sind. Weiterhin muss der in alle vier Richtungen verbleibende Raum ohne Rest durch den Skalierungsfaktor Scl_1 des ersten Ringes teilbar sein. Wenn diese Bedingung nicht erfüllt ist, muss die Position und/oder die Größe des Fokus angepasst werden. Dabei wird bei Abweichungen in Richtung $left$ bzw. top der Fokus nach links bzw. oben verschoben; bei Abweichungen in Richtung *right* oder *bot* wird der Fokus vergrößert. Das führt bei einem Zoom-Faktor von $f_{zoom} = 1$ immer zu einer gültigen Lösung. Bei größerem f_{zoom} und nicht ganzzahliger Teilbarkeit einer Ausgangsbilddimension durch f_{zoom} ist obige Bedingung nicht erfüllbar; in diesem Fall müssen beim Originalbild am Rand Zeilen bzw. Spalten angefügt oder weggelassen werden.

Wenn die Nebenbedingung (5.5) verletzt ist, werden die Ergebnisse $Wid_{r,i}$ aus Gleichung (5.3) iterativ so lange um eins erhöht, bis sie Bedingung (5.5) erfüllen. Das führt immer zu einer Unterteilung, bei der Breite und Höhe der Ringe Vielfache der entsprechenden Skalierungsfaktoren Scl_i sind. Die Nebenbedingung kann zu leichten Abweichungen von den in $Ratio_i$ vorgegebenen Breitenverhältnissen der Ringe führen; sie stellt jedoch sicher, dass keine Pixelzeilen oder -spalten weggelassen werden.

Aus den $Pos_{r,i}$ im Originalbild können nun durch Anwenden der Verkleinerungsfaktoren leicht die korrespondierenden Koordinaten $\overline{Pos}_{r,i}$ in der FISHEYE-Darstellung berechnet werden.

$$\overline{Pos}_{left,nBelt-1} = 0 \tag{5.6a}$$

$$\overline{Pos}_{top,nBelt-1} = 0 \tag{5.6b}$$

$$\overline{Pos}_{r,k} = \overline{Pos}_{r,k+1} + \frac{Wid_{r,k+1}}{Scl_{k+1}} \quad \text{für } r \in \{left, top\} \tag{5.7}$$

$$\overline{Pos}_{right,0} = \overline{Pos}_{left,0} - 1 + \frac{Pos_{right,0} - Pos_{left,0} + 1}{Scl_0} \tag{5.8a}$$

$$\overline{Pos}_{bot,0} = \overline{Pos}_{top,0} - 1 + \frac{Pos_{bot,0} - Pos_{top,0} + 1}{Scl_0} \tag{5.8b}$$

$$\overline{Pos}_{r,k} = \overline{Pos}_{r,k-1} + \frac{Wid_{r,k}}{Scl_k} \quad \text{für } r \in \{right, bot\} \tag{5.9}$$

Die Darstellung des RECHTECKIGEN FISHEYE-VIEWS auf dem Bildschirm geschieht derart, dass basierend auf den in (5.4) berechneten Koordinaten die Bilddaten aus dem Ausgangsbild extrahiert und in korrekter Skalierung auf den korrespondierenden Bildschirmkoordinaten dargestellt werden. Auf diese Art wird jedes durch den RECHTECKIGEN FISHEYE-VIEW erzeugte Teilrechteck einzeln behandelt.

Durch $Pos_{r,i}$ und $\overline{Pos}_{r,i}$ wird also eine Koordinatentransformation definiert, die als FISHEYETransformation bezeichnet werden soll. Diese Transformation ist nur innerhalb der Fokusregion eindeutig invertierbar, in den Kontextrechtecken existiert lediglich eine Pseudo-Inverse. Das heißt, hier wird für einen in FISHEYE-Koordinaten gegeben Punkt $\overline{P}_0$ ein rechteckiger Bildbereich geliefert, der s_X Pixel breit und s_Y Pixel hoch ist. Die Werte für s_X und s_Y entsprechen dabei dem Skalierungspaar des FISHEYE-transformierten Teilrechtecks, in dem $\overline{P}_0$ liegt. Welche der Pixelkoordinaten letztendlich als inverser Wert geliefert werden, hängt von der Anwendung ab. Eine pragmatische Lösung besteht darin, die Koordinaten des mittleren Pixels (bzw. eines der mittleren Pixel) zurückzugeben.

5.4.2.3 Automatische Steuerung der Kontextierung

Die Forderung nach Steuerbarkeit der Kontextierung kann durch die Möglichkeit, Größenverhältnis und Sichtbarkeit der Kontextringe festzulegen, erfüllt werden. Die komplexen Zusammenhänge zwischen o.g. Steuerparametern, Verzerrung und Platzbedarf des Kontextes sowie die notwendigen Interaktionen lenken den Nutzer jedoch stark von seiner eigentlichen Aufgabe, der Explorierung des Bildes durch Verschieben des Fokus, ab. Daher soll hier ein Ansatz vorgestellt werden, der die Parameter *Ring-Sichtbarkeit* und *Verhältniswerte* automatisch berechnet, so dass bestimmte Rahmenbedingungen eingehalten werden und der Nutzer sich voll auf seine eigentliche Arbeitsaufgabe konzentrieren kann.

Folgende Rahmenbedingungen erscheinen sinnvoll:

1. Die FISHEYE-Darstellung sollte die verfügbare Darstellungsfläche auf dem Bildschirm nicht überschreiten. Die bei einer Überschreitung zur Anzeige notwendigen Scrollbars erschweren die Übersicht, erfordern zusätzliche Bedienschritte und führten in Benutzertests zu geringer Akzeptanz der Darstellungstechnik.
2. Die Darstellung sollte die verfügbare Bildschirmfläche möglichst vollständig ausnutzen.
3. Der dem Fokus benachbarte Ring sollte den kleinstmöglichen Skalierungsfaktor aufweisen, mit dem Forderung (1) erfüllbar ist, damit die größtmögliche Detailliertheit der Kontextbereiche in der Nähe des Fokus sichergestellt ist.

Zur Erfüllung der Forderungen (1) und (2) ist es notwendig, zwei getrennte Listen $Ratio_x$ und $Ratio_y$ von Verhältniswerten einzuführen, eine für die X- und eine weitere für die Y-Richtung. Statt $sclRatio_i$ wird in Gleichung (5.2) dann analog $sclRatio_{x,i}$ und $sclRatio_{y,i}$ verwendet. Gleichung (5.3) benutzt $sclRatio_{x,i}$ für $r \in \{left, right\}$ und $sclRatio_{y,i}$ für $r \in \{top, bot\}$. Da die Verhältniswerte in X- und Y-Richtung voneinander unabhängig sind, soll im Folgenden nur die Berechnung der Werte für X diskutiert werden; die Berechnung für Y erfolgt analog.

Als Ausgangsgrößen für die automatische Anpassung werden die für alle Kontextringe (also ohne Fokus) insgesamt verfügbare Breite x_O im Originalbild und x_F in der FISHEYE-Darstellung sowie die Skalierungsfaktoren Scl_i benötigt. Die automatische Anpassung ist immer dann erforderlich, wenn sich eine dieser Größen ändert. Das ist der Fall bei einer Änderung der Größe der Darstellungsfläche (x_F ändert sich), der Fokusgröße (x_O und x_F ändern sich) oder des Zoom (Scl_i, x_O und x_F ändern sich). Die Verhältniswerte $Ratio_{x,i}$ sollen nun so berechnet werden, dass sie die Breite der einzelnen Ringe in FISHEYE-Koordinaten widerspiegeln:

$$x_O = \sum_{i=1}^{nBelt-1} Scl_i \cdot Ratio_{x,i} \tag{5.10}$$

$$x_F = \sum_{i=1}^{nBelt-1} Ratio_{x,i} \tag{5.11}$$

Dieses Gleichungssystem ist nur für $nBelt = 3$ oder zwei ausgewählte Kontextringe[4] eindeutig lösbar:

$$n = nBelt - 1 \tag{5.12}$$

$$Ratio_{x,n} = \frac{x_O - x_F \cdot Scl_{n-1}}{Scl_n - Scl_{n-1}} \tag{5.13}$$

$$Ratio_{x,n-1} = x_F - Ratio_{x,n} \tag{5.14}$$

Führt man die folgende zusätzliche Bedingung für die beiden inneren Ringe ein:

$$Ratio_{x,n-2} = Ratio_{x,n-1} \tag{5.15}$$

so erhält man aus den Gleichungen (5.10), (5.11) und (5.15) ein System, das auch für drei Ringe eindeutig lösbar ist.

Die Lösungen ergeben sich dann wie folgt (n wie in Gleichung (5.12)):

$$Ratio_{x,n} = \frac{2x_O - x_F(Scl_{n-1} + Scl_{n-2})}{2Scl_n - Scl_{n-1} - Scl_{n-2}} \tag{5.16}$$

$$Ratio_{x,n-1} = \frac{x_F - Ratio_{x,n}}{2} \tag{5.17}$$

$$Ratio_{x,n-2} = Ratio_{x,n-1} \tag{5.18}$$

Falls $Ratio_{x,i} <= 0$ gilt, so wird der entsprechende Ring als unsichtbar angenommen. Die Lösungen der Gleichungen (5.13) und (5.14) bzw. (5.16), (5.17) und (5.18) haben einen von drei Typen:

Typ 1: $Ratio_{x,n-1} \geq 0 \wedge Ratio_{x,n} \geq 0$. In diesem Fall sind Forderung (1) und (2) beide erfüllt.

Typ 2: $Ratio_{x,n-1} < 0 \wedge Ratio_{x,n} > 0$. In diesem Fall sind Forderung (1) und (2) beide nicht erfüllt, der RECHTECKIGE FISHEYE-VIEW ist breiter als der zur Verfügung stehende Bildschirmplatz.

[4] In diesem Fall wird für alle anderen Ringe $Ratio_{x,i} = 0$ angenommen.

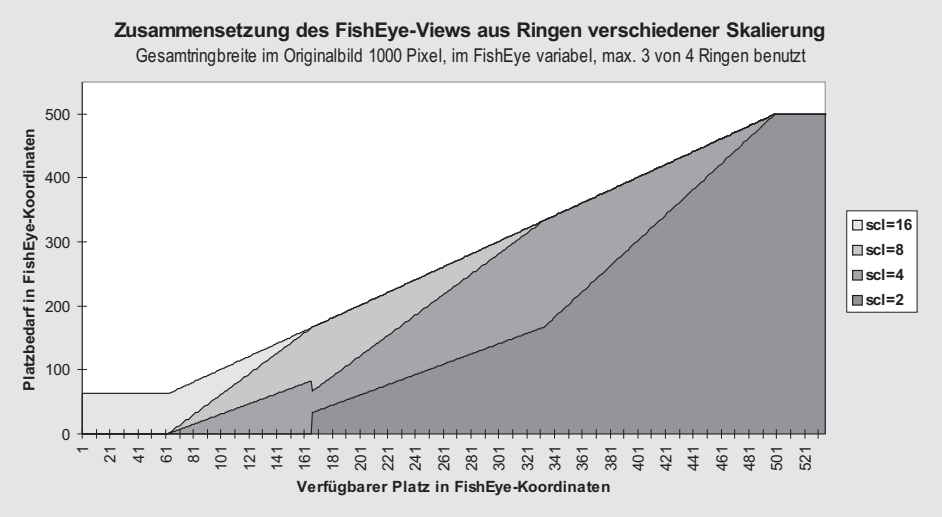


Abbildung 5.4: Automatische Berechnung der Kontextierung bei einer Gesamtbreite aller Ringe von 1000 Pixeln im Originalbild und von 1 bis 535 Pixeln in der FISHEYE-Darstellung.

Typ 3: $Ratio_{x,n-1} > 0 \wedge Ratio_{x,n} < 0$. In diesem Fall ist Forderung (1), aber nicht Forderung (2) erfüllt, der RECHTECKIGE FISHEYE-VIEW ist schmaler als der zur Verfügung stehende Bildschirmplatz.

Bei $nBelts = 3$ sind Lösungen der Typen 2 und 3 zwar unbefriedigend, können jedoch nicht weiter verbessert werden. Falls $nBelts \geq 4$ ist[5], kann in den meisten Fällen eine Lösung vom Typ 1 wie folgt in mehreren Schritten ermittelt werden: $nBelts$ wird als eine maximale Anzahl von Kontextringen angenommen; für die meisten praktischen Anwendungen ist $nBelts \in \{4,5\}$ ausreichend[6]. Ein iteratives Lösen der Gleichungen (5.16) bis (5.18) wählt dann drei aufeinander folgende Kontextringe aus, deren Kombination die Forderungen (1) und (2) erfüllt. Dazu wird nach den Gleichungen (5.12) sowie (5.16) bis (5.18) eine initiale Lösung berechnet. Ist diese vom Typ 1, handelt es sich um das gesuchte Ergebnis. Eine Lösung vom Typ 2 zeigt an, dass zu wenig Platz auf dem Bildschirm zur Verfügung steht. Da der Ring mit dem höchsten Index auch den höchsten Skalierungsfaktor und damit den geringsten Platzbedarf aufweist, ist eine solche Lösung nicht mehr iterativ verbesserbar. Interessant ist eine Lösung vom Typ 3, da sie nicht die gesamte Darstellungsfläche ausfüllt und daher das Potenzial für eine Verbesserung bietet. Dazu wird iterativ in den Gleichungen (5.16) bis (5.18) so lange n um eins vermindert und eine neue Lösung berechnet, bis diese vom Typ 1 oder $n = 3$ ist. Stehen mit $n = 3$ nur noch zwei Ringe zur Verfügung, so erfolgt ein letzter Lösungsversuch mit den Gleichungen (5.13) und (5.14).

Diagramm 5.4 illustriert einige Lösungen, die dieser Algorithmus für ein variables x_F und ein festes x_O liefert. Auf der horizontalen Achse ist x_F als gewünschter Platzbedarf abgetragen, auf der vertikalen Achse der resultierende Platzbedarf nach Ausführung des Algorithmus. Die praktische Entsprechung dieses Szenarios ist das Ändern der Größe der Darstellungsfläche durch den Nutzer, z.B. durch Verkleinern eines Fensters. Für Werte von $x_F < 63$ liefert bereits die erste Iteration eine Lösung vom Typ 2; der Platzbedarf der Darstellung mit einem Ring des höchsten Skalierungsfaktors ist größer als die Darstellungsfläche. Im Bereich von $63 \leq x_F \leq 500$ findet das Iterationsverfahren Lösungen vom Typ 1, und der RECHTECKIGE FISHEYE-VIEW füllt die Darstellungsfläche aus. Für $x_F > 500$

[5] Bei $nBelts > 4$ ist hierbei das Gleichungssystem unterbestimmt.

[6] Eine höhere Anzahl erfordert bei der Vorbereitung des Bildes zur Übertragung zu viele unabhängige Zerlegungsstufen und erzeugt damit stärkere Artefakte, vgl. Abschnitt 4.4.3.

bricht die Iteration mit einer Lösung vom Typ 3 ab, wenn $n = 3$ gilt. In diesem Fall ist die FISHEYE-Darstellung kleiner als die Bildschirmfläche.

5.4.2.4 Interaktionsfunktionalität

Wie in 5.4.1 beschrieben, muss der Nutzer wichtige Parameter des RECHTECKIGEN FISHEYE-VIEWS direktmanipulativ beeinflussen können, um den Bildinhalt schnell und komfortabel explorieren zu können. Insbesondere muss die Steuerung von Größe und Position der Fokusregion einfach und intuitiv sein, so dass der Nutzer ihn interessierende Bildbereiche in voller Detailliertheit betrachten kann. Dazu werden die Interaktionstechniken *Move*, *Resize* und *Jump* verfügbar gemacht. Sie wurden mit einem *Zoom+Pan*-Modus kombiniert, der durch Abschalten aller Kontextringe und Beeinflussen des Zoom-Faktors bei Bedarf das schnelle Umschalten zu einer zwar platzintensiveren, dafür aber verzerrungsfreien Bilddarstellung ermöglicht, die konventionell durch Scrollen bzw. Pannen verschoben und durch Hinein- bzw. Herauszoomen in der Größe geändert werden kann (vgl. 5.3). Die Einstellung der Kontextierungsparameter (Verhältniswerte bzw. Sichtbarkeit der Ringe) erfolgt entweder nach dem in 5.4.2.3 vorgestellten automatischen Ansatz oder über Dialogboxen, die die manuelle Einstellung von Ring-Sichtbarkeit und Verhältniswerten erlauben.

Die Umsetzung der Techniken erfordert es, spezifizierte Bildschirm- oder FISHEYE-Koordinaten unter Nutzung der Pseudo-Inversen der FISHEYE-Transformation (siehe 5.4.2.2) in Bildkoordinaten umzurechnen. Im Folgenden sollen die direktmanipulativen Interaktionsfunktionen detaillierter vorgestellt werden.

Move. Diese Funktion erlaubt es dem Nutzer, die Position des Fokus im Originalbild in kleinen Schritten zu ändern, z.B. durch Ziehen der Maus bzw. durch Betätigen von Bewegungstasten. Eine Bewegung des Fokus beeinflusst nicht den Platzbedarf der Darstellung, so dass die Fokusgröße und die Kontextierungsparameter nicht modifiziert werden. Die Move-Funktion wird verwendet, wenn sich das Interesse des Nutzers in ein dem Fokus benachbartes Gebiet verlagert.

Jump. Mithilfe dieser Funktion kann der Fokus schnell an eine – möglicherweise weit entfernte – neue Position gesetzt werden. Dazu reicht es aus, dass ein Punkt angegeben wird, bevorzugt der Mittelpunkt des neuen Fokus. Diese Funktion kann beispielsweise durch einen Mausklick ausgelöst werden. Wie *Move* ändert sie die Darstellungsgröße nicht.

Resize. Eine schnelle, intuitive Einstellung der Fokusgröße wird durch diese Funktion ermöglicht, die in Analogie zum Ändern der Größe eines Bildschirmfensters z.B. durch Klicken auf die Begrenzung des Fokus verbunden mit dem Ziehen der Maus umgesetzt werden kann. Diese Funktion ändert die Darstellungsgröße oder löst eine Neuberechnung der automatischen Kontextierung aus.

Jump+Resize. Die Funktion *Jump+Resize* kombiniert das beliebige Positionieren des Fokus mit der Definition einer neuen Fokusgröße. Dazu wird ein Rechteck auf dem Bildschirm spezifiziert, das die neue Fokusregion angibt.

Zoom. Diese Funktion erlaubt das Verdoppeln bzw. Halbieren des Zoom-Faktors für den Fokus.

View Panning. Auch wenn es sich beim RECHTECKIGEN FISHEYE-VIEW um eine Platz sparende Darstellungstechnik handelt, ist es nicht immer garantiert, dass die Darstellung auf die verfügbare Anzeigefläche passt. In diesem Fall erlaubt Panning als Alternative zum Scrollen ein schnelleres Verschieben des sichtbaren Bildausschnitts.

5.4.2.5 RoI/LoD-basierte Übertragung

Ein Entwurfsziel für den RECHTECKIGEN FISHEYE-VIEW war die einfache Integrierbarkeit des in Kapitel 4 vorgestellten RoI/LoD-basierten Übertragungsmechanismus. Zur effizienten Nutzung der verfügbaren Bandbreite ist es erforderlich, die Bilddaten für die verschiedenen Teilrechtecke des Gitters in der Auflösung zu übertragen, in der sie für die Darstellung benötigt werden. Dazu wird für jedes Teilrechteck eine eigene RoI definiert, deren lokaler LoD die entsprechenden Werte für die X- und Y-Auflösung spezifiziert. Durch Nutzung des XY-unabhängigen Zerlegungsschemas (siehe 4.4.2) können die erforderlichen Skalierungsfaktoren exakt angegeben werden. Die LoD-Dimensionen *Genauigkeit* und *Farbe* haben keinen Einfluss auf das räumliche Layout des RECHTECKIGEN FISHEYE-VIEWS. Sie werden auf Maximalwerte gesetzt und dienen zur progressiven Verfeinerung.

Um eine frühe Erkennbarkeit der Fokusregion im Verlauf der Übertragung zu sichern, werden die Prioritäten der RoIs in Abhängigkeit von den Ringen, zu denen sie gehören, so gesetzt, dass der Fokus die höchste und der äußerste Ring die niedrigste Priorität erhält. Durch die hohe Priorisierung des Fokus im Verlauf der progressiven Verfeinerung ist eine frühe Erkennbarkeit wichtiger Bildbereiche sichergestellt.

Jede der in 5.4.2.4 beschriebenen Interaktionen *Move*, *Resize* und *Jump* spezifiziert ein neues Gitter, das unter Nutzung des Kommando-Mechanismus (vgl. 4.5.2.3) vom Clientrechner zum Server übertragen und anschließend zwischen Server und Client synchronisiert wird. Dabei wird zuerst die Datenübertragung für alle RoIs des aktiven Gitters durch Senden des Steuerkommandos `STOPROI(ID_ALL_ROIS)` gestoppt. Danach erfolgt die Definition der RoIs des neuen Gitters. Dazu wurde ein Kommando `DEFFISH` in die RoI/LoD-Bibliothek integriert, das statt der getrennten Definition jeder einzelnen RoI nur die Ringrechtecke spezifiziert, auf deren Basis der Kommandoprozessor die erforderlichen RoIs erzeugen kann.

5.4.2.6 Optimierung für interaktives Antwortverhalten

Die Interaktivität des RECHTECKIGEN FISHEYE-VIEWS ist die Voraussetzung, dass der Benutzer trotz der Verzerrungen das Bild erkunden und seinen Inhalt verstehen kann. Daher ist es wichtig, dass jede Aktion des Nutzers zu einer schnellen Antwort des Systems führt. Nur durch ein Antwortverhalten, das den Nutzer nicht bremst, kann dieser ein *„Gefühl“* für die verzerrte Darstellung entwickeln.

Das ist besonders wichtig für die Funktionen *Move* und *Resize*, die nur effektiv sind, wenn jede kleine Positions- oder Größenänderung in einer zügig aktualisierten Darstellung resultiert. Daher erfolgt das Bildschirmupdate ringsequenziell und *unterbrechbar*. Zunächst wird die Fokusregion neu gezeichnet, gefolgt vom ersten Kontextring und so weiter. Die Unterbrechbarkeit ermöglicht den vorzeitigen Abbruch dieser Aktualisierungssequenz, wenn währenddessen eine der Funktionen *Move* oder *Resize* erneut ausgeführt wird.

Genügt der Performance-Gewinn durch das unterbrechbare Zeichnen nicht, so kann statt dessen die von älteren Window-Systemen beim Verschieben von Fenstern bekannte Methode des *Ghost-Feedback* eingesetzt werden. Hierbei wird statt des Fokus-Inhalts nur ein Rechteck über der FISHEYE-Darstellung verschoben, das Position und Größe des potenziellen neuen Fokus repräsentiert. Abbildung B.11 zeigt ein Beispiel. Im Unterschied zum Verschieben von Fenstern auf dem Bildschirm muss der Ghost beim Verschieben seine Größe in Abhängigkeit von der Skalierung des überdeckten Gebietes ändern.

Zur Erzeugung des RECHTECKIGEN FISHEYE-VIEWS müssen große Teile des Originalbildes skaliert werden. Da diese Operation zeitaufwändig sein kann, ist es vorteilhaft, die

Bilddaten *redundant*[7] in verschiedenen Skalierungsstufen abzuspeichern und während der Anzeigeaktualisierung zur Verkürzung der Antwortzeit die richtig skalierte Version wiederzuverwenden. Die Erzeugung der vorskalierten Versionen kann entweder in einem Präprozess oder durch Wiederverwendung skalierter Bilddaten aus einem früheren Interaktionsschritt geschehen.

Hinsichtlich der Erzeugung von Redundanz muss ein Kompromiss zwischen dem zusätzlichen Speicherbedarf und der eingesparten Rechenleistung geschlossen werden. Wenn für jedes mögliche Skalierungspaar eine Kopie des Bildes gespeichert wird, ist die Redundanz sehr hoch. Zum Beispiel würde die vollständiger Speicherung aller Auflösungskombinationen des Gitters in Abbildung 5.2 in einer Redundanz von 206.25% resultieren. Für eine wachsende Anzahl von Kontextringen divergiert die Redundanz bei vollständiger Speicherung. Es ist jedoch möglich, die Redundanz auf $33.\overline{3}\%$ unabhängig von der Anzahl der Kontextringe zu senken, wenn man – analog zur Gauß-Pyramide – nur für solche Skalierungspaare (s_X, s_X) eine verkleinerte Version des Bildes speichert, für die $s_X = s_Y$ gilt, und für die fehlenden Paare die skalierte Version auf Anforderung daraus generiert.

Für diese Generierung gibt es zwei Möglichkeiten: Vergrößern eines Bildes niedriger Auflösung, wenn eine hohe Rechengeschwindigkeit gefordert ist, oder Verkleinern eines Bildes hoher Auflösung, wenn es auf hohe Bildqualität ankommt. Während der Interaktion ist Geschwindigkeit wichtiger als Qualität; daher kommt hier die erste Möglichkeit zum Einsatz. Wenn beispielsweise ein Teilrechteck mit einer Skalierung von $(2,4)$ benötigt wird, so kann es durch Vergrößern des entsprechenden mit Skalierung $(4,4)$ gespeicherten Teilrechtecks erzeugt werden. Interagiert der Benutzer nicht mit dem System, kann die Darstellung durch Verkleinern des Originalbildes in höherer Qualität neu berechnet werden.

Im Verlauf der Übertragung sind – bedingt durch die verschiedenen Skalierungsfaktoren in den unterschiedlichen Teilrechtecken des Gitters – nicht alle Teile des Bildes in allen Auflösungsstufen auf der Clientseite vorhanden. Weiterhin ist die Berechnung der inversen Wavelet-Transformation zu langsam für interaktive Antwortzeiten beim Verschieben bzw. Vergrößern des Fokus. Daher ist es erforderlich, eine effiziente Bilddatenverwaltung zu nutzen, die für einen Bildteil nur dann Speicherplatz belegt, wenn er auch verfügbar ist.

Dazu wurde eine *lokale Bilddatenverwaltung* realisiert (siehe Abbildung 5.5), die verfügbare Bildteile in verschiedenen Skalierungsstufen redundant vorhält. Um die schwach besetzten Bildmatrizen effizient speichern zu können, werden sie in Kacheln unterteilt. Nur diejenigen Kacheln belegen Speicherplatz, für die bereits Bilddaten übertragen wurden[8]. Da nur der Fokus mit voller Auflösung vorliegt, müssen in der höchsten Auflösungsstufe („Grundfläche“ der Pyramide in Abbildung 5.5) auch nur die Kacheln gespeichert werden, die den Fokus überlappen. In der nächstniedrigeren Auflösung müssen Daten für Fokusregion und ersten Kontextring gespeichert werden usw., und in der niedrigsten Auflösungsstufe sind Daten für das gesamte Bild verfügbar. Verschieben oder Vergrößern des Fokus initiiert neue Übertragungen, in deren Verlauf jeweils einige neue Kacheln hinzugefügt werden. Die Funktionen zur Aktualisierung des Bildschirms fordern aus der lokalen Bilddatenverwaltung Daten zur Anzeige an. Ist dabei eine benötigte Kachel nicht in der erforderlichen Auflösung enthalten, so werden die benötigten Daten durch Heraufskalieren aus vorhandenen niedriger skalierten Kacheln erzeugt. Durch dieses Schema ist ein schnelles Bildschirmupdate ebenso gesichert wie eine effiziente Speicherung.

In längeren Pausen zwischen den Nutzerinteraktionen wird die für schnelle Bildschirmaktualisierung genutzte Darstellung niedriger Qualität durch eine langsamer zu berechnende *Qualitätsdarstellung* ersetzt. Dazu wird für jedes Teilrechteck die inverse Wavelet-Transformation mit entsprechender Skalierung auf die Daten im Koeffizientenpuffer des übertragenen Bildes angewendet (vgl. 4.6). Das Ergebnis dieser Rücktransformation wird

[7] Die redundante Speicherung erfolgt auf Clientseite. Nach wie vor ist jedoch die Übertragung redundanzfrei.
[8] Dabei kann es vorkommen, dass Bilddaten nur für einen Teil der Kachel vorhanden sind.

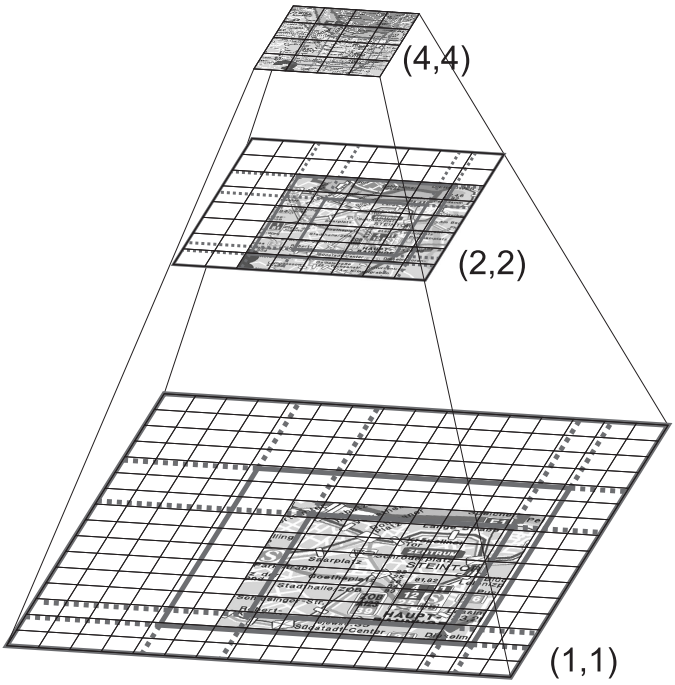


Abbildung 5.5: Redundante lokale Bilddatenverwaltung.

sowohl auf den Bildschirm gezeichnet als auch zur Wiederverwendung in die lokale Bilddatenverwaltung geschrieben.

Durch Nutzung der lokalen Bildverwaltung mit $33.\overline{3}\%$ Redundanz ist es gelungen, auch die rechenintensiven Interaktionstechniken *Move* and *Resize* mit interaktivem Antwortzeitverhalten zu realisieren.

5.4.2.7 Untersuchungen zu multiplen Foki

Bisher wurde davon ausgegangen, dass der Nutzer zu einem Zeitpunkt nur für einen Bereich des Bildes Interesse hat und daher nur eine Fokusregion erforderlich ist. Es sind aber Anwendungsfälle denkbar, in denen multiple Foki wünschenswert sind. Ein Beispiel sind Videokonferenzen mit mehreren Teilnehmern, wobei die den Betrachter besonders interessierenden Konferenzteilnehmer in jeweils einer Fokusregion dargestellt werden, während die anderen im Kontextbereich abgebildet werden. Weitere Beispiele sind eine Anwendung, in der zwei Bildbereiche miteinander verglichen werden sollen, oder die Darstellung von Anfangs- und Endpunkt einer Stadtfahrt.

Abbildung 5.6 zeigt ein Beispiel für einen multifokalen RECHTECKIGEN FISHEYE-VIEW mit 4 Fokusregionen. Problematisch ist die Tatsache, dass sich die Fokusregionen gegenseitig beeinflussen und somit zusätzlich zu den vom Benutzer definierten *expliziten* Foki weitere *implizite* Foki entstehen können. Aus Abbildung 5.6 ist nicht erkennbar, ob sie mit 2, 3, 4 oder sogar mehr als 4 expliziten Fokusregionen erstellt worden ist. Implizite Fokusregionen entstehen immer dann, wenn zwei explizite Foki Anteil an denselben Zeilen bzw. Spalten im Bild haben. Überlappen sich zwei Foki, so ist das Ergebnis eine neue, größere Fokusregion. Abbildung 5.7 stellt die möglichen Fälle dar, die bei der Überlappung zu betrachten sind.

Zur Kontextierung der resultierenden multiplen Foki ist es erforderlich, das Bild in recht-

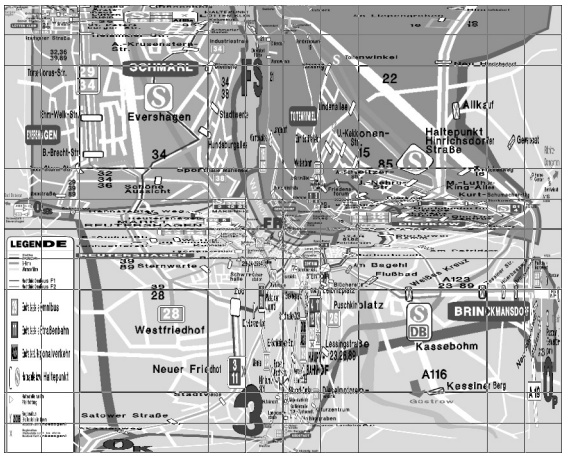
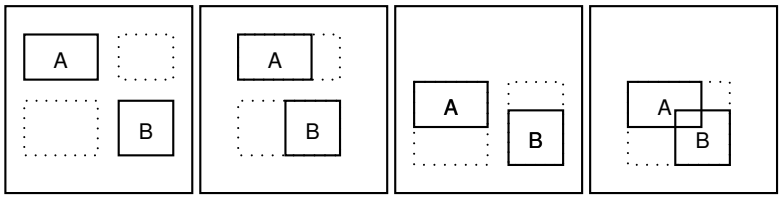


Abbildung 5.6: Beispiel für einen multifokalen RECHTECKIGEN FISHEYE-VIEW mit 4 Fokusregionen.



(a) Keine Überlappung (b) Überlappung in x (c) Überlappung in y (d) Überlappung in $x \wedge y$

Abbildung 5.7: Mögliche gegenseitige Beeinflussungen zweier Foki.

eckige Teilbilder mit jeweils einem Fokus aufzuteilen, die Teilbilder zu kontextieren und die resultierenden Teil-FISHEYE-Views wieder zu einer Darstellung zusammenzusetzen.

Auf Grund der entstehenden impliziten Foki und des damit verbundenen visuellen Overheads erscheint der multifokale RECHTECKIGE FISHEYE-VIEW nur dann geeignet für die Anwendung, wenn das Szenario ohne Überlappung der Foki auskommt, wie das beispielsweise bei einer Videokonferenz möglich wäre. Für das zweite skizzierte Szenario, den Vergleich zweier Bildausschnitte, bietet sich eher die Darstellung der interessierenden Ausschnitte in nebeneinander angeordneten Fenstern an.

5.4.3 Der RECHTECKIGE FISHEYE-VIEW als reine Darstellungstechnik

Die Grundidee des RECHTECKIGEN FISHEYE-VIEWS wurde bisher für eine kombinierte Übertragung und Darstellung von Rasterbildern konkretisiert, was verschiedene Einschränkungen zur Folge hatte. Die Technik bietet jedoch das Potenzial für eine umfassende Funktionalität bei der Darstellung der existierenden Vielfalt grafischer Datentypen.

Dazu werden neben Bitmaps auch Vektordaten betrachtet. Die Kontextringe können frei skaliert werden, da die Zweierpotenzen als Skalierungsfaktoren nur für die waveletbasierte Übertragung erforderlich sind. Daraus ergibt sich die Möglichkeit, die Anzahl der Kontextringe stark zu erhöhen und ihre Breite auf ein Pixel zu reduzieren, um ein allmähliches Abfallen der Skalierung vom Fokus in Richtung Bildrand zu realisieren.

Clipping-Kontextierung. Zur Erzeugung einer Darstellung mit wenigen, frei skalierten Kontextringen ist die Viewing-Pipeline eines Grafik-Subsystems (z.B. das Windows-GDI) wie folgt nutzbar: Zunächst wird für die Bounding Box jedes darzustellenden Primitivs ermittelt, mit welchen Teilrechtecken des Skalierungsgitters diese eine nicht-leere Durchschnittsmenge aufweist. Eine solche Vorverarbeitung beschleunigt die nachfolgende Ausgabe, die für jedes Teilrechteck des RECHTECKIGEN FISHEYE-VIEWS separat erfolgt. Dazu werden dem Grafiksystem zunächst die Teilrechteckkoordinaten als Clip-Rechteck mitgeteilt, und die Viewing-Transformation wird entsprechend des Skalierungspaars des Teilrechtecks gesetzt. Mit diesen Einstellungen werden alle Primitive gerendert, deren Bounding Box das aktuelle Rechteck überlappt. Abbildung B.12 zeigt ein Beispiel für die erzielbare Bildqualität. Die Rechenzeit dieses Verfahrens ist vor allem von der Anzahl der auszugebenden Primitive abhängig.

Glatte Kontextierung. Ein Kontextbereich mit vielen sehr schmalen Kontextringen, deren Skalierung vom Fokus in Richtung Bildrand allmählich abfällt, soll als *glatte Kontextierung* bezeichnet werden. Das geometrische Layout entspricht einem RECHTECKIGEN FISHEYE-VIEW mit nur einem Kontextring wie in 5.4.2.1 eingeführt. Während jedoch dort pro Teilrechteck des Kontextes die Skalierung für alle Pixel gleich ist, können hier alle Pixel eine unterschiedliche Skalierung aufweisen.

Diese nicht-uniforme Skalierung wird von den meisten Grafiksystemen nicht direkt unterstützt. Deshalb müssen in einem der Bildschirmausgabe vorangehenden Zwischenschritt die grafischen Primitive auf eine Hintergrundbitmap gerendert werden. Aus dem Gebiet des *Image Warping* sind effiziente separierbare Verfahren (wie *Fant's Resampling-Algorithmus* [Wol90, S. 153ff]) bekannt, um eine solche Bitmap mit räumlich variierender Skalierung auf eine andere Bitmap oder ein Rasterausgabegerät abzubilden.

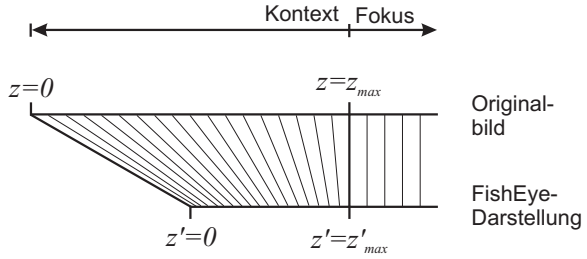


Abbildung 5.8: Funktionsprinzip der glatten Kontextierung.

Da die Algorithmen separierbar sind, erfolgt die Skalierung in zwei unabhängigen Durchläufen pro Teilrechteck, einem horizontalen und einem vertikalen. Zur Veranschaulichung der Arbeitsweise genügt es, die Pixel auf einer Strecke zwischen Bildrand und Fokusrand zu betrachten. Diese Strecke kann entweder waagerecht oder senkrecht verlaufen. Die Variable z beschreibt die Position eines Pixels auf dieser Strecke, wobei $z = 0$ immer dem Schnittpunkt der Strecke mit dem Bildrand entspricht. Abbildung 5.8 illustriert das Prinzip. Um in der Nähe des Fokus eine geringe und in der Nähe des Bildrandes eine starke Verkleinerung zu erreichen, wird eine Exponentialfunktion verwendet, wobei der Parameter ε zur Steuerung der Verzerrung genutzt werden kann. Die Position z' eines Pixels auf o.g. Strecke in der FISHEYE-Darstellung ergibt sich dann aus dessen Position z im Originalbild wie folgt:

$$z' = z^{\varepsilon} \cdot \frac{z'_{max}}{(z_{max})^{\varepsilon}} \tag{5.19}$$

Abbildung B.13 zeigt einen RECHTECKIGEN FISHEYE-VIEW der Größe 512x356 Pixel, der aus einer Hintergrundbitmap der Größe 1024x713 Pixel durch glatte Kontextierung mit $\varepsilon = 1.5$ erzeugt wurde.

Bewertung. Bei glatter Kontextierung ist der Rechenzeitbedarf höher als bei der Clipping-Kontextierung. Die Rechenzeit hängt bei ersterem Verfahren von der Größe der Hintergrundbitmap und bei letzterer Methode von der Anzahl der grafischen Primitive ab (vgl. Tabelle C.2). Unterbrechbares Zeichnen bringt bei beiden Verfahren maßgebliche Antwortzeitverkürzungen.

Im Unterschied zur skalierten Ausgabe in den meisten Grafik-Subsystemen interpoliert das Verfahren zur glatten Kontextierung beim Skalieren, was in optisch ansprechenderen Ergebnissen resultiert (vgl. Abbildungen B.13 und B.12).

5.4.4 Ergebnisse

Ein wesentlicher Vorteil des RECHTECKIGEN FISHEYE-VIEWS ist seine effiziente Übertragung und die Platz sparende Darstellung im Vergleich zum Ausgangsbild. Im Folgenden soll versucht werden, das Einsparpotenzial zu quantifizieren und es an Hand eines praktischen Übertragungsbeispiels zu veranschaulichen.

Platzeinsparung. Abhängig von der exakten Konfiguration des Skalierungsgitters kann die Platzeinsparung substanziell sein: Der RECHTECKIGE FISHEYE-VIEW in Abbildung 5.2 (rechts) erfordert nur 17.7% der Darstellungsfläche, die die Anzeige des Originalbildes in Abbildung 5.2 (links) benötigt. Die Parameter, die den Platzbedarf beeinflussen, sollen im Folgenden einzeln betrachtet werden.

Ring-Sichtbarkeitskonfigurationen. Jedem Ring ist ein Flag zugeordnet, über das dessen Sichtbarkeit ein- bzw. ausgeschaltet werden kann. Abbildung B.14(a) illustriert die Platzersparnis bei verschiedenen Sichtbarkeitskonfigurationen der Kontextringe. Der RECHTECKIGE FISHEYE-VIEW benötigt umso weniger Platz, je mehr Kontextringe mit großen Skalierungsfaktoren und je weniger Ringe mit kleinen Skalierungsfaktoren angezeigt werden.

Verhältniswerte. Abbildung B.14(b) gibt einen Eindruck von der Platzersparnis unter Verwendung verschiedener Größenverhältnisse. RECHTECKIGE FISHEYE-VIEWS, die Ringen mit hohen Skalierungsfaktoren durch größere Verhältniszahlen mehr Platz zuweisen, beanspruchen insgesamt weniger Platz.

Fokus-Zoomfaktor. Die mögliche Platzersparnis bei Verwendung verschiedener Zoomfaktoren für den Fokus illustriert Abbildung B.14(c). Darstellungen mit höheren Zoomfaktoren beanspruchen weniger Platz.

Die Einflussfaktoren „Verhältniswert“ und „Ring-Sichtbarkeit“ werden im bereits beschriebenen Algorithmus zur automatischen Anpassung der Größe des RECHTECKIGEN FISHEYE-VIEWS so berechnet, dass die Darstellung möglichst nicht größer als die verfügbare Anzeigefläche wird.

Bandbreiteneinsparung. Neben der Einsparung von Bildschirmplatz führt der Einsatz der FISHEYE-Technik auch zur Einsparung von Übertragungsbandbreite, da aufgrund der Kopplung mit dem RoI/LoD-basierten Übertragungsverfahren nur die Bilddaten übertragen werden, die auch für die Anzeige erforderlich sind, und da durch die Übertragung differenzieller Daten nach Nutzerinteraktionen keine Daten doppelt übertragen werden müssen.

Durch Nutzung des XY-unabhängigen Zerlegungsschemas (siehe 4.4.2) können die erforderlichen Skalierungsfaktoren exakt angegeben werden. Im Gegensatz dazu hätte man bei Verwendung der dyadischen Zerlegung in den Bereichen, in denen sich die Auflösungen in X- und Y-Richtung um einen Faktor größer als zwei unterscheiden, nicht benötigte Daten

Methode	Koeffizienten	%	Zeit
ganzes Bild übertragen	1048576	100%	145.6 s
RECHTECKIGER FISHEYE-VIEW, dyadische Dekomposition	393216	38%	54.6 s
RECHTECKIGER FISHEYE-VIEW, neues Dekompositionsschema	262144	25%	36.4 s

Tabelle 5.3: Übertragungszeiten verschiedener Dekompositionsalternativen für den RECHTECKIGEN FISHEYE-VIEW[10].

übertragen und die entsprechenden Bildbereiche nach der Übertragung auf das erforderliche Maß verkleinern müssen (vgl. 4.4.1). Tabelle 5.3 vergleicht die theoretischen Übertragungszeiten bei Nutzung des RECHTECKIGEN FISHEYE-VIEWS und XY-unabhängiger sowie dyadischer Dekomposition mit der Zeit für die Übertragung des ganzen Bildes. Es ist erkennbar, dass durch die Verwendung der XY-unabhängigen Dekomposition der Bandbreitenbedarf gegenüber der dyadischen Dekomposition von 38% auf 25%, also nochmals um ein Drittel, gesenkt werden konnte.

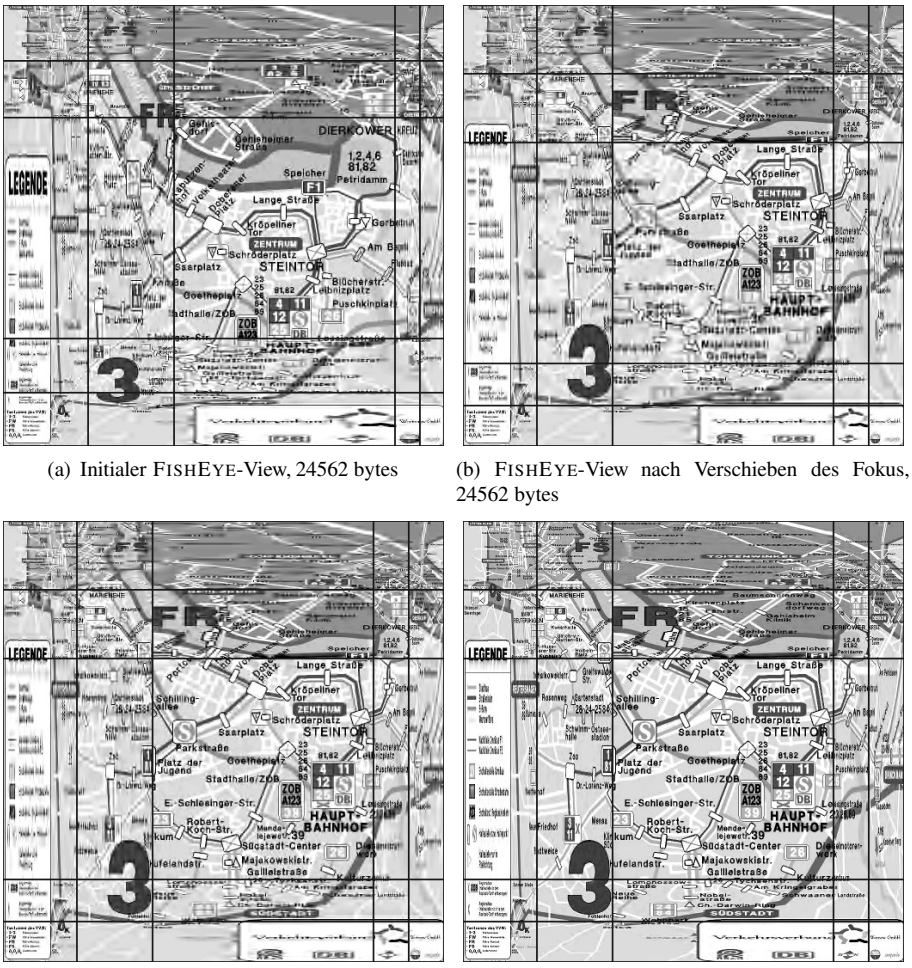
Als Beispielszenario soll nun eine simulierte Übertragung der gescannten Rostocker Nahverkehrskarte beschrieben werden. Die simulierte Bandbreite beträgt 7200 Bits/s, was etwa der Nettobandbreite einer TCP/IP-Verbindung über GSM mit 9600 Bits/s Bruttobandbreite entspricht. Bildschirmabzüge sind in Abbildung 5.9 abgedruckt.

Das Originalbild hat eine Größe von 1024x1024 Pixeln, der Fokus von 256x256 Pixeln. Zwei Kontextringe mit den Skalierungsfaktoren $Scl_1 = 2$ und $Scl_2 = 4$ umgeben den Fokus. Durch die gewählten Verhältniswerte $Ratio_1 = Ratio_2 = 1$ haben beide Kontextringe dieselbe Breite im resultierenden RECHTECKIGEN FISHEYE-VIEW, der mit einer Größe von 512x512 Pixeln nur 25% der für die Darstellung des Originalbildes notwendigen Displayfläche benötigt. Theoretisch werden durch das unterliegende Übertragungsschema auch nur 25% der Bandbreite für die Übertragung benötigt. Da die Komprimierbarkeit des Bildes lokal variiert, weichen die praktischen Ergebnisse geringfügig von diesem Wert ab.

Bevor die Übertragung gestartet werden kann, muss die initiale Position des Fokus bestimmt werden. Das kann durch die Nutzung von Kontextwissen erfolgen, in diesem Beispiel etwa durch Bestimmung der Position des Nutzers oder durch Kenntnis seiner Reisepläne. In diesem Szenario wird der Fokus im Stadtzentrum positioniert. Abbildung 5.9(a) zeigt das übertragene Bild nach 27 Sekunden. Obwohl bisher nur ein geringer Teil der Bilddaten übertragen wurde, ist die Information im Fokusbereich bereits erkennbar, und auch ein Überblick über grobe Merkmale der Umgebung ist bereits möglich. Im Vergleich dazu würde die Übertragung des Gesamtbildes mit derselben Detaillierungsstufe, die der Fokus nach 27 Sekunden erreicht hat, 115 Sekunden dauern. Somit erfordert die Übertragung mit dem RECHTECKIGEN FISHEYE-VIEW bis zu diesem Zeitpunkt nur 23.4% der Bandbreite für die Übertragung des gesamten Bildes. Würde man im Gegensatz dazu 27 Sekunden lang den Anfang des kodierten Datenstromes des Gesamtbildes über eine gleich langsame Verbindung übertragen, wäre die Darstellung vollkommen unleserlich (vgl. Abbildung 5.10).

Nach 27 Sekunden Übertragungszeit bewegt nun der Nutzer den Fokus nach links und nach unten (siehe Abbildung 5.9(b)). Die Darstellungssoftware ändert sofort das Layout der Bildschirmdarstellung auf der Basis der bereits vorliegenden Daten; der linke und der untere Teil des neuen Fokus werden zunächst aus den in niedriger Auflösung bereits verfügbaren Daten rekonstruiert. Außerdem wird eine Anfrage an den Server gesendet, die das neue Gitter spezifiziert. Danach sind nur 6 Sekunden Übertragungszeit (zuzüglich Latenzzeit)

[10] Originalbildgröße 1024x1024, Fokusregion 256x256, Größe des FishEye-Views 512x512, Breite des Rings mit halber Auflösung: 128 Pixel, Breite des Rings mit Viertelauflösung: 256 Pixel; Übertragung von 1 bpp über einen Kanal mit 7200 bps.



(a) Initialer FISHEYE-View, 24562 bytes

(b) FISHEYE-View nach Verschieben des Fokus, 24562 bytes

(c) FISHEYE-View mit neuem Fokus, 30091 bytes

(d) FISHEYE-View mit neuem Fokus, 79617 bytes

Abbildung 5.9: Übertragungssimulation mit einem RECHTECKIGEN FISHEYE-VIEW der Größe 512x512 Pixel bei einer Originalbildgröße von 1024x1024 Pixeln.

erforderlich, um durch Übertragung differenzieller Daten die neu zum Fokus hinzugekommenen Teile auf den Level of Detail des alten Fokus zu bringen (siehe Abbildung 5.9(c)). Die weitere progressive Verfeinerung erfolgt automatisch, und etwa 88 Sekunden nach dem Start der Übertragung hat die Erkennbarkeit des gesamten RECHTECKIGEN FISHEYE-VIEWS einen Stand erreicht, der auch durch Übertragung weiterer Daten nicht mehr verbessert werden kann (obwohl die Kompressionsartefakte noch reduzierbar sind). Die Übertragung des Gesamtbildes mit dem durch den Fokus zu diesem Zeitpunkt erreichten LoD würde 321 Sekunden dauern – der RECHTECKIGE FISHEYE-VIEW benötigt lediglich 27% dieser Zeit.

Mit dem RECHTECKIGEN FISHEYE-VIEW ist es somit gelungen, eine Anzeigetechnik zu entwickeln, die durch die Nutzung des Fokus-und-Kontext-Konzeptes Darstellungsfläche auf dem Bildschirm einspart und dieses Einsparpotenzial durch Anwendung des RoI/LoD-Schemas auch auf die Übertragungsbandbreite ausdehnt.

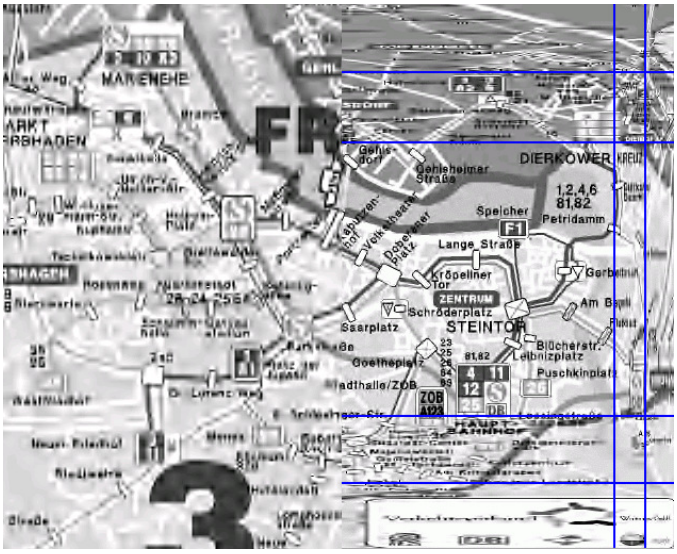


Abbildung 5.10: Qualitätsvergleich nach der Übertragung von 24562 Bytes für den RECHTECKIGEN FISHEYE-VIEW aus Abbildung 5.9(a) (rechts) und das ganze Bild (links). Die Abbildung stellt aneinander grenzende Ausschnitte aus FISHEYE-View und Gesamtbild dar.

Kapitel 6

Verfeinerung der Farbinformation von Farbtabellenbildern

6.1 Motivation und Grundidee

Bisher wurden Verfahren zur Kodierung und Übertragung von Graustufen- und Echtfarbbildern betrachtet. Farbtabellenbilder erfordern bei der progressiven Kodierung eine besondere Behandlung. Die etablierten Verfahren interlaced GIF und PNG verfeinern die räumliche Auflösung dieser Bildklasse in mehreren Durchläufen, die als Levels of Detail auffassbar sind. Bereits in Abschnitt 2.6 wurde darauf hingewiesen, dass bei diesen Verfahren fast alle LoDs übertragen werden müssen, um Texte und andere Forminformationen, wie feine Umrisse, erkennen zu können.

In Online-Diensten werden Farbtabellenbilder häufig mit Texten versehen und zur Navigation eingesetzt. Um die aus der ausschließlich räumlichen Verfeinerung resultierende lange Antwortzeit zu verkürzen, kann eine progressive Verfeinerung der Farbtiefe eingesetzt werden. Sie bietet die Möglichkeit, bereits in frühen LoDs feine Details mit hohem Kontrast zu erkennen, so dass der Nutzer beispielsweise bei einer Online-Sitzung schneller auf die gewünschte Information zugreifen kann, ohne auf die vollständige Übertragung der Bilder der Navigationsstruktur warten zu müssen.

Existierende Systeme und Methoden zur progressiven Übertragung der Farbinformation von Echtfarbbildern und zur verlustbehafteten Kompression von Farbtabellenbildern sowie die Besonderheiten von Farbtabellenbildern wurden in 2.6 diskutiert. Hier soll die im Rahmen der Dissertation entwickelte Methode MOVICOLORQ (*MoVi Color Quantized Format*, vgl. auch [Rau00a]) beschrieben werden, die eine progressive Verfeinerung der Farbinformation von Farbtabellenbildern ermöglicht. Ziele beim Entwurf der Methode waren:

1. eine *frühe Erkennbarkeit* feiner Details mit hohem Kontrast,
2. eine *hohe Kompressionsrate* sowie
3. eine *progressive Verfeinerbarkeit der Farbinformation*.

Problematisch bei Farbtabellenbildern ist, dass das Pixelfeld lediglich Indizes in eine im Allgemeinen unsortierte Farbtabelle speichert und so eine progressive, bitebenenweise Übertragung der Indizes nicht möglich ist. Vor der Übertragung müssen deshalb die fehlenden

Korrelationen zwischen Pixelwerten und Farbeigenschaften durch *Sortierung der Farbtabelle* erzeugt werden. Dieser Schritt resultiert in einem Pixelfeld, das ähnliche Farben durch ähnliche Indexwerte und Farben mit hohem Kontrast durch sehr unterschiedliche Indexwerte repräsentiert. Nach der Sortierung bietet es sich an, auf das Indexfeld ein *Vorverarbeitungsverfahren* anzuwenden, um die Interpixelredundanz zur Erreichung einer hohen Kompressionsrate ausnutzen zu können. Das derart behandelte Pixelfeld kann nun *bitebenenweise* kodiert werden, um die Verfeinerbarkeit zu realisieren. Der Übertragung jeder Bitebene muss eine Tabelle von *Mischfarben* für die jeweilige Ebene voran gestellt werden, die für jeden nach Übertragung dieser Bitebene gültigen Indexwert einen Eintrag enthält. Um eine noch frühere Erkennbarkeit des Bildes zu erreichen, sollte zusätzlich ein einfaches *Interlacing-Schema* verwendet werden.

Somit lässt sich die entwickelte Vorgehensweise in folgenden Schritten zusammenfassen:

1. Sortierung der Farbtabelle
2. Anwendung des Vorverarbeitungsverfahrens
3. Für jede Bitebene
   (a) Berechnung einer Mischfarbtabelle für alle nach Übertragung der aktuellen Bitebene möglichen Farbindizes durch Zusammenfassen von Farben aus der globalen Tabelle
   (b) Kodierung der Mischfarbtabelle für die aktuelle Bitebene
   (c) Kodierung des aktuellen Bits aller Pixelwerte; dies erfolgt bei Verwendung des Interlacing-Schemas in mehreren Durchläufen.

Nachfolgend wird dieses Verfahren in das LoD-Modell eingeordnet und auf die einzelnen Kodierungsschritte detaillierter eingegangen.

6.2 Unterstützte Dimensionen des LoD-Raumes

Levels of Detail für Farbtabellenbilder mit sortierter Farbtabelle benutzen die folgenden Dimensionen im LoD-Raum:

X-Auflösung: Diese Dimension legt die Unterabtastung in X-Richtung fest und kann durch ein Interlacing-Schema oder durch eine Quadtree-Kodierung realisiert werden.

Y-Auflösung: Analog für die Y-Richtung.

Genauigkeit: Diese Dimension legt die Genauigkeit der Farbdarstellung, also die Farbtiefe, fest. Die einzelnen Koordinatenwerte der Dimension entsprechen den Bitebenen der Koeffizienten, die einzeln übertragen werden müssen.

Farbe: Da keine getrennte Behandlung von Luminanz und Chrominanz erfolgt, hat diese Dimension einen Wertebereich der Mächtigkeit 1.

Ein Interlacing-Schema stellt eine frühere Verfügbarkeit gröber aufgelöster Daten im Übertragungsverlauf sicher. Es ist jedoch bekannt, dass Interlacing meist die Interpixelredundanz verringert und sich daher ungünstig auf die Kompressionsrate der nachfolgenden Kodierungsschritte auswirkt. Das PNG-Verfahren liefert beispielsweise für die 516 Bilder des Testsets *All*[1] ohne Interlacing um 32.3% kleinere Dateien als mit Interlacing (vgl. Tabelle 6.1 auf Seite 113). Daher ist abzuwägen, wie viele und wie große Interlacing-Schritte zur Anwendung kommen sollen.

[1] Siehe 6.4.1.

Für das in dieser Arbeit entwickelte Format wurde eine kleine Interlacing-Schrittweite gewählt, um gemäß Entwurfsziel (1) noch eine Erkennbarkeit feiner Details zu ermöglichen und um Ziel (2) zu erfüllen, also keine zu große Verschlechterung der Kompressionsrate in Kauf nehmen zu müssen. Das Interlacing-Schema besteht aus zwei Durchläufen mit einer Schrittweite von 1 in X-Richtung und von 2 in Y-Richtung. Damit reduzieren sich die verfügbaren LoD-Dimensionen auf Farbtiefe und Y-Auflösung.

Interlacing erfolgt optional, so dass der Nutzer zwischen früherer Verfügbarkeit einer vergröberten Darstellung und höherer Kompressionsrate wählen kann.

6.3 Kodierung

6.3.1 Sortierung der Farbtabelle

Für die Farbtabellensortierung wurden in der Literatur verschiedene Methoden beschrieben (vgl. 2.6.1). Leider gibt es keine Aussagen, die die Güte der einzelnen Methoden zuverlässig vergleichen. Daher wurden die Verfahren *Y-Sortierung* [ZL93] und *Closest-Pair-Sortierung* [PTC94, PT94] implementiert und für verschiedene Testbilder exemplarisch visuell verglichen.

Bei der *Y-Sortierung* werden die Farben der Farbtabelle in den YC_bC_r-Farbraum konvertiert; der Y-Anteil dient als Sortierkriterium. Da dieser Anteil die Helligkeit einer Farbe annähert, ist hierbei eine gute Erfüllung des Entwurfszieles „frühe Erkennbarkeit feiner Details mit hohem Kontrast“ zu erwarten. Das *Closest-Pair-Sortierverfahren* (im Folgenden kurz CP-Sortierung genannt) arbeitet im *RGB*-Raum und geht davon aus, dass ähnliche Farben in benachbarten Einträgen der Farbtabelle gespeichert werden sollten. Das CP-Verfahren wurde bereits in Abschnitt 2.6.1 beschrieben.

Es ergab sich, dass die Auswahl des Sortierverfahrens für optimale Ergebnisse bildabhängig getroffen werden muss. Für viele Bilder gilt jedoch, dass die Erkennbarkeit der ersten Detaillierungsstufen mit der Y-Sortierung besser ist als mit der CP-Sortierung (vgl. Abbildung 6.1 oben). Im Gegensatz dazu liefert in vielen Fällen das CP-Verfahren geringere Farbverfälschungen in späteren LoDs (vgl. Abbildung 6.1 unten). Die Bildabhängigkeit illustriert Abbildung B.25. Hier ist im Gegensatz zu Abbildung 6.1 die Erkennbarkeit in allen LoDs bei Nutzung der Y-Sortierung auf Grund des höheren Kontrastes besser als bei Verwendung des CP-Verfahrens, obwohl die dritte Detaillierungsstufe bei Verwendung von Closest-Pair die geringeren Farbverfälschungen aufweist.

Um die Stärken beider Sortiermethoden zu kombinieren, wird eine hybride Sortierung (im Folgenden als *YCP-Sortierung* bezeichnet) eingeführt. Hierbei wird die Menge der Farben eines Bildes mit n Bitebenen zunächst mit dem Y-Verfahren für die höchsten y Bitebenen vorsortiert. Für die verbleibenden $n-y$ Stufen wird die Farbtabelle dann in mehrere Teiltabellen gesplittet, die mit der CP-Methode sortiert werden. Die YCP-Sortierung vereinigt damit die Vorteile der Y- und der CP-Sortierung und ermöglicht über den Parameter y eine bildabhängige Steuerung. Als Sonderfälle sind sowohl Y-Sortierung ($y = n$) als auch CP-Sortierung ($y = 0$) möglich. Der günstigste Wert für y ist bildabhängig, die pragmatische Lösung $y = 1$ liefert jedoch in vielen Fällen zufrieden stellende Ergebnisse. Wenn im Folgenden von YCP-Sortierung die Rede ist, bezieht sich das immer auf die Parametrisierung mit $y = 1$.

Eine flexiblere Lösung wäre das interaktive Setzen von y: Dem Bildautor werden nebeneinander zwei Versionen des Bildes angezeigt, eine mit Y-Sortierung und eine mit CP-Sortierung, beginnend mit $y = n$. Durch „Weiterschalten“ von y kann der Autor beide Bilder vergleichen und entscheiden, ab welchem Bit die Farbverfälschung der Y-Sortierung unangenehmer ist als der geringere Kontrast der CP-Sortierung.

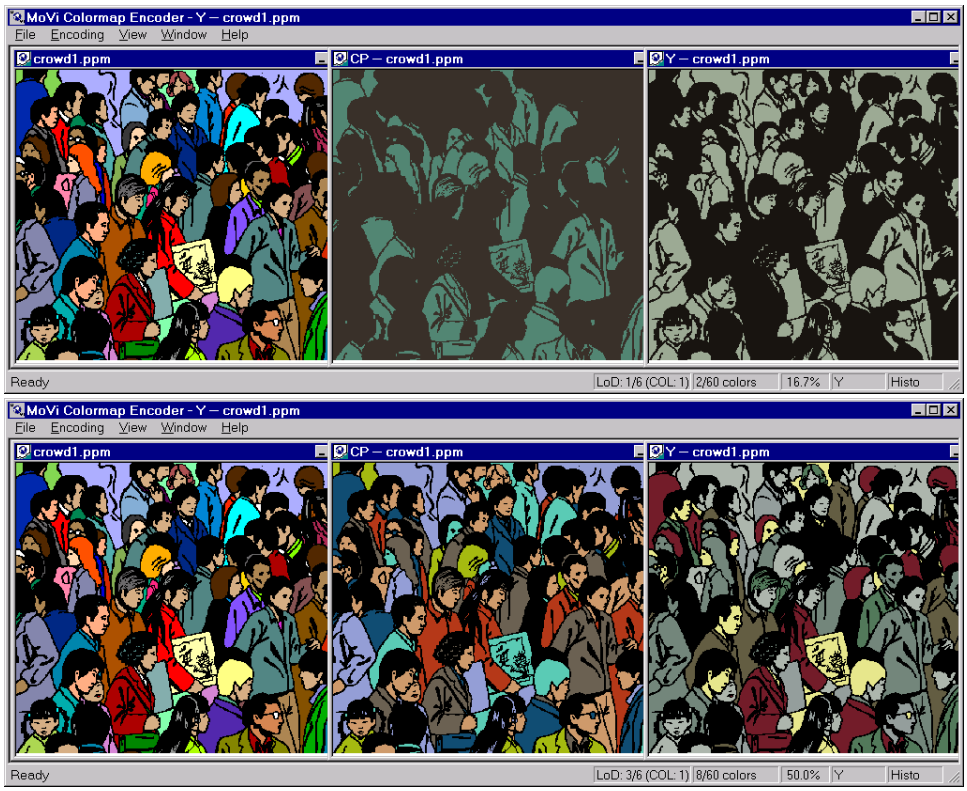


Abbildung 6.1: Vergleich der Bildqualität nach Übertragung der ersten (oben) und dritten (unten) Bitebene bei Nutzung der Y- (rechts) und der CP-Sortierung (Mitte) mit dem Original (links).

6.3.2 Vorverarbeitungsverfahren

Vorverarbeitungsverfahren können nach der Sortierung der Farbtabelle eingesetzt werden, um die Interpixelredundanz zur effektiveren Komprimierung des Pixelfeldes auszunutzen. Hierbei bieten sich zwei verschiedene Vorgehensweisen an: die Anwendung von Prädiktionsverfahren (vgl. 2.4.5.3) oder der Einsatz des BCQ-Verfahrens (vgl. 2.4.5.4). Während bei Prädiktionsverfahren viele Werte mit kleinem Betrag (idealerweise Null) erzeugt werden, kodieren Quadtrees homogene quadratische Bereiche effizient. In Voruntersuchungen wurde eine Auswahl zwischen diesen beiden Alternativen getroffen. Hierbei ergab sich für die Bildklasse „Farbtabellenbilder“ eine klare Überlegenheit der Prädiktionsverfahren: BCQ mit nachfolgender arithmetischer Kodierung erzeugte um 60% größere Dateien als eine Kombination aus Prädiktion und Golomb-Kodierung.

Im PNG-Grafikformat wird die DPCM-Prädiktion zur weiteren Effizienzsteigerung mit einer zeilenweisen Auswahl des Prädiktors aus fünf Alternativen kombiniert (siehe 2.4.5.3). Für das entwickelte Verfahren soll das Auswahlschema von PNG entlehnt werden; jedoch ist DPCM nicht für die bitebenenweise Übertragung geeignet. Da kleine Differenzen erst in späteren Durchläufen der sukzessiven Approximation kodiert werden, kann in frühen Übertragungsstufen durch das Fehlen negativer Werte mit kleinem Betrag eine Bereichsüberschreitung der Pixelwerte auftreten. Das heißt, Pixel können Indizes annehmen, die ungültige Farbtabelleneinträge referenzieren. Deshalb muss eine Prädiktionsfunktion gefunden werden, der bitebenenweise arbeitet. Hat das aktuelle Bit bei vorhersagendem und aktuellem Pixel den gleichen Wert, so soll diese Funktion für das betrachtete Bit den Wert 0 liefern, sonst den Wert 1. Der Operator XOR weist diese gewünschte Eigenschaft auf.

Zum Zweck der Anzeige des Bildes muss die Prädiktion rückgängig gemacht werden. Da die XOR-Funktion ihre eigene Umkehrfunktion darstellt, wird obiges Verfahren sowohl zur Berechnung der Prädiktion bei der Kodierung als auch zu deren Rückgängigmachen bei der Dekodierung verwendet.

Um zu entscheiden, welche der fünf Prädiktor-Alternativen zur Berechnung der Prädiktion für die aktuelle Zeile die Günstigste ist, wird zeilenweise für jede Alternative eine Prädiktion durchgeführt, die folgende bitebenenweise Kodierung der Zeile simuliert und so die Anzahl der kodierten Bits geschätzt, ohne Daten auszugeben. Aufgrund der Beschränkung auf eine Zeile liefert dieses Verfahren eine obere Abschätzung für die Bitanzahl. Für jede Zeile wird der Prädiktor ausgewählt, welcher die geringste Anzahl kodierter Bits liefert.

Falls Interlacing verwendet wird, erfolgt die Prädiktion in zwei Durchläufen, wobei im ersten Durchlauf nur die Zeilen mit gerader Y-Koordinate betrachtet werden dürfen.

6.3.3 Berechnung der Mischfarbtabelle

6.3.3.1 Zusammenfassung von Farben

Vor dem Kodieren einer Bitebene der Pixeldaten ist jeweils die Spezifikation einer gültigen Farbtabelle notwendig. Jede dieser Farbtabellen hat $m_i \leq 2^i$ Einträge, wobei $i \geq 1$ für die Nummer der aktuellen Bitebene steht, beginnend mit dem höchstwertigsten Bit als $i = 1$. Wenn ein Bild $m_{i_{max}}$ Farben enthält, weist die letzte (globale) Farbtabelle $m_{i_{max}}$ Einträge auf, und i_{max} Bitebenen müssen kodiert werden, wobei

$$i_{max} = \lceil \log_2 m_{i_{max}} \rceil. \tag{6.1}$$

Der j-te Eintrag ($0 \leq j < m_i$) der i-ten ($1 \leq i < i_{max}$) Farbtabelle wird durch Zusammenfassung der Einträge mit den Indizes $2 \cdot j$ und $2 \cdot j + 1$ der $i+1$-ten Farbtabelle erzeugt. Jede Farbe wird vor der Zusammenfassung mit der Anzahl der Pixel gewichtet, die diese Farbe aufweisen.

Falls der Index $2 \cdot j + 1$ auf einen Eintrag nach dem Ende der $i+1$-ten Farbtabelle verweist, wird der Eintrag mit Index $2 \cdot j$ unverändert aus der $i+1$-ten in die i-te Farbtabelle übernommen, da kein „Partner“ zum Zusammenfassen existiert.

Das kann bei Bildern, bei denen $m_{i_{max}}$ nur geringfügig größer ist als eine Zweierpotenz, dazu führen, dass eine Farbe in allen Farbtabellen unverändert auftritt, während alle anderen Farben gemischt werden. Dabei handelt es sich immer um die Farbe am Ende der Tabelle, also – je nach gewählter Sortierung - meist um die Hellste. Die Farbtabelle für die erste Bitebene enthält dann diese Farbe sowie das gewichtete Mittel aller anderen Farben. Falls die einzeln auftretende Farbe nur selten im Bild vorkommt oder der Mischfarbe aller anderen Farben sehr ähnlich ist, zeigt der erste LoD nur wenige, oft unwichtige Details bzw. einen sehr geringen Kontrast. Diese Problemsituation sollte automatisch erkannt und korrigiert werden.

Eine pragmatische Lösung dazu stellt die Umkehrung der Sortierfolge der Farbtabelle dar, so dass nun eine andere Farbe (die Dunkelste) unverändert übernommen wird. Dazu wird nach der Sortierung zunächst ein Kontrastmaß γ für die erste Bitebene ermittelt, dann die Sortierung umgekehrt und der damit erzielbare Kontrastwert γ_{rev} mit dem vorherigen Wert verglichen. Die Alternative, deren γ-Wert größer ist, wird verwendet. Mit dieser einfachen Vorgehensweise lassen sich in vielen Fällen zufrieden stellende Ergebnisse erzielen. Das Maß γ wird wie folgt berechnet (w_{pic} und h_{pic} repräsentieren die Dimensionen des Bildes, $P_{x,y}$ den Wert des Pixels an der Position (x,y) und h den größten Index in die globale Farbtabelle, der eine Zweierpotenz ist):

$h = 2^{i_{max}-1}$ (siehe Gleichung (6.1))
$\gamma = 0$
FOR ($x = 1(1)w_{pic} - 1$) FOR ($y = 1(1)h_{pic} - 1$)
BEGIN
  IF ($P_{x,y} < h$)
  THEN
    IF ($P_{x-1,y} \geq h$) $\gamma = \gamma + 1$
    IF ($P_{x,y-1} \geq h$) $\gamma = \gamma + 1$
  ELSE
    IF ($P_{x-1,y} < h$) $\gamma = \gamma + 1$
    IF ($P_{x,y-1} < h$) $\gamma = \gamma + 1$
  ENDIF
END

Eine weitere mögliche Variante wäre es, die am häufigsten im Bild auftretende Farbe zu suchen und diese an das Ende der Farbtabelle zu setzen. Der Nachteil hierbei wäre, dass diese Farbe aus der Sortierfolge „herausgerissen“ würde. Um dies zu mindern, kann man zusätzlich die Sortierfolge umkehren, wenn die Farbe näher am Anfang der Farbtabelle als an deren Ende steht.

Die dritte mögliche Lösung besteht darin, die Farbtabelle durch Einfügen von Fülleinträgen am Tabellenanfang derart zu manipulieren, dass h die Tabelle in zwei etwa gleich große Teile unterteilt. Dadurch entstehen die beiden Farben der ersten Mischfarbtabelle jeweils durch Zusammenfassen etwa gleich vieler Einträge. Die Anzahl der Fülleinträge m_{fill} errechnet sich wie folgt:

$$m_{fill} = \frac{2^{i_{max}} - m_{i_{max}}}{2} \tag{6.2}$$

Diese Lösung führt jedoch zu größeren Indizes im Pixelfeld und damit zu schlechterer Komprimierbarkeit der Datei.

6.3.3.2 Behandlung starker Farbverfälschungen

Die Übertragung der ersten Detaillierungsstufen eines Bildes ist möglicher Weise mit störenden starken Farbverfälschungen verbunden, obwohl die Formen im Bild bereits erkennbar sind. Für die Formerkennung ist die Farbinformation jedoch nicht notwendig. Bei inakzeptablen Verfälschungen können die Mischfarbtabellen der obersten Bitebenen daher auf Grauwerte beschränkt werden. Das Umschalten von Grauwert- zu Farbdarstellung lässt sich über einen Parameter steuern, der im Folgenden mit c bezeichnet wird. Die Mischfarbtabellen der ersten bis c-ten Bitebene enthalten Grauwerte, ab der $c+1$-ten Ebene werden dann *RGB*-Tripletts übertragen. Das Setzen von c kann bei der Bilderzeugung durch den Bildautor erfolgen, aber auch während der Übertragung durch den Bildbetrachter. Prinzipiell ist ebenso eine Automatisierung in Abhängigkeit von einem Verfälschungsmaß denkbar. Dieses Problem wäre im Rahmen weiterer Arbeiten zu untersuchen.

6.3.4 Bitebenenweise Kodierung

Die Kodierung des Pixelfeldes erfolgt bitebenenweise, um eine Verfeinerung in der Farbtiefe zu unterstützen.

Während des Prädiktionsschrittes wurden in allen Pixelwerten bitebenenweise diejenigen Bitpositionen auf 1 gesetzt, an denen sich Prädiktorwert und aktuelles Pixel unterscheiden. Durch die Interpixelredundanz tritt das höchstwertigste 1-Bit in der Regel in einer

niedrigen Bitebene auf, so dass in den oberen Bitebenen eine Häufung von 0-Bits anzutreffen ist. Für die auf das höchstwertigste 1-Bit eines Koeffizienten folgenden Bits gilt, dass 1-Bits und 0-Bits nahezu gleich verteilt sind. Zur Ausnutzung dieser unterschiedlichen Charakteristiken bietet sich somit eine getrennte Behandlung von *Signifikanz-* und *Verfeinerungsinformation* an, wie sie auch in progressive JPEG [JPGa], SPIHT [SP96b] sowie dem Zerotree-Verfahren [Sha93] verwendet wird. Signifikanzinformation teilt dem Dekoder das höchstwertigste 1-Bit eines bisher mit 0 belegten Pixels mit; Verfeinerungsinformation repräsentiert ein weiteres Bit für ein von 0 verschiedenes Pixel.

Zur Kodierung der Signifikanzinformation kommt der RLR-Enkoder (Run-Length / Rice Enkoder, siehe 2.4.3.2) zum Einsatz. Bei der Kodierung werden solche Pixel übersprungen, die bereits in früheren Bitebenen als signifikant kodiert worden sind, und deren aktuelles Bit als Verfeinerungsinformation gepuffert. Nach der Ausgabe eines Kodewortes werden die gepufferten Verfeinerungsbits der übersprungenen Pixel unkodiert an den Datenstrom angehängt. Falls Interlacing verwendet wird, erfolgt die Kodierung einer Bitebene in zwei Durchläufen, wobei zunächst nur die Zeilen mit gerader, danach nur die Zeilen mit ungerader Y-Koordinate betrachtet werden.

Ein RLR-Enkoder kann durch Beeinflussung des Lauflängen-Parameters k im Verlauf der Kodierung an die Charakteristik der Datenquelle angepasst werden. Malvar [Mal99] schlägt vor, k nach der Ausgabe eines 0-Bits um 1 zu inkrementieren und nach Ausgabe eines mit 1 beginnenden Kodewortes um 1 zu dekrementieren. Bei Verwendung zweier verschiedener Parameter k_{inc} und k_{dec} für Inkrement und Dekrement sind jedoch höhere Kompressionsraten erzielbar als bei Nutzung des von Malvar vorgeschlagenen Schemas. Die Abbildungen B.20 und B.21 zeigen die Abhängigkeit der komprimierten Dateigröße von den Parametern für das Lauflängeninkrement und –dekrement der RLR-Kodierung für die Testbilder in Abbildung B.19. Die beiden Parameter k_{inc} und k_{dec} wurden zur besseren Interpretierbarkeit der Grafik wie folgt in einen kombinierten Parameter k_{comb} „gefaltet“: $k_{comb} = 10 \cdot k_{inc} + k_{dec}$. Ein Wert von $k_{dec} = 0$ drückt dabei ein variables Dekrement aus: Die aktuelle Lauflänge wird auf den Wert gesetzt, der zur Ausgabe der aktuellen Anzahl n an 0-Bits benötigt worden wäre. Der Kurvenverlauf ist für die meisten Bilder ähnlich: Die Peaks liegen jeweils bei $k_{dec} = 1$; zwischen den Peaks ist die Kurve relativ flach. Das globale Minimum ist immer bei Werten von $k_{comb} \leq 29$ zu finden. Bei geditherten Bildern oder Bildern mit vielen Farben wie in Abbildung B.21 tritt eine stärkere Streuung zwischen den Kompressionsraten der verschiedenen Sortierverfahren auf.

Da das Minimum häufig bei $k_{comb} = 14$ liegt, werden $k_{inc} = 1$ und $k_{dec} = 4$ als Standardwerte verwendet. Statt fester Standardwerte wäre auch die Nutzung von Optimierungstechniken zum Finden eines globalen Minimums der Dateigröße möglich. Jeder Optimierungsschritt würde allerdings eine Neukodierung des Bildes mit einem anderen Wert von k_{comb} bedingen. Eine solche Optimierung bringt jedoch nur eine geringe Verbesserung (vgl. Abbildung B.15) von weniger als 1% der Dateigröße. Dem steht im Kodierprozess ein hoher Rechenaufwand für die Optimierung gegenüber.

6.4 Evaluierung

Im Folgenden sollen die mit der vorgeschlagenen Kodierungstechnik erzielbaren Ergebnisse vorgestellt und mit den Standardverfahren interlaced GIF und PNG verglichen werden. Hierzu werden als Kriterien die visuelle Qualität im Verlauf einer simulierten Übertragung sowie die erzielbare Kompressionsrate herangezogen. Außerdem wird die Abhängigkeit der Kompressionsrate von der Farbanzahl und dem Vorhandensein von Dithering-Mustern untersucht.

6.4.1 Die Testdatenbasis

Bilder verschiedener Quellen bilden die Testdatenbasis für die Evaluierung:

- Das Testset *Internet* enthält 32 Karikaturen, Screendumps, Karten und Navigationsbilder aus dem Internet.
- Die 534 Clipart-Bilder des Verzeichnisses „cartoons“ der CorelDraw8-CD-ROM bilden das Testset *Cartoons*. Dieses zerfällt in die Untermengen *Cartoons_dither* mit 117 geditherten Bildern und *Cartoons_nodither* mit 417 Bildern ohne Dithering.
- Das Testset *All* mit 516 Bildern besteht aus den Bildern der Testsets *Internet* und *Cartoons_nodither* sowie dem Inhalts des Verzeichnisses „the_east“ von der CorelDraw8-CD-ROM.

Tabelle C.3 zeigt anhand von Histogrammen die Zusammensetzung der Testsets hinsichtlich der Anzahl von Bitebenen. In den Vergleichen werden durchgängig die folgenden Abkürzungen für die Bildkodierungsverfahren verwendet: *iMCQ* für interlaced MoviColorQ, *MCQ* für MoviColorQ ohne Interlacing, *iGIF* für interlaced GIF, *iPNG* für interlaced PNG sowie *PNG* für PNG ohne Interlacing.

6.4.2 Visuelle Vergleiche

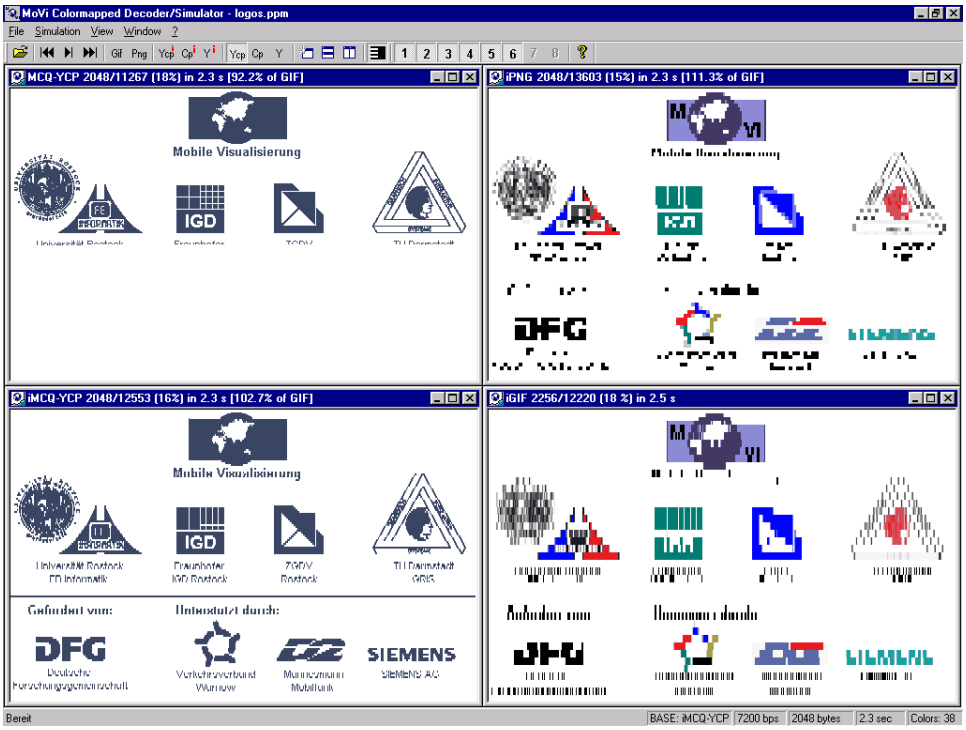
Zur visuellen Qualität der Bilder kann festgestellt werden, dass das Entwurfsziel einer frühen Erkennbarkeit feiner Details erreicht wurde. Prinzipiell lassen sich folgende Aussagen machen:

- Feine Details mit hohem Kontrast sind bei MoviColorQ bedeutend früher im Übertragungsverlauf erkennbar als bei den etablierten Standardverfahren iGIF und iPNG.
- Das Interlacing-Schema in iMCQ bringt bei der Übertragung der ersten Bitebene Vorteile, da schneller erste Informationen über alle Pixel im Bild vorliegen.
- Features mit großer räumlicher Ausdehnung und geringem Kontrast sind bei MoviColorQ später erkennbar als bei iGIF und iPNG.
- In den ersten Übertragungsstufen können starke Farbverfälschungen auftreten, falls hier nicht Grauwerte genutzt werden.

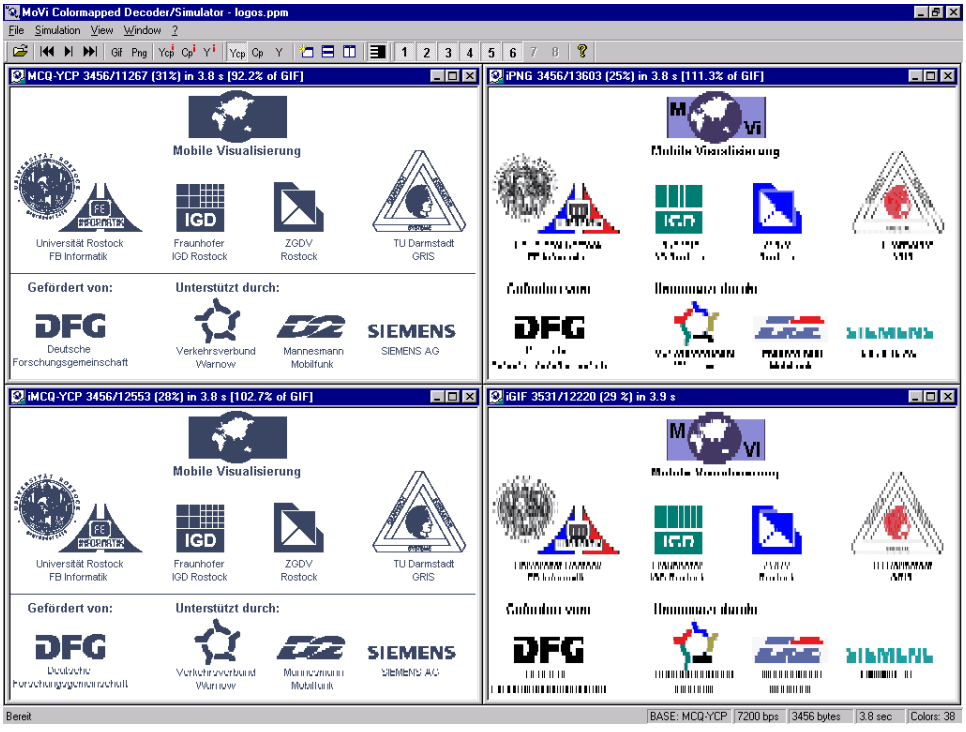
Abbildung 6.2 illustriert die ersten beiden Aussagen. Weitere Vergleichsbilder befinden sich in Anhang B.9.4. Vier Bildformate werden anhand einer simulieren Übertragung mit 7200 bps verglichen: MCQ (oben links), iMCQ (unten links), iPNG (oben rechts) und iGIF (unten rechts). In den Titelbalken der Fenster ist jeweils die bereits übertragene Dateigröße, die Gesamtgröße der komprimierten Datei, der bereits übertragene prozentuale Anteil, die simulierte Übertragungszeit und die Gesamtdateigröße im Verhältnis zu iGIF ablesbar.

Bei Nutzung von iMCQ sind bereits nach 2.3 s (Abbildung 6.2(a)) gröber aufgelöste Daten für die erste Bitebene aller Pixel verfügbar, so dass die Form der Logos bereits erkennbar ist. Im Gegensatz dazu erlauben iGIF und iPNG zu diesem Übertragungszeitpunkt noch keine Formerkennung. Nach 3.8 s ist die erste Bitebene in MCQ vollständig übertragen. Hier sind alle Texte bereits lesbar. In den iGIF- und iPNG-Pendants sind zwar einige der Formen zu erkennen, jedoch noch immer keine Texte lesbar. Dies erfordert im Fall des iGIF-Bildes eine Übertragungszeit von 13.6 s.

Abbildung 6.3 zeigt die Farbverfälschungen, die in frühen Übertragungsstufen durch MoviColorQ verursacht werden können. Dabei wird das Bild aus Abbildung 6.2 als MCQ (oben links), MCQ mit Graustufenfarbtabelle (unten links) sowie iGIF (oben rechts) nach einer Übertragungszeit von 6.8 s dem Originalbild (unten rechts) gegenübergestellt.



(a) Übertragung von 2048 Bytes: Erste Bitebene ist vollständig bei interlaced MOVICOLORQ



(b) Übertragung von 3456 Bytes: Erste Bitebene ist vollständig bei non-interlaced MOVICOLORQ

Abbildung 6.2: Übertragungssimulation einer Navigationsbitmap mit vier Techniken: MOVICOLORQ (iMCQ und MCQ), iPNG sowie iGIF.

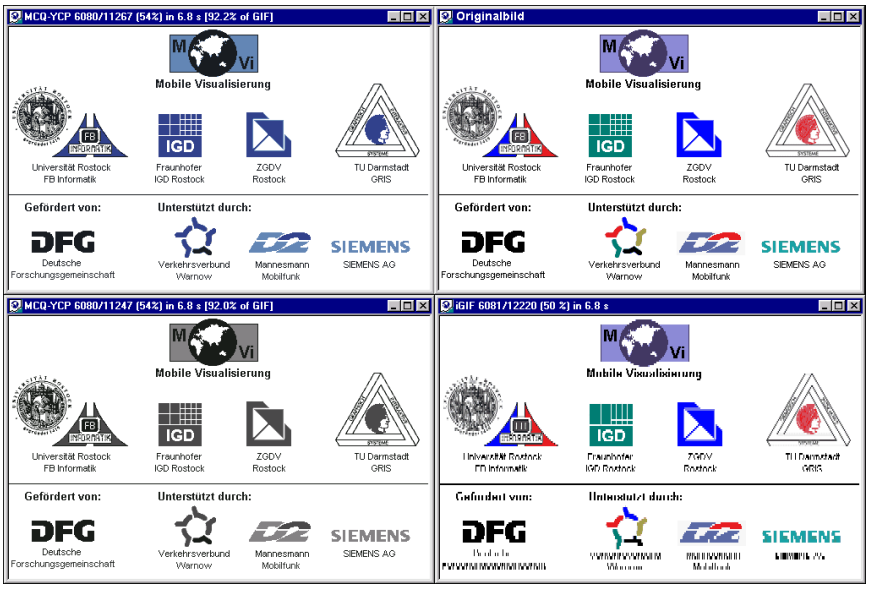


Abbildung 6.3: Farbverfälschungen in frühen Übertragungsstufen bei MOVICOLORQ. Oben links: MCQ mit RGB-Farbtabelle, Unten links: MCQ mit Graustufenfarbtabelle, Oben rechts: Originalbild, Unten rechts: iGIF.

Auch iMCQ-Bilder mit relativ wenigen Bitebenen (vgl. Abbildung B.28 mit 4 Ebenen) zeigen bereits nach Übertragung eines Interlacing-Passes der ersten Bitebene erkennbare Konturen. Geringe Farbunterschiede mit größerer räumlicher Ausdehnung (wie das unterlegte Straßennetz in Abbildung B.29) werden jedoch bei iGIF und iPNG früher im Übertragungsverlauf sichtbar.

Die Konturen in Bildern mit glatten Farbübergängen sind mit MOVICOLORQ ebenfalls eher zu erkennen als mit iPNG und iGIF, wie die 256-Farben-Version des populären Lena-Testbildes in Abbildung B.26 illustriert. In späteren Stufen der Übertragung ist die Erkennbarkeit solcher Bilder bei iPNG und iGIF besser, da hier keine Farbverfälschungen auftreten (vgl. Abbildung B.27). Für dieses Bild wäre allerdings das verlustbehaftete JPEG das am besten geeignete Kompressionsverfahren (vgl. Anhang B.1).

Bilder mit Farbverläufen als Hintergrund liefern in der ersten MOVICOLORQ-Übertragungsstufe (vgl. Abbildung B.33) einen einfarbigen Hintergrund. Feine Details sind auch hier eher erkennbar als bei iGIF und iPNG. In weiteren Übertragungsstufen wird bei MOVICOLORQ-kodierten Bildern mit Farbverläufen eine Bänderstruktur sichtbar (siehe Abbildung B.34).

6.4.3 Kompressionsraten

Obwohl eine frühe Erkennbarkeit feiner Details im Verlauf der Bildübertragung eine nützliche Eigenschaft ist, muss sich ein neues Bildkodierungsverfahren hinsichtlich der Kompressionsrate an existierenden Methoden messen und hier mindestens eine ähnliche Leistungsfähigkeit bieten. Um die Kompressionsleistung des MOVICOLORQ-Verfahrens im Vergleich zu iGIF und iPNG zu ermitteln, wurden Experimente mit den in 6.4.1 beschriebenen Testsets durchgeführt. Tabelle 6.1 listet die Summe der komprimierten Dateigrößen der Bilder des Testsets *All* für MOVICOLORQ mit und ohne Interlacing auf und stellt sie den mit iGIF, iPNG und PNG erreichbaren Größen gegenüber.

Methode	iMCQ	MCQ	iGIF	iPNG	PNG
Gesamt-Dateigröße [Bytes]	2198307	1877666	2230259	2750039	1859936
Größe im Vergleich zu iGIF	98.6%	84.2%	100.0%	123.3%	83.4%
Größe im Vergleich zu iPNG	79.9%	68.3%	81.1%	100.0%	67.7%

Tabelle 6.1: Vergleich der komprimierten Gesamt-Dateigröße für verschiedene Kompressionsmethoden. Testset: *All*.

Die folgenden Aussagen können für die verwendeten Bilder getroffen werden:

- Im Vergleich zu iPNG komprimiert die neue Methode generell besser, MCQ liefert 68.3% und iMCQ 79.9% der mit iPNG erreichbaren Dateigröße.
- Die Kompressionsraten von interlaced MoviColorQ und iGIF sind ähnlich.
- Ohne Interlacing komprimiert MoviColorQ etwa 15% besser als iGIF.
- Noninterlaced PNG ist eigentlich nicht mit MoviColorQ vergleichbar, da es keine progressive Verfeinerung gestattet und so effizienter komprimieren kann als interlaced PNG. Trotzdem bietet MoviColorQ ohne Interlacing eine ähnliche Kompressionsrate wie noninterlaced PNG und ermöglicht zusätzlich eine progressive Verfeinerung

Bei allen Testsets hat das gewählte Sortierverfahren nur einen marginalen Einfluss auf die Kompressionsrate (siehe Diagramm B.15).

Negativ wirkt sich die Nutzung des Interlacing-Schemas in MoviColorQ auf die Kompressionseffizienz aus, wie die Ergebnisse in den Tabellen 6.1, C.4 und C.5 zeigen. Es kann festgestellt werden, dass der Verzicht auf Interlacing für die drei Testdatensätze eine um 9 bis 15% geringere Dateigröße liefert. Die schlechtere Kompression mit Interlacing ist auch anhand der visuellen Qualität in späteren Bitebenen erkennbar. Hier liefert eine Übertragung mit Interlacing die verfeinerte Informationen für alle Pixel später als ohne Interlacing. Abbildung B.29 illustriert diesen Fakt: Die zweite Bitebene des noninterlaced-MoviColorQ-Bildes (oben links) wurde bereits vollständig übertragen, während im interlaced-MoviColorQ-Bild der zweite Durchlauf durch diese Bitebene noch nicht abgeschlossen ist (unten links). Dies wird in Form von Streifen in dem Bildbereich sichtbar, der im zweiten Interlacing-Pass noch nicht traversiert wurde.

Der Farbverlauf in den Abbildungen B.33 bis B.35 wird durch das Prädiktionsschema von MoviColorQ und PNG bedeutend besser als von GIF komprimiert. Erstaunlicher Weise wirkt sich in diesem Fall das Interlacing bei MoviColorQ günstig aus – die Kompressionsrate ist mit Interlacing höher als ohne. Abbildung B.35 zeigt einen Vergleich der Bildqualität nach Abschluss der Übertragung einer Datenmenge, die der kleinsten erreichbaren Dateigröße (iMCQ) entspricht.

6.4.4 Einfluss von Farbanzahl und Dithering auf die Kompressionsrate

Die Testsets zur Ermittlung der Kompressionsrate enthalten im Einklang mit dem intendierten Einsatzgebiet des MoviColorQ-Verfahrens vor allem Bilder mit wenigen Farben. Für Bilder mit steigender Farbanzahl sinkt die Kompressionsrate. Die folgenden Experimente sollen diesen Zusammenhang quantifizieren.

Für einen ersten Test wurde das Testset *All* in Klassen von Bildern mit gleicher Bitebenenanzahl zerlegt. Tabelle C.3 zeigt die Anzahl der Bilder je Klasse. Für jede Klasse wurde die Gesamtgröße der komprimierten Dateien für die Verfahren MCQ, iMCQ, iGIF, iPNG und PNG ermittelt. Die Diagramme B.16, B.17 und B.18 im Anhang stellen die Ergebnisse

grafisch dar. Für Bilder mit wenigen Farben komprimieren die MOVICOLORQ-Verfahren besser als iGIF und iPNG; für Bilder mit vielen Farben ist die Kompressionsrate vergleichbar.

In einem zweiten Test wurde der Einfluss der Farbanzahl auf die Komprimierbarkeit eines einzelnen Bildes geprüft. Dazu wurde aus einem Stadtplan mit 18 Farben (für einen Ausschnitt siehe Abbildung B.19 links) durch Glättung und Verkleinerung ein Bild mit 244 Farben erzeugt (Abbildung B.19 rechts). Dieses Bild wurde dann mit dem Programm `ppmquant` aus der netPBM-Distribution [P+] auf verschiedene Farbanzahlen farbquantisiert und nachfolgend mit MCQ, iMCQ, iGIF und iPNG komprimiert. Die Diagramme B.22 bis B.24 im Anhang zeigen die komprimierte Dateigröße in Abhängigkeit von der Farbanzahl für alle getesteten Farben und zur besseren Erkennbarkeit zusätzlich für die untersten 64 Farben sowie bitebenenweise. Die Ergebnisse können wie folgt interpretiert werden:

- Die MOVICOLORQ-Varianten komprimieren das Bild bei hoher Farbanzahl schlechter, bei niedriger Farbanzahl besser als iGIF und iPNG.
- Damit wird deutlich, dass die komprimierte Dateigröße bei MOVICOLORQ schneller mit sinkender Farbanzahl fällt (bzw. die Kompressionsrate steigt) als bei iGIF und iPNG. Interessant ist, dass stärkere Sprünge in der Kompressionsrate zwar in der Nähe, jedoch nicht immer exakt an der Position von Bitebenenwechseln auftreten (z.B. in Abbildung B.22 bei 130, 65, 32, 16).
- Obwohl iGIF und iPNG das Bild bei hoher Farbanzahl besser komprimieren als MOVICOLORQ, gibt es keine auffälligen visuellen Unterschiede zwischen dem vollständig übertragenen iGIF-Bild und dem bis dahin mit gleicher Dateigröße teilweise übertragenen MOVICOLORQ-Bild (Abbildung B.32).

	Methode	iMCQ	MCQ	iGIF	iPNG	PNG
Cartoons dither	Gesamtgröße [Bytes]	818137	755946	653888	811193	610420
	Größenvergleich: iGIF	125.1%	115.6%	100.0%	124.1%	93.4%
	Größenvergleich: iPNG	100.9%	93.2%	80.6%	100.0%	75.2%
Cartoons nodither	Gesamtgröße [Bytes]	877510	745447	866549	1221850	812085
	Größenvergleich: iGIF	101.3%	86.0%	100.0%	141.0%	93.7%
	Größenvergleich: iPNG	71.8%	61.0%	70.9%	100.0%	66.5%

Tabelle 6.2: Der Einfluss von Dithering-Mustern im Bild auf die Kompressionsleistung.

Einen weiteren negativen Einfluss hat das Vorhandensein von Dithering-Mustern im Bild. Das Testset *Cartoons* wurde in eine Gruppe von Bildern mit Dithering-Mustern und eine andere Gruppe ohne solche Muster unterteilt. Die erreichte Kompression ist in Tabelle 6.2 dargestellt. Es ist ersichtlich, dass Dithering-Muster bei den MOVICOLORQ-Varianten zu einem Anstieg der komprimierten Dateigröße im Vergleich zu iGIF, PNG und iPNG führen.

Diese Ergebnisse legen nahe, auf die Nutzung von Dithering-Verfahren zur Senkung der Farbanzahl zu verzichten.

6.5 Verbesserungsmöglichkeiten

Möglichkeiten zur Verbesserung der Methode MOVICOLORQ wurden teilweise bereits in den entsprechenden Abschnitten erwähnt. Sie sollen hier noch einmal kompakt zusammengefasst werden.

Es hat sich gezeigt, dass der Einsatz von Interlacing zu einer schnelleren Erkennbarkeit in der ersten Bitebene des Bildes führt. In den folgenden Bitebenen verringert es jedoch

die Kompressionsrate. Daher sollte in einer verbesserten Version von MOVICOLORQ das Interlacing-Schema nur für die erste Bitebene eingesetzt werden.

Die Empfindlichkeit gegenüber Dithering-Mustern legt eine weitere Verbesserungsmöglichkeit nahe. Dithering verkürzt die Folgen von 0-Bits durch Erzeugen künstlicher Muster im Bild, was sich ungünstig auf die Kompressionsrate der RLR-Kodierung auswirkt. Es ist zu vermuten, dass Methoden, die Muster von Symbolen effizient kodieren (z.B. LZ77), in diesem Fall eine Verbesserung bringen. Um erste Hinweise über die Gültigkeit dieser Vermutung zu erhalten, wurden je zwei geditherte und zwei ungeditherte Bilder betrachtet. Für Signifikanz- und Verfeinerungsinformation wurden jeweils drei Alternativen der Kodierung eingesetzt:

- RAW: Ausgabe ohne Kodierung (in MOVICOLORQ für Verfeinerungsinformation genutzt),
- RLR: Golomb-Kodierung (in MOVICOLORQ für Signifikanzinformation genutzt),
- GZ: `gzip`-Kodierung (LZ77 kombiniert mit Huffman-Kodierung).

Für die ungeditherten Bilder erwies sich die bei MOVICOLORQ getroffene statische Zuordnung (RLR-Kodierung der Signifikanz- und unkodierte Ausgabe der Verfeinerungsinformation) als nahezu optimal. Die dynamische Wahl einer der drei Kodierungsalternativen für jeden Pass und jede Bitebene brachte nur eine geringe Verbesserung der Kompressionsrate (siehe Tabelle C.6). Bei der Kodierung von Bildern mit Dithering-Mustern ermöglichte die dynamische Auswahl eine Steigerung der Kompressionsrate von 10.8% bzw. 23.1% (siehe Tabelle C.7). Dies wurde durch Nutzung der GZ-Kodierung für alle Verfeinerungs- und die erste Signifikanz-Bitebene sowie der RLR-Kodierung für die verbleibenden Signifikanz-Bitebenen erreicht. Weiter führende Experimente müssen diesen Zusammenhang für eine größere Anzahl von Bildern bestätigen.

Bei der Erzeugung von MOVICOLORQ-Dateien werden die Parameter y zur Umschaltung von Y- auf CP-Sortierung und c zur Umschaltung von Graustufen- auf Farbdarstellung bisher manuell durch den Bildautor gesetzt. In weiter führenden Arbeiten sollte ein Mechanismus entwickelt werden, der diese Parameter in Abhängigkeit von Verzerrungsmaßen automatisch mit Werten belegt.

Kapitel 7

Schlussbetrachtungen

7.1 Zusammenfassung der Ergebnisse

Diese Arbeit befasst sich mit der Problematik der Bildübertragung unter den Bedingungen eingeschränkter Ressourcen in mobilen Umgebungen. Existierende Methoden der adaptiven Bildübertragung ermöglichen durch Steuerung der Kodierung die Anpassung von Größe und Bandbreitenbedarf eines Bildes vor der Übertragung an die aktuelle Ressourcensituation, den Kontext. Hierbei ergeben sich aus der Ressourcenknappheit mobiler Umgebungen vor allem dann Probleme, wenn zur Bildübertragung Standardverfahren wie z.B. JPEG oder GIF eingesetzt werden. Die Realisierung eines adaptiven Bildservers[1] [Rau97] zeigte diese Defizite auf: Die Steuerung der kodierten Dateigröße arbeitet relativ ungenau, und ein geänderter Bedarf des Nutzers bedingt eine erneute Kodierung und die redundante Übertragung des Bildes.

Eine Steuerung während laufender Bildübertragung gemäß dem Bedarf des Nutzers offeriert daher zusätzliche Einsparungspotenziale für die knappen Ressourcen *Übertragungsbandbreite* und *Bildschirmplatz*. Dazu sollte für jede Region im Bild die erforderliche Detaillierungsstufe vor und während der Übertragung definierbar sein.

Es wurde ein formales Modell zur Beschreibung des Übertragungsbedarfes mittels RoIs und LoDs und zur Realisierung einer redundanzfreien Verfeinerung entwickelt und dessen Umsetzbarkeit auf der Basis verschiedener Bildkodierungsverfahren geprüft. Realisierungen erfolgten für Graustufenbilder auf der Basis von Wavelets durch Entwicklung eines flexiblen Dekompositionsschemas und Erweiterung des Embedded-Zerotree-Verfahrens sowie für Farbtabellenbilder in Form einer neuen progressiven Kodierungsmethode.

Letztere ermöglicht die progressive Verfeinerung in der Dimension „Farbtiefe“, die im Gegensatz zu interlaced GIF und PNG, den etablierten Kodierverfahren für Farbtabellenbilder, eine frühzeitige Erkennbarkeit feiner Details während der Übertragung ermöglicht und damit vorteilhaft für Navigationsbilder in Online-Diensten eingesetzt werden kann. Die mit der neuen Methode erreichbaren Kompressionsraten sind mit denen von interlaced GIF und PNG vergleichbar bzw. ihnen überlegen.

Auf der Grundlage des entwickelten Wavelet-Verfahrens wurden mehrere Anwendungen realisiert und evaluiert. Dabei wurde Wert auf die Unterstützung der beiden Benutzerrollen „Bildautor“ und „Bildbetrachter“ gelegt, so dass die Angabe des Bedarfs sowohl vor Beginn als auch im Verlauf einer Übertragung möglich ist. Die entwickelte kombinier-

[1] Der Bildserver ist unter `http://www.informatik.uni-rostock.de/Projekte/movi/IIS/` erreichbar.

te Übertragungs- und Darstellungstechnik „RECHTECKIGER FISHEYE-VIEW“ ermöglicht durch Kombination einer Fokusregion mit verkleinerten Kontextbereichen die Platz sparende Darstellung großer Bilder auf kleinen Displays und deren effiziente Übertragung über schmalbandige Netze.

7.2 Ausblick

Mögliche Erweiterungen und Optimierungen der Verfahren im Detail wurden zum großen Teil bereits in den entsprechenden Abschnitten diskutiert. Hier sollen die Verbesserungsmöglichkeiten noch einmal zusammengefasst werden. Die vorauszusehende Verbreitung von JPEG2000 motiviert darüber hinaus weiter führende Forschungsarbeiten zur Umsetzung von RoIs und LoDs auf der Grundlage dieses neuen Standards, die kurz skizziert werden sollen.

Obwohl im RoI/LoD-Modell sowohl Echtfarbbilder als auch polygonale RoIs berücksichtigt werden, steht deren implementierungstechnische Unterstützung in LIBROILOD noch aus. Performancemessungen ergaben weiterhin, dass für die Übertragung größerer Bilder bzw. RoIs die Übertragungsschritte teilbar gestaltet werden sollten, um eine kürzere Latenzzeit bei der Übertragungssteuerung zu erreichen.

Verbesserungsmöglichkeiten der Farbtabellenmethode MOVICOLORQ wurden bereits in Abschnitt 6.5 dargelegt. Da MOVICOLORQ seine Stärken vor allem in den ersten Bitebenen während der Übertragung ausspielt, wäre darüber hinausgehend eine hybride Methode aus MOVICOLORQ und z.B. PNG denkbar. Dabei sollten die höchstwertigsten Bits in MOVICOLORQ kodiert und die verbleibenden Bitebenen als Differenzbild in PNG übertragen werden. Für die praktische Nutzung des MOVICOLORQ-Formates ist es erforderlich, dieses in einer breiten Palette von WWW-Browsern verfügbar zu machen. Dies kann durch eine Java-Implementierung geschehen.

Der neue Standard JPEG2000 enthält in seinem Baseline-Modus [JPG99a] nur rudimentäre RoI-Unterstützung in Form der *MaxShift*-Methode (vgl. 2.5.2.2). Diese ist vor allem für die Archivierung medizinischer Bilder gedacht, aber zur interaktiven Steuerung der Bildübertragung in mobilen Umgebungen nicht geeignet. Dafür ermöglicht die Struktur eines JPEG2000-Datenstromes dessen Analyse ohne Dekodierung. Durch das Layer-Konzept des Datenstromes und die Aufteilung des Bildes in separat kodierte Blöcke ist die Transkodierung eines JPEG2000-Bildes möglich, um eine RoI- und LoD-Unterstützung zu realisieren. Dabei kann die Zugehörigkeit von Wavelet-Koeffizienten zu partitionierenden Teilmengen – im Gegensatz zu der spannenbasierten Realisierung in dieser Arbeit – nicht pro Spanne, sondern nur noch pro Block von Koeffizienten angegeben werden. Diese Zuordnung führt zu einer gröberen Behandlung der RoI-Geometrie, bei der in der Regel die Unterstützung rechteckiger RoIs ausreichend sein wird. Dies erhöht den Bandbreitenbedarf, verringert jedoch die Belastung der Serverkomponente, da keine Neukodierung des Bildes erforderlich ist. Im Extremfall kann möglicherweise ganz auf eine dedizierte RoI/LoD-Serversoftware verzichtet werden, wenn das verwendete Netzprotokoll einen wahlfreien Zugriff auf Bereiche der zu übertragenden JPEG2000-Datei erlaubt. Eine solche Realisierung wird sicherlich hinsichtlich der Latenzzeit nicht optimal sein. Sie gestattet es jedoch beim Vorhandensein einer geeigneten Client-Software, JPEG2000-Bilder von beliebigen Servern im Netz RoI- und LoD-basiert zu übertragen, ohne dass Änderungen an der Server-Software erforderlich sind.

Anhang A

Publikationen zum Thema RoI

Quelle	Basis-Algorithmus	RoI-Form	RoI-Signalisierung	Speicher-Overhead	Verfeinerungsdimensionen	Priorisierung	Interaktiv	multipel	überlappend	Kompression
[HJWJG83]	Knowlton	rechteckig	2 Punkte	◇	Auflösung	exklusiv	✓	✓	?	-
[Loh84]	DCT	rechteckig	2 Punkte	□	Qualität	exklusiv	✓	-	-	-
[Sha96a]	EZW	beliebig	?	◇	Pixelamplitude	Amplitudenverhältnis	-	?	-	✓
[FSZ97]	SPIHT	kreisförmig	Mittelpunkt + Wichtigkeitsfunktion	◯	Genauigkeit	Bitebenen	✓	✓	✓	✓
[AF98]	SPIHT	?	?	?	Genauigkeit	Bitebenen	✓	-	-	✓
LuRaWave [LWF]	LuRa-Wave	rechteckig	?	◇	Qualität	Qualitätsunterschied	-	✓	✓	✓
JPEG Teil III	progressive JPEG	rechteckig	Eckpunkt, Breite, Höhe	◇	Spektralbänder, Genauigkeit, Farbkomponenten	implementierbar	✠	✓	-	✓
JPEG2000: MaxShift	EBCOT	beliebig	Bit-Offset	◇	Bitrate, Farbkomponenten	-	-	-	-	✓
JPEG2000: Transcoding	EBCOT	Blöcke	nicht erforderlich	◇	Auflösung, Genauigkeit, Farbkomponenten	implementierbar	✠	✠	✠	✓
JPEG2000: Skalierung	EBCOT	rechteckig / beliebig	2 Punkte / Bitmaske	? / ◯	Genauigkeit, Farbkomponenten	Bitebene	?	✓	✓	✓
[CY97], [CYY97]	Wavelets	kreisförmig / rechteckig	Mittelpunkt/ 2 Punkte	◯	Auflösung	exklusiv	✓	✓	✓	-
[Duc97]	Wavelets	kreisförmig	Mittelpunkt	◯	Auflösung	-	-	✓	✓	-

Legende: ✓ vorhanden ✠ realisierbar - nicht vorhanden ? keine Angabe ◇ niedrig □ mittel ◯ hoch

Quelle	Basis-Algorithmus	RoI-Form	RoI-Signalisierung	Speicher-Overhead	Verfeinerungs-dimensionen	Priorisierung	interaktiv	multipel	überlappend	Kompression
[RR96]	MPEG 2	rechteckig, elliptisch	Mittelpunkt	◇	Auflösung, Framerate	Bitrate	✓	-	-	✓
[KCKH95]	DCT	beliebig	Bitmaske	◯	Kompressions-verlust ja/nein	-	-	✓	✓	✓
[SL97]	EZW	elliptisch mit Randabfall	Mittelpunkt, Radius, Exzentrizität, Randabfall	◇	Genauigkeit	Skalierung	-	✓	?	✓
[SWL97]	Wavelet Packets + CQT	kreisförmig	Punkt, Radius	◇	Genauigkeit	?	-	✓	?	✓
[YLLC97]	Wavelets + GSTP	?	?	?	Genauigkeit	-	?	✓	✓	✓
[NC98]	S+P, SPIHT	beliebig	Bitmaske	◯	Kompressions-verlust ja/nein	Bitrate für Hintergrund	-	-	-	✓
Ziel	Wavelets, eingebettet	beliebig	geometrisch	◇	Auflösung, Genauigkeit, Farbkomponenten	mehrstufig	✓	✓	✓	✓

Legende: ✓ vorhanden ✠ realisierbar - nicht vorhanden ? keine Angabe ◇ niedrig □ mittel ◯ hoch

Anhang B

Bilder und Diagramme

B.1 Abhängigkeit der Kompressionsrate von der Farbtiefe bei Nutzung von Standardverfahren

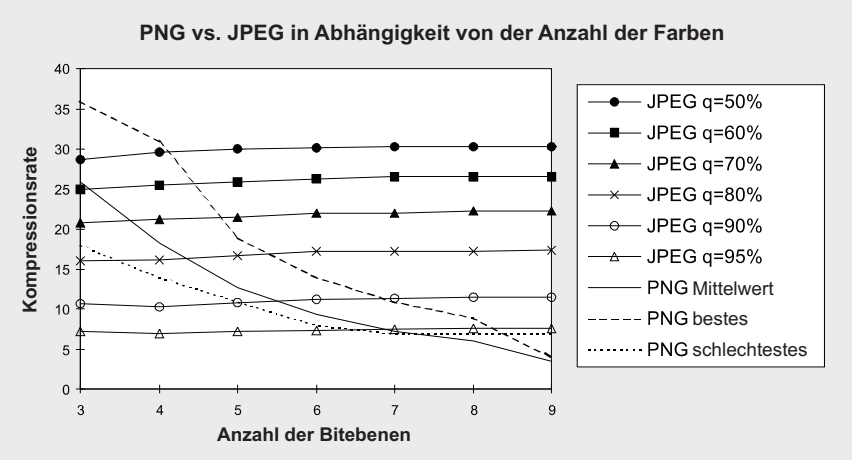
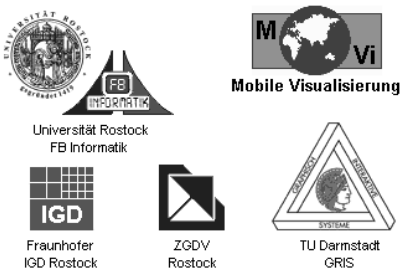


Abbildung B.1: Kompressionsrate von JPEG und PNG für Bilder unterschiedlicher Farbtiefe.

Eine verbreitete Meinung ist, dass der verlustbehaftete JPEG-Algorithmus in jedem Fall eine höhere Kompression erreicht als die verlustfreien Methoden GIF und PNG. Das ist jedoch falsch. JPEG wurde für fotorealistische Bilder mit kontinuierlichen Farbverläufen entwickelt und komprimiert diese am besten. Bilder (insbesondere Strichzeichnungen) mit wenigen Farben können oft mit verlustfreien Methoden stärker als mit JPEG komprimiert werden. Das soll an Hand eines Tests belegt werden: Sechs Echtfarbbilder wurden auf eine Farbanzahl von 2^2 bis 2^9 (2 bis 9 Bitebenen) quantisiert und mit PNG sowie mit JPEG bei unterschiedlichen Qualitätseinstellungen komprimiert. Die Kurven in Abbildung B.1 zeigen die durchschnittliche Kompressionsrate für die sechs Bilder. Neben den Durchschnittskurven sind zwei Kurven für das am besten und das am schlechtesten mit PNG komprimierbare Bild eingezeichnet. Es ist zu erkennen, dass die JPEG-Kompressionsrate unabhängig von der Anzahl der Farben ist. Für Bilder mit vielen Farben ist JPEG am besten geeignet. PNG komprimiert Bilder mit wenigen Farben besser als JPEG.

B.2 Graustufen-Testbilder

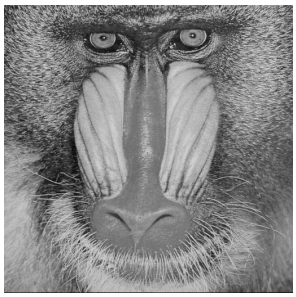


(a) logos

(b) lena

(c) barbara

(d) boat



(e) mandrill

(f) goldhill

(g) peppers

Abbildung B.2: Graustufen-Testbilder. Das Bild „logos“ist 496x309 Pixel, alle weiteren Bilder sind 512x512 Pixel groß.

B.3 Qualitätsvergleiche komprimierter Bilder

B.3.1 Spektrale Selektion vs. Sukzessive Approximation

Abbildung B.3: Vergleich von spektraler Selektion allein (links, 0.276 bpp, DC, AC1, AC2) mit spektraler Selektion und sukzessiver Approximation (rechts, 0.275 bpp, DC Bits 7-1, AC1-AC5 Bits 7-2).

B.3.2 JPEG vs. Wavelets

Die JPEG-Bilder der folgenden Abbildungen wurden mit dem JPEG-Codec der Independent JPEG Group, Version 6a, im progressiven JPEG-Modus und mit optimierten Huffman-Tabellen erzeugt. Die Wavelet-Bilder wurden mit dem MOVIWAVE-Codec mit folgenden Parametern generiert: 5/3-Wavelet, 5 Zerlegungsstufen, 1 unabhängige Zerlegungssstufe.

Abbildung B.4: Vergleich der mit JPEG- (links) und Wavelet-Kodierung (rechts) erzielbaren Bildqualität bei 0.05bpp (1:160).

Abbildung B.5: Vergleich der mit JPEG- (links) und Wavelet-Kodierung (rechts) erzielbaren Bildqualität bei 0.128bpp (1:62.5).

Abbildung B.6: Vergleich der visuellen Bildqualität bei gleicher PSNR von 27.32 dB. Links: JPEG-Kodierung, 0.128 bpp. Rechts: Wavelet-Kodierung, 0.069 bpp.

B.4 Kompressionsbeispiele zum BCQ-Verfahren

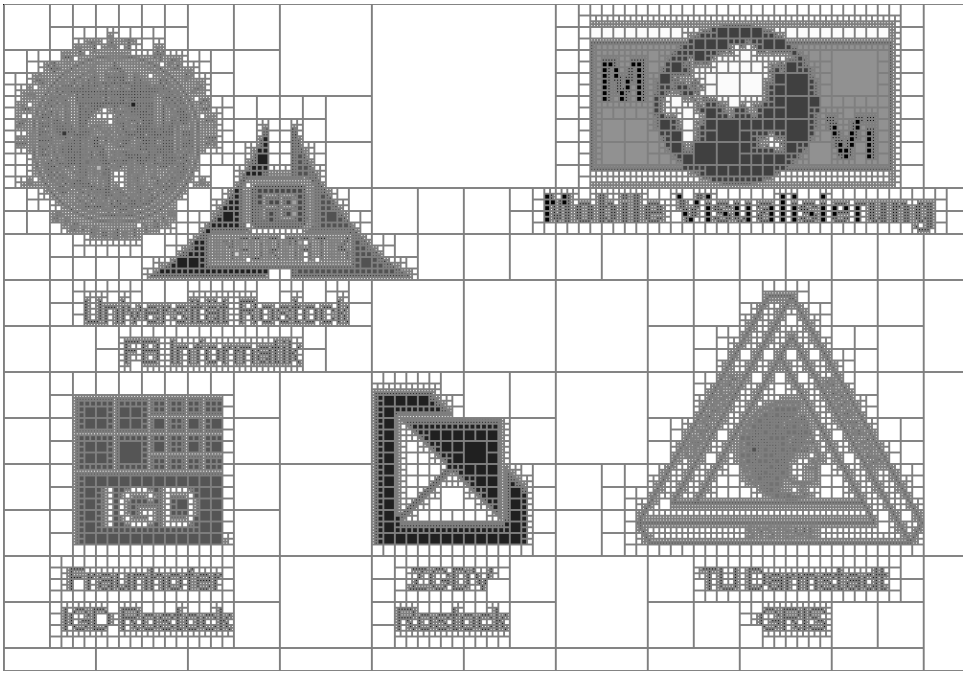
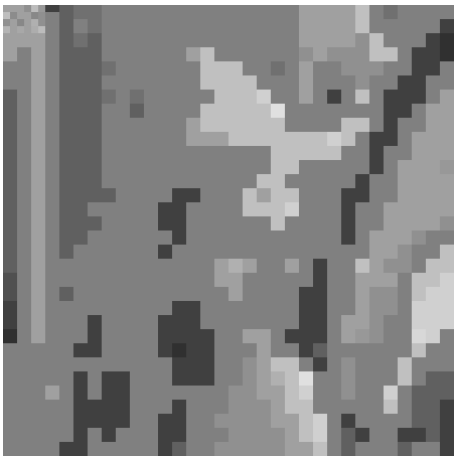


Abbildung B.7: BCQ-Zerlegung des Testbildes „logos“ (Abbildung B.2(a)).



BCQ, 6 levels (410 Bytes)

MOVIWAVE, 0.0125 bpp (410 Bytes)

BCQ, 7 levels (1692 Bytes)

MOVIWAVE, 0.052 bpp (1692 Bytes)

BCQ, 8 levels (7285 Bytes)

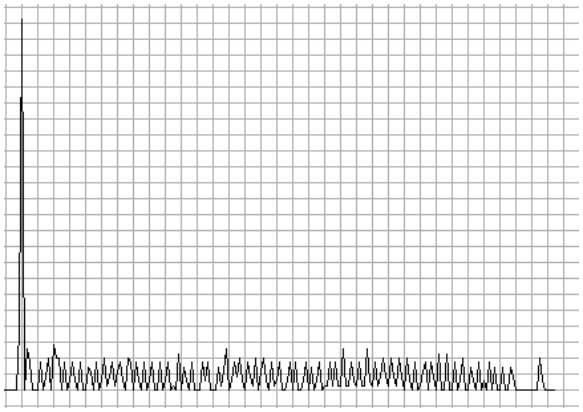
MOVIWAVE, 0.222 bpp (7285 Bytes)

BCQ, 9 levels (35400 Bytes)

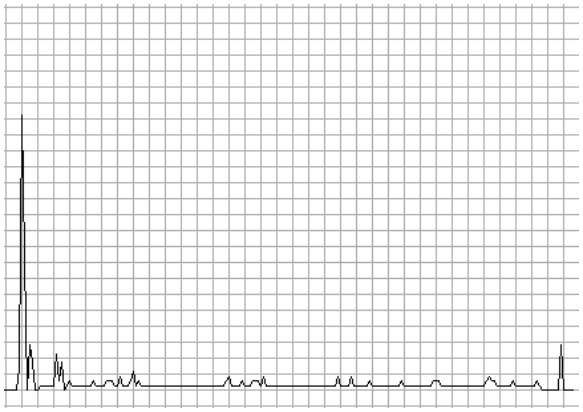
MOVIWAVE, 1.08 bpp (35400 Bytes)

Abbildung B.8: Vergleich der visuellen Bildqualität des Testbildes „lena“ bei der Kodierung mit BCQ (links) und mit dem MOVIWAVE-Format (rechts) bei jeweils gleicher Bitrate.

B.5 Latenzzeit der Client-Server-Kommunikation



(a) HTTP+CGI



(b) Sockets

Abbildung B.9: CPU-Belastung eines ansonsten unbelasteten Servers (INTEL PENTIUM II, 450 MHz, 256 MB RAM) bei einer dreiminütigen Übertragung einer 1024x1024 Pixel großen RoI im LAN.

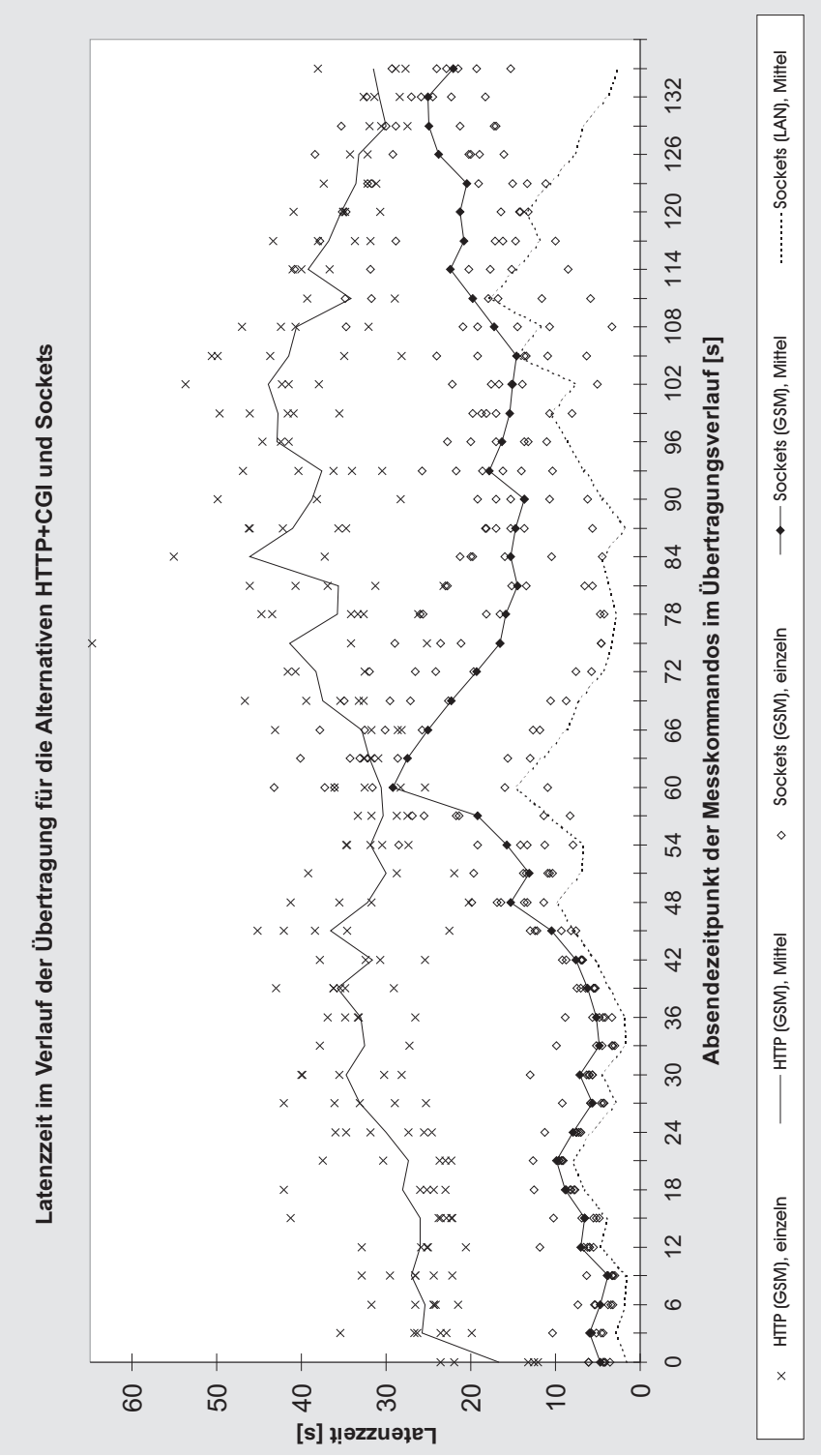


Abbildung B.10: Latenzzeiten der Kommunikationsalternativen HTTP+CGI und Sockets im Übertragungsverlauf.

B.6 Ghost-Feedback beim RECHTECKIGEN FISHEYE-VIEW

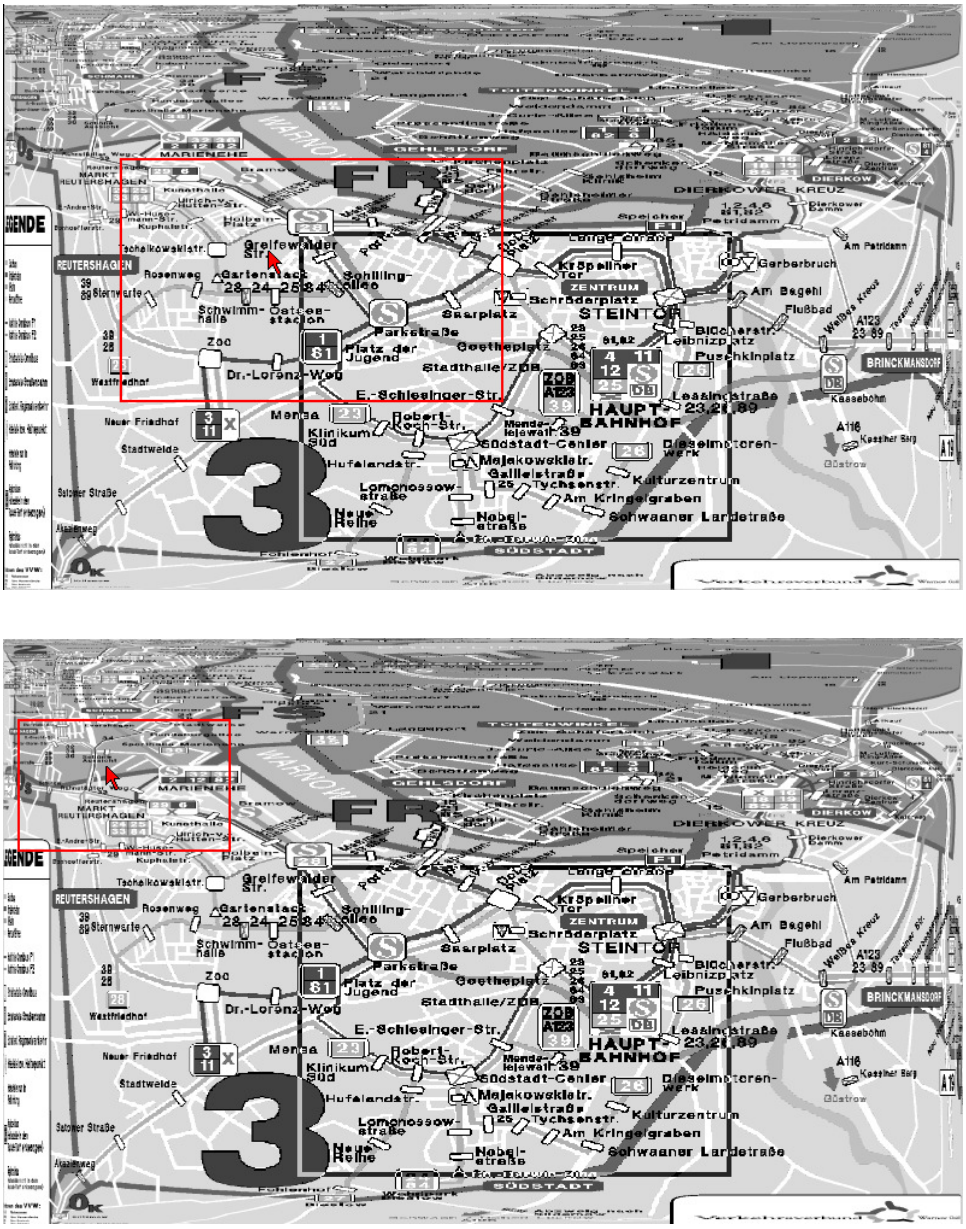


Abbildung B.11: Ghost-Feedback zur Verbesserung des Antwortverhaltens beim Bewegen des Fokus. Das rot dargestellte Ghost-Rechteck zeigt den Bildbereich, der im Fokus nach Abschluss der Verschiebung dargestellt wird.

B.7 Der RECHTECKIGE FISHEYE-VIEW als reine Anzeigetechnik

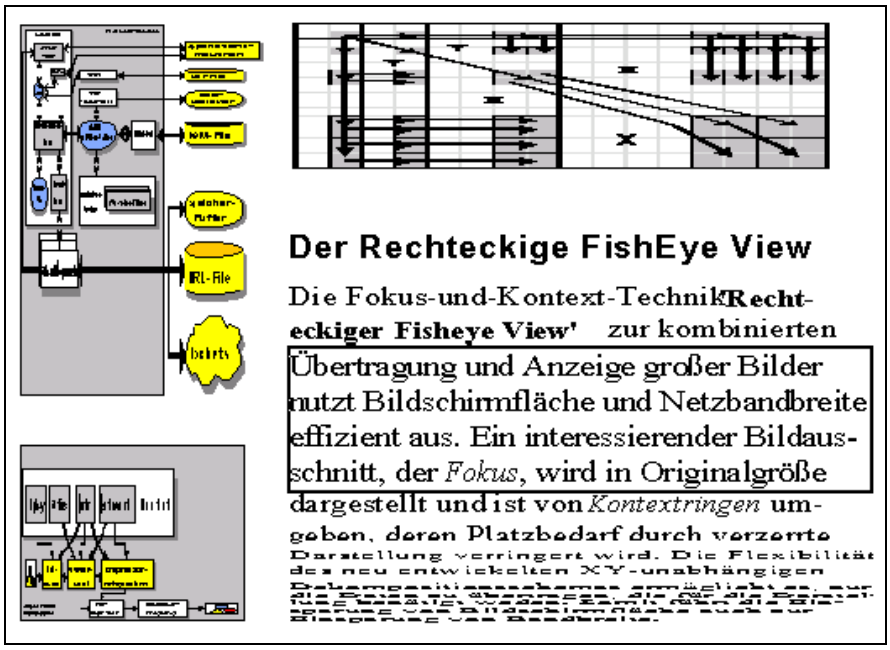


Abbildung B.12: FISHEYE-Darstellung einer Vektorgrafik durch Clipping-Kontextierung mit drei Kontextringen. Größe des Darstellungsfensters: 512x357 Pixel.

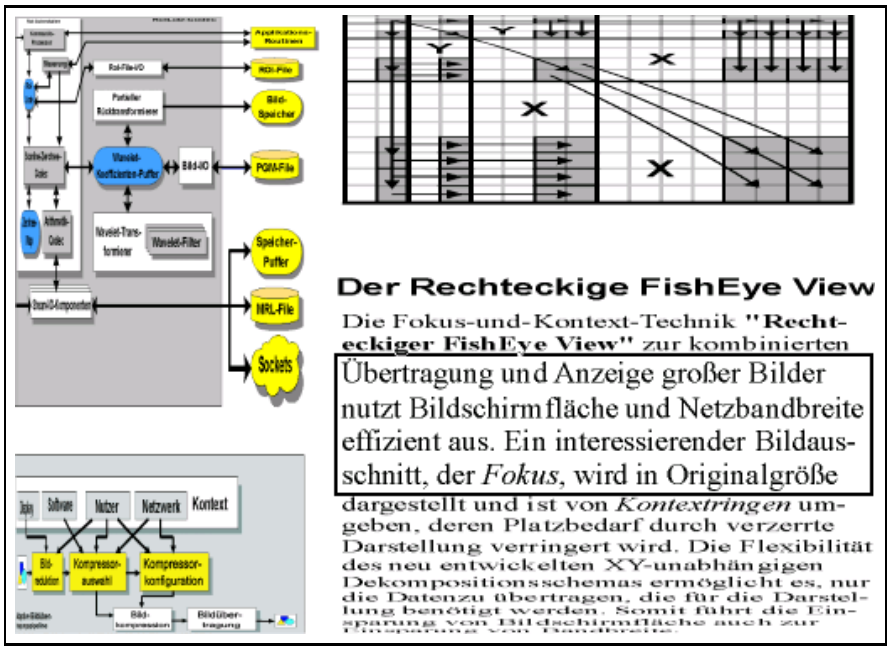
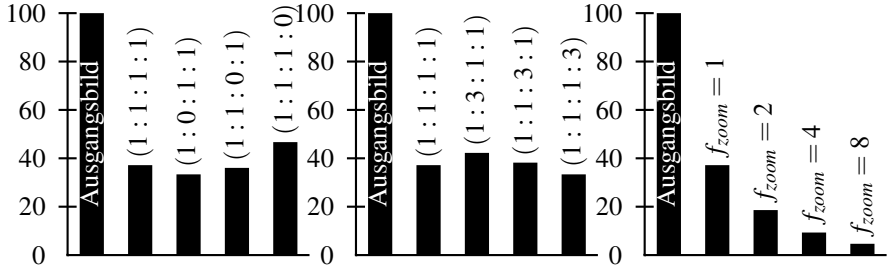


Abbildung B.13: FISHEYE-Darstellung durch glatte Kontextierung mithilfe einer Hintergrundbitmap der Größe 1024x713 Pixel und $\varepsilon = 1.5$. Größe des Darstellungsfensters: 512x357 Pixel.

B.8 Platzersparnis durch den RECHTECKIGEN FISHEYE-VIEW



(a) Verschiedene Sichtbarkeitskonfigurationen: 2er, 4er und 8er Ring. Sichtbarkeit in Klammern; erste Zahl steht für Fokus, letzte für 8er Ring.

(b) Verschiedene Größenverhältnisse: 2er, 4er und 8er Ring. Verhältnisse in Klammern; erste Zahl steht für Fokus, letzte für 8er Ring.

(c) Verschiedene Zoomstufen: 3 Ringe mit jeweils doppelter Skalierung als vorheriger Ring.

Abbildung B.14: Platzersparnis durch den RECHTECKIGEN FISHEYE-VIEW. Angaben in Prozent. Bei allen Berechnungen lag ein Ausgangsbild in der Größe $10\,000 \times 10\,000$ Pixel mit einem Fokus von $2\,002 \times 2\,002$ Pixeln zugrunde. Die Platzersparnis bezieht sich somit auf die Höhe bzw. Breite. Diese Aussagen gelten für alle RECHTECKIGEN FISHEYE-VIEWS mit gleichem Verhältnis zwischen den Größen des Fokus zu denen des Ausgangsbildes.

B.9 Progressive Übertragung von Farbtabellenbildern

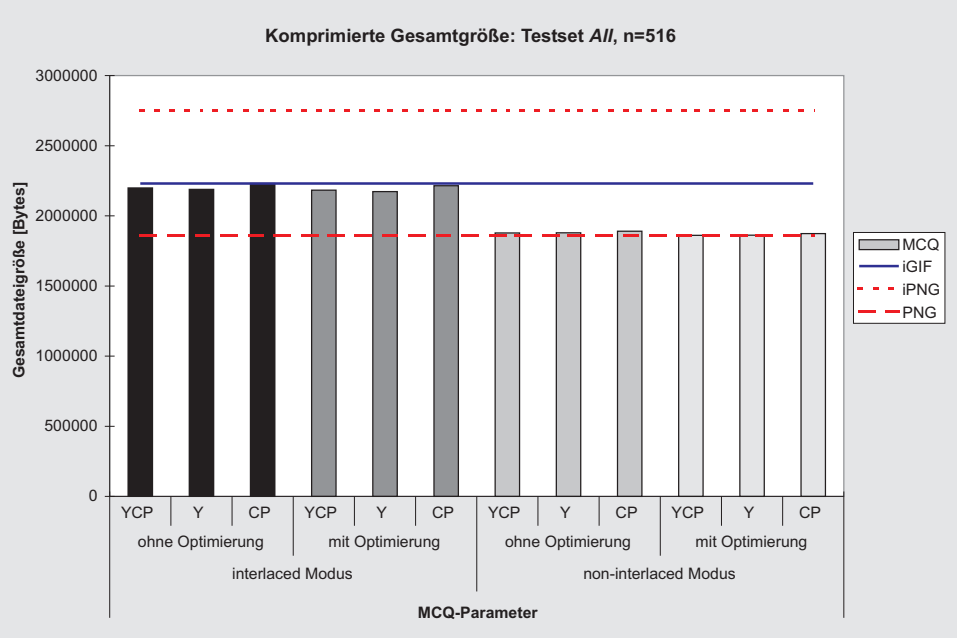


Abbildung B.15: Gesamtgrößen aller komprimierten Bilder des Testsets *All* bei Nutzung von iGIF, iPNG, PNG im Vergleich zu MoviColorQ (MCQ) mit unterschiedlichen Sortierverfahren, mit und ohne Interlacing sowie mit und ohne Optimierung der Parameter k_{inc} und k_{dec}.

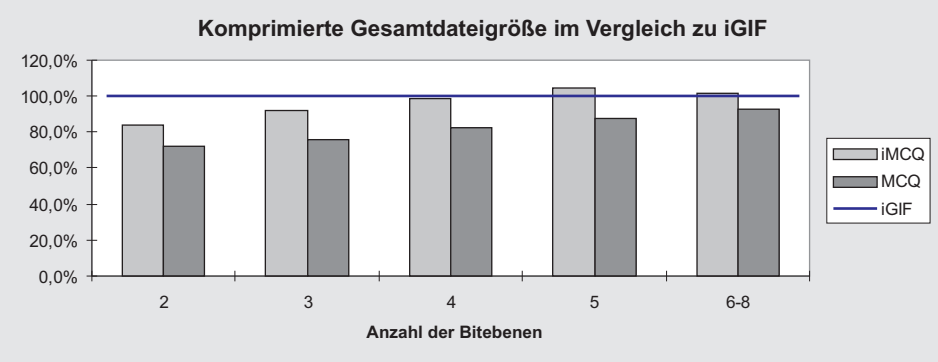


Abbildung B.16: Gesamtgröße der komprimierten Dateien des Testsets *All* bei Benutzung von MoviColorQ (MCQ und iMCQ) im Vergleich zu iGIF ($\widehat{=}100\%$) abhängig von der Anzahl der Bitebenen.

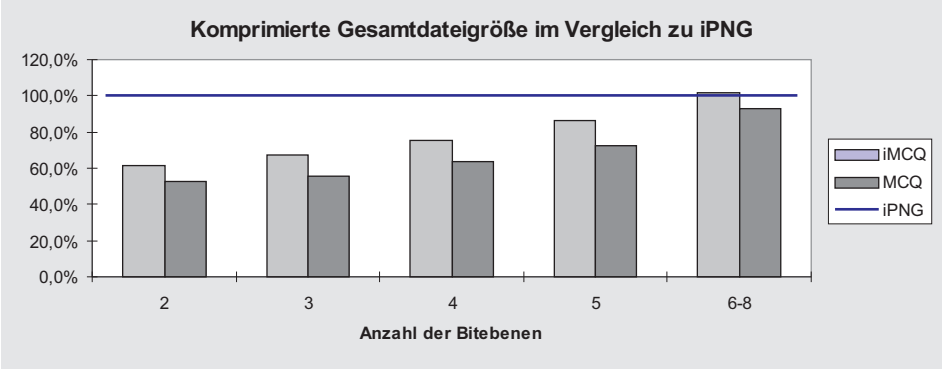


Abbildung B.17: Gesamtgröße der komprimierten Dateien des Testsets *All* bei Benutzung von MOVICOLORQ (MCQ und iMCQ) im Vergleich zu iPNG ($\widehat{=}100\%$) abhängig von der Anzahl der Bitebenen.

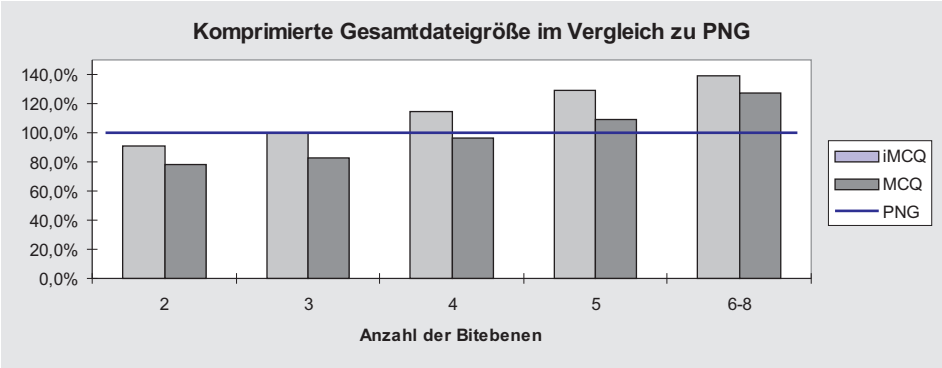


Abbildung B.18: Gesamtgröße der komprimierten Dateien des Testsets *All* bei Benutzung von MOVICOLORQ (MCQ und iMCQ) im Vergleich zu PNG ($\widehat{=}100\%$) abhängig von der Anzahl der Bitebenen.

B.9.1 Abhängigkeit der komprimierten Dateigröße von den Parametern für Lauflängeninkrement und -dekrement der RLR-Kodierung

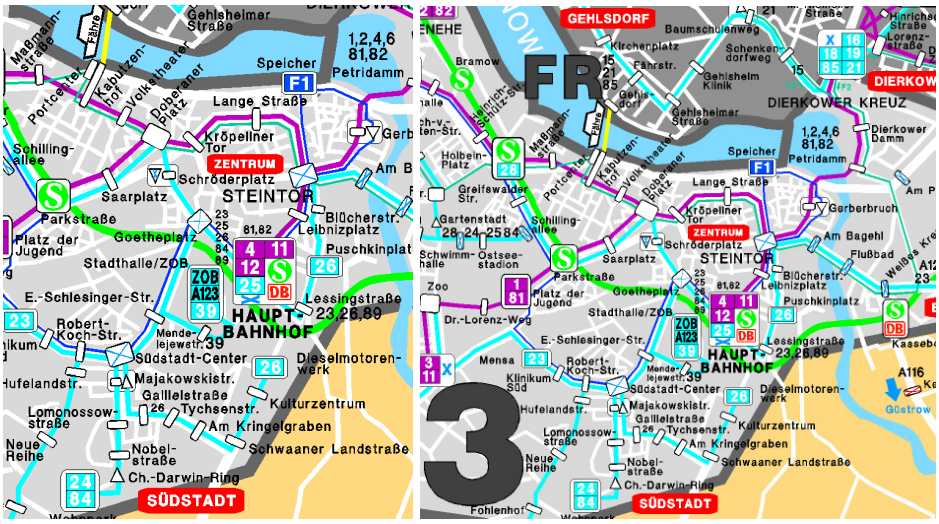


Abbildung B.19: Die Testbilder „innenst18“ (18 Farben, links) und „innenst256“ (244 Farben, rechts).

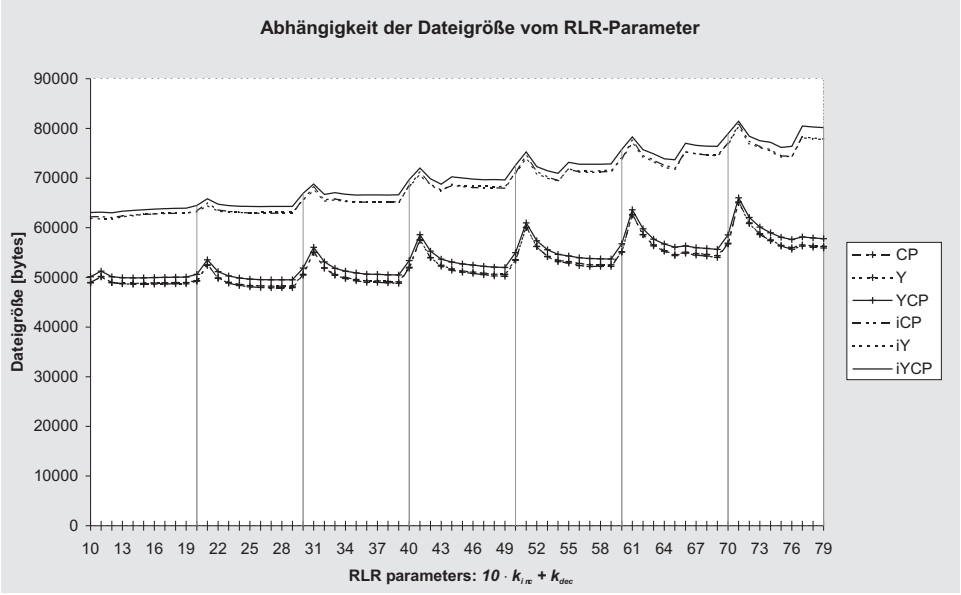


Abbildung B.20: Komprimierte Dateigröße in Abhängigkeit von den RLR-Parametern k_{inc} und k_{dec} für das Testbild „innenst18“ für verschiedene Sortierverfahren mit (iCP, iY, iYCP) und ohne Interlacing (CP, Y, YCP).

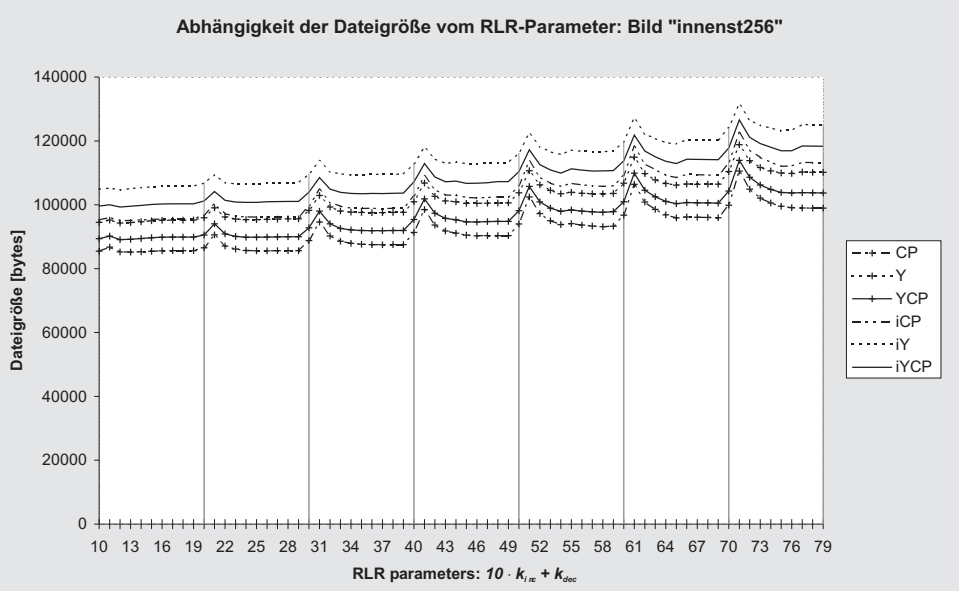


Abbildung B.21: Komprimierte Dateigröße in Abhängigkeit von den RLR-Parametern k_{inc} und k_{dec} für das Testbild „innenst256“ für verschiedene Sortierverfahren mit (iCP, iY, iYCP) und ohne Interlacing (CP, Y, YCP).

B.9.2 Abhängigkeit der komprimierten Dateigröße von der Anzahl der Farben im Bild

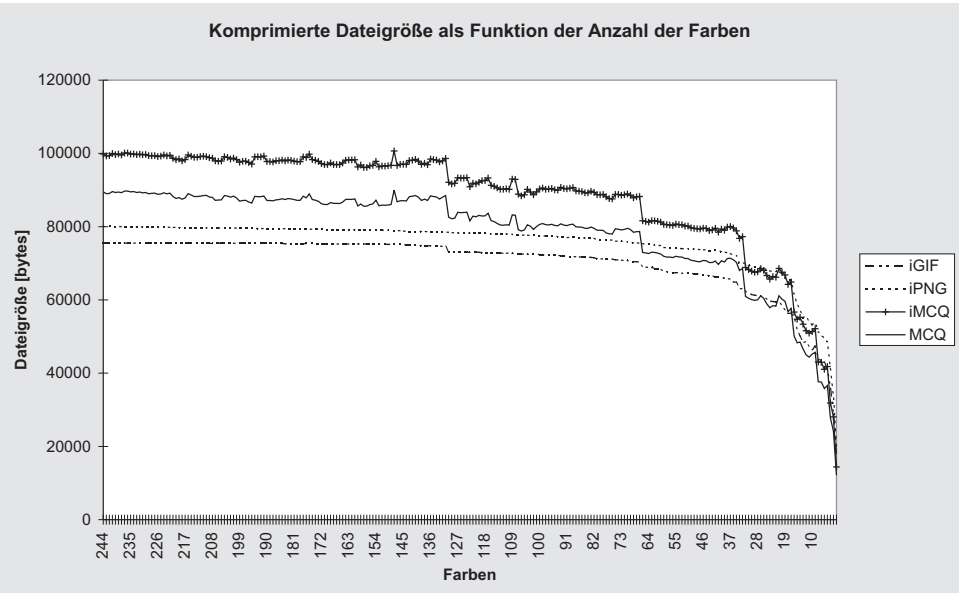


Abbildung B.22: Komprimierte Dateigröße in Abhängigkeit von der Anzahl der Farben im Bild für das Beispiel aus (Abbildung B.19 rechts).

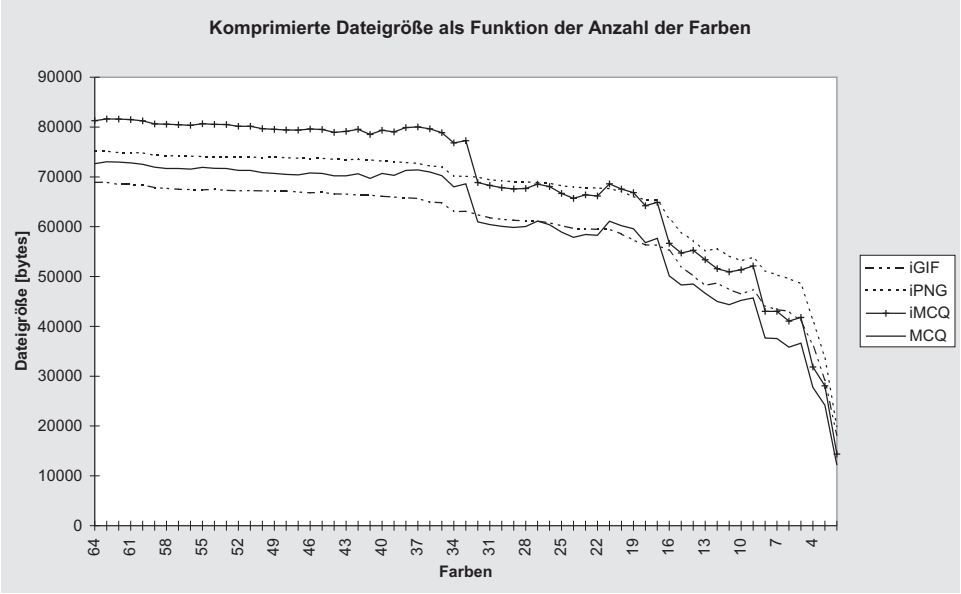


Abbildung B.23: Komprimierte Dateigröße in Abhängigkeit von der Anzahl der Farben im Bild für das Beispiel aus (Abbildung B.19 rechts) – unterste 64 Farben dargestellt.

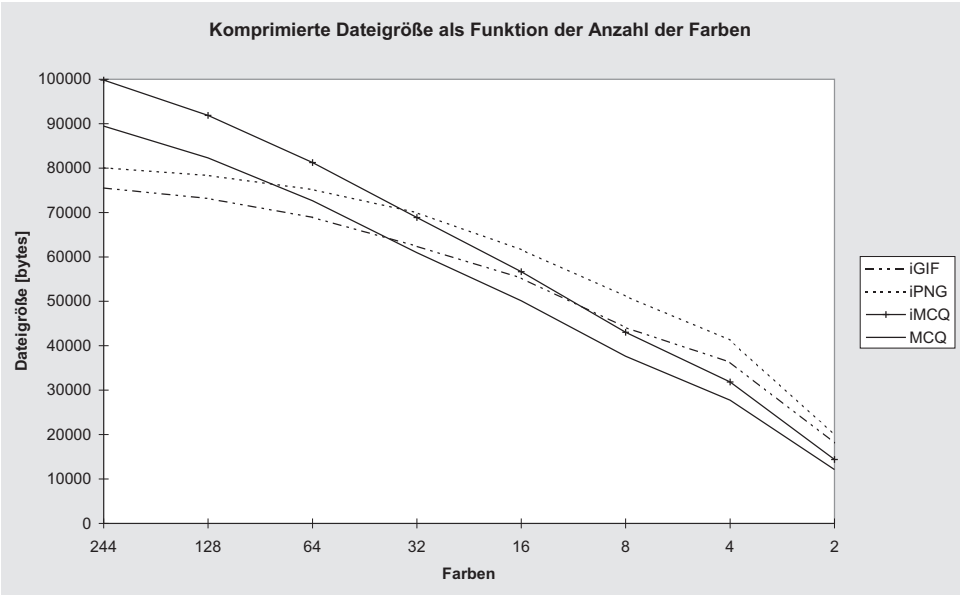


Abbildung B.24: Komprimierte Dateigröße in Abhängigkeit von der Anzahl der Farben im Bild für das Beispiel aus Abbildung B.19 rechts – Bitebenenwechsel in der Farbanzahl dargestellt.

B.9.3 Vergleich von Y-Sortierung und CP-Sortierung

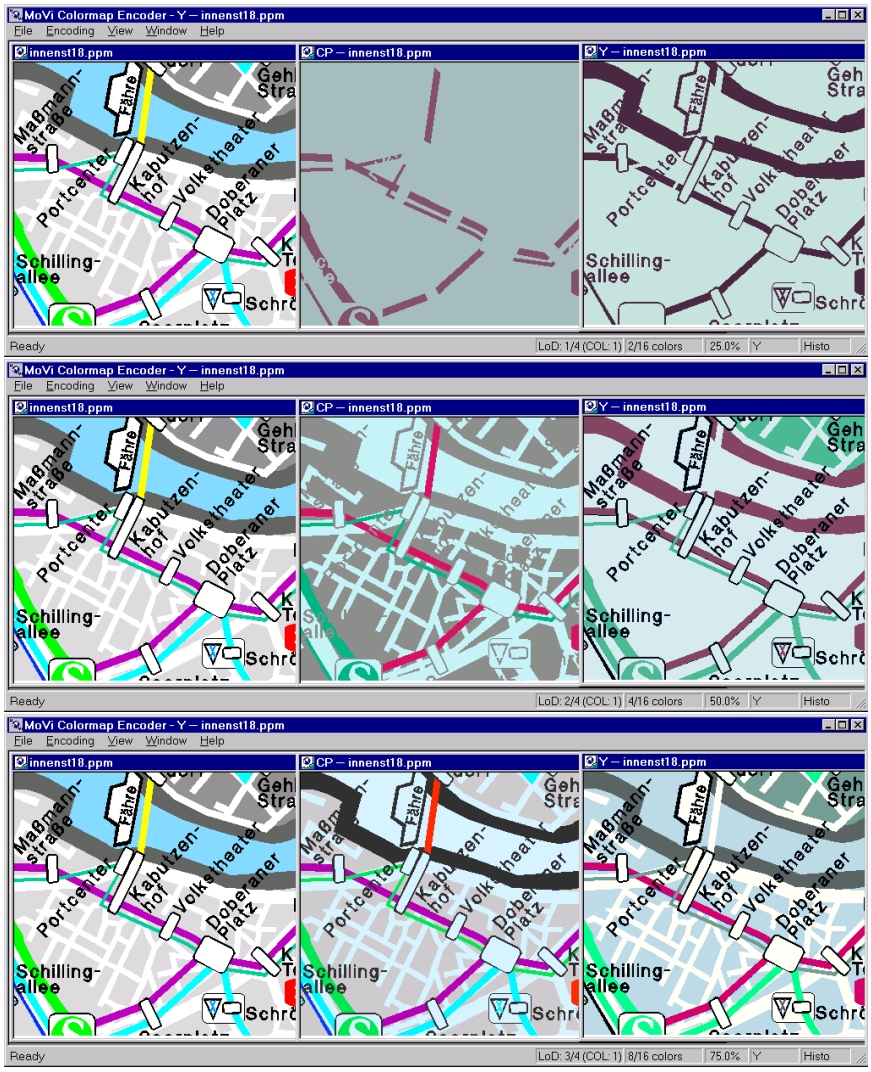


Abbildung B.25: Vergleich der Bildqualität nach Übertragung von einer (oben), zwei (Mitte) und drei (unten) Bitebenen bei Nutzung der Y- (rechts) und der CP-Sortierung (Mitte) für das Testbild „innenst18“. Das Originalbild ist jeweils links dargestellt.

B.9.4 Bildbeispiele

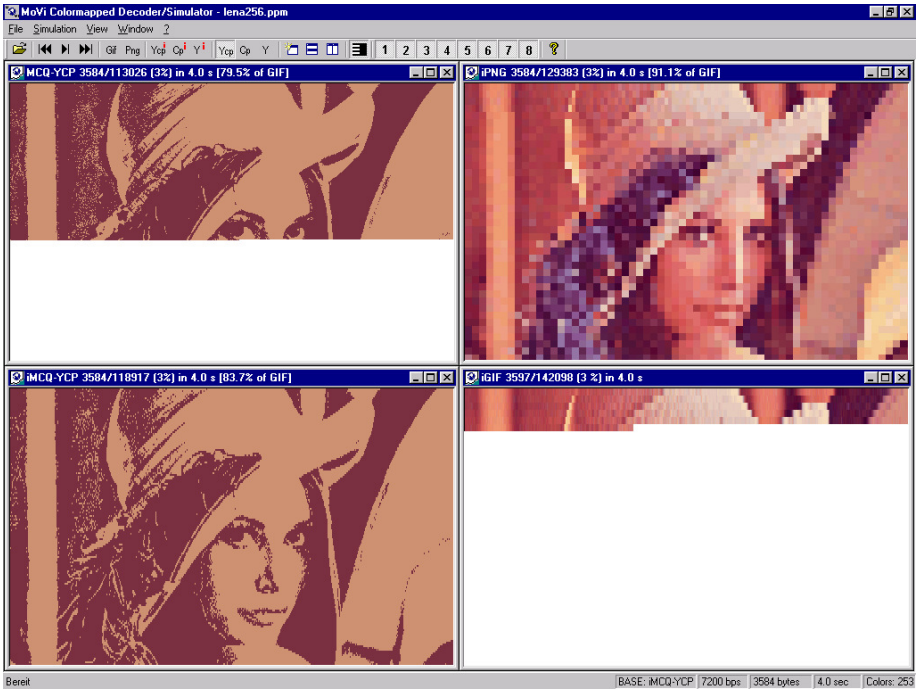


Abbildung B.26: 256-Farben-Version des Lena-Testbildes nach Übertragung des ersten Passes der ersten Bitebene mit Interlacing.

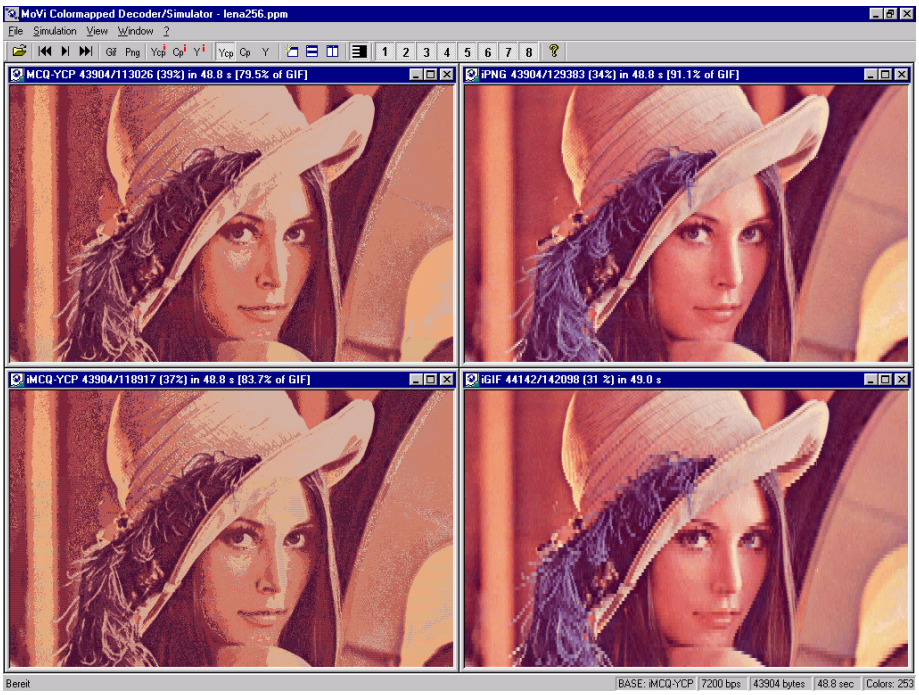


Abbildung B.27: 256-Farben-Version des Lena-Testbildes in einer späteren Stufe der Übertragung.

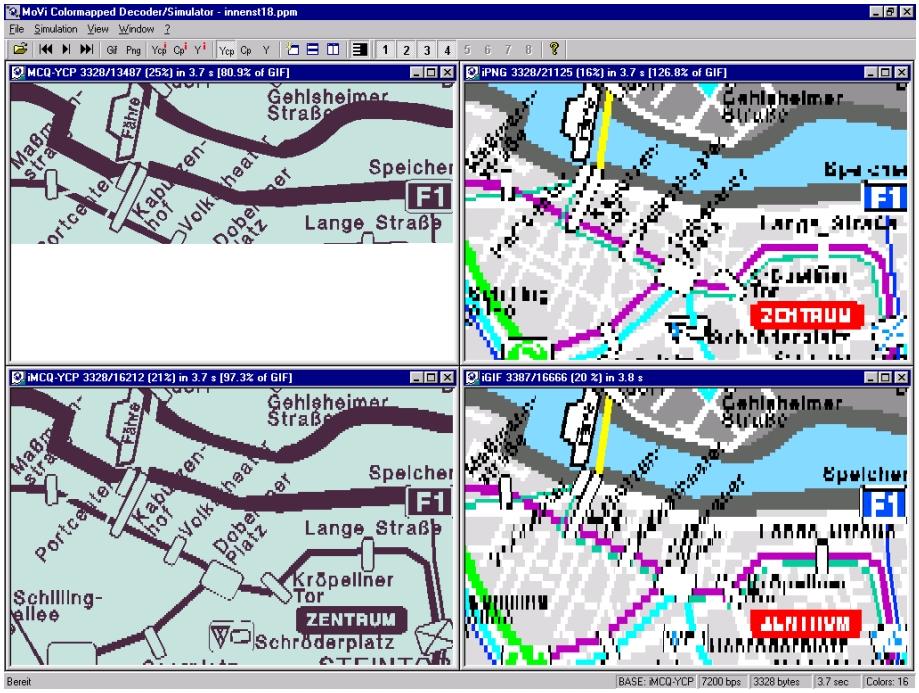


Abbildung B.28: Bild mit wenigen Bitebenen (16 Farben, 4 Bitebenen). Für iMCQ (unten links) wurde der ersten Interlacing-Pass der ersten Bitebene vollständig übertragen..

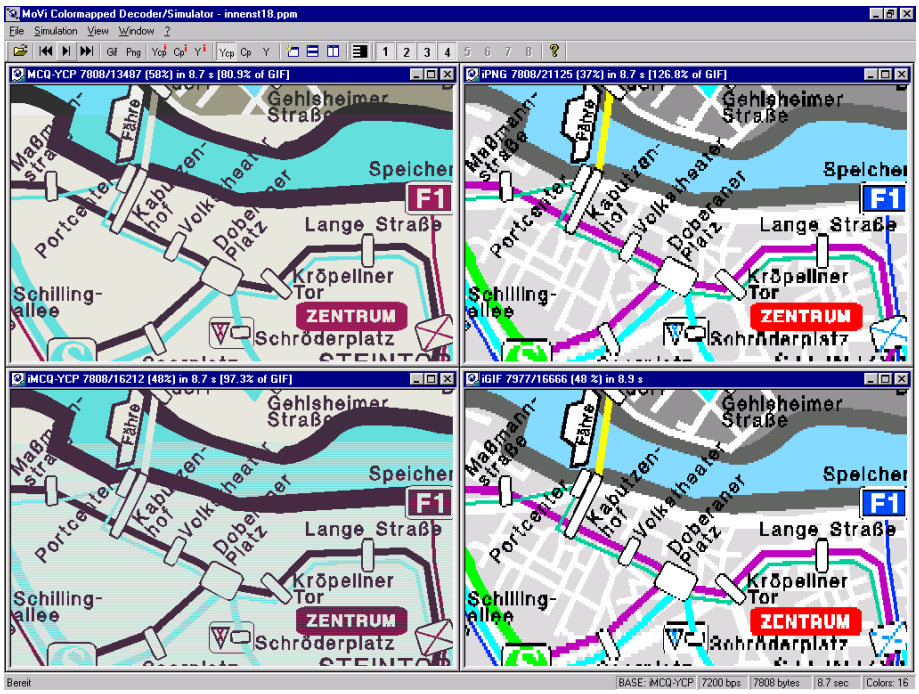


Abbildung B.29: Bessere Erkennbarkeit geringer Farbunterschiede mit größerer räumlicher Ausdehnung (Straßennetz) bei iGIF und iPNG; Streifen durch schlechtere Kompression von iMCQ im Vergleich zu MCQ in späteren Übertragungsschritten.

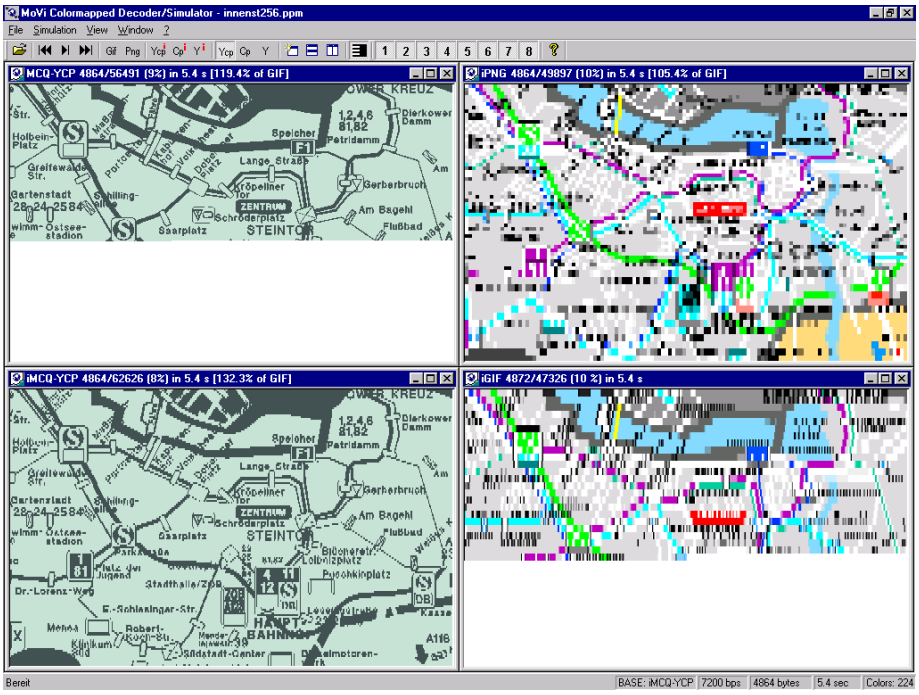


Abbildung B.30: Bild mit 224 Farben nach Abschluss der Übertragung der ersten Bitebene bei MoviColorQ mit Interlacing (unten links).

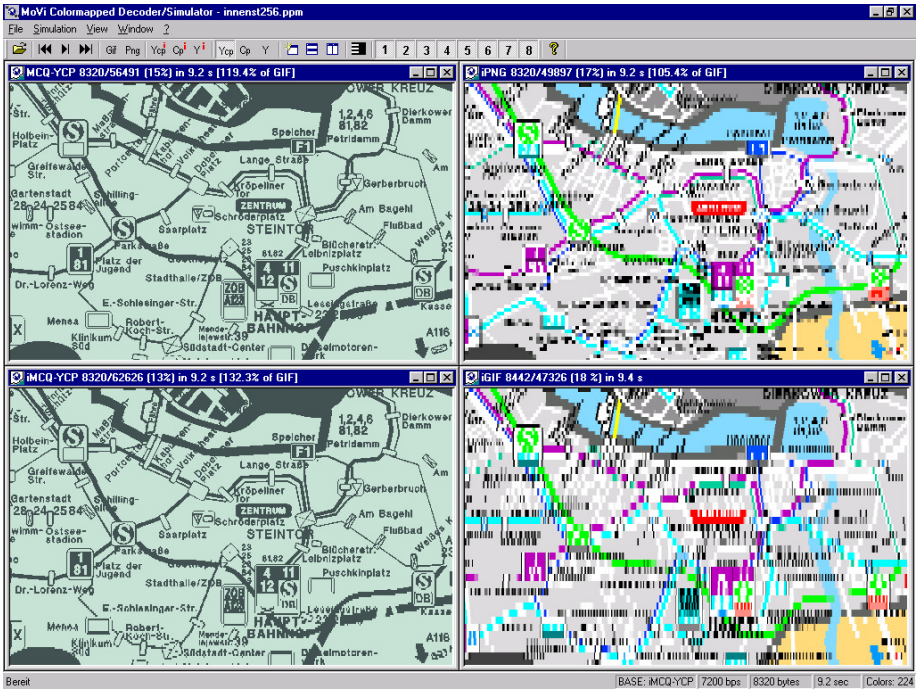


Abbildung B.31: Bild mit 224 Farben nach Abschluss der Übertragung der ersten Bitebene bei MoviColorQ ohne Interlacing (oben links).

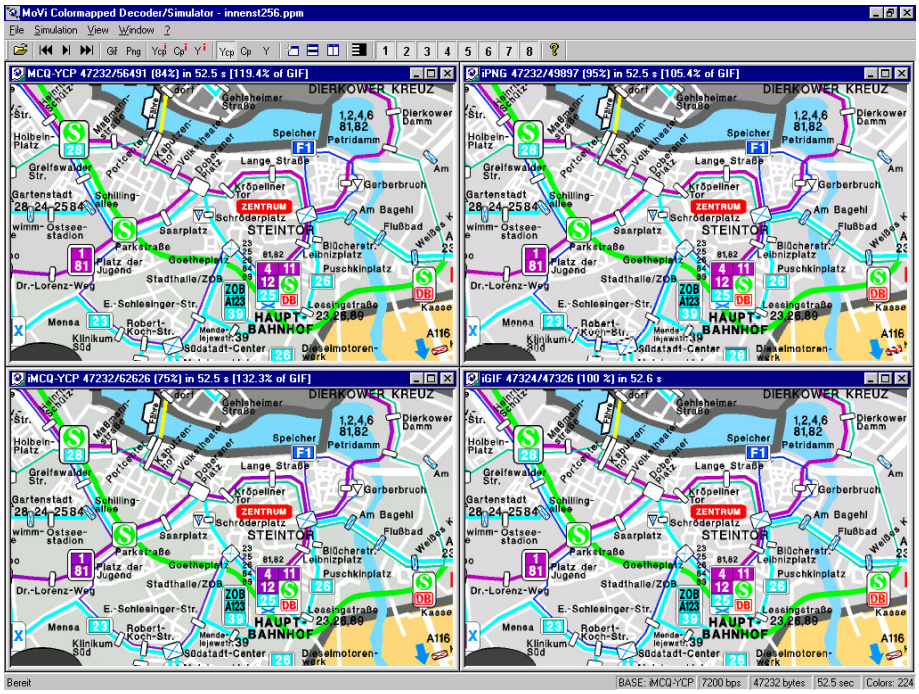


Abbildung B.32: Vollständig übertragenes GIF-Bild mit 224 Farben (unten rechts) und geringe visuelle Unterschiede zu den noch nicht vollständig übertragenen MoviColorQ-Pendants (links).

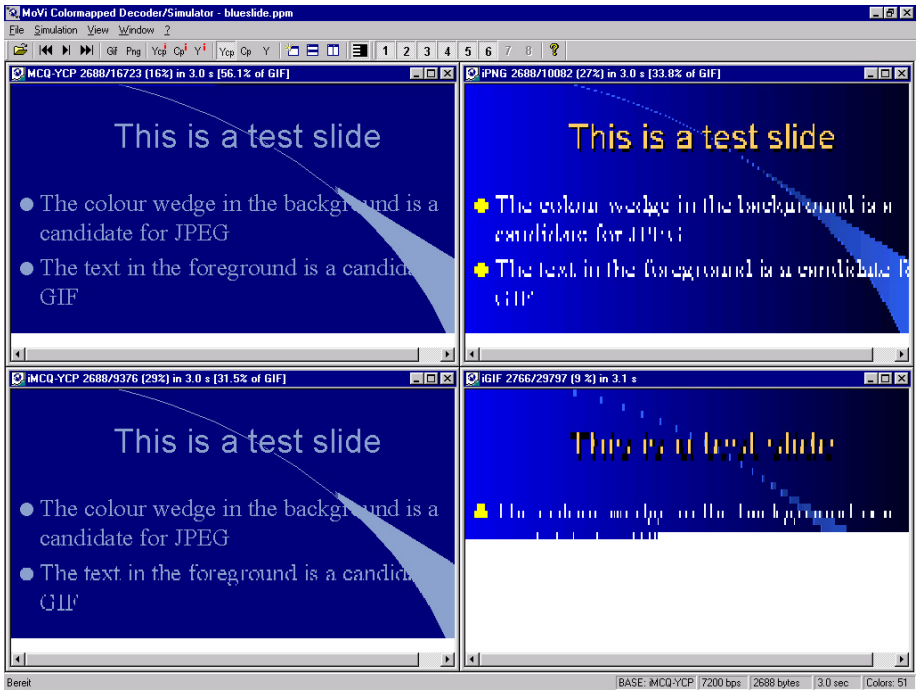


Abbildung B.33: Erste Übertragungsstufe eines Bildes mit Farbverlauf.

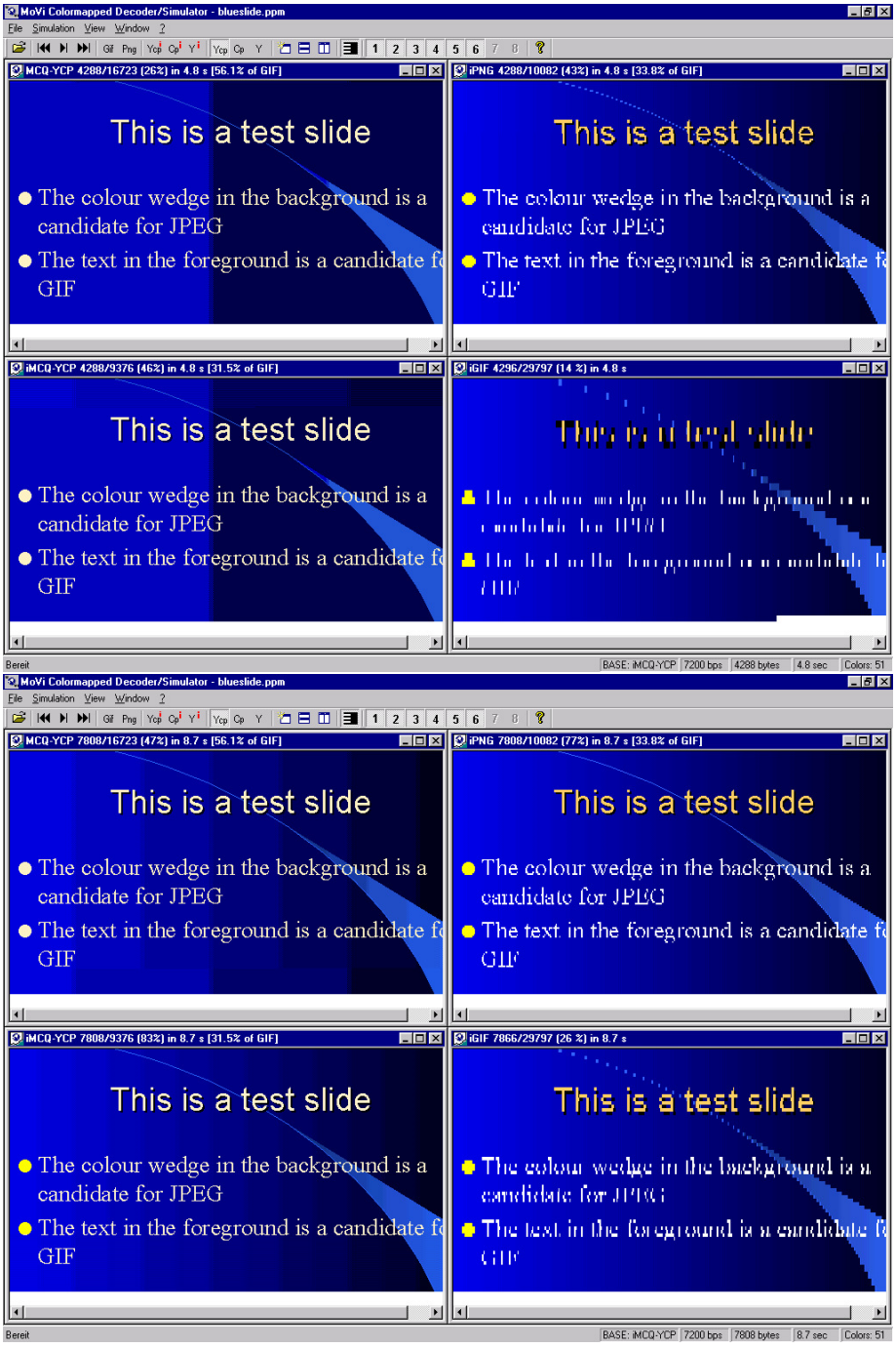


Abbildung B.34: Spätere Übertragungsstufen eines Bildes mit Farbverlauf.

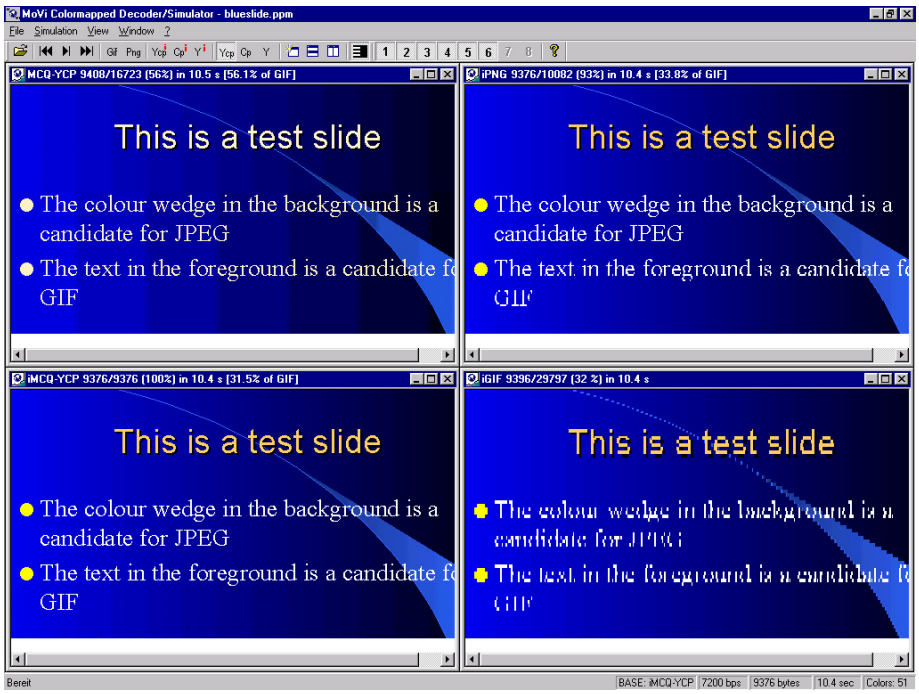


Abbildung B.35: Bildqualität eines Bildes mit Farbverlauf nach Übertragung der kleinsten erreichbaren Dateigröße (MoviColorQ mit Interlacing) im Vergleich zu MoviColorQ ohne Interlacing, iPNG und iGIF.

Anhang C

Messwerttabellen

C.1 Client-Server-Kommunikation

Absendefolge	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Ankunftsfolge	1	3	2	4	6	5	8	7	9	11	10	**??**	13	14
Absendefolge	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Ankunftsfolge	12	19	16	15	17	18	22	20	21	24	23	35	25	26
Absendefolge	29	30	31	32	33	34	35	36	37	38	39	40	41	
Ankunftsfolge	33	27	32	28	29	30	31	34	36	37	38	39	40	

Tabelle C.1: Ankunftsreihenfolge der Kommandos bei einer HTTP+CGI-Übertragung über GSM.

C.2 Zeiten für den Bildaufbau beim RECHTECKIGEN FISH-EYE-VIEW als reine Anzeigetechnik

Methode	**Komplexität**	**Zeichnen unterbrechbar**	**Zeit**
Glatte Kontextierung	1000x1000 Pixel	nein	3.6 s
Glatte Kontextierung	2000x2000 Pixel	nein	9.5 s
Clipping-Kontextierung	143 verschiedene Primitive	nein	0.166 s
Clipping-Kontextierung	10800 Polylines mit 169400 Punkten	nein	1.8 s
Glatte Kontextierung	1000x1000 Pixel	ja	0.043 s
Glatte Kontextierung	2000x2000 Pixel	ja	0.09 s
Clipping-Kontextierung	143 verschiedene Primitive	ja	0.062 s
Clipping-Kontextierung	10800 Polylines mit 169400 Punkten	ja	0.333 s

Tabelle C.2: Performance des RECHTECKIGEN FISHEYE-VIEWS als reine Darstellungstechnik: Benötigte Zeit[2] für einen Bildaufbau beim Verschieben des Fokus für die beiden vorgeschlagenen Methoden *Glatte Kontextierung* und *Clipping-Kontextierung* in Abhängigkeit von der Komplexität der Daten und der Unterbrechbarkeit des Zeichnens.

[2]Die Zeiten wurden auf einem INTEL Pentium mit 200MHz, 32MB RAM, einer Displaygröße von 800x600 Pixeln und einer Fokusgröße von 400x300 Pixeln unter Windows 98 gemessen.

C.3 Progressive Übertragung von Farbtabellenbildern

C.3.1 Bitebenen-Histogramme

	Bitebenen	2	3	4	5	6	7	8	Gesamt
Anzahl von Bildern	*All*	64	213	174	36	17	5	7	516
	Cartoons	64	193	155	91	29	2	0	534
	Cartoons_dither	1	3	19	72	20	2	0	117
	Cartoons_nodither	63	190	136	19	9	0	0	417
	Internet	1	1	8	7	4	4	7	32

Tabelle C.3: Bitebenenhistogramme der Testsets.

C.3.2 Kompressionsratenvergleiche

Methode	iMCQ	MCQ	iGIF	iPNG	PNG
Gesamt-Dateigröße [Bytes]	1695647	1501393	1520437	2033043	1422505
Größe im Vergleich zu iGIF	111.5%	98.7%	100.0%	133.7%	93.6%
Größe im Vergleich zu iPNG	83.4%	73.8%	74.8%	100.0%	70.0%

Tabelle C.4: Vergleich der komprimierten Gesamt-Dateigröße für verschiedene Kompressionsmethoden. Testset: *Cartoons*.

Methode	iMCQ	MCQ	iGIF	iPNG	PNG
Gesamt-Dateigröße [Bytes]	567359	517081	569304	555811	405401
Größe im Vergleich zu iGIF	99.7%	90.8%	100.0%	97.6%	71.2%
Größe im Vergleich zu iPNG	102.1%	93.0%	102.4%	100.0%	72.9%

Tabelle C.5: Vergleich der komprimierten Gesamt-Dateigröße für verschiedene Kompressionsmethoden. Testset: *Internet*.

C.3.3 Dynamische Auswahl des Kodierverfahrens

	Bild: alien17			Bild: awards		
Bit-ebene	**Signi-fikanz**	**Verfei-nerung**	**% von MCQ**	**Signi-fikanz**	**Verfei-nerung**	**% von MCQ**
7						
6						
5				RLR		100.0%
4				RLR	GZ	99.6%
3	RLR		100.0%	RLR	RAW	100.0%
2	RLR	RAW	100.0%	RLR	GZ	97.8%
1	RLR	RAW	100.0%	RLR	RAW	100.0%
0	RLR	GZ	65.1%	RLR	RAW	100.0%
Gesamt			95.4%			99.6%

Tabelle C.6: Steigerung der Kompression von MOVICOLORQ durch dynamische Auswahl des Kodierverfahrens[4] (ungeditherte Bilder). Prozentwerte geben die prozentuale Dateigröße im Vergleich zum Original-MCQ-Verfahren an.

	Bild: alien16			Bild: win		
Bit-ebene	**Signi-fikanz**	**Verfei-nerung**	**% von MCQ**	**Signi-fikanz**	**Verfei-nerung**	**% von MCQ**
7						
6						
5						
4				GZ		89.7%
3	GZ		89.8%	RLR	GZ	80.2%
2	RLR	GZ	73.7%	RLR	GZ	71.3%
1	RLR	GZ	94.1%	RLR	GZ	64.2%
0	RLR	GZ	91.9%	RLR	GZ	61.9%
Gesamt			89.2%			76.9%

Tabelle C.7: Steigerung der Kompression von MOVICOLORQ durch dynamische Auswahl des Kodierverfahrens (geditherte Bilder). Prozentwerte geben die prozentuale Dateigröße im Vergleich zum Original-MCQ-Verfahren an.

[4] RAW bedeutet „Unkodierte Ausgabe“.

Anhang D

Spezifikation des ASCII–RoI–Formates

D.1 Zweck

Dieses Format dient zur Speicherung von RoIs (Regions of Interest) für das Autorenwerkzeug, zum Datenaustausch und als Steuerdatei für den serverseitigen Bildkodierer.

D.2 Syntax

```
<File> :: <Header> <RoI_Spec> {<RoI_Spec>}
                   <number_of_auth_cmds> {<AuthorCmd> }

<Header> ::
"ROI19"  <EOL>
<path_and_name_of_image_file> <EOL>
<number_of_levels> <number_of_ir_levels> <filter_number> <EOL>
<number_of_RoIs> <EOL>

<AuthorCmd> ::
<step_cnt> <keyword> <params> <EOL>

<RoI_Spec> ::
@ NAME <name> <EOL>
@ LLOD <x_res> <y_res> <precision> <color><EOL>
@ PRIO <RoI_priority>
<Footprint_spec>

<Footprint_spec> :: <Polygon> | <Rectangle> | <Ellipse>

<Polygon> ::
@ POLY <number_of_vertices> <EOL>
<X1> <Y1> <EOL>
...
<Xn> <Yn>  <EOL>

<Rectangle> ::
@ RECT <X1> <Y1> <X2> <Y2>  <EOL>

<Ellipse> ::
@ ELLIPSE <X_center> <Y_center> <dx> <dy> <EOL>
```

D.3 Bemerkungen

Alle Elemente werden durch Whitespace (Leerzeichen, TAB, Zeilenende) getrennt. Leerzeilen sind nicht erlaubt.

`ROI19` kennzeichnet die Version.

Kommentarzeilen beginnen mit # in der ersten Spalte. Sie dürfen nicht im Header und nicht innerhalb eines `<Polygon>` auftreten.

Attributzeilen beginnen mit `@` in der ersten Spalte, gefolgt von Whitespace, einem Schlüsselwort und Parametern. Jede Attributzeile muss mit einen Zeilenendezeichen abgeschlossen werden.

Alle X– und Y–Koordinaten sind Integer-Werte. Es gilt $0 \leq x < \mathit{image\_width}$ und $0 \leq y < \mathit{image\_height}$.

`<path_and_name_of_image_file>` gibt Pfad und Namen der Bilddatei an. Der Pfad darf relativ sein; in diesem Fall muss beim Start des Decoders gesichert werden, dass das richtige Arbeitsverzeichnis eingestellt ist. Die Bilddatei muss existieren.

`<number_of_levels>` gibt die Anzahl der Wavelet-Zerlegungsstufen an (>2).

`<number_of_ir_levels>` gibt die Anzahl der unabhängigen Zerlegungsstufen an (>=1; 1 entspricht klassischer Wavelet-Zerlegung).

`<filter_number>` gibt den Code des zu nutzenden Wavelet-Filters an (analog zum Datenstrom-Header von MVWCodec). Default-Wert ist 8 (WT_HYB_53HAAR).

Die Attribute `@ NAME`, `@ LLOD` und `@ PRIO` sind OPTIONAL. Wird ein optionales Attribut angegeben, so sind seine vollständigen Argumente anzugeben.

Genau eine `<Footprint_spec>` je RoI ist VORGESCHRIEBEN.

`@ NAME` gibt der RoI einen Namen, unter dem sie im Autorensystem ansprechbar ist. Es ist auch denkbar, den Namen zu übertragen, um vordefinierte RoIs für den Nutzer intuitiv zu benennen. Der Namensstring wird nicht in Anführungszeichen eingeschlossen. Er darf keine Leerzeichen und keine Tabulatoren enthalten.

`@ LLOD` gibt eine minimale Spezifikationsmenge von Ortsvektoren aus $\mathbb{L}$ der Mächtigkeit 1 für den Ziel-LoD dieser RoI an. Spezifikationsmengen mit mehr Ortsvektoren sind aktuell nicht implementiert. Dabei entspricht `<?_res>` der Differenz aus `#<number_of_levels>` und Log2 des Nenners des Auflösungsfaktors (`?_res`>=0). `<precision>` ist die maximale Genauigkeit in Bits, vom msb angefangen. 0<precision<=16 <color> hat den Wert 0 (Y-Kanal) oder 1 (Cb+Cr-Kanal).

`@ PRIO` gibt die Priorität für die akuelle RoI an. `<RoI_priority>` ist die Scheduling-Priorität für die RoI. Sie wird als INTEGER definiert und als Bitplane-Differenz interpretiert. Verschiedene RoIs dürfen die gleiche Priorität haben.

AuthorCmd: `<step_cnt>` gibt den Schritt des RoI-Schedulers an, nach dem Cmd ausgeführt werden soll. `<keyword>` ist das Schlüsselwort des Kommandos. `<params>` sind die Parameter des Kommandos. Die `<EOL>` – Zeichen in der Kommandobeschreibung sind optional. Ein solches Zeichen kann auch durch andere Whitespace-Zeichen ersetzt werden. Eine Kommandobeschreibung muss jedoch am Anfang einer Zeile beginnen. Der Parameter `<roi_idx>` selektiert die RoI, auf die sich das Kommando bezieht. Bei Angabe von `roi_idx=65535` wird das Kommando auf alle definierten RoIs angewendet. STOPALL und STOPROI(65535) haben also dieselbe Semantik; STOPALL wird nur aus Gründen der Kompatibilität mit älteren Versionen unterstützt.

Aktuell werden folgende Kommandos unterstützt:

```
STOPALL          (keine Parameter)
STOPROI          <roi_idx>
CONTROI          <roi_idx>
LLODMOD          <roi_idx> <lodResX> <lodResY> <lodPrec> <lodColr>
PRECINC          <roi_idx> <pos_increment>
XRESINC          <roi_idx> <pos_increment>
YRESINC          <roi_idx> <pos_increment>
PRECPRIOMOD      <roi_idx> <precPrio>
PRECPRIOINCDEC   <roi_idx> <pos_or_neg_increment>

DEFRECT          <name> <EOL>
                 <lodResX> <lodResY> <lodPrec> <lodColr> <precPrio> <EOL>
                 <left> <top> <right> <bottom>

DEFELIPS         <name> <EOL>
                 <lodResX> <lodResY> <lodPrec> <lodColr> <precPrio> <EOL>
                 <left> <top> <right> <bottom>

DEFPOLY          <name> <EOL>
                 <lodResX> <lodResY> <lodPrec> <lodColr> <precPrio>
                                                          <n> <EOL>
                 <X1> <Y1> <EOL>
                 ...
                 <Xn> <Yn>

DEFFISH          <N> <EOL>
                 <XWire1> <YWire1> <EOL>
                 ...
                 <XWireN> <YWireN> <EOL>
                 <Subsample1> <EOL>
                 ...
                 <Subsample_n/2> <EOL>
                 <Prio1> <EOL>
                 ...
                 <Prio_n/2>
                 Anmerkung:  Prio1, Subsample1: äußerster Ring
                             Prio_n/2, Subsample_n/2: Fokus
                             N gerade

APPLICATION      <type_flag> <length> <0..255> {<0..255>}
```

D.4 Beispiel

Diese RoI-Datei definiert RoIs für die Bilddatei c:\temp\image.pgm. Es werden 4 Zerlegungsstufen verwendet, der Parameter für die XY-unabhängige Zerlegung ist 1 (dyadisches Schema). Das Standard-Wavelet-Filter mit dem Code 8 wird benutzt (5/3+Haar; siehe Hilfetext von MOVIWAVECODEC für eine Liste aller gültigen Codes).

Die RoI-Datei definiert 3 RoIs: ein Hintergrundrechteck, eine Ellipse und ein unregelmäßiges konkaves Viereck. Das Hintergrundrechteck weist sowohl in X- ails auch in Y-Richtung nur die halbe Auflösung auf. Die Ellipse wurde höher priorisiert.

Ein vordefiniertes Kommando zur Erhöhung der Priorität des Hintergrundes nach 100 Übertragungsschritten steht am Ende der Datei.

```
------------------------------------------------
ROI19
c:\temp\image.pgm
4 1 8
3
@ NAME Background RoI
@ LLOD 3 3 16 0
```

```
@ PRIO 0
@ RECT 0 0 511 511
@ NAME A Polygon
@ LLOD 4 4 16 0
@ PRIO 0
@ POLY 4
322 35
488 40
414 155
388 85
@ NAME An ellipse
@ LLOD 4 4 16 0
@ PRIO 3
@ ELLIPSE 184 232 187 223
1
100 PRECPRIOINCDEC 0 2
-------------------------------------------------
```

Literatur

[ABMD92] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[AF98] E. Atsumi and N. Farvardin. Lossy/lossless region-of-interest image coding based on set partitioning in hierarchical trees. In *Proc. IEEE International Conference on Image Processing (ICIP98)* [ICI98].

[AJ81] E. H. Adelson and Burt P. J. Image data compression with the Laplacian pyramid. In *Proc. Conference on Pattern Recognition and Image Processing*, pages 218–223, Dallas, Texas, 1981. IEEE Computer Society Press, Los Angeles. `http://www-bcs.mit.edu/people/adelson/publications/`.

[AP94] P. Astheimer and M.L. Pöche. Level-of-detail generation and its application in virtual reality. In S. Feiner et al., editors, *VRST '94. Virtual Reality and Software Technology*, Singapore, 1994.

[ASH87] E. H. Adelson, E. Simoncelli, and R. Hingorani. Orthogonal pyramid transforms for image coding. In *Proc. SPIE, Visual Communications and Image Processing II*, volume 845, pages 50–58, Cambridge, MA, October 27-29 1987. `http://www-bcs.mit.edu/people/adelson/publications/acrobat/orthogonal87.pdf`.

[AV] Algo Vision Mediatec GmbH. `http://www.algovision.com/`.

[BA83] P. Burt and E.H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, COM-31:532–540, 1983. `http://www-bcs.mit.edu/people/adelson/publications/`.

[BDH96] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, December 1996.

[BGSK97] M. Boliek, J.D. Gormish, E.L. Schwartz, and A. Keith. Next generation image compression and manipulation using CREW. In *Proc. of IEEE International Conference on Image Processing*, Santa Barbara, CA, October 1997.

[BR95] A. Binstock and J. Rex. *Practical Algorithms for Programmers*. Addison-Wesley, Reading, MA, 1995.

[BS95] C.W. Brown and B.J. Shepherd. *Graphics file formats: reference and guide*. Manning Publications, Greenwich, Conn., 1995.

[C+99] J. Coutaz et al. CoMedi: Using computer vision to support awareness and privacy in mediaspaces. In *Proc. CHI*, volume Extended Abstracts, pages 13–14, Pittsburgh, PA, USA, May 15-20 1999. `http://www-prima.inrialpes.fr/MediaSpace/CHI99/`.

[CBCC98] J. Coutaz, F. Bérard, E. Carraux, and J. Crowley. Early experience with the mediaspace CoMedi. In *Proc. IFIP Working Conference on Engineering for Human-Computer Interaction*, Heraklion, Crete, 1998. `http://iihm.imag.fr/publs/1998/EHCI98\_CoMedi.pdf`.

[CDF92] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45:485–560, 1992.

[CDSBL98] R. C. Calderbank, I. Daubechies, W. Sweldens, and Y. Boon-Lock. Wavelet transforms that map integers to integers. *Journal of Applied and Computational Harmonic Analysis*, 5(3):332–369, 1998. `http://cm.bell-labs.com/cm/ms/who/wim/papers/papers.html`.

[CF97] Hongyang Chao and Paul Fisher. An approach of integer reversible wavelet transform for lossless image compression. In *Advances in Computational Mathematics – Proc. Guangzhou International Symposium on Computational Mathematics*, Guangzhou, China, 11-15 August 1997. `http://www.compsci.com/~chao/Publication/`.

[CGI] CGI: Common gateway interface. `http://www.w3.org/CGI/`.

[CGV96] P.C. Cosman, R.M. Gray, and M. Vetterli. Vector quantization of image subbands – a survey. *IEEE Transactions on Image Processing*, 5(2):202–225, February 1996. `http://www.code.ucsd.edu/cosman/subband.ps`.

[Cha94] S.F. Chang. Some new algorithms for processing images in the transform compressed domain. In *Proc. SPIE Symposium on Visual Communications and Image Processing*, 1994. `ftp://ftp.ctr.columbia.edu/CTR-Research/advent/public/papers/94/chang94e.ps`.

[CLR91] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. McGraw Hill, New York, 1991.

[CM95] S.-F. Chang and D.G. Messerschmitt. Manipulation and compositing of MC-DCT compressed video. *IEEE Journal of Selected Areas in Communications, Special Issue on Intelligent Signal Processing*, pages 1–11, Jan. 1995.

[CO98] C. Chrysafis and A. Ortega. Line based, reduced memory, wavelet image compression. In *Proc. Data Compression Conference (DCC98)* [DCC98]. `http://biron.usc.edu/~chrysafi/dcc98.pdf`.

[CO99a] C. Chrysafis and A. Ortega. Line based reduced memory wavelet image compression. Submitted to IEEE Transactions on Image Processing, `http://biron.usc.edu/~chrysafi/line\_based99.pdf`, January 1999.

[CO99b] Christos Chrysafis and Antonio Ortega. An algorithm for low memory wavelet image compression. In *Proc. IEEE International Conference on Image Processing (ICIP99)* [ICI99].

[CP97] S. W. Chiang and L. M. Po. Adaptive lossy LZW algorithm for palettized image compression. *IEE Electronics Letters*, 33(10):852–854, May 8 1997.

[CPB93] Y. Chen, H. Peterson, and W. Bender. Lossy compression of palettized images. In *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, volume 5, pages 325–328, 1993.

[CY97] E.C. Chang and C.K. Yap. A wavelet approach to foveating images. In *Proc. ACM Symposium on Computational Geometry*, volume 13, pages 397–399, 1997. `ftp://cs.nyu.edu/pub/local/yap/visual/foveated.ps.gz`.

[CYY97] E.C. Chang, C.K. Yap, and T.J. Yen. Realtime visualization of large images over a thinwire. In *Proc. IEEE Visualization*, Phoenix, Arizona, Oct. 19 - 24 1997. `ftp://cs.nyu.edu/pub/local/yap/visual/thinwire.ps.gz`.

[Dau88] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41:909–996, 1988.

[DCC98] *Proc. Data Compression Conference (DCC)*, Snowbird, Utah, March 30 – April 1 1998.

[DCC99] *Proc. Data Compression Conference (DCC)*, Snowbird, Utah, March 29 – 31 1999.

[Deu96a] P. Deutsch. RFC 1951: DEFLATE compressed data format specification version 1.3. `http://www.ietf.org/rfc/rfc1951.txt`, May 1996.

[Deu96b] P. Deutsch. RFC 1952: GZIP file format specification version 4.3. `http://www.ietf.org/rfc/rfc1952.txt`, May 1996. Siehe auch: [Deu96a].

[DK91] M.J. Dürst and T.L. Kunii. Progressive transmission increasing both spatial and gray scale resolution. In *Proc. International Conference on Multimedia Information Systems*, pages 175–186, Singapore, 1991. McGraw-Hill.

[DLMP93] N.D. Degan, R. Lancini, P. Migliorati, and S. Pozzi. Still images retrieval from a remote database: the system Imagine. *Signal Processing: Image Communication*, 5, 1993.

[Dre87] H.M. Dreizen. Content-driven progressive transmission of gray-scale images. *IEEE Transactions on Communications*, 35(3):289–296, March 1987.

[Duc97] A.T. Duchowski. Representing multiple regions of interest with wavelets. In *Proc. SPIE*, volume 3309, page 975ff., 1997. `http://www.cs.tamu.edu/research/vislab/docs/vcip98.ps.gz`.

[Dür93] M.J. Dürst. Progressive image transmission for multimedia applications. In N. Magnentat Thalmann and D. Thalmann, editors, *Virtual Worlds and Multimedia*, pages 57–68. John Wiley & Sons, Chichester, UK, 1993.

[Dür97] M.J. Dürst. The progressive transmission disadvantage. *IEEE Transactions on Information Theory*, 43(1):347–350, Jan. 1997.

[EA95] A. Eleftheriadis and D. Anastassiou. Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video. In *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 95–106, Durham, New Hampshire, April 1995. Reprinted in Lecture Notes in Computer Science Series, Vol. 1018, Springer, 1995.

[EFK95] J. L. Encarnação, M. Frühauf, and T. Kirste. Mobile visualization: Challenges and solution concepts. In *Proc. CAPE'95*, Bejing, China, May 1995.

[EKS96] J. L. Encarnação, T. Kirste, and R. Strack. Visualisierung und Interaktion im Zeitalter des Mobile Computing. *it+ti – Informationstechnik und Technische Informatik*, 38(3):41–47, Juni 1996.

[Ern94] Bruno Ernst. *Der Zauberspiegel des M. C. Escher*. Taschen Verlag, Köln, 1994.

[FB96] A Fox and E.A. Brewer. Reducing WWW latency and bandwidth requirements by real-time distillation. In *Proc. 5th International World Wide Web Conference*, Paris, France, May 6-10 1996. `http://www.igd.fhg.de/www/www95/proceedings/posters/subject.html`.

[FCG] FastCGI homepage. `http://www.fastcgi.com`.

[Fei95] Andreas Feininger. *Das ist Fotografie*. Verlag Fotografie, Schaffhausen, 1995.

[Fis94] Y. Fisher. *Fractal Image Compression: Theory and Application*. Springer, New York, 1994.

[FJS96] A. Finkelstein, C.E. Jacobs, and D.H. Salesin. Multiresolution video. In *Proc. SIGGRAPH '96*, pages 281–290, New Orleans, Louisiana, August 4-6 1996.

[Fou95] A. Fournier. Wavelets and their applications in computer graphics. SIGGRAPH Course Notes, 1995. `http://www.cs.ubc.ca/nest/imager/contributions/bobl/wvlt/download/notes.ps.Z.saveme`.

[FPX97] FlashPix format specification, Version 1.0.1. The Digital Imaging Group, 1996, 1997. Specification available from `http://www.digitalimaging.org/i\_flashpix.html`.

[Fro97] Konrad Froitzheim. *Multimedia-Kommunikation, 1. Auflage*. dpunkt Verlag für digitale Technologie, Heidelberg, 1997. ISBN 3-920993-61-6.

[FS93] T.A. Funkenhouser and H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics Proceedings, Annual Conference Series*, 1993.

[FSZ97] T. Frajka, P.G. Sherwood, and K. Zeger. Progressive image coding with spatially variable resolution. In *Proc. IEEE International Conference on Image Processing*, volume 1, page 53ff., Santa Barbara, California, October 1997.

[Fur82] G.W. Furnas. The fisheye view: a new look at structured files. Technical memorandum 82-11221-22, Bell Labs, Oct 18 1982. `http://www.si.umich.edu/~furnas/POSTSCRIPTS/FisheyeOriginalTM.ps`.

[FvDFH90] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics – Principles and Practice, Second Edition*. Addison-Wesley, 1990. ISBN 0-201-12110-7.

[GAW90] R.S. Gentile, J.P. Allebach, and E. Walowit. Quantization of color images based on uniform color spaces. *Journal Imaging Tech.*, 16(1):11–21, Feb. 1990.

[GIF87] Spezifikation des GIF87a-formates. http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/GIF87a.txt, 1987.

[GIF89] Spezifikation des GIF89a-formates. http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/GIF89a.txt, 1989.

[Gol66] S. W. Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, IT-12:399–401, July 1966.

[GP90] M. Gervautz and W. Purgathofer. A simple method for color quantization: Octree quantization. In A.S. Glassner, editor, *Graphics Gems*, volume I, pages 287–293. Academic Press, San Diego, 1990.

[GR98] J.R. Goldschneider and E.A. Riskin. Joint optimal bit allocation and best-basis selection for wavelet packet trees. In *Proc. IEEE ICASSP'98*, page 2693ff., Seattle, Washington, USA, May 12-15 1998.

[Gra72] R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972. beschrieben in [CLR91].

[GT88] H. Gharavi and A. Tabatabai. Subband coding of monochrome and color images. *IEEE Transactions on Circuits and Systems*, 35(2), 1988.

[Haa10] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. *Math. Ann.*, 69:331–371, 1910.

[HDG92] Y. Huang, H.M. Dreizen, and N.P. Galatsanos. Prioritized DCT for compression and progressive transmission of images. *IEEE Transactions on Image Processing*, 1:477–487, Oct. 1992.

[Hec82] P. Heckbert. Color image quantization for frame buffer display. *ACM, Computer Graphics (Proc. Siggraph)*, 16(3):297–307, July 1982.

[HJWJG83] F.S. Hill Jr., S. Walker Jr., and F. Gao. Interactive image query system using progressive transmission. *Computer Graphics*, 17(3), 1983.

[HK97] I. Höntsch and L. Karam. Apic: Adaptive perceptual image coding based on subband decomposition with locally adaptive perceptual weighting. In *Proc. IEEE International Conference on Image Processing (ICIP97)* [ICI97].

[Huf52] D. Huffman. A method for the construction of minimum redundancy codes. In *Proc. IRE 40*, pages 1098–1101, 1952.

[ICI97] *Proc. IEEE International Conference on Image Processing (ICIP)*, Santa Barbara, California, Oct. 26-29 1997.

[ICI98] *Proc. IEEE International Conference on Image Processing (ICIP)*, Chicago, Illinois, October 4-7 1998.

[ICI99] *Proc. IEEE International Conference on Image Processing (ICIP)*, Kobe, Japan, Oct. 25-28 1999.

[IIP97] Internet imaging protocol, Version 1.05, 1997. http://www.digitalimaging.org/i_iip.html.

[IK96] T. Imielinski and H.F. Korth, editors. *Mobile Computing*. Kluwer Academic Publishers, Boston, 1996.

[IQW98] Intel quickweb technology. Whitepaper, Intel Corporation, 1998. http://www.intel.com/quickweb/.

[ISK98] J. In, S. Shirani, and F. Kossentini. JPEG compliant efficient progressive image coding. In *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP98)*, pages 2633–2636, Seattle, Washington, USA, May 12-15 1998.

[Jar73] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2:18–21, 1973. Beschrieben in [CLR91].

[JBG93] ITU-Recommendation T.82 – Information technology - Coded representation of picture and audio information - progressive bi-level image compression. Beuth Verlag, Abteilung Auslandsnormservice, Berlin, March 1993. http://www.itu.int/itudoc/itu-t/rec/t.html.

[JPGa] Digital Compression and Coding of Continuous Tone Still Images, Part 1, Requirements and Guidelines, ISO/IEC IS 10918-1. Reprinted in [PM93].

[JPGb] Digital Compression and Coding of Continuous Tone Still Images, Part 2, Compliance Testing, ISO/IEC IS 10918-2. Reprinted in [PM93].

[JPG96] ITU-Recommendation T.84 – Information Technology – Digital compression and coding of continuous-tone still images, Extensions. Beuth Verlag, Abteilung Auslandsnormservice, Berlin, July 1996. `http://www.itu.int/itudoc/itu-t/rec/t.html`.

[JPG99a] JPEG 2000 image coding system. Committee Draft CD15444-1, Version 1.0, ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8), 9 December 1999. `http://www.jpeg.org/JPEG2000.htm`.

[JPG99b] JPEG 2000 requirements and profiles. Draft, revision 5.0, ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8), 1999. `http://www.jpeg.org/public/wg1n1271.pdf`.

[K+98] Thomas Kirste et al. Projekt Mobile Visualisierung (MoVi), Projektphase II. Projektbericht, DFG-Forschergruppe Mobile Visualisierung, Rostock und Darmstadt, 1998.

[KCKH95] M.-B. Kim, Y.-D. Cho, D.-K. Kim, and N.-K. Ha. On the compression of medical images with regions of interest (ROIs). In *Proc. SPIE Vis. Comm. and Img. Proc.*, volume 2501 Part 1, pages 733–744, Taipei, Taiwan, May 1995.

[KCKK96] Jae D. Kim, Sang-Tae Choi, Jong-Seog Koh, and Soon-Hong Kwon. Progressive imaging on the internet via WWW browser using adaptive block truncation coding. In *Proc. IS&T/SPIE Symposium*, San Jose, 1996.

[Kea98] T.A. Keahey. The nonlinear magnification homepage. `http://www.cs.indiana.edu/hyplan/tkeahey/research/nlm/nlm.html`, Oct. 26 1998. Version 1.5.

[KG96] P. Kortum and W.S. Geisler. Implementation of a foveated image coding system for image bandwidth reduction. In *Proc. SPIE, Human Vision and Electronic Imaging*, volume 2657, pages 350–360, 1996.

[Kno80] K. Knowlton. Progressive transmission of gray-scale and binary pictures by simple, efficient, and lossless encoding schemes. *Proc. IEEE*, 68(7):885–896, July 1980.

[Knu85] D.E. Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6(2):163–180, June 1985.

[KR96] T.A. Keahey and E.L. Robertson. Techniques for non-linear magnification transformations. In *Proc. IEEE Symposium on Information Visualization, IEEE Visualization*, pages 38–45, October 1996. `http://www.cs.indiana.edu/hyplan/tkeahey/research/papers/infovis.96.html`.

[L+] Thomas G. Lane et al. The public domain JPEG codec of the independent JPEG group, V 6a. `ftp://ftp.uu.net/graphics/jpeg`.

[Lau99] J. Laun. PTI - Ein System zur stufenweisen Übertragung von Echtfarbenbildern im WWW. Schriftliche Arbeit im Wettbewerb Jugend Forscht '99, 1999. `http://server.hvzgymn.wn.schule-bw.de/pti/`.

[LCK95] J. Li, P.-Y. Cheng, and C.C.J. Kuo. On the improvements of embedded zerotree wavelet (EZW) coding. In *Proc. SPIE*, volume 2501 Part 3, pages 1490–1501, Taiwan, May 1995.

[Lee96] H.J. Lee. Imaging for the Internet: High-resolution graphics with FlashPix and the PsiFi Internet Imaging Protocol. *Web Techniques, Online Magazine*, December 1996. `http://www.webtechniques.com/archives/1996/12/lee/`.

[Lee97] H.J. Lee. Too many pixels, not enough bandwidth, June 1997. `http://www.webreview.com/97/06/06/feature/index.html`.

[LJ83] G. G. Langdorn Jr. An adaptive run-length coding algorithm. *IBM Technical Disclosure Bulletin*, 26:3783–3785, December 1983.

[LKB98] W. Li, J. H. Kasner, and M. Boliek. Region of interest coding. Document number N892, ISO/IEC JTC 1/SC 29/WG 1 (ITU-T SG8), June 1998.

[Loh84] H. Lohscheller. A subjectively adapted image communication system. *IEEE Transactions on Communications*, 32:1316–1322, December 1984.

[LPBD95] V. Lalich-Petrich, G. Bhatia, and L. Davis. Progressive image transmission capability (PROTRAC) on the World Wide Web. In *Proc. 3rd Int. WWW Conference*, Darmstadt, Germany, 1995. Posters and Demos, `http://www.umiacs.umd.edu/users/lpv/HTML/PROTRAC/protrac.html`.

[LRP95] J. Lamping, R. Rao, and P Piroli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. CHI*, 1995. `http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/jl_bdy.htm`.

[LSI] Lightning strike wavelet image compression. `http://www.infinop.com/`.

[Lub95] J. Lubin. A visual discrimination model for imaging system design and evaluation. Technical report, David Sarnoff Research Center, Princeton, NJ, February 1995.

[LWF] Bildkompressionsverfahren LuRaWave. `http://www.luratech.com`.

[Mal89] S.G. Mallat. A theory for multiresolution signal decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[Mal99] H. S. Malvar. Fast progressive wavelet coding. In *Proc. Data Compression Conference (DCC99)* [DCC99], pages 336–343. `http://www.research.microsoft.com/~malvar/papers/index.htm`.

[MB94] A.J. Maeder and D.M. Bell. Modeling colour refinement for progressive image coding. In *Proc. Second IS&T/SID Color Imaging Conference*, pages 111–114, November 1994.

[MC97] D. Marpe and H.L. Cycon. Efficient pre-coding techniques for wavelet-based image compression. Presented at Picture Coding Symposium '97, 1997. `ftp://ftp.rz.fhtw-berlin.de/pub/fhtw/fb3/pcs97.ps.gz`.

[MeV] MeVis Technology. `http://www.mt.mevis.de/`,`http://www.mevis.de/`.

[Mey93] Y. Meyer. *Wavelets: Algorithms & Applications*. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1993. Translated and revised by Robert D. Ryan.

[MFSW97] M. Merz, K. Froitzheim, P. Schulthess, and P. Wolf. Iterative transmission of media streams. In *Proc. ACM Multimedia 97*, Seattle, USA, November 8-14 1997. `http://www.acm.org/sigmm/MM97/papers/merz/text.htm`.

[Mil95] T. Milde. *Videokompressionsverfahren im Vergleich, 1. Auflage*. dpunkt Verlag für digitale Technologie, Heidelberg, 1995.

[MJ91] M. Malak and Baker J. Am image database for low bandwidth communication links. In *Proc. Data Compression Conference*, Snowbird, Utah, March 1991.

[MKA96] M. Miyahara, K. Kotani, and V. R. Algazi. Objective picture quality scale (PQS) for image coding. Technical report, Center for Image Processing and Integrated Computing, 1996. Submitted to IEEE Trans. on Communications. `http://info.ece.ucdavis.edu/scripts/bbl2html`.

[Mül98] Wolfgang Müller. *Effizienter Einsatz grundlegender Darstellungsprimitive zur Informationsvisualisierung*. Dissertation, Technische Hochschule Darmstadt, Fachbereich Informatik, 1998.

[MTS97] N. Magnentat Thalmann and V. Skala, editors. *Proc. WSCG'97 – The Fifth International Conference in Central Europe on Computer Graphics and Visualization*, Plzen, Czech Republic, February 10-14 1997.

[NB95] B. Natarajan and Vasudev Bhaskaran. A fast approximate algorithm for scaling down images in the DCT domain. In *Proc. IEEE International Conference on Image Processing*, pages 241–243, Piscataway, N.J., October 1995. IEEE Press.

[NC98] D. Nister and C. Christopoulos. Lossless region of interest with a naturally progressive still image coding algorithm. In *Proc. IEEE International Conference on Image Processing (ICIP98)* [ICI98], pages 856–859.

[OCAV97] Antonio Ortega, Fabio Carignano, Serge Ayer, and Martin Vetterli. Soft caching: Web cache management for images. In *IEEE Signal Processing Society Workshop on Multimedia*, Princeton, NJ, June 1997. `http://sipi.usc.edu/~ortega/SoftCaching/mmsp97.html`.

[Ohm95] J.-R. Ohm. *Digitale Bildcodierung. Repräsentation, Kompression und Übertragung von Bildsignalen*. Springer, Berlin Heidelberg, 1995.

[OTW+99a] Erik Ordentlich, David Taubman, Marcelo Weinberger, Gadiel Seroussi, and Michael Marcellin. Memory efficient scalable line-based image coding. Technical Report HPL-1999-1, HP Laboratories, Hewlett-Packard, 1999. `http://www.hpl.hp.com/techreports/1999/HPL-1999-1.html`.

[OTW+99b] Erik Ordentlich, David Taubman, Marcelo Weinberger, Gadiel Seroussi, and Michael Marcellin. Memory efficient scalable line-based image coding. In *Proc. Data Compression Conference (DCC99)* [DCC99].

[OWS97] Erik Ordentlich, Marcelo Weinberger, and Gadiel Seroussi. A low-complexity modeling approach for embedded coding of wavelet coefficients. Technical Report HPL-97-150, HP Laboratories, Hewlett-Packard, 1997. `http://www.hpl.hp.com/techreports/97/HPL-97-150.html`.

[OWS98] Erik Ordentlich, Marcelo Weinberger, and Gadiel Seroussi. A low-complexity modeling approach for embedded coding of wavelet coefficients. In *Proc. Data Compression Conference (DCC98)* [DCC98].

[OZV94] A. Ortega, Z. Zhang, and M. Vetterli. A framework for the optimization of a multiresolution remote image retrieval system. In *Proc. IEEE Infocom*, pages 672–679, Toronto, June 1994.

[P+] Jef Poskanzer et al. The NetPBM graphics library and tools. `ftp://sunsite.unc.edu/pub/Linux/apps/graphics/convert/netpbm-8.0.tar.gz`, `ftp://ftp.cc.gatech.edu/pub/linux/apps/graphics/convert/netpbm-8.0.tar.gz`.

[Pae91] A.W. Paeth. Image file compression made easy. In James Arvo, editor, *Graphics Gems II*. Academic Press, San Diego, 1991.

[PB99] S. Pigeon and Y. Bengio. Binary pseudowavelets and applications to bilevel image processing. In *Proc. Data Compression Conference (DCC99)* [DCC99]. `http://www.iro.umontreal.ca/~pigeon/pub/dcc99.ps`.

[PG88] W. Purgathofer and M. Gervautz. A simple method for color quantization: Octree quantization. In N.; Magnenat-Thalmann and D. Thalmann, editors, *New Trends in Computer Graphics*, pages 219–231. Springer-Verlag, Berlin, 1988.

[PIC] Pegasus imaging corporation. `http://www.pegasusimaging.com/`, `http://www.jpg.com/imagetech\_jpeg.htm`.

[PM93] W.B. Pennebaker and J.L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.

[PNG] Spezifikation des PNG-formates. `http://www.wco.com/~png/`, `http://www.w3.org/TR/REC-png`.

[PT94] Lai Man Po and Wen-Tao Tan. Block address predictive color quantisation image compression. *IEE Electornic Letters*, 30(2):120–121, 20 Jan 1994.

[PTC94] Lai Man Po, Wen-Tao Tan, and Chi-Ho Chan. Address predictive color quantization image compression for multimedia applications. In *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, volume 5, pages 289–292, 1994.

[R+99] R. Rösel et al. Verfahren zur mehrdimensionalen, diskreten Wavelet-Transformation und Transformationseinheit zur Durchführung des Verfahrens. Patentschrift DE 197 44 407 C 1, Deutsches Patentamt, 1999.

[Rau96a] S. Lange, U. Rauschenbach, and H. Schumann. Alternatives for the presentation of information in a mobile environment. In *Proc. IMC'96 Workshop on Information Visualization and Mobile Computing*, Rostock, Germany, February 1996.

[Rau96b] U. Rauschenbach and T. Kirste. A presentation model for mobile information visualization. *Computers and Graphics*, 20(5), 1996.

[Rau96c] U. Rauschenbach, R. Schultz, and H. Schumann. Quality and resource controlled transmission of images. In B. Urban, editor, *Multimedia '96 – Proceedings of the EUROGRAPHICS workshop in Rostock*, Rostock, Germany, May 28-30 1996. Springer Wien NewYork. ISBN 3-211-82876-1.

[Rau97] U. Rauschenbach and H. Schumann. Adaptive image transmission. In Magnentat Thalmann and Skala [MTS97].

[Rau98a] U. Rauschenbach. Progressive image transmission using levels of detail and regions of interest. In *Proc. CGIM'98 – IASTED Conference on Computer Graphics and Imaging*, Halifax, Nova Scotia, Canada, June 1-4 1998.

[Rau98b] U. Rauschenbach and H. Schumann. Flexible embedded image communication using levels of detail and regions of interest. In *Proc. IMC '98 – Interactive Applications of Mobile Computing*, Rostock, Germany, November 24-25 1998.

[Rau99a] U. Rauschenbach. The Rectangular Fish Eye View as an efficient method for the transmission and display of large images. In *Proc. IEEE International Conference on Image Processing (ICIP99)* [ICI99].

[Rau99b] U. Rauschenbach and H. Schumann. Demand-driven image transmission with levels of detail and regions of interest. *Computers and Graphics*, 23(6):857–866, December 1999.

[Rau00a] U. Rauschenbach. Compression of palettized images with progressive coding of the color information. In *Proc. SPIE Visual Communications and Image Processing*, Perth, Australia, 20-23 June 2000.

[Rau00b] U. Rauschenbach, S. Jeschke, and H. Schumann. General Rectangular FishEye Views for 2D graphics. Submitted to IMC'2000 - Workshop on Intelligent Interactive Assistance and Mobile Computing, Rostock-Warnemünde, Germany, November 9-10 2000.

[Rau00c] U. Rauschenbach, T. Weinkauf, and H. Schumann. Interactive focus and context display of large raster images. In *Proc. WSCG'2000, The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, Plzen, Czech Republic, February 7-11 2000.

[Rau01] U. Rauschenbach, R. Rosenbaum, and H. Schumann. A flexible polygon representation of multiple overlapping regions of interest for wavelet-based image coding. To be submitted to IEEE International Conference on Image Processing ICIP2001, Thessaloniki, Greece, October 7-10 2001.

[RC98] J. Rogers and P. Cosman. Robust wavelet zerotree image compression with fixed-length packetization. In *Proc. IEEE Data Compression Conference*, Snowbird, Utah, March 30 - April 1 1998. `http://lark.ucsd.edu/~jkrogers/packets/`.

[RR96] T.H. Reeves and J.A. Robinson. Adaptive foveation of MPEG video. In *Proc. 4th ACM International Multimedia Conference*, 1996.

[SA91] E. Simoncelli and E. H. Adelson. Subband transforms. In J. Woods, editor, *Subband Image Coding*, pages 143–192. Kluwer Academic Publishers, Norwell, MA, 1991. `http://www-bcs.mit.edu/people/adelson/publications/`.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2), 1984.

[SB97] Eero P. Simoncelli and Robert W. Buccigrossi. Embedded wavelet image compression based on a joint probability model. In *Proc. IEEE International Conference on Image Processing (ICIP97)* [ICI97]. `http://www.cns.nyu.edu/~eero/publications.html`.

[SC99] Skodras and C. Christopoulos. The JPEG 2000 tutorial at [ICI99], 1999. `http://etro.vub.ac.be/~chchrist/jpeg2000\_contributions.htm`.

[Sha93] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, Dec. 1993.

[Sha95] J.M. Shapiro. Apparatus and method for compressing information. U.S. Patent 5,412,741, May 1995.

[Sha96a] J.M. Shapiro. Apparatus and method for emphasizing a selected region in the compressed representation of an image. U.S. Patent 5,563,960, October 1996.

[Sha96b] J.M. Shapiro. A fast technique for identifying zerotrees in the EZW algorithm. In *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, volume 3, page 1455ff., 1996.

[SJT79] K. R. Sloan Jr. and S. L. Tanimoto. Progressive refinement of raster images. *IEEE Transactions on Computers*, C-28(11):871–874, November 1979.

[SL97] A. Signoroni and R. Leonardi. Progressive ROI coding and diagnostic quality for medical image compression. In *Proc. SPIE*, volume 3309, pages 674–685, 1997.

[SMG84] A. Sanz, C. Munoz, and N. Garcia. Approximation quality improvement techniques in progressive image transmission. *IEEE Journal on Selected Areas in Communications*, SAC-2(2):359–373, March 1984.

[SN97] G. Strang and T. Nguyen. *Wavelets and filter banks*. Wellesley: Wellesley-Cambridge Press, Mass., 1997. ISBN 0-9614088-7-1.

[SP96a] A. Said and W.A. Pearlman. An image multiresolution representation for lossless and lossy image compression. *IEEE Transactions on Image Processing*, 5:1303–1310, Sept. 1996.

[SP96b] A. Said and W.A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[SR93] B.C. Smith and L.A. Rowe. Algorithms for manipulating compressed images. *Computer Graphics and Applications*, 13(5):34–42, September 1993.

[SS95] B. Shen and I.K. Sethi. Inner-block operations on compressed images. In *Proc. ACM Multimedia 95*, San Francisco, California, 1995. `http://www.cs.wayne.edu/~bos/acm.html`.

[SSM97] T. Strutz, H. Schwarz und E. Müller. Bildcodierung durch hierarchische Prädiktion im Wavelet-Bereich. *FREQUENZ*, 51(3-4):106–115, 1997.

[SSTR93] M. Sarkar, S.S. Snibbe, O. Tversky, and S.P.. Reiss. Stretching the rubber sheet. In *Proc. ACM Symposium on User Interface Software and Technology*, 1993. `http://www.cs.brown.edu/publications/techreports/reports/CS-93-39.html`.

[Ste91] J.E. Steinhart. Scanline coherent shape algebra. In J. Arvo, editor, *Graphics Gems*, volume II, pages 31–45. Academic Press, San Diego, 1991.

[Str97] T. Strutz. *Untersuchungen zur skalierbaren Kompression von Bildsequenzen bei niedrigen Bitraten unter Verwendung der dyadischen Wavelet-Transformation*. Dissertation, Universität Rostock, Fakultät für Ingenieurwissenschaften, 1997. Veröffentlicht bei: Shaker Verlag, Aachen-Maastricht, 1998, ISBN 3-8265-3600-2.

[Swe95] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A.F. Laine and M. Unser, editors, *Proc. SPIE: Wavelet Applications in Signal and Image Processing III*, volume 2569, pages 68–79, 1995. `http://cm.bell-labs.com/cm/ms/who/wim/papers/papers.html`.

[Swe96a] W. Sweldens. The lifting scheme: a custom design construction of biorthogonal wavelets. *Journal of Applied and Computational Harmonic Analysis*, 1996.

[Swe96b] W. Sweldens. Wavelets and the lifting scheme: A 5 minute tour. *Zeitschrift für Angewandte Mathematik und Mechanik*, 76 (Suppl. 2):41–44, 1996. `http://cm.bell-labs.com/cm/ms/who/wim/papers/papers.html`.

[SWL97] D. Shin, H.-H. Wu, and J.-C. Liu. A region of interest (ROI) based wavelet compression scheme for medical images. In *Proc. SPIE*, volume 3031, pages 790–798, 1997.

[SZB95] E.L. Schwartz, A. Zandi, and M. Boliek. Implementation of compression with reversible embedded wavelets. In *Proc. SPIE 40th Annual Meeting*, volume 2564, San Diego, CA, July 1995.

[Tan79] S.L. Tanimoto. Image transmission with gross information first. *Computer Graphics and Image Processing*, 9(1):72–76, Jan. 1979.

[Tau99a] David Taubman. High performance scalable image compression with EBCOT. Submitted to IEEE Transactions on Image Processing, August 1999.

[Tau99b] David Taubman. High performance scalable image compression with EBCOT. In *Proc. IEEE International Conference on Image Processing (ICIP99)* [ICI99].

[TBC97] S. Thillainthan, D. Bull, and N. Canagarajah. Robust embedded zerotree wavelet coding algorithm. In *Proc. SPIE*, volume 3309, page 58ff., 1997.

[TZ94] D. Taubmann and A. Zakhor. Multirate 3d subband coding of video. *IEEE Trans. Img. Proc.*, 3(5):572–588, September 1994.

[Tzo87] K.-H. Tzou. Progressive image transmission: A review and comparison of techniques. *Optical Engineering*, 26:581–589, July 1987.

[Ver95] O.A. Verevka. Color image quantization in windows systems with local k-means algorithm. In *Proc. VI-th Western Computer Graphics Symposium*, pages 74–79, March 1995.

[Wat93] A.B. Watson. Dct quantization matrices visually optimized for individual images. In *Proc. SPIE*, volume 1913, pages 202–216, 1993.

[Wel84] T. A. Welch. A technique for high-performance data compression. *IEEE Computer*, 17(6):8–19, June 1984.

[WI] Summus wavelet image format. `http://www.summus.com/`.

[Wic94] M.V. Wickerhauser. *Adapted Wavelet Analysis from theory to software*. A.K. Peters Ltd, 1994.

[WNC87] I.H. Witten, R.M. Neal, and J.G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[Wol90] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, 1990.

[WR94] P. Waldemar and T.A. Ramstad. Subband coding of color images with limited palette size. In *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, volume 5, pages 353–356, April 1994.

[WS99] M. J. Weinberger and G. Sapiro. From LOCO-I to the JPEG-LS standard. In *Proc. IEEE International Conference on Image Processing (ICIP99)* [ICI99].

[Wu97] Xiaolin Wu. High order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression. Technical report, The University of Western Ontario, 1997. `http://www.csd.uwo.ca/faculty/wu/ececow.ps`.

[WVOC97] Claudio Weidmann, Martin Vetterli, Antonio Ortega, and Fabio Carignano. Soft caching: Image caching in a rate-distortion framework. In *Proc. IEEE International Conference on Image Processing (ICIP97)* [ICI97]. `http://lcavwww.epfl.ch/~weidmann/softcache/index.html`.

[YGHS96] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd. Filters: QoS support mechanisms for multipeer communications. *IEEE Journal on Selected Areas in Computing (JSAC) special issue on Distributed Multimedia Systems and Technology*, 14(7):1245–1262, September 1996.

[YLLC97] T. Yu, N. Lin, S.J. Liu, and A.K. Chan. A region-of-interest based transmission protocol for wavelet-compressed medical images. In *Proc. SPIE*, volume 3078, pages 56–64, 1997.

[YMGH96] N. Yeadon, A. Mauthe, F. Garcia, and D. Hutchison. QoS filters: Addressing the heterogeneity gap. In *Proc. Interactive Multimedia Systems and Services (IDMS96)*, Berlin, 4-6 March 1996.

[ZASB95] A. Zandi, J.D. Allen, E.L. Schwartz, and M. Boliek. CREW: compression with reversible embedded wavelets. In *Proc. IEEE Data Compression Conference*, Snowbird, Utah, March 1995.

[ZBS+95] A. Zandi, M. Boliek, E.L. Schwartz, J.D. Gormish, and J.D. Allen. CREW lossless/lossy image compression. Contribution to ISO/IEC JTC 1.29.12 CRC-TR-9524, ISO/IEC JTC1/SC29/WG1, N196, June 30 1995. `http://yellow.crc.ricoh.com/pub/pub.compression.html\#crew`.

[Zha97] J. Zhang. Medium support in mobile computing. In *Proc. Conference on Distributed Multimedia Systems*, Vancouver, Canada, July 24-25 1997.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.

[ZL93] A. Zaccarin and B. Liu. A novel approach for coding color quantized images. *IEEE Transactions on Image Processing*, 2(4):442–453, 1993.

[ZV97] J. Zhang and T.P. Vuong. Using wavelet transformation to support an efficient and adaptive image data handling within a mobile environment. In Magnentat Thalmann and Skala [MTS97].

Thesen

1. Übertragung und Darstellung von Rasterbildern erlangen mit der zunehmenden Verbreitung portabler Rechner und GSM-basierter Datenfunkeinrichtungen auch in mobilen Umgebungen wachsende Bedeutung. In solchen Umgebungen stellen geringe Übertragungsbandbreite und eingeschränkte Displayfläche begrenzende Faktoren für Transfer und Anzeige großer Bilder dar, zu deren Handhabung neue Konzepte erforderlich sind.

2. Existierende Systeme zur adaptiven Bildübertragung ermöglichen durch Steuerung der Kodierung die Anpassung von Größe und Bandbreitenbedarf eines Bildes vor der Übertragung an die aktuelle Ressourcensituation, den Kontext. Reaktionen auf Kontextänderungen während laufender Übertragung sind jedoch nicht möglich. *Bedarfsgesteuerte Bildübertragung* erlaubt es dem Benutzer darüber hinaus, vor und während der Übertragung für wählbare Regionen im Bild (so genannte *Regions of Interest*) die benötigte Detaillierungsstufe (*Level of Detail*) festzulegen. Die Umsetzung dieses Konzeptes erfolgt durch hybride Lösungsansätze, bei der Bildkompressionsverfahren mit Regions of Interest und Levels of Detail kombiniert werden.

3. Das entwickelte formale Modell für Regions of Interest und Levels of Detail in der Rasterbildübertragung ermöglicht eine exakte Spezifikation des Übertragungsbedarfes und bildet die Grundlage für die Umsetzung einer redundanzfreien Verfeinerung. Eine Region of Interest (RoI) repräsentiert dabei eine zusammenhängende Pixelmenge im Bild. Die zu übertragende Detaillierungsstufe (LoD) einer RoI kann als Kombination der Merkmale *X-Auflösung*, *Y-Auflösung*, *Genauigkeit* und *Farbe* spezifiziert werden. Für die Übertragung ist jede RoI einzeln priorisierbar.

4. Als Basis der bedarfsgesteuerten Bildübertragung wird eine Erweiterung des Embedded-Zerotree-Wavelet-Kompressionsverfahrens vorgeschlagen. Dieses bietet gegenüber dem JPEG-Standard die Vorteile einer höheren Bildqualität bei niedrigen Bitraten, einer höheren erreichbaren Kompressionsrate, einer feinstufigen Skalierung der Bitrate und einer einfacheren Integration von RoIs durch bessere Lokalisierung. Es erfordert jedoch einen höheren Rechenaufwand.

5. Bei Verwendung des etablierten dyadischen Wavelet-Zerlegungsschemas kann die X-Auflösung minimal das Halbe und maximal das Doppelte der Y-Auflösung betragen. Für bestimmte Bildübertragungsanwendungen ist jedoch eine höhere Flexibilität erforderlich. Das vorgeschlagene *XY-unabhängige Dekompositionsschema* ermöglicht größere Abweichungen zwischen X- und Y-Auflösung, die für bestimmte Applikationen Effizienzgewinne bringen. Da das neue Schema auch zusätzliche Kosten hinsichtlich Rechenzeit, Speicherplatz und Artefakten verursacht, wird es nur in solchen Anwendungen benutzt, in denen der Effizienzgewinn die Kosten überwiegt.

6. Der entwickelte und implementierte waveletbasierte MOVIROILOD-Algorithmus zur bedarfsgesteuerten Übertragung von Graustufen- und Echtfarbbildern ermöglicht es, mithilfe von *RoI-Schedulern* Kodierung und Dekodierung zu steuern. Dabei gestattet

er die Einbettung von Steuerkommandos in den kodierten Datenstrom, um auch während einer laufenden Übertragung flexibel auf Bedarfsänderungen zu reagieren. Der Übertragungsbedarf kann somit sowohl im Vorfeld der Übertragung als auch während laufender Übertragung spezifiziert werden, indem durch eine *Verfeinerungsanforderung* neue RoIs definiert oder die Detaillierungsstufe bzw. Priorität existierender RoIs modifiziert werden.

7. Aufbauend auf die implementierte Software-Bibliothek LIBROILOD sind verschiedene Client-Server-Applikationen zur bedarfsgesteuerten Bildübertragung entworfen und realisiert worden. Die Benutzerrollen Bildautor und Bildbetrachter in Online-Diensten können auf der Basis von LIBROILOD mit einem Autoren- und einem Betrachtungswerkzeug unterstützt werden. Während ersteres vielfältige Einstellmöglichkeiten zur Aufbereitung von Bildern im Vorfeld der Übertragung bietet, unterstützt letzteres mit einer minimalistischen, schnell zu bedienenden Benutzungsschnittstelle die Signalisierung eines geänderten Bedarfs während laufender Übertragung.

8. Es wurde die Fokus-und-Kontext-Technik RECHTECKIGER FISHEYE-VIEW zur kombinierten Übertragung und Anzeige großer Bilder entwickelt, die Bildschirmfläche und Netzbandbreite effizient ausnutzt. Ein interessierender Bildausschnitt, der *Fokus*, wird in Originalgröße dargestellt und ist von *Kontextringen* umgeben, deren Platzbedarf durch verzerrte Darstellung verringert wird. Die Flexibilität des neu entwickelten XY-unabhängigen Dekompositionsschemas ermöglicht es, nur die Daten zu übertragen, die für die Darstellung benötigt werden. Somit führt die Einsparung von Bildschirmfläche auch zur Einsparung von Bandbreite.

9. Die entwickelte waveletbasierte Bildübertragung ist für Farbtabellenbilder ineffizient. Bilder dieser Klasse werden heute meist in den Bilddateiformaten interlaced GIF und interlaced PNG übertragen. Diese etablierten Formate gestatten eine progressive Verfeinerung der Auflösung. Das erfordert die Übertragung eines großen Teils der Datenmenge, bevor feine Details im Bild erkennbar sind. Für die in Online-Diensten häufig zur Navigation eingesetzten grafischen Menüs bedingt dies in mobilen Umgebungen eine lange Übertragungszeit und damit lange Wartezeiten für den Nutzer, bevor die Navigationsmenüs benutzbar sind.

10. Es wurde das neue Kodierungsverfahren MOVICOLORQ für Farbtabellenbilder entwickelt, das durch eine Verfeinerung der Farbtiefe diesen Nachteil nicht aufweist. Feine Details mit hohem Kontrast sind früh im Verlauf der Übertragung erkennbar, so dass eine Verkürzung der Wartezeit erzielbar ist. Dabei sind die erreichbaren Kompressionsraten für Farbtabellenbilder denen der Standardverfahren interlaced PNG und GIF überlegen bzw. gleichwertig.

11. In dieser Arbeit wurden generelle Lösungen zur Übertragung von Rasterbildern mit Regions of Interest und Levels of Detail in mobilen Umgebungen entwickelt. Speziell wurde Wert auf die Einsparung von Übertragungsbandbreite bei der Bildübertragung und von Bildschirmfläche bei der Präsentation der übertragenen Bilder gelegt. Das entwickelte formale Modell ist auf verschiedene Bildkodierungsverfahren anwendbar. Neben der vorgestellten eigenen Umsetzung für waveletkodierte Bilder ist beispielsweise auch eine Realisierung auf der Basis des neuen Bildkodierungsstandards JPEG2000 möglich.