Zusammenfassung

Diese Arbeit präsentiert einen effizienten Ansatz zur Kompression von Bewegtbilddaten mit Methoden der fraktalen Bildkodierung, beruhend auf einer Least-Squares Approximation von Bildblöcken. Der Ansatz umfaßt eine Quadtreepartitionierung, ein Konzept nichtkontrahierender Abbildungen zur Kodierung aufeinanderfolgender Teilbilder einer Bildsequenz und eine Farbtransformation zur Kodierung von Farbbildern. Die prototypische Implementierung zeigt die Realisierbarkeit des Ansatzes. Eine 101 Farbbilder umfassende Testsequenz läßt sich mit einer Kompressionsrate von 1:50 kodieren, jedoch verbunden mit einer deutlichen Degradation der Bildqualität.

Inhaltsverzeichnis

1 Einführung und Aufgabenstellung

1.1 Motivation

Seit Erfindung des Films gegen Ende des 19. Jahrhunderts, mehr noch nach der Entwicklung der Fernsehtechnologie seit den 20er-Jahren dieses Jahrhunderts, wurden bewegte Bilder immer mehr zum Bestandteil der Wahrnehmung und Erfahrung des (zivilisierten) Menschen. Die Bedeutung des Mediums Bewegtbilder für die Unterhaltung, Information und Kommunikation nimmt weiter zu. Indizien dafür sind: die steigende Anzahl von Fernsehkanälen, der Ausbau der entsprechenden Infrastruktur (Satelliten, Kabelnetze) und die massive Entwicklung neuer Technologien wie HDTV, Bildtelefon und Multimedia auf Computern.

Bei der Übertragung oder Speicherung digitalisierter Bildsequenzen fallen große Datenmengen an, verbunden mit entsprechend hohen Übertragungs- bzw. Speicherungskosten. Der Nutzen einer Bilddatenkompression läßt sich durch folgendes Beispiel anschaulich machen: ein Übertragungskanal mit einer für die Übertragung eines (unkomprimierten) Fernsehprogrammes ausreichenden Bandbreite, kann bei einer Kompression von 10:1 10 Sender gleichzeitig übertragen (ohne Ton und Synchronisiationsdaten zu berücksichtigen). Andererseits wird es auf Kanälen mit geringer Bandbreite (z.B. beim Bildtelefon) durch Kompression erst möglich, Bewegtbilder zu übermitteln.

Verlustfreie Kompressionsverfahren (Lauflängenkodierung, Huffman- und Arithmetische Kodierung, oder Varianten der Lempel-Ziv Kodierung etc.), die die statistische Redundanz in den Daten ausnützen, erreichen Kompressionsraten von 1:2 bis 1:5. Für Bilddaten läßt sich eine höhere Kompression mit verlustbehafteten Verfahren erreichen, die versuchen, für das menschliche Wahrnehmungssystem nichtwesentliche Information wegzulassen, oder eine spezifizierbar akzeptable Degradation zuzulassen. Wichtige Verfahren der verlustbehafteten Bildkompression sind DCT-basierte Verfahren (Diskrete CosinusTransformation, z.B. JPEG und MPEG), Vektor- Quantisierung (VQ), Wavelet Transformation und fraktale Kodierung.

1.2 Aufgabe

Die fraktale Bildkodierung approximiert ein gegebenes Bild durch ein fraktales Bild. Fraktale können für den Betrachter ein komplex erscheinendes Aussehen besitzen, auch wenn sie mathematisch nur durch wenige Formeln definiert sind. Ziel der fraktalen Bildkodierung ist es, einen fraktalen Code zu berechnen, dessen Attraktor (Fixpunkt der Iteration) ein gegebenes Bild möglichst gut annähert und dessen Informationsgehalt im Falle fraktaler Kompression deutlich geringer ist, als der des Orginalbildes.

Ein mögliches Verfahren zur Bestimmung eines fraktalen Codes (Collage-Ansatz) erfordert menschliche Interaktion: es besteht darin, jedes Motiv eines Bildes durch verkleinerte Kopien seiner selbst zu überdecken. Die verkleinerten Kopien stellen mathematisch gesehen kontrahierende affine Selbstabbildungen des Bildes dar. Die Theorie der fraktalen Bildkodierung sichert, daß die iterierte Anwendung dieser Abbildungen auf ein beliebig wählbares Startbild gegen einen Attraktor konvergiert, der das zu kodierende Bild approximiert.

Die publizierten automatischen Verfahren arbeiten dagegen blockbasiert. Ein Grauwertbild wird mit einem Partitionierungsschema in Blöcke (Dreiecke, Rechtecke, Quadrate usw.) geteilt, die nichtüberlappend das Bild vollständig überdecken. Für jeden dieser Blöcke (Ranges) wird je ein (größerer) Ursprungsblock (Domain) im Bild gesucht. Die Blocktransformationen, die den fraktalen Code darstellen, bestehen aus zwei Teilen: einem geometrischen Anteil, der eine affine Abbildung vom Domainblock auf den Rangeblock ist und Translation, Verkleinerung, Drehungen und Spiegelungen umfassen kann; und aus einem Anteil, der die Grauwerte des Domainblockes abbildet, so daß der Fehler, zwischen den transformierten Grauwerten des Domainblockes und denen des Rangeblockes klein wird.

Fraktale Bildkompression nützt die Selbstähnlichkeit von Bildteilen aus, um Bilddaten zu komprimieren. Das fraktale Bild ist skalierungsunabhängig, d.h. es läßt sich in beliebiger Größe wiedergeben. Während die Kodierungsphase sehr teuer sein kann, ist die Dekodierung relativ schnell und einfach möglich. Die Kodierergebnisse, insbesondere das Verhältnis von Kompressionsgrad zu gemachten Fehler, sind in verschiedenen Publikationen zur fraktalen Bildkodierung zum Teil besser als bei Standardverfahren wie JPEG, allerdings sind sie nur mit einem höheren Kodieraufwand erreichbar.

Das Ziel dieser Arbeit ist die Entwicklung eines fraktalen Kodierers und eines zugehörigen Dekodierers für Bewegtbilder. Ein großes Interesse besteht daran, einen effizienten und (eventuell) parallelisierbaren Algorithmus anzugeben. Die Implementierung dient zum Beweis der Realisierbarkeit.

2 Literatur zur fraktalen Bildkodierung

Das folgende Kapitel gibt einen Überblick über einen Teil der publizierten Literatur zur fraktalen Bildkodierung und versucht die historische Entwicklung dieser Technologie nachzuvollziehen. Dabei werden zentrale Begriffe aus der Theorie der fraktalen Bildkodierung wie "Attraktor", "Iterierte Funktionensysteme", "Partitionierung" usw. zunächst nur verwendet; eine exakte Definition folgt in Abschnitt 4.1.1.

2.1 Vorgeschichte

J. Hutchinson zeigte in einer Arbeit von 1981 [Hutchinson 81], daß Iterierte Funktionensysteme (IFS), deren Funktionen einer Kontraktivitätsbedingung genügen, einen eindeutigen Attraktor besitzen. Die Attraktoren, die in der Regel fraktale Mengen bilden, werden durch die Funktionen des IFS eindeutig beschrieben. Die Existenz und Eindeutigkeit des IFS-Attraktors gestattet die Erzeugung eines breiten Spektrums verschiedener Fraktale. Bemerkenswert an diesen Fraktalen ist, daß manche von ihnen natürlichen Objekten wie Pflanzen oder Wolken ähneln.

Begründer der Fraktalen Bildkodierung ist Michael F. Barnsley vom Georgia Institute of Technologie. Barnsley und seine Kollegen beschäftigten sich seit Mitte der 80er-Jahre mit IFS zur Erzeugung von Fraktalen [Barnsley et al. 85] und weckte das Interesse am inversen Problem, für ein gegebenes Bild ein IFS zu finden, dessen Attraktor das Bild approximiert.

Das Problem der Erzeugung von Fraktalen aus IFS war leicht zu lösen im Vergleich zum inversen Problem. Beispiele für die verschiedenen Algorithmen zur Erzeugung (d.h. auf einem Rechner Sichtbarmachen) von Attraktoren sind der deterministische Algorithmus, der Random-Walk-Algorithmus oder der Escape-Time-Algorithmus [Barnsley 88b].

Ein wichtiger Meilenstein für die Lösung des inversen Problems war der Beweis eines Collage-Theorems [Barnsley et al. 86]. Es besagt folgendes: überdeckt man ein Objekt mit affinen Verkleinerungen seiner selbst, so besitzt das IFS, das durch diese selbstaffinen Abbildungen festgelegt ist, einen Attraktor, der das Ausgangsobjekt approximiert.

Der nächste Schritt war die Erweiterung der Iterierten Funktionensysteme zu RIFS (Recurrent IFS) [Barnsley et al. 89] und der Beweis eines entsprechenden Collage-Theorems für RIFS. Global operierende IFS erzeugen immer exakt selbstaffine Attraktoren, während bei der Beschreibung natürlicher Objekte oft nur eine bedingte Selbstaffinität oder diese nur in Teilen der Objekte vorhanden ist. RIFS erweitern IFS um eine Menge von diskreten Zuständen, deren Übergänge mit Wahrscheinlichkeiten gewichtet sind. Anschaulich ausgedrückt, lassen sich damit einzelne Bildmotive unabhängig voneinander kodieren.

Diese Vorraussetzungen ermöglichen einen interaktiven Algorithmus zur fraktalen Bildkodierung: ein Bild wird in möglichst selbstähnliche Teile segmentiert, diese werden dann mit Collagen approximiert; anschließend werden die Attraktoren zu einem Gesamtbild zusammengesetzt. Gespeichert werden nur die Koeffizienten der Abbildungen. Vorteil des Ansatzes sind eine hohe Kompressionsrate und eine fraktale Beschreibung des Bildes zu erhalten, die unmittelbar mit dem Bildinhalt korreliert ist. Nachteile sind der hohe Aufwand und surrealistisch wirkende Approximationen.

Mit diesem Ergebnis traten Barnsley und seine Mitarbeiter im Jahre 1988 an eine breite Öffentlichkeit. In einem Artikel in der Computer-Fachzeitschrift "Byte" [Barnsley et al. 88b] unter dem Titel "A Better Way to Compress Images" stellte Barnsley zusammen mit Alan D. Sloan den damaligen Stand der fraktalen Bildkodierung vor. Mit den in Aussicht gestellten Kompressionsraten von 10 000 zu 1 und mehr und durch den in diesem Artikel angedeuteten Einsatz von IFS für die Bildanaylse, gewann die fraktale Bildkodierung ein breites Interesse (belegt durch die ständig steigende Zahl der Publikationen in den darauffolgenden Jahren).

Kritisch angemerkt muß hierbei werden, daß es bis zu diesem Zeitpunkt keinen vollautomatischen Algorithmus zur Berechnung der IFS-Codes für beliebige Eingabebilder gab. Zudem war der Rechenaufwand für die automatisierbaren Teile (das Finden von Collagen) groß. Barnsley und Sloan geben für die Kodierung eines komplexen Farbbildes (ohne zu definieren, was der Begriff "komplex" in diesem Zusammenhang bedeutet) einen Kodierungsaufwand von 100 Stunden auf einer Workstation mit 2 Prozessoren vom Typ Motorola 68020 an.

Bereits im Jahre 1987 hatten Barnsley und Sloan die Firma "Iterated Systems Inc." mit dem Ziel gegründet, fraktale Technologien kommerziell zu nutzen. Sie halten 2 U.S.-Patente [Barnsley et al. 90, Barnsley et al. 91], die die Grundlagen eines vollautomatischen Kodierungsverfahren patentrechtlich schützen.

Es gab verschiedene Publikationen zu der bis dahin vorhandenen Technologie der fraktalen Bildkodierung. S. F. Reddaway, A. Wilson und A. Horn erzeugten IFS-Fraktale auf einem massiv parallelen (SIMD) Rechner mit effizienten Algorithmen für diese Rechnerarchitektur [Reddaway et al. 88]; sie diskutieren auch Methoden zur fraktalen Kompression auf SIMD Rechnern (Collage-Ansatz, Momenten-Theorie), ohne jedoch einen dieser betrachteten Ansätze auszuführen.

Daneben gab es eine Reihe von Publikationen, die in der Nachfolge Barnsleys standen, aber keine neuen Erkenntnisse brachten. Beispiele aus dem deutschsprachigen Raum sind hierfür: [Gipser 90a, Gipser 90b, Meneghini et al. 91] in Zürich und [Graf 92] in Passau.

Erwähnenswert ist noch eine Arbeit von Masayuki Kawamata, Hiroaki Kanbara und Tatsuo Higuchi [Kawamata et al. 92]. Sie schlugen den Einsatz von speziellen Korrelationsoperatoren vor, die die affinen Transformationen der IFS berücksichtigen bzw. integrieren. Durch Berechnung einer Autokorrelationsfunktion lassen sich die Koeffizienten der geometrischen Abbildungen eines IFS gewinnen. Der Ansatz scheint aber nicht weitergeführt worden zu sein.

2.2 Automatische Kodierung

Den ersten vollautomatischen Algorithmus zur fraktalen Bildkodierung veröffentlichte 1989 Arnaud Jacquin, ein Doktorand Barnsleys, in seiner Dissertation am Georgia Institute of Technologie [Jacquin 89]. Der Ansatz Jacquins operiert auf Graustufenbildern bzw. aus mathematischer Sicht auf Grauwertfunktionen. Jacquin definiert eine selbstähnliche Abbildung der Grauwertfunktion eines Bildes blockweise: das zu kodierende Bild wird in nichtüberlappende Rangeblöcke R_k geteilt; zu jedem dieser Blöcke wird ein geeigneter (zu R_k affiner) Ursprungsblock D_k (Domainblock) für eine Blocktransformation gesucht.

Eine Blocktransformation zwischen zwei Blöcken setzt sich aus einer geometrischen (affinen) Komponente zusammen, die den Ursprungsblock auf den Bildblock hin verschiebt, verkleinert, dreht oder spiegelt und einer Transformation der Grauwerte, die einen additiven und einem multiplikativen Anteil umfaßt. Diese Grauwerttransformation läßt sich also schreiben als:

$$g' = a\,g + b$$

wobei g der Grauwert eines Punktes des Orginalblocks ist, g' der des entsprechenden Punktes auf dem Bildblock (nach geometrischer Transformation) und a, b die Koeffizienten der Grauwerttransformation sind. Während die geometischen Anteile als IFS aufgefaßt werden können, sind die Grauwerttransformationen ein zusätzliches Konzept gegenüber dem IFS-Ansatz.

Um geeignete Domainblöcke effizient zu finden, teilt Jacquin die Bildblöcke in drei Klassen ("edge", "midrange" und "shade"); die Suche nach geeigneten Domainblöcken beschränkt sich auf Blöcke derselben Klasse (wobei für Blöcke mit nahezu konstantem Grauwert nicht gesucht werden muß). Der Pool von Domainblöcken für Rangeblöcke der Seitenlänge B umfaßt bei Jacquin alle quadratischen Bildblöcke mit Seitenlänge $2B$, deren Seiten parallel zu den Rändern des Bildes sind und deren Ecken auf einem vorgegebenen (groben) Gitter liegen. Die Suche nach geeigneten Domainblöcken bleibt immer innerhalb einer Region um den Rangeblock, damit sich der Aufwand für die Suche in Grenzen hält. Jacquin vergleicht den Pool von Domainblöcken mit dem "Codebook" der Vektor-Quantisierung (VQ) und charakterisiert seinen Ansatz zur fraktalen Bildkodierung als Vektor-Quantisierung mit virtuellem "Codebook", das beim Kodieren aus dem Bild genommen wird und beim Dekodieren während der Iteration immer mehr verbessert wird.

Jacquin veröffentlichte in den folgenden Jahren mehrere Zusammenfassungen seiner Arbeit [Jacquin 90, Jacquin 92]. Darüberhinaus enthält [Jacquin 93] einen Überblick über die Erweiterungen, die sein Ansatz durch anderer Forscher erhielt.

Jacquins Arbeit war ein sehr wichtiger Meilenstein in der Enwicklungsgeschichte der fraktalen Bildkodierung. Sie ermöglichte einen vollautomatischen Kodierer, jedoch nur für Kompressionsraten in der Größenordnung von 1:10 – 1:30. Nahezu alle später publizierten Ansätze automatischer Kodierer gehen auf die Arbeit Jacquins zurück, gehören also zur Klasse der auf Grauwertbildern operierenden Blockkodierer.

2.3 Erweiterungen

Yuval Fisher arbeitete zusammen mit E. W. Jacobs und R. D. Boss seit ca. 1989 an fraktaler Bildkodierung. Sie erweiterten den Ansatz Jacquins um mehrere Punkte:

- statt festen Blockgrößen benutzten sie Quadtree-, HV- oder Dreieck- Partitionierung.
- für die Grauwerttransformationen wurden die Koeffizienten mit einem Least-Squares Kriterium gefunden [Fisher 92].
- die Kontraktivitätsbedingung konnte insofern gelockert werden als auch eventuell kontrahierende Transformationen (τ ist evtl. kontrahierend, wenn τ^n kontrahierend ist für ein $n \in \mathbb{N}$) einen eindeutigen Attraktor liefern [Fisher et al. 92].
- Verbesserung der Bildblock-Klassifikation. Blöcke werden in eine von bis zu 72 Klassen eingeteilt, um die Komplexität der Suche nach Domainblöcken zu verringern [Jacobs et al. 92].
- Nachbearbeitung des wiederhergestellten Bildes zur Vermeidung von Blockartefakten.

Fishers Buch über fraktale Bildkmpression ist für September 1994 angekündigt. Es umfaßt eine Einführung von Fisher zum Thema Kompression, eine Reihe aktueller Artikel zu Spezialthemen sowie lauffähige Beispielprogramme. Die Programme, das Inhaltsverzeichnis und Literaturliste des Buches wurden vorab veröffentlicht [Fisher 94]. Obwohl noch nicht erschienen, wird Fishers Buch vielfach zitiert und gelobt. Es ist zu erwarten, daß es zum Standardwerk für fraktale Kompression werden wird.

Mark Beaumont kodierte statt dem Orginalbild ein Differenzbild, das durch Subtraktion des mittleren Grauwerts je Bildblock gebildet wird [Beaumont 91]. Den mittleren Grauwert kann der Dekodierer aus den Koeffizienten der Grauwerttransformation zurückberechnen; dieser verursacht also kein zusätzliches Datenaufkommen. Der Dekodierer erhält ein aus den mittleren Grauwerten der Blöcke interpoliertes Bild als Startbild und erzielt dadurch eine schnelle Konvergenz, da das Ausgangsbild für die Iteration schon nahe am Attraktor liegt.

Die Arbeit von Geir E. Øien, Skalg Lepsøy und Tor A. Ramstad [Øien et al. 91] interpretiert die auf Blöcken definierten Grauwertfunktionen als Punkte eines Vektorraums der Dimension $B \times B$ wobei B die Seitenlänge eines Bildblocks bezeichne. Die Koeffizienten der Grauwerttransformation wurden durch Lösen eines Satzes von Orthogonalitätsgleichungen bestimmt.

Frank Dudbridge entwickelte in seiner Dissertation [Dudbridge 92] einen Ansatz, der ganz auf Suche nach geeigneten Ursprungsblöcken verzichte. Dudbridge benutzt allgemeine Funktionen für die Grauwerttransformation, also nicht nur Funktionen mit 2 Koeffizienten wie Jacquin. Die optimalen Koeffizienten der auf quadratischen Bildblöcken operiernden Grauwerttransformationen werden mit einem Least-Squares Kriterium analytisch bestimmt. Der geometrische Anteil der Blocktransformation beschränkt sich bei Dudbridge auf die Mittelung eines Bildblocks und Abbildung auf die 4 Teilblöcke. Der fraktale Kode, bestehend aus kontrahierenden Blocktransformationen, beschreibt eine invariante Grauwertfunktion, die das Orginalbild approximiert. Der Vorteil dieses Ansatzes liegt im geringen Kodieraufwand, mit dem Nachteil einer schlechteren Relation zwischen Kompressionsrate und Bildqualität.

Zusammen mit D. Monro publizierte Dudbridge mehrere kurze Artikel über einen Spezialfall dieses Ansatzes [Monro et al. 92b, Monro et al. 92a], der eine Grauwerttransformation mit 4 Koeffizienten benutzt:

$$g' = a\,x + b\,y + c\,g + d$$

wobei x und y die Koordinaten des abgebildeten Punktes darstellen. Monro vereinigt in späteren Publikationen [Monro 93b, Monro et al. 93] den Ansatz Dudbridges mit dem Jacquins, indem er eine Suche nach geeigneten Domainblöcken integriert. Er nennt diesen Ansatz “Bath Fractal Transform” und untersucht insbesondere eine Kosten - Nutzen Relation verschiedener Kombinationen von Suche und Komplexität der Grauwerttransformation [Monro 93a].

Carsten Frigaarg, Jess Gade, Thomas T. Hemmingsen und Torben Sand erweiterten die Klassifikation der Bildblöcke: statt einem Satz fester Kategorien werden Bildblöcke in einen kontinuierlichen Eigenschaftsraum abgebildet, der für verschiedene Parameter definiert werden kann. Der Abstand im Eigenschaftsraum zweier Blöcke ist ein Maß für die Ähnlichkeit der Blöcke gemäß der gewählten Parameter. Nach der Vorabklassifikation aller Domainblöcke werden bei der Suche nur die ersten n naheliegendsten Domainblöcke für jeden Rangeblock betrachtet.

Dietmar Saupe verringerte die algorithmische Komplexität der Suche durch Anwendung effizienter Algorithmen zur Suche von nächsten Nachbarn in mehrdimensionalen Räumen [Saupe 94], die mit einem Aufwand von $O(n \log n)$ durchgeführt werden kann (n = Anzahl der Bildblöcke). Dabei werden die Bildblöcke als Vektoren in einem entsprechenden Vektorraum aufgefaßt. Saupe veröffentlichte darüberhinaus eine Literaturübersicht [Saupe et al. 94], die eine Sammlung ausgewählter Abstracts und eine umfangreiche Literaturliste umfaßt.

Das Kodieren im Frequenzraum ist ein weiterer Ansatz. Kai Uwe Barthel und Thomas Voyé [Barthel et al. 94] kodierten mit einer modifizierten Grauwertabbildung im DCT-Frequenzraum und erreichten dadurch eine höhere Kompression und bessere Konvergenz als bei Kodierung im Ortsraum. Y. Zhao und B. Yuan [Zhao et al. 94] kodierten zusätzlich zur fraktalen Kodierung im DCT-Frequenzraum die Differenz zwischen fraktaler Approximation und Orginalbild mit einem DCT-Kompressionsverfahren und erreichten damit eine sehr gute Bildqualität bei hoher Kompressionsrate.

Die von Fisher vorgeschlagenen Dreieckspartitionen wurden von F. Davoine und J-M. Chassery aufgegriffen [Davoine et al. 94b, Davoine et al. 94a], die mit einer adaptiven Delaunay Triangulation ein gegebenes Bild in Dreiecke teilen, die gut ineinander transformiert werden können.

Lucia Vences und Osaac Rudomin setzten genetische Algorithmen zur fraktalen Kodierung ein [Vences et al. 94]. Sie erreichten damit einen Geschwindigkeitsvorteil gegenüber einem Algorithmus ohne Klassifikation von Bildblöcken.

Andreas Molisch entwickelte in seiner Studienarbeit [Molisch 94] an der Fakultät Informatik der Universität Stuttgart einen fraktalen Kodierer und Dekodierer für Farbbilder. Eine kurze Zusammenfassung dieser Arbeit befindet sich in Abschnitt 3.1.

2.4 Videokompression

Die Geschichte der fraktalen Videokompression erhält als Hauptthema dieser Arbeit einen eigenen Abschnitt. Viele Autoren erwähnen die Möglichkeit zur Erweiterung ihres Ansatzes auf 3-dimensionale Daten. Spezielle Publikationen zur Videokompression sind aber im Vergleich zu denen der Einzelbildkompression selten.

Barnsley erwähnte die Möglichkeit zu fraktalem Video in [Barnsley et al. 88b]. Zu dieser Zeit verfügte er über eine Sequenz von IFS-Codes, deren Attraktoren sich verändernde Wolken darstellten. Zur Visualisierung der IFS-Attraktoren wurde Speziealhardware benötigt, die IFS-Codes über eine serielle Schnittstelle mit geringer Bandbreite erhielt.

Eine naheliegende Erweiterung von Einzelbildkodierung auf Bildsequenzkodierung schlug Beaumont [Beaumont 91] vor: statt 2-dimensionale Bildblöcke können 3-dimensionale Blöcke aus der zu kodierenden Bildsequenz als Domain- und Rangeblöcke einer fraktalen Transformation benutzt werden.

Dieser Ansatz wurde von W. O. Cochran, J C. Hart und P. J. Flynn für Volumendaten aufgegriffen [Cochran et al. 94]. Für Volumendaten lassen sich Selbstähnlichkeiten verschiedener Orte bzw. Blöcke eines 3-dimensionalen Bildes ausnutzen.

Für Videokompression wurde dieses Verfahren nicht eingesetzt, weil bei Bewegtbildern die Korrelation zwischen aufeinanderfolgenden Einzelbildern stärker ausgenutzt werden kann als die Selbstähnlichkeit von Volumenblöcken. Das ohnehin aufwendige Verfahren der Suche nach geeigneten Domainblöcken wird beim Übergang in die nächsthöhere Dimension um eine weitere Größenordnung aufwendiger.

Die übrigen publizierten Ansätze zur Videokompression erweitern das Verfahren der Einzelbildkodierung auf folgende Weise: statt für Rangeblöcke eines Bildes Domainblöcke im selben Bild zu suchen, werden diese im vorangegangenen Bild der Sequenz gesucht. Dadurch wird eine Abbildung von einem Teilbild der Sequenz auf das jeweils nächste definiert, so daß bei der iterativen Anwendung der Transformationen in jedem Schritt ein Teilbild der Sequenz erzeugt wird.

Zusammen mit L. P. Hurd und M. A. Gustavus veröffentlichte Barnsley eine 2 Seiten umfassende Übersicht über ein implementiertes Verfahren zur fraktalen Videokompression [Hurd et al. 92]. Das erste Bild einer Sequenz wird mit der fraktalen Kompression für Einzelbilder oder mit jeder beliebigen anderen Kompressionstechnik kodiert. Die nachfolgenden Teilbilder werden sowohl mit kontrahierenden fraktalen Abbildungen als auch nichtkontrahierenden ("Carry-forward") Abbildungen kodiert. Für jeden Punkt der Bilder wird nur eine begrenzte Anzahl von Carry-forward Abbildungen zugelassen; dadurch konvergiert die approximierte Bildfolge auch im Falle eines Übertragungsfehlers gegen die kodierte Sequenz nach einer endlichen Anzahl von Schritten.

Zusammem mit D. Rogovin und T.P. Shen veröffentlichte Y. Fisher kürzlich eine Arbeit über Videokompression [Fisher et al. 94], die auf seinem Kodierungsschema für Einzelbilder aufbaut und für eine Beispielsequenz Kompressionsraten von 1 : 25 bis 1 : 244 bietet.

Ebenfalls zum Thema fraktale Videokompression arbeiten D. Monro, J. Nicholls und D. Wilson. Die beiden Publikationen [Wilson et al. 94, Monro et al. 94] waren bis zur Abgabe dieser Arbeit nicht beschaffbar.

2.5 Weitere Literatur und sonstige Quellen

An der Fakultät Informatik der Universität Stuttgart wurde im Sommersemester 1993 ein Hauptseminar zu Thema “Fraktale und Bildkodierung” abgehalten mit zwei Vorträgen zur fraktalen Bildkodierung. [Harrer 93] stellte die Grundlagen der Kodierung mit IFS und einfachen Transformationen dar, während [Meyer 93] auf Jacquins Ansatz basierende Blockkodierer behandelte.

Eine umfangreiche Quelle zur Theorie der Iterierten Funktionensysteme und der fraktalen Bildkodierung ist Barnsleys Buch “Fractals Everywhere” [Barnsley 88b].

Barnsley und Hurd veröffentlichten 1992 ein Buch [Barnsley et al. 92] speziell zur Bildkompression, mit dem Titel “Fractal image compression”. Sie erklären darin die mathematischen Grundlagen der IFS-Kodierung und der blockbasierten Kodierung, geben jedoch keinen konkreten (effizienten) Algorithmus an.

Es gab eine Reihe von Artikeln, die einen groben Überblick über fraktale Bildkompression (meist Jacquins Schema) geben. Darunter sind: [Ammeter et al. 93, Anson 93, Dettmer 92]. Daneben erschienen mehrere Artikel in Computermagazinen, die das kommerzielle Produkt von Barnsleys Firma “Iterated Systems Inc.” betrachteten [Apiki 93, Baran 93, Kesy 93].

Ein sehr wichtiges Medium für aktuelle Information aus Wissenschaft und Forschung ist das INTERNET. Anhang A zu dieser Arbeit gibt einen Überblick über Informationsquellen zur fraktalen Bildkodierung im INTERNET.

3 Lösungsansatz

Zweck dieses Abschnittes ist es, den gewählten Ansatz zur Lösung der Aufgabe zu skizzieren, ohne auf mathematische Vorraussetzungen und Details der Lösung einzugehen.

3.1 Vorrangegangene Arbeiten

Die Studienarbeit von Andreas Molisch [Molisch 94] war die erste Arbeit zum Thema "Fraktale Bildkodierung" an der Fakultät Informatik in Stuttgart. Ihre Aufgabe war die Entwicklung eines fraktalen Kodierers für Einzelbilder. Der in der Arbeit implementierte Ansatz basiert auf dem Jacquins, erweitert um HV-Partitionierung, ein spezielles Least-Squares Kriterium zur Bestimmung der Koeffizienten der Grauwerttransformationen, einer lokal eingeschränkten Suche und einer Nachbearbeitung des Attraktorbildes, jedoch ohne Klassifikation von Bildblöcken wie bei Jacquin. Der Ansatz bietet sehr gute Kompressionsraten bei einer sehr guten Bildqualität, ist aber bei der Kodierung aufwendig.

3.2 Ausgangspunkt und Entscheidungen

Das Ziel dieser Studienarbeit ist die Entwicklung eines Kompressionsverfahrens Video- bzw. Bildsequenzdaten. Der zu erwartende Aufwand bei der Kodierung von Bewegtbildern ist proportional zur Anzahl der Einzelbilder in der Sequenz. Ein effizienter Algorithmus ist deshalb besonders für Bewegtbilder interessant.

Ein effizientes Verfahren stellt die Arbeit von Dudbridges [Dudbridge 92] vor: durch Verzicht auf die aufwendige Suche nach geeigneten Domainblöcken für die Blocktransformationen ist eine Kodierung mit einem Aufwand proportional zur Fläche des Bildes möglich.

Diesen Ansatz als Ausgangspunkt zu wählen, war eine der Grundentscheidungen der Studienarbeit. Aus folgenden weiteren Gründen ist Dudbridges Arbeit interessant für Bewegtbildkodierung: darin wird ein Konzept der Blockapproximation herausgearbeitet, das sich auch für die Kodierung aufeinanderfolgender Teilbilder eignen kann, und darüberhinaus im Vergleich mit Jacquins Ansatz neuer und weniger bekannt ist.

Folgender Typ eines fraktalen Blockkodierers wird in dieser Arbeit implementiert: die Blocktransformationen setzen sich aus einem geometrischen und einem Grauwert-Anteil zusammen; der geometrische Teil ist entweder die identische Abbildung oder eine Abbildung auf die Teilblöcke eines Domainblocks, während der Grauwert-Anteil durch Funktionen mit 4 Koeffizienten beschrieben wird. Die Koeffizienten der Grauwerttransformation werden durch ein allgemeines Least-Squares Verfahren berechnet.

3.3 Erweiterungen und Arbeitshypothesen

In dieser Studienarbeit wird der Ansatz Dudbridges um folgende Punkte erweitert:

- Einsatz einer Quadtreepartitionierung. Die zugrundeliegende Hypothese ist, daß Teile des Bildes mit geringer Bildaktivität (z.b. einfarbige Flächen, Bereiche mit wenig Kanten, sich nicht bewegende Bereiche einer Sequenz) mit größeren Blöcken günstiger kodiert werden können.

- Erweiterung für die Kodierung von Farbbildern durch Farbtransformation $RGB \rightarrow YCrCb$ (1 Intensitätswert und 2 Farbwerte pro Bildpunkt). Dies geschieht unter der Hypothese, daß die transformierten Teilbilder weniger korreliert sind, als die RGB Teilkomponenten und daß eine Dezimierung der Farbinformation bereits eine Kompression bringt, ohne das Bild wahrnehmbar zu verändern.

- Statt ausschließlich verkleinernde Blocktransformationen (im geometrischen Anteil der Abbildung) einzusetzen, werden auch nicht verkleinernde Blockapproximationen verwendet. Diese haben insbesondere ihren Sinn bei der Abbildung von einem Teilbild der Sequenz auf das nächste Teilbild, da sie für Blöcke ohne Bewegung eine identische Abbildung, also ein Durchreichen der Information sind. Die zugrundeliegende Hypothese ist, daß mit nicht verkleinernden Abbildungen eine ähnlich gute Approximation möglich ist, wie nach Dudbridge für verkleinernde. Daneben kann sich aus den nichtverkleinernden Abbildungen ein Geschwindigkeitsvorteil bei der Dekodierung ergeben.

- Bei Bedarf werden weniger als 4 Koeffizienten für die Grauwerttransformationen abgespeichert. Beispielsweise läßt sich die identische Abbildung ohne Koeffizienten beschreiben, eine Abbildung auf einen konstanten Grauwert durch nur einen Koeffizienten. Ziel ist eine bessere Kompressionsrate zu erreichen.

Die Lösung der Aufgabe besteht aus einem Kodierer und einem Dekodierer. Der Kodierer erhält die einzelnen Bilder der Sequenz als Eingabe und gibt den dafür berechneten fraktalen Code aus. Der Dekodierer nimmt diesen Code als Eingabe und rekonstruiert daraus die approximierten Bilder. Dabei stellt der Dekodierer aus dem fraktalen Code den Zusand des Kodierer exakt wieder her; das bedeutet, daß Kodierer und Dekodierer dieselben Blocktransformationen auf denselben Ausgangsbildern anwenden.

Der fraktale Code besteht aus Informationen zur Bildgeometrie, einer Beschreibung der Bildpartition und aus den Koeffizienten der Grauwerttransformationen. Pro Bildblock der Partition werden bis zu 4 quantisierte Koeffizienten gespeichert.

4 Lösung

Dieses Kapitel umfaßt den Hauptteil der Studienarbeit - die Lösung der Aufgabenstellung; es behandelt die Grundlagen und Implementierungsaspekte eines fraktalen Videokodierers bzw. Dekodierers.

Die Unterkapitel stellen verschiedene Teilaspekte der Lösung dar; im Vordergrund stehen dabei zunächst die theoretischen Zusammenhänge und später mehr die Umsetzung in Programme. Das erste Teilkapitel behandelt die Grundlagen der fraktalen Bildkodierung in einer sehr allgemeinen Darstellung, die Konzepte von Barnsley, Jacquin und Dudbridge umfaßt. Zunächst werden der IFS-Ansatz nach Barnsley dargestellt, zur Kodierung von global selbstaffinen Schwarzweißbildern. Das nächste wichtige Konzept ist die blockbasierte Kodierung von Graustufenbildern nach Jacquin (mit Verallgemeinerungen). Ein Abschnitt behandelt das Least-Squares Verfahren nach Dudbridge, welches ein zentrales Konzept zur Approximation von Bildblöcken darstellt. Weitere Abschnitte erörtern das für die Implementierung gewählte Quadtree-Partitionierungsschema, die Farbtransformation zur Kodierung von Farbbildern und die Erweiterungen des Konzeptes zur Kodierung von Bildsequenzen.

Die Teile am Ende dieses Kapitels behandeln Aspekte der (prototypischen) Implementierung des fraktalen Kodiereres und Dekodierers, insbesondere den verwendeten Algorithmus, die Modulstruktur der Implementierung sowie Verfahren zurn Quantisierung und Speicherung der Transformationskoeffizienten.

4.1 Fraktale Bildkodierung und Erweiterungen

4.1.1 Fraktale Bildkodierung mit IFS

Dieser Abschnitt enthält mathematischen Grundlagen der fraktalen Bildkodierung mit iterierten Funktionensystemem (IFS). Eine umfangreiche Darstellung der IFS-Theorie findet sich in [Barnsley 88b].

Die grundlegenden Begriffe der fraktalen Bildkodierung sind die Begriffe Metrik, vollständiger metrischer Raum, IFS und Attraktor. Dieser Begriffe werden in diesem Abschnitt definiert.

Definition 1 *Metrik und metrischer Raum*

X sei eine Menge. Eine Metrik d auf X mißt den Abstand von je zwei Elementen aus X. Exakt ist eine Metrik eine Abbildung, die je zwei Werten aus X eine positive reelle Zahl zuweist, für die folgende Axiome gelten:

1. $d(x, y) \geq 0$

2. $d(x,y) = 0 \Longleftrightarrow x = y$

3. $d(x,y) = d(y,x)$

4. $d(x,y) \leq d(x,z) + d(z,y) \qquad \forall z \in X$

Ein metrischer Raum ist ein Vektorraum X versehen mit einer Metrik d auf X. Schreibweise: (X,d).

Ein metrischer Raum heißt vollständig, wenn jede Cauchy-Folge konvergiert und der Grenzwert der Folge im Raum liegt.

Für die fraktale Bildkodierung mit IFS ist folgender metrischer Raum von Bedeutung:

Beispiel 1 *$X(\mathcal{S}, d_H)$ = Raum der Schwarzweißbilder $\mathcal{S}$, exakt ist das die Menge der kompakten Teilmengen von $\mathrm{I\!R}^2$, versehen mit dem Hausdorffabstand d_H zweier Bilder als Metrik. Der Hausdorffabstand $d_H(A,B)$ zweier Elemente $A, B \in \mathcal{S}$ ist definiert durch die kleinste Zahl ε, so daß A in einer ε-Umgebung von B und umgekehr B in einer ε-Umgebung von A liegt.*

Um den Begriff des IFS definieren zu können benötigt man den Begriff der kontrahierenden Abbildung:

Definition 2 *Kontrahierende Abbildung*

Sei (X,d) ein metrischer Raum. Eine Abbildung $w: X \to X$ heißt kontrahierend in mit Kontraktionsfaktor (oder Lipschitzkonstante) $s < 1$, wenn

$$\forall_{x,y \in X} \; d(w(x), w(y)) < s\, d(x,y) \tag{1}$$

d.h. wenn die Abbildung Abstände verringert. Die Konstante s ist ein Maß für die Kontraktivität.

Damit läßt sich nun der Begriff "IFS" auf folgende Weise definieren:

Definition 3 *IFS (Iteriertes Funktionensystem) und Attraktor*

Sei (X,d) ein vollständiger metrischer Raum, W sei eine endliche Menge von kontrahierenden Selbstabbildungen auf X.

$$W = \{w_k \mid w_k: X \to X; \quad k = 1, \ldots, n\} \tag{2}$$

Sei eine Iterationsvorschrift auf Elementen $A \in X$ definiert durch:

$$A' = \bigcup_{k=1}^{N} w_k(A) \tag{3}$$

Dann ist das Paar (X, W) mit der obengenannten Iterationsvorschrift ein IFS.

Die Iteration erzeugt für ein beliebig gewähltes Startelement $A_0 \in X$ eine Folge $(A_n)_{n=0}^{\infty}$ in X, die sich rekursiv schreiben läßt als:

$$A_0, \quad A_1 := \bigcup_{k=1}^{N} w_k(A_0), \quad \ldots \quad , A_{n+1} := \bigcup_{k=1}^{N} w_k(A_n), \quad \ldots \tag{4}$$

Falls die Folge konvergiert, bezeichnet man den Grenzwert A_∞ als Attraktor des IFS.

Die Frage nach der Existenz und Eindeutigkeit von Attraktoren konnte Hutchinson positiv beantworten [Hutchinson 81]. Er zeigte die Kontraktivität des Iterationsoperators $w = \bigcup_{k=1}^{N} w_k$ (Hutchinson-Operator) im Raum der Bilder mit dem Hausdorffabstand als Metrik. Die Existenz und Eindeutigkeit des Attraktors bzw. Fixpunkt der rekursiven Folge folgt damit aus dem Banachschen Fixpunktsatz.

Für die Bildkodierung mit IFS wird $(\mathcal{S}, d_H)$, die Menge der möglichen Schwarzweißbilder zusammem mit dem Hausdorffabstand als metrischer Raum gewählt. Die Attraktoren der IFS sind oft Fraktale, also Objekte, deren fraktale Dimension keine natürliche Zahl ist (siehe [Peitgen et al. 92] für eine Übersicht über die verschiedenen Möglichkeiten, den Begriff "fraktale Dimension" zu definieren). Das erklärt auch den den Namen "fraktale Bildkodierung" für den hier beschriebenen Ansatz. Nicht alle Attraktoren sind Fraktale, z.B. ist ein Quadrat kein Fraktal, aber ein Attraktor von 4 Abbildungen, die das Quadrat auf seine 4 Teilquadrate abbilden.

Die Bildkodierung verwendet für die Abbildungen w_k affine Transformationen, die sich aus einem linearen Teil und einer Translation zusammensetzen. Die lineare Abbildung entspricht einer Multiplikation der Punkte mit einer Matrix und umfaßt Spiegelung, Rotation, und Skalierung; die Translation entspricht der Addition eines Vektors. Die affinen Transformationen lassen sich in der folgenden Matrixschreibweise darstellen:

$$w_k(A) := \begin{pmatrix} a_k & b_k \\ c_k & d_k \end{pmatrix} A + \begin{pmatrix} e_k \\ f_k \end{pmatrix} \tag{5}$$

Die Matrix- bzw. Vektoroperationen im Raum der Bilder sind dabei als Vereinigung der Operationen auf den Einzelpunkten des Bildes definierbar.

Die Attraktoren der IFS sind die kodierten Bilder. Der fraktale Code eines Attraktors besteht aus N Transformationen, die mit jeweils den 6 Koeffizienten der Matrix und des Vektors eindeutig spezifiziert werden können.

Beispiel 2 *Barnsleys IFS-Farn*

Barnsley konnte einen Farn durch ein IFS mit 4 affinen Transformationen kodieren, deren Koeffizienten in der folgenden Tabelle aufgelistet sind:

a	b	c	d	e	f
0.849	0.037	-0.037	0.849	0.075	0.1380
0.197	-0.226	0.226	0.197	0.4	0.049
-0.150	0.283	0.260	0.237	0.575	-0.084
0.0	0.0	0.0	0.16	0.5	0.0

Die Größe des Farns wird von diesen 24 Koeffizienten nicht festgelegt. Man kann also auch nicht von einer bestimmten Kompressionsrate sprechen; vielmehr läßt sich durch Vergrößerung des Farnes jede beliebige Kompressionsrate erreichen.

Folgendes Collage-Theorem Barnsleys ermöglicht es, fraktale Codes für eine Reihe von Ausgangsbilder zu finden: die Aussage ist, daß ein IFS, dessen Abbildungen ein gegebenes Bild wenig verändern einen Attraktor besitzt, der nahe an diesem gegebenen Bild ist. Der maximale Fehler hängt ab von der Kontraktivität der Abbildungen des IFS und von der Güte der Collage.

Satz 1 *Collage-Theorem (Barnsley)*

Gegeben sei ein Bild $A \in \mathcal{S}$ (d.h. eine kompakte und abgeschlossene Teilmenge von $\mathbb{R}^2$). Sei $(\mathcal{S}, W)$ ein IFS mit $W = \{w_k \mid k = 1, \ldots, n\}$. Seien s_k, $k = 1, \ldots, n$ die Kontraktionsfaktoren der Abbildungen w_k und $s := \max\{s_k\}$. d_H bezeichne den Hausdorffabstand.

Falls für ein $\varepsilon > 0$

$$d_H(A, \bigcup_{k=1}^{n} w_k(A)) < \varepsilon \tag{6}$$

(d.h. das IFS stellt eine Collage des Bildes A dar, die Abbildung $\bigcup_{k=1}^{n} w_k(A)$ ist nahe am Bild selbst.)

Dann gilt für den Attraktor A_∞ des IFS

$$d_H(A, A_\infty) < \frac{\varepsilon}{1 - s} \tag{7}$$

Die Aussage folgt direkt aus der Fehlerabschätzung des Banachschen Fixpunktsatzes (siehe [Peitgen et al. 92], oder für einen direkten Beweis: [Barnsley 88b]). Ein Beweis des Banachschen Fixpunktsatzes befindet sich in [Harrer 93].

4.1.2 Fraktale Bildkodierung mit Blocktransformationen

Um von einem interaktiven Algorithmus zu einem automatischen Algorithmus zu kommen, definierte Jacquin die Selbstabbildungen eines Bildes nicht global, sondern blockweise. Ein weiterer wichtiger Unterschied zum IFS-Ansatz ist der zugrundeliegende metrische Raum. Die den Abbildungen eines IFS entsprechenden Blocktransformationen operieren auf einem Raum von Grauwertfunktionen, versehen mit der "Root Mean Square" Metrik oder "RMS"-Metrik.

Ziel dieses Ansatzes ist es, eine invariante Funktion auf den Grauwerten zu finden, die ein gegebenes Grauwertbild approximiert (und deren Komplexität geringer ist als die der gegebenen Grauwertfunktion). Zunächst werden die benötigten Begriffe definiert, ein Collage-Theorem für invariante Funktionen angegeben und mit diesen Hilfsmitteln die Schritte angeführt, die bei der Kodierung gemacht werden müssen.

Dieser Ansatz beschränkt sich auf für Grauwertbilder. Abschnitt 4.1.5 behandelt eine mögliche Erweiterung auf Farbbilder.

Definition 4 *Grauwertfunktion*

Sei A eine Bildfläche, d.h eine eine kompakte und abgeschlossene Teilmenge von $\mathbb{R}^2$. Eine Grauwertfunktion $f\colon A \to \mathbb{R}$ weist den Punkten der Bildfläche einen Grauwert zu. Sie ist die mathematische Beschreibung eines Grauwertbildes.

$\mathcal{F}_A$ bezeichne den Raum der über A definierten Grauwertfunktionen.

Die RMS-Metrik auf dem Raum der Grauwertfunktionen mißt den Fehler zwischen zwei Bildern durch Auswertung der quadratischen Differenz der Grauwertfunktionen. Sie wird durch die folgende Integralformel definiert:

Definition 5 *RMS (Root Mean Square) Metrik*

Seien $f, g \in \mathcal{F}_A$ Grauwertfunktionen.

Die RMS-Metrik (oder auch L_2-Metrik) d_{rms} ist definiert als:

$$d_{rms}(f,g) = \sqrt{\frac{\int_A (f-g)^2 \, d\mathcal{L}}{\int_A \, d\mathcal{L}}} \tag{8}$$

Falls f und g nur auf einer echten Teilmenge von A definiert sind (wie zum Beispiel für einzelne Bildblöcke), wird nur über den Definitionsbereich integriert. Damit ist der RMS-Fehler auch für Teilbilder und Bildblöcke definiert. $\mathcal{L}$ steht für das Lebesguesche Maß auf $\mathbb{R}^2$.

Die RMS-Metrik läßt sich auch für Bilder berechnen, deren Grauwertfunktion nur eine endliche Teilmenge von $\mathbb{R}$ annimmt, also insbesondere für digitalisierte Bilder.

Es lassen sich weitere Metriken auf den Raum der Grauwertbilder definieren, allerdings ist die RMS-Metrik die wichtigste für die konkrete Berechnung, denn sie ermöglicht ein Least-Squares Verfahren zur Approximierung von Bildblöcken.

Definition 6 *Partition, Rangeblock*

Sei A die Grundfläche eines Grauwertbildes, d.h. A ist der Definitionsbereich der Grauwertfunktion. Eine Partition P von A ist eine endliche Menge von paarweise disjunkten, zusammenhängenden und kompakten Teilmengen (Blöcke) von A, deren Vereinigung A ergibt.

$$P = \{R_k \mid R_k \subset A, k = 1, \ldots, N\} \tag{9}$$

$$\left(\bigcup_{k=1}^{N} R_k = A\right) \;\wedge\; \left(\forall_{i,j}\; i \neq j \Rightarrow R_i \cap R_j = \emptyset\right) \tag{10}$$

Die Blöcke R_k der Partition heißen Rangeblöcke.

Der Name Rangeblöcke für die Blöcke der Partition erklärt sich daraus, daß jeder dieser Blöcke Bild einer geometrischen Transformation ist. Range wird also im Sinne von Wertebereich einer Abbildung verwendet. Diese Abbildung wird als Blocktransformation bezeichnet, weil ihr Definitionsbereich auf einen Block des Bildes eingeschränkt ist und sie diesen immer auf einen anderen Block abbildet.

Definition 7 *Blocktransformation, Domainblöcke*

Sei $A \subset \mathbb{R}^2$ eine Bildfläche, $f \in \mathcal{F}_A$ eine Grauwertfunktion auf A, sei $P = \{R_k \mid k = 1, \ldots, N\}$ eine Partition von A.

Die auf den Rangeblock R_k abbildende Blocktransformation t_k ist definiert durch:

$$\begin{array}{rccc} t_k\colon & D_k \times \mathbb{R} & \to & R_k \times \mathbb{R} \\ & (x, y, f(x, y)) & \mapsto & (w_k(x, y), f'_k(w_k(x, y))) \end{array} \tag{11}$$

D_k ist ein zum Rangeblock R_k affiner Block dessen Koordinaten auf die des Rangeblocks abgebildet werden. Die auf $D_k \subset A$ eingeschränkte Grauwertfunktion f wird auf eine über R_k definierte Grauwertfunktion f'_k abgebildet. Ein solcher Block D_k heißt Domainblock.

Der geometrische Anteil der Blocktransformation, die Abbildung

$$w_k\colon \mathbb{R}^2 \to \mathbb{R}^2$$

bildet den Domainblock D_k affin auf den Rangeblock R_k ab und entspricht der affinen Abbildung eines IFS (Gleichung 5), das blockweise arbeitet.

$$w_k(D_k) := \begin{pmatrix} a_k & b_k \\ c_k & d_k \end{pmatrix} D_k + \begin{pmatrix} e_k \\ f_k \end{pmatrix} = R_k \tag{12}$$

Neben dem geometrischen Anteil w_k besteht die Blocktransformation t_k aus einem auf der Grauwertfunktion f operierenden Anteil v_k

$$\forall_{p \in D_k}\ v_k(p, f(p)) = f'_k(w_k(p)) \tag{13}$$

der $f|_{D_k}$ auf eine auf R_k definierte Grauwertfunktion f'_k abbildet, mit dem Ziel, daß der Fehler zwischen $f|_{R_k}$ und $f'_k|_{R_k}$ möglichst klein wird.

Alle Blocktransformationen t_k zusammen definieren eine Abbildung τ, die auf $\mathcal{F}_A$ operiert und damit eine Grauwertfunktion auf A auf eine andere solche abbildet. Die Funktion τ spielt bei der Blockkodierung die Rolle, die der Hutchinson-Operator bei der IFS-Kodierung spielt.

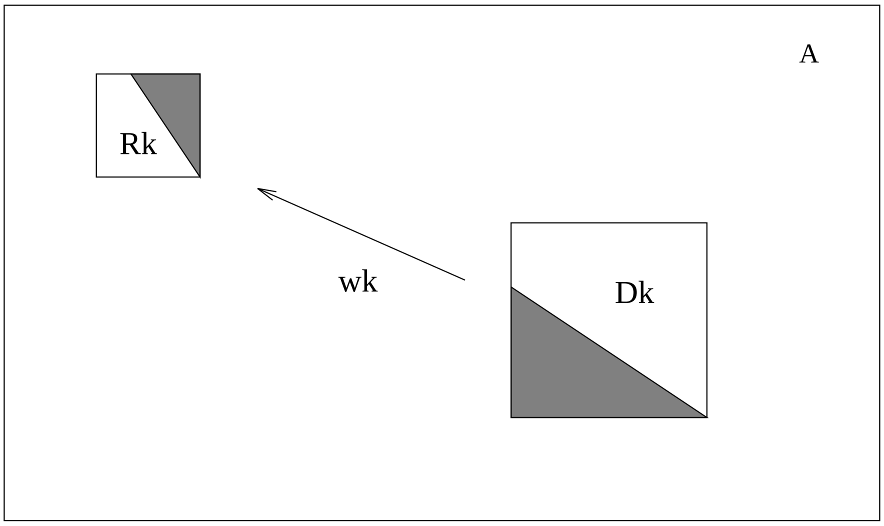

Abbildung 1: Domainblock D_k und Rangeblock R_k auf einer Bildfläche A

Abbildung 1 zeigt ein Beispiel für einen Range- und einen Domainblock und für die Blocktransformation zwischen diesen Blöcken. Die affine Abbildung w_k, die den Domainblock D_k auf den Rangeblock R_k abbildet, besteht aus einer Translation, einer Verkleinerung und einer Spiegelung an der 1. Diagonalen. Die auf der Grauwertfunktion operierende Abbildung v_k ist in diesem Beispiel die Identische Abbildung, weil die Grauwerte nicht verändert werden.

Für die Grauwerttransformation v_k existieren folgende Ansätze:

Jacquin [Jacquin 89] (und damit die meisten Autoren zur fraktalen Bildkodierung) verwendet eine lineare Funktion mit zwei Koeffizienten $\alpha_1^{(k)}$ und $\alpha_2^{(k)}$ (mit

$(x,y) \in D_k$):

$$v_k(x,y,f(x,y)) = f'_k(w_k(x,y)) = \alpha_1^{(k)}\, f(x,y) + \alpha_2^{(k)} \tag{14}$$

Der Koeffizient $\alpha_1^{(k)}$ bewirkt eine Kontrast-Skalierung des Grauwertes, $\alpha_2^{(k)}$ eine Helligkeitsänderung.

Dudbridge und Monro [Monro et al. 92a] verwenden Grauwerttransformationen mit 4 Koeffizienten $\alpha_1^{(k)}, \ldots, \alpha_4^{(k)}$:

$$v_k(x,y,f(x,y)) = \alpha_1^{(k)}\, x + \alpha_2^{(k)}\, y + \alpha_3^{(k)}\, f(x,y) + \alpha_4^{(k)} \tag{15}$$

Grauwerttransformationen höherer Ordung werden in neueren Arbeiten von Monro untersucht [Monro 93a, Monro 93b]. Diese Transformationen sind allgemeine Polynome in den formalen Parametern x, y und $f(x,y)$.

Die Koeffizienten $\{\alpha_j^{(k)}\}$, die die beste Approximation des Rangeblocks (bezüglich des RMS-Fehlers) liefern, können analytisch mit einem Least-Squares Kriterium bestimmt werden; siehe dazu Abschnitt 4.1.4.

Der Begriff Attraktor spielt für die Kodierung mit Blocktransformationen nur mehr die folgende Rolle: die geometrischen Anteile w_k der Blocktransformationen t_k besitzen als IFS aufgefaßt trivialerweise die gesamte Bildfläche A als Attraktor, falls die einzelnen w_k kontrahierend in $(\mathbb{R}^2, |\cdot|)$ sind. Diese Aussage ergibt sich aus der Eigenschaft der Rangeblöcke, die gesamte Bildfläche zu überdecken. Das Analogon zum Attraktor als approximiertes fraktales Bild bei der Kodierung mit IFS ist hier die auf dem Attraktor A definierte invariante Grauwertfunktion.

Definition 8 *Invariante Funktionen*

Eine Grauwertfunktion $F\colon A \to \mathbb{R}$ auf einem Attraktor A heißt invariant unter einer Transformation τ, falls τ F auf sich selbst abbildet. Insbesondere gilt für die durch Blocktransformationen definierten τ:

$$\forall_{p\in A}\ \ (p \in R_k)\ \Rightarrow\ F(p) = \tau\circ F(p) = t_k({w_k}^{-1}(p), F({w_k}^{-1}(p))) \tag{16}$$

Die invariante Funktion ist das, was aus einem frakalen Code wiederhergestellt werden kann; sie ist also das approximierte fraktale Bild. Für invariante Funktionen gilt das folgende Collage-Theorem:

Satz 2 *Collage Theorem für invariante Funktionen*

Sei f eine Grauwertfunktion auf A (f entspricht dem Orginalbild) und τ eine kontrahierende Grauwerttransformation, die sich aus Blocktransformationen t_k zusammensetzt. Sei s Kontraktionsfaktor von τ. Dann gilt für die durch τ festgelegte invariante Funktion F (entspricht dem kodierten Bild) folgende Abschätzung:

$$d_{rms}(F,f) \leq \frac{1}{1-s} d_{rms}(f, \tau \circ f) \tag{17}$$

Aus dem Fehler bei der Transformation von f unter τ läßt damit sich der maximale Fehler zwischen f und der invarianten Funktion F berechnen. Damit erhält man eine Abschätzung für den maximalen Fehler zwischen Orginalbild und kodiertem Bild. Dieses Collage-Theorem ergibt sich wie das Collage-Theorem für IFS direkt aus dem Banachschen Fixpunktsatz. Zur Kontraktivität des Operators τ siehe Anmerkung 1.

Das Ziel der fraktalen Bildkodierung mit Blocktransformationen besteht aus folgenden Schritten (Skizze eines allgemeinen Kodieralgorithmus'):

1. Partitionierung des Bildes.

2. Für jeden Rangeblock R_k der Partition eine geeignete Blocktransformation finden, die den Fehler zwischen der Grauwertfunktion auf R_k und der transformierten Grauwertfunktion auf R_k möglichst klein macht. Dieser Schritt umfaßt die Suche nach einem geeigneten Domainblock D_k und die Berechnen der Koeffizienten der Grauwerttransformationen v_k.

3. Der fraktale Code besteht dann aus einer Beschreibung der Partitionierung und aus einer Liste von Blocktransformationen $\{t_k\}$. (Anm.: es brauchen nicht alle 6 Parameter der geometrischen Abbildung gespeichert werden, es reicht die Angabe der Geometrie des Domainblocks (Größe, Lage) und die durchgeführten Drehungen und Spiegelungen, um die affine Abbildung exakt zu beschreiben.) Die Blocktransformationen beinhalten insbesondere die Koeffizienten der Grauwerttransformationen v_k.

Falls der Fehler bei der Approximation der Rangeblöcke klein ist, läßt sich aus dem fraktalen Code eine invariante Funktion rekonstruieren, die das Ausgangsbild approximiert. Eine Fehlerabschätzung liefert das Collage-Theorem für invariante Funktionen.

Folgende Punkte lassen sich dazu anmerken:

Anmerkung 1 *Die Kontraktivität der Transformation τ bezüglich der RMS-Metrik wird durch die Kontraktivität der Grauwerttransformationen v_k bestimmt. Die Kontraktivität der v_k bestimmt sich durch die Koeffizienten $\alpha_j^{(k)}$ der Monome, die von f abhängen.*

Z.B. für v_k aus Formel (14) mit

$$v_k(f) = \alpha_1^{(k)} f + \alpha_2^{(k)}$$

spielt nur der Koeffizient $\alpha_1^{(k)}$ *eine Rolle für die Kontraktivität in der RMS-Metrik:*

Beweis: Seien f, g *Grauwertfunktionen. Dann gilt:*

$$
\begin{aligned}
d_{rms}(\tau \circ f, \tau \circ g) &= \sqrt{\frac{\int_A (\tau \circ f - \tau \circ g)^2 \, d\mathcal{L}}{\int_A \, d\mathcal{L}}} \\
&= \sqrt{\frac{\sum_i \int_{R_k} (\alpha_1^{(k)} f + \alpha_2^{(k)} - (\alpha_1^{(k)} g + \alpha_2^{(k)}))^2 \, d\mathcal{L}}{\int_A \, d\mathcal{L}}} \\
&= \sqrt{\frac{\sum_i (\alpha_1^{(k)})^2 \int_{R_k} (f-g)^2 \, d\mathcal{L}}{\int_A \, d\mathcal{L}}} \\
&\leq \sqrt{\frac{\max_i \{(\alpha_1^{(k)})^2\} \sum_i \int_{R_k} (f-g)^2 \, d\mathcal{L}}{\int_A \, d\mathcal{L}}} \\
&\leq \max_i \{ \mid \alpha_1^{(k)} \mid \} \, d_{rms}(f, g)
\end{aligned}
\tag{18}
$$

Wenn alle $\alpha_1^{(k)}$ *mit Betrag kleiner 1 gewählt werden, ist auch* τ *kontrahierend in der RMS-Metrik.*

Anmerkung 2 *Jacquin zeigte, daß man auch ohne vollständige metrische Räume auskommen kann. Unvollständige metrische Räume erlauben zwar keine Aussage über Fixpunkte, in der Praxis genügen jedoch Punkte, die nicht wandern, d.h. ab einer Iterationsstufe immer innerhalb einer* ε*-Umgebung bleiben. Mit einer Hilfskonstruktion kommt man auch mit Pseudometriken aus: eine Pseudometrik ist definiert wie eine Metrik, wobei in Axiom 2 nur eine Richtung gilt:*

$$x = y \Longrightarrow d(x, y) = 0 \tag{19}$$

Die Umkehrung gilt nicht, d.h. die Pseudometrik kann unterscheidbaren Punkten auch den Abstand 0 zuordnen. Diese Situation tritt insbesondere in Pixelbildern auf, bei denen Punkte mit verschiedenen Koordinaten, die auf ein Pixel fallen, einen Abstand 0 zugewiesen bekommen.

Jacquin zeigte [Jacquin 89], daß aus einem pseudometrischen Raum immer ein metrischer Raum konstruiert werden kann (durch Faktorisierung mit dem Ideal, das durch die Punkte mit Abstand 0 gebildet wird).

Anmerkung 3 *Denkbar sind auch Partitionierungen mit überlappenden Blöken. Die Eigenschaft nichtüberlappender Partitionierungen, daß jeder Punkt des approximierten Bildes genau einen Ursprungspunkt bezüglich der Transformation* τ *besitzt, geht dadurch verloren. Daraus können sich bei der Iteration der Transformation Schwierigkeiten ergeben.*

In dieser Arbeit werden nur nichtüberlappende Partitionierungen betrachtet.

4.1.3 Partitionierung

Die Partitionierung des Bildes ist eine wichtige Teilaufgabe der Berechnung eines fraktalen Codes. Sinnvolle Partitionierungsverfahren arbeiten adaptiv: abhängig vom Bildinhalt (bzw. dem Verlauf der Grauwertfunktion) werden Blöcke von verschiedener Größe erzeugt. Teile des Bildes, die wenig Details enthalten, können mit großen Blöcken approximiert werden, während Bildteile mit Kanten und vielen Details durch kleine Blöcke approximiert werden müssen, um einen tolerierbaren Fehler nicht zu überschreiten.

Sinnvolle Vorgaben für die Partitionierung und damit wichtige Parameter für die Kodierung sind: der maximal zugelassene Fehler pro Blocktransformation oder die maximale Anzahl der Rangeblöcke. Im einen Fall wird die Bildqualität vorgegeben, im anderen Fall die Kompressionsrate, die im wesentlichen von der Anzahl der Rangeblöcke abhängt.

Das Partitionierungsschema von Monro und Dudbridge, deren Approximationsverfahren Grundlage dieser Arbeit ist, arbeitet sehr statisch: Domainblöcke haben eine vorgegebene feste Größe von 8×8 Pixel. Jeder Domainblock wird auf 4 nichtüberlappende Teile seiner selbst transformiert. Die 4 Rangeblöcke können in der Größe variieren. Dudbridge findet einen geeigneten Splitpunkt S des Domainblocks durch eine binäre Suche, jedoch ohne damit immer den optimalen Punkt zu finden. Abbildung 2 zeigt als Beispiel einen Domainblock, der in 4 (in diesem Fall rechteckige) Rangeblöcke geteilt ist.

Während dieses Partitionierungsverfahren für die Einzelbildkodierung gute Ergebnisse liefert, ist es für die Kodierung von Bildsequenzen nicht geeignet, weil für Bereiche der Sequenz ohne Bewegung, Verkleinerungen nichts bringen. Solche Bereiche lassen sich dagegen gut mit großen Blöcken ohne geometrische Transformation kodieren.

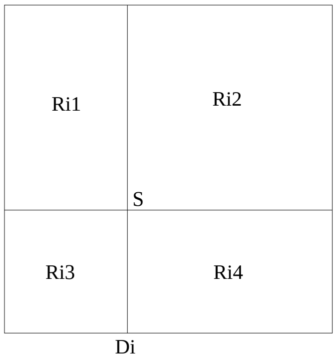

Abbildung 2: Partitionierung eines Domainblocks bei Dudbridge und Monro

Der Kodierer dieser Studienarbeit verwendet eine adaptive Quadtreepartitionierung mit quadratischen Blöcken, die den maximal zulässigen RMS-Fehler r als Vorgabe des rekursiven Algorithmus' erhält. Die Partitionierung entsteht dabei während der Kodierung: zunächst wird versucht, einen Block mit einer Blocktransformation zu approximieren. Abhängig von der Anzahl der Teilblöcke, in denen der Fehler den Wert r überschreitet, werden 2 Fälle unterschieden:

1. Der Fehler ist in 0-2 Teilblöcken größer als r. Dann wird die gefundene Blocktransformation ausgegeben, und die Blöcke, bei denen der Fehler zu groß ist, werden notiert. (Abbruch der Rekursion falls es 0 Teilblöcke sind.)

2. Der Fehler ist in 3 oder 4 Teilblöcken größer als r, d.h. die gefundene Blocktransformation ist nicht gut genug. Dann wird die Blocktransformation verworfen und alle 4 Teilblöcke werden notiert.

Auf die notierten Blöcken wird rekursiv dasselbe Verfahren angewendet, bis die minimale Blockgröße erreicht ist.

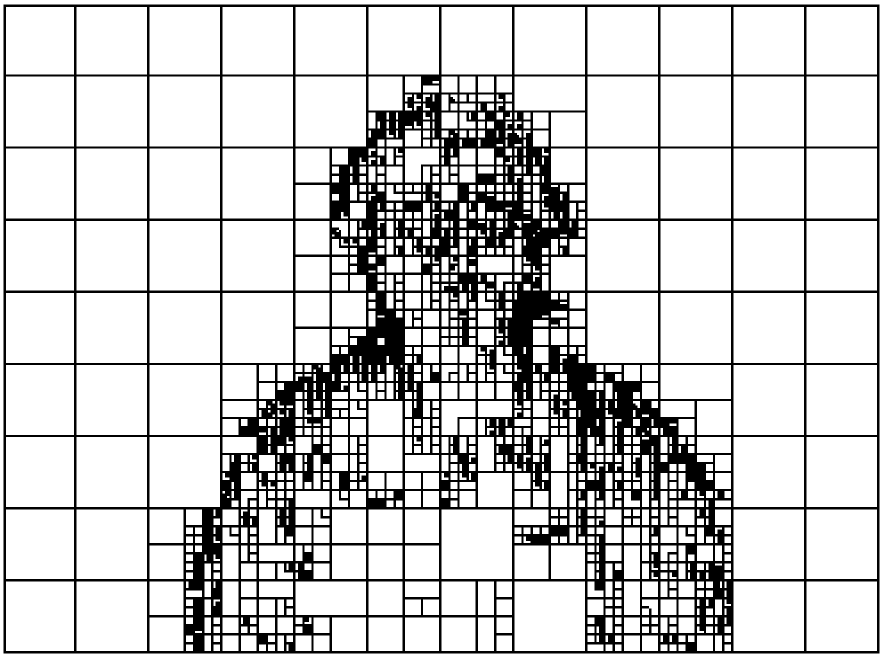

Abbildung 3: Partitionierung eines Teilbildes einer Bildsequenz

Das implementierte Verfahren beginnt nicht mit der gesamten Bildfläche; vielmehr erhält der Algorithmus neben dem RMS-Fehler die Größe der Startblöcke (B_{max}) und die kleinste mögliche Blockgröße (B_{min}) (bzw. die maximale Tiefe der Rekursions).

Eine Partitionierung eines Teilbildes (Frame) einer Testsequenz zeigt Abbildung 3. Begonnen wurde mit einer Blockgröße von 32×32 Pixel. Der Hintergrund des Frames, der sich in der Sequenz nicht bewegt, kann mit der größten Blockgröße kodiert werden. Die sich bewegenden Teile des Vordergrundes erfordern kleinere Blöcke für die Approximation.

4.1.4 Least-Squares Approximation

Dieser Abschnitt behandelt die Methode der Least-Squares Minimierung zur Berechnung optimaler Koeffizienten der Grauwerttransformationen v_k, d.h. Koeffizienten $\alpha_j^{(k)}$, die den RMS-Fehler $d_{rms}(f|_{R_k}, f_i')$ für jeden Rangeblock minimieren. Die Approximation erfolgt immer blockweise, auch wenn im folgenden nicht ausdrücklich darauf verwiesen wird.

Der Einsatz einer Least-Squares Approximation für die fraktale Bildkodierung geht auf F. Dudbridges Arbeit [Dudbridge 92] zurück. Die vorliegende Arbeit folgt dem Ansatz Dudbridges und verwendet für die Blockapproximation Grauwerttransformationen mit 4 Koeffizienten:

$$v_k(x, y, f(x, y)) = \alpha_1^{(k)}\, x + \alpha_2^{(k)}\, y + \alpha_3^{(k)}\, f(x, y) + \alpha_4^{(k)}$$

Mit der besseren Approximationsleistung der Transformation mit 4 Koeffizienten gegenüber der mit 2 Koeffizienten rechtfertigt Dudbrigde den Verzicht auf die Suche nach Domainblöcken. Die Effizienz des Verfahrens dieser Studienarbeit resultiert aus dem Verzicht auf Suche.

Die Least-Squares Minimierung basiert auf der Eigenschaft der RMS-Metrik, das Minimum an einer Nullstelle des Gradienten (bezüglich der Koeffizienten $\alpha_j^{(k)}$ von v_k) anzunehmen, weil alle von den Koeffizienten abhängenden Terme quadratisch eingehen. Der Name Least-Squares (kleinste Quadrate) erklärt sich daraus, daß die quadratischen Terme der RMS-Metrik minimiert werden.

Die Minimierung umfaßt die folgenden Schritte:

1. Die partiellen Ableitungen der RMS-Metrik nach den Koeffizienten $\alpha_j^{(k)}$ werden bestimmt und gleich 0 gesetzt:

$$\frac{\partial d(f|_{R_k}, f_i')}{\partial \alpha_j^{(k)}} = 0 \tag{20}$$

2. Für die n unbekannten Koeffizienten $\alpha_j^{(k)}$, $j = 1, \ldots, n$ erhält man so n Gleichungen, die linear in den Koeffizienten sind. Durch Lösen des Linearen Gleichungssystems bestehend aus diesen n Gleichungen, lassen sich die optimalen Koeffizienten bestimmen, die den Fehler minimieren. Falls das Gleichungssystem keine eindeutige Lösung hat ist eine Sonderbehandlung nötig.

Um diese grobe Skizze des Verfahrens zu veranschaulichen führt der folgender Teil dieses Abschnittes die Umformungen und Berechnungen durch, die für die Grauwerttransformation v_k mit 4 Koeffizienten benötigt werden.

Für jede Blocktransformation t_k ergibt sich folgender Ansatz ($(x,y) \in D_k$):

$$\frac{\partial}{\partial \alpha_j^{(k)}} d(f(w_k(x,y)), \alpha_1^{(k)}\,x + \alpha_2^{(k)}\,y + \alpha_3^{(k)}\,f(x,y) + \alpha_4^{(k)}) = 0 \tag{21}$$

In die Definition der RMS-Metrik eingesetzt erhält man (Quadratwurzel und der Nenner fallen weg):

$$\frac{\partial}{\partial \alpha_j^{(k)}} \int_{D_k} [f(w_k(x,y)) - (\alpha_1^{(k)}\,x + \alpha_2^{(k)}\,y + \alpha_3^{(k)}\,f(x,y) + \alpha_4^{(k)})]^2 \, d\mathcal{L} = 0 \tag{22}$$

Ausquadriert ergibt sich folgende (längere) Formel für die zu integrierende Funktion:

$$\begin{aligned}
[\,] \;&:=\; [f(w_k(x,y)) - (\alpha_1^{(k)}\,x + \alpha_2^{(k)}\,y + \alpha_3^{(k)}\,f(x,y) + \alpha_4^{(k)})]^2 \\
&=\; [f(w_k(x,y))^2 + (\alpha_1^{(k)}x)^2 + (\alpha_2^{(k)}y)^2 + (\alpha_3^{(k)}f(x,y))^2 + (\alpha_4^{(k)})^2 \\
&\quad -2(\alpha_1^{(k)}xf(w_k(x,y)) + \alpha_2^{(k)}yf(w_k(x,y))) \\
&\quad -2(\alpha_3^{(k)}f(x,y)f(w_k(x,y)) + \alpha_4^{(k)}f(w_k(x,y))) \\
&\quad +2(\alpha_1^{(k)}\alpha_2^{(k)}xy + \alpha_1^{(k)}\alpha_3^{(k)}xf(x,y) + \alpha_1^{(k)}\alpha_4^{(k)}x) \\
&\quad +2(\alpha_2^{(k)}\alpha_3^{(k)}yf(x,y) + \alpha_2^{(k)}\alpha_4^{(k)}y + \alpha_3^{(k)}\alpha_4^{(k)}f(x,y))]
\end{aligned} \tag{23}$$

Eingesetzt ergibt das folgende 4 Formeln:

$$\begin{aligned}
\frac{\partial}{\partial \alpha_1^{(k)}} \int_{D_k} [\,]\, d\mathcal{L} &= 2\int_{D_k} [\alpha_1^{(k)}x^2 + \alpha_2^{(k)}xy + \alpha_3^{(k)}xf(x,y) + \alpha_4^{(k)}x - xf(w_k(x,y))]\, d\mathcal{L} \\
\frac{\partial}{\partial \alpha_2^{(k)}} \int_{D_k} [\,]\, d\mathcal{L} &= 2\int_{D_k} [\alpha_1^{(k)}xy + \alpha_2^{(k)}y^2 + \alpha_3^{(k)}yf(x,y) + \alpha_4^{(k)}y - yf(w_k(x,y))]\, d\mathcal{L} \\
\frac{\partial}{\partial \alpha_3^{(k)}} \int_{D_k} [\,]\, d\mathcal{L} &= 2\int_{D_k} [\alpha_1^{(k)}xf(x,y) + \alpha_2^{(k)}yf(x,y) + \alpha_3^{(k)}f(x,y)^2 + \alpha_4^{(k)}f(x,y) \\
&\qquad -f(x,y)f(w_k(x,y))]\, d\mathcal{L} \\
\frac{\partial}{\partial \alpha_4^{(k)}} \int_{D_k} [\,]\, d\mathcal{L} &= 2\int_{D_k} [\alpha_1^{(k)}x + \alpha_2^{(k)}y + \alpha_3^{(k)}f(x,y) + \alpha_4^{(k)} - f(w_k(x,y))]\, d\mathcal{L}
\end{aligned} \tag{24}$$

Da Summe und Integral vertauschbar sind, ergeben sich 4 lineare Gleichungen in den Koeffizienten $\alpha_1^{(k)}, \ldots, \alpha_4^{(k)}$. Zur Bestimmung der optimalen Koeffizienten ist folgendes Gleichungssystem zu lösen (wobei $\int f$ eine abkürzende Schreibweise für $\int_{D_k} f \, d\mathcal{L}$ ist):

$$\begin{bmatrix} \int x^2 & \int xy & \int xf(x,y) & \int x \\ \int xy & \int y^2 & \int yf(x,y) & \int y \\ \int xf(x,y) & \int yf(x,y) & \int (f(x,y))^2 & \int f(x,y) \\ \int x & \int y & \int f(x,y) & \int 1 \end{bmatrix} \begin{bmatrix} \alpha_1^{(k)} \\ \alpha_2^{(k)} \\ \alpha_3^{(k)} \\ \alpha_4^{(k)} \end{bmatrix} = \begin{bmatrix} \int xf(w_k(x,y)) \\ \int yf(w_k(x,y)) \\ \int f(x,y)f(w_k(x,y)) \\ \int f(w_k(x,y)) \end{bmatrix} \quad (25)$$

Die Approximation der Grauwertfunktion des Rangeblocks durch v_k umfaßt die Berechung der Integrale aus Formel (25) und die anschließende Lösung des Gleichungssystems.

Anmerkung 4 *In dieser Darstellung werden die Integrale über den Domainblöcken D_k gebildet. Für die Berechnung ist es genausogut möglich, die Integrale über den Rangeblöcken zu bilden. Statt $f(x,y)$ taucht dann $f({w_k}^{-1}(x,y))$ in den Gleichungen auf und statt $f(w_k(x,y))$ $f(x,y)$. Für kontrahierende w_k besitzen die Rangeblöcke eine kleinere Fläche als die Domainblöcke; daher ist die Berechnung über den Rangeblöcken weniger aufwendig, dafür aber ungenauer, weil bereits gemittelte bzw. interpolierte Werte eingehen.*

Bislang wurde die Integrierbarkeit der Grauwertfunktion vorrausgesetzt. Die in der Praxis relevanten Pixelbilder besitzen immer integrierbare Grauwertfunktionen. Die Grauwertfunktion eines Pixelbildes ist auf den Pixeln (kleine quadratische Flächen) konstant, damit stückweise stetig und damit integrierbar. Die Koordinaten des Pixelbildes sind so gewählt, daß Pixelgrenzen mit ganzzahligen Koordinaten korrespondieren: ein Pixel ist dann eine Fläche mit den Koodinaten $[n, n+1) \times [m, m+1)$ mit $n, m \in \mathbb{N}$. Die Integrale aus Gleichung (25) können für Pixelbilder auf folgende Weise berechnet werden:

Ohne Beschränkung der Allgemeinheit seien die Koordinaten des quadratischen Domainblocks aus dem Intervall $[0, B)$. Die Integrale über D_k lassen sich wegen der stückweisen Stetigkeit durch Summen von Integralen über den Pixelflächen ersetzen. Da die Grauwertfunktion auf den Pixeln konstant ist, ist die Integration dort leicht möglich:

Z. B. kann das Integral $\int_{D_k} xf(x,y)\, d\mathcal{L}$ auf folgende Art berechnet werden:

$$\begin{aligned} \int_{D_k} xf(x,y)\, d\mathcal{L} &= \int_0^B \int_0^B xf(x,y)\, dxdy \\ &= \sum_{j=0}^{B-1} \sum_{i=0}^{B-1} f(i,j) \int_j^{j+1} \int_i^{i+1} x\; dxdy \\ &= \sum_{j=0}^{B-1} \sum_{i=0}^{B-1} f(i,j) \int_j^{j+1} \left[\tfrac{1}{2}x^2\right]_{x=i}^{i+1}\; dy \end{aligned}$$

$$= \sum_{j=0}^{B-1} \sum_{i=0}^{B-1} f(i,j)(i+\tfrac{1}{2}) \tag{26}$$

Die übrigen Integrale können analog bestimmt werden. Es ist zweckmäßig, eine implizite Koordinatentransformation des Domainblocks auf den Koordinaten_ursprung zu machen. Damit können auch die von der Grauwertfunktion f unabhängigen Integrale für jede Blockgröße a priori berechnet werden. Die Berechnung am Koordinatenursprung vermeidet auch einen sonst möglichen Zahlenüberlauf. Beispiel 3 zeigt das resultierende Gleichungssystem für quadratische Blöcke.

Beispiel 3 *Die Seitenlänge eines quadratischen Domainblocks sei B. Dann erhält man folgende Matrix auf der linken Seite des Gleichungssystems:*

$$\begin{bmatrix} \frac{1}{3}B^4 & \frac{1}{4}B^4 & \sum_{i,j=0}^{B-1}(i+\frac{1}{2})f(i,j) & \frac{1}{2}B^3 \\ \frac{1}{4}B^4 & \frac{1}{3}B^4 & \sum_{i,j=0}^{B-1}(j+\frac{1}{2})f(i,j) & \frac{1}{2}B^3 \\ \sum_{i,j=0}^{B-1}(i+\frac{1}{2})f(i,j) & \sum_{i,j=0}^{B-1}(j+\frac{1}{2})f(i,j) & \sum_{i,j=0}^{B-1}(f(i,j))^2 & \sum_{i,j=0}^{B-1}f(i,j) \\ \frac{1}{2}B^3 & \frac{1}{2}B^3 & \sum_{i,j=0}^{B-1}f(i,j) & B^2 \end{bmatrix} \tag{27}$$

Für den Vektor auf der rechten Seite des Gleichungssystems erhält man:

$$\begin{bmatrix} \sum_{i,j=0}^{B-1}(i+\frac{1}{2})f(w_k(i,j)) \\ \sum_{i,j=0}^{B-1}(j+\frac{1}{2})f(w_k(i,j)) \\ \sum_{i,j=0}^{B-1}f(i,j)f(w_k(i,j)) \\ \sum_{i,j=0}^{B-1}f(w_k(i,j)) \end{bmatrix} \tag{28}$$

Ein Kodierer muß die 8 verschiedenen Summen berechnen, die von $f(i,j)$ oder $f(w_k(i,j))$ abhängen.

Der noch offene Schritt des Least-Squares Verfahrens ist die Lösung des linearen Gleichungssystems. In der Implementierung zu dieser Studienarbeit wird ein Gauß-Seidel Algorithmus benutzt, um die Lösung zu berechnen (falls sie existiert).

Anmerkung 5 *Spezialfall $v_k(f) = \alpha_1^{(k)} f + \alpha_2^{(k)}$*
Für Grauwerttransformationen mit 2 Koeffizienten, wie Jacquin sie benützt, läßt sich das Gleichungssystem leicht statisch lösen:
Man erhält folgende Ausdrücke für die Koeffizienten $\alpha_1^{(k)}$ und $\alpha_2^{(k)}$:

$$\alpha_1^{(k)} = \frac{\sum_{i,j=0}^{B-1} f(i,j) f(w_k(i,j))}{\sum_{i,j=0}^{B-1} (f(i,j))^2 - \frac{1}{B^2}\left(\sum_{i,j=0}^{B-1} f(i,j)\right)^2} \tag{29}$$

$$\alpha_2^{(k)} = \frac{1}{B^2}\left(\sum_{i,j=0}^{B-1} f(w_k(i,j)) - \alpha_1^{(k)} \sum_{i,j=0}^{B-1} f(i,j)\right) \tag{30}$$

Diese Formeln werden unter anderem auch in [Fisher 92] angegeben.

In dieser Studienarbeit werden nur quadratische Blöcke eingesetzt. Für Blöcke anderer Geometrie müssen die angegebenen Formeln erweitert werden. Leicht sind die Berechnungen, wenn die Kanten der Blöcke parallel zu den Bildseiten verlaufen und auf Pixelgrenzen fallen. Komplizierter wird die Integration, wenn Kanten beliebig verlaufen, wie es sich bei der Dreieck-Partitionierung nicht vermeiden läßt. Der Aufwand wird zwar größer, aber die Integrale bleiben berechenbar.

Ein Problem tritt auf, wenn das Gleichungssystem nicht lösbar ist. Beispiel: falls alle Pixel im Domainblock den Grauwert 0 besitzten, wird die dritte Spalte und die dritte Zeile von Matrix (27) zu Null und damit das Gleichungssystem unlösbar. Es ist von vornherein schwer abzuschätzen, für welche Grauwertkombinationen das Gleichungssystem unlösbar wird, aber wahrscheinlich genügt es nicht, für unlösbare Gleichungssysteme konstanten Grauwert 0 anzunehmen. Für fraktale Verfahren, die nach den besten Domainblöcken suchen ist die Wahrscheinlichkeit gering, daß alle möglichen Domainblöcke ein unlösbares Gleichungssystem hervorrufen. Für solche Verfahren ist das Problem ignorierbar.

In einem Verfahren, das ohne Suche arbeitet, wie bei dem Verfahren von Dudbridge und Monro, ist für unlösbare Gleichungssysteme eine Sonderbehandlung nötig. Die spezielle Sonderbehandlung dieser Arbeit wird im Abschnitt 4.2 dargestellt.

4.1.5 Farbbildkodierung

Alle bisher dargestellten Begriffe und Verfahren zur fraktalen Bildkodierung beziehen sich entweder auf Schwarzweißbilder (IFS-Ansatz) oder auf Graustufenbilder (Blocktransformations-Ansatz). Dieser Abschnitt stellt eine Erweiterung des blockbasierten Ansatzes zur Kodierung von Farbbildern vor.

Definition 9 *RGB-Farbbild*

Sei G eine Menge von Grauwerten, A die Grundfläche eines Bildes.

Ein RGB-Farbbild ist eine Funktion,

$$f\colon A \to G \times G \times G$$

die jedem Punkt aus A ein 3-Tupel (r, g, b) von Grauwerten zuweist. Die Werte (r, g, b) entsprechen den Intensitäten der drei Teilfarben rot, grün und blau.

Für digitalisierte Bilder ist die Menge G endlich; meist werden die Graustufen mit 8-Bit Werten quantisiert, so daß die Menge G die natürlichen Zahlen von 0 bis 255 umfaßt.

Da die Projektion auf die Teilkomponenten des RGB-Farbbildes eine Grauwertfunktion ist, ist ein naheliegender Ansatz zur Kodierung von Farbbildern, jede Komponente unabhängig voneinander zu kodieren.

Die in dieser Arbeit verwendete Methode dagegen arbeitet mit einer Farbtransformation, wie sie beispielsweise auch beim Farbfernsehen oder bei DCT-basierten Kodierern verwendet wird. Die Farbkanäle für rot, grün und blau werden in einen Luminanzkanal Y (Helligkeit), und 2 Chrominanzkanäle Cr und Cb (Farben) umgerechnet. Das menschliche Wahrnehmungssystem ist empfindlicher für Helligkeitsschwankungen als für Farbschwankungen, so daß die Crominanzkanäle mit geringerer Auflösung abgetastet werden können als der Luminanzkanal, ohne wahrnehmbare Fehler im Bild hervorzurufen. In dieser Arbeit werden je 2×2 Pixel der Crominanzen zu einem Wert gemittelt, dadurch wird bereits eine Kompression von 1 : 2 gegenüber dem RGB-Bild erreicht.

Konkret arbeitet der Kodierer dieser Studienarbeit mit folgender Farbtransformation $RGB \to YCrCb$, die als CCIR[1] Recommendation 601 Standard für die Kodierung von Komponentenvideo ist [Zaccarin et al. 93] und auch für die MPEG Kodierung verwendet wird.

$$\begin{aligned} Y' &= 0.299\,R + 0.587\,G + 0.114B \\ Cr' &= 0.7132\,(R - Y) \\ Cb' &= 0.5643\,(B - Y) \end{aligned} \tag{31}$$

Der Standard sieht folgende Skalierung vor, damit die Eregebnisse mit 8 Bit quantisiert werden können:

$$Y = \frac{219}{256} Y' + 16.5 \tag{32}$$

[1] Comité Consultatif International des Radiocommunications.

$$Cr = \frac{224}{256} Cr' + 128.5 \qquad (33)$$

$$Cb = \frac{224}{256} Cb' + 128.5 \qquad (34)$$

Für RGB-Bilder mit einer Genauigkeit von 8 Bit pro Farbe liegen die Werte für alle möglichen RGB-Kombinationen in folgenden Intervallen:

$$Y \in [16 \ldots 235],\ Cr, Cb \in [16 \ldots 240]$$

Um das kodierte RGB-Bild wiederherzustellen, wendet der Dekodierer die inversen Operationen auf $YCrCb$ an, wobei für 4 Y-Werte je ein Cr- und ein Cb-Wert eingeht.

4.1.6 Bildsequenzen

Bisher wurde gezeigt, wie mithilfe von Blocktransformationen und einem Least-Squares Kriterium eine kontrahierende Selbsttransformation eines Grauwertbildes definiert werden kann. Mit den gleichen Konzepten kann auch eine Transformation von einem Teilbild (Frame) einer Sequenz auf das Nachfolgebild definiert werden.

Bei der Einzelbildkodierung benötigt man kontrahierende Selbstabbildungen, weil ein Attraktor, also der Fixpunkt einer Folge benötigt wird, um ein Bild zu approximieren. Nach dem Banachschen Fixpunktsatz konvergiert eine rekursiv definierte Folge immer, wenn die die Folge definierende Abbildung kontrahierend ist.

Die Kodierung von Bildsequenzen benötigt dagegen keine kontrahierenden Transformationen, weil die Approximation eines Frames nicht aus sich heraus, sondern vom Vorgänger aus definiert wird.

Definition 10 *Bildsequenz, Referenzframe, Transformationsframe*

Ohne Beschränkung der Allgemeinheit sei eine Bildsequenz (oder ein Video, ein Film) eine (nicht notwendig endliche) Folge $f_1, f_2, \ldots \in \mathcal{F}_A$ *von Grauwertfunktionen (Frames).*

Für die Folge der kodierten Frames $f'_1, f'_2, \ldots \in \mathcal{F}_A$ *mit* $f`_i \approx f_i$ *lassen sich zwei Frametypen unterscheiden: erstens "Referenzframes"* f'_i*, die nicht von ihren Vorgänger-Frames* f'_{i-1} *abhängen und zweitens "Transformationsframes"* f'_j*, die durch eine Grauwerttransformation aus ihren Vorgängern* f'_{j-1} *approximiert werden:*

$$f'_j = \tau_j \circ f'_{j-1}$$

τ_i *ist eine (blockweise definierte) Transformation von Frame* f_{j-1} *nach Frame* f_j*.*

Für Farbvideos die Kanäle Y, Cr und Cb als unabhängige Teilsequenzen kodiert (wie in Abschnitt 4.1.5), so daß es genügt, nur Sequenzen von Grauwertfunktionen zu betrachten.

Das Kodierungsverfahren für Transformationsframes besteht aus den beiden Phasen: Partitionierung und Finden von Blocktransformationen, die den Frame gut approximieren. Im Unterschied zur Einzelbildkodierung werden die Domainblöcke ausschließlich aus vorrangegangenen Frame genommen. Ein weiterer Unterschied ist, daß der geometrische Anteil der Blocktransformation nicht kontrahierend sein muß. Im Gegenteil: Ausschnitte der zu kodierenden Filmsequenz, in denen keine Bewegung vorhanden ist, lassen sich mit kontrahierenden Abbildungen nicht effizient kodieren.

Bei der Wahl des Kodierungsschemas für Referenzframes ist man nicht eingeschränkt. Neben einem fraktalen Schema für Einzelbilder ist ebenso ein DCT-basiertes Verfahren denkbar, oder auch, die Referenzframes unkomprimiert zu übertragen. Die Kodierung der nachfolgenden Frames wird dadurch wenig beeinflußt.

Die fraktalen Verfahren zur Einzelbildkodierung besitzen ein Problem: die Approximation muß durch Iteration wiederhergestellt werden. Eine Iteration der Grauwertfunktion ist etwa so aufwendig, wie die Transformation von einem Frame auf den nächsten. Wenn 10-mal iteriert werden muß, um das Einzelbild wiederherzustellen, hätte der Dekodierer in derselben Zeit 10 Frames dekodieren können. Viele derartige selbstaffine Referenzframes verzögern also die Dekodierung der Sequenz und erfordern zusätzlichen Aufwand, um die Teilbilder synchron anzeigen zu können.

Die Implementierung dieser Arbeit sieht daher ein modifiziertes fraktales Schema zur Kodierung von Referenzframes vor. Das Verfahren ist analog zur Kodierung von Transformationsframes: im Referenzframe wird eine Reihenfolge der zu kodierenden Blöcke $b_1, \ldots, b_n$ eingeführt. Zur Kodierung von Rangeblöcken in b_i, $i > 1$ werden nur Domainblöcke aus den bereits kodierten Teilen des Bildes zugelassen, also aus b_j, $j < i$. Kontrahierende Transformationen und eine Iteration sind nur in Block b_1 nötig, während die übrigen Blöcke, für deren Kodierung auch nichtkontrahierenden Transformationen verwendet werden können, exakt nach einer Anwendung der Transformation dekodiert sind.

Eine weitere Möglichkeit ist es, auf die selbstaffine Kodierung des Referenzframes ganz zu verzichten und stattdessen die Approximation von fest vorgegebenen Bildblöcken aus vorzunehmen, im Sinne einer Kombination von Vektor-Quantisierung und Least Squares Approximation. Für diesen Ansatz sind ebenfalls nichtkontrahierende Transformationen möglich.

Der erste Frame einer Sequenz muß trivialerweise ein Referenzframe sein. Bei den nachfolgenden Frames sind zwar auch Mischformen möglich, allerdings spielen diese in der vorliegenden Arbeit keine Rolle. Ein Frame ist hier entweder

Referenzframe und unabhängig von seinem Vorgänger, oder er ist Ergebnis einer Transformation.

In vielen Anwendungen von Videokompressionsverfahren ist es nötig, kodierte Filme von beliebigen Stellen ab, dekodieren zu können. Beispielsweise muß der Empfänger eines komprimierten Fernsehprogrammes in der Lage sein, beim Einschalten die kodierte Bildfolge ohne übermäßige Verzögerung wiederherzustellen.

Die fraktale Kodierung von Sequenzen erlaubt zwei Ansätze dazu: entweder wird nach einer bestimmten Anzahl von Frames immer ein Referenzframe kodiert, oder es wird die Kontraktivität der Grauwerttransformationen v_k ausgenützt. Im letzten Fall sorgt der Kodierer dafür, jeden Punkt der Bildfläche nach einer endlichen Anzahl von "Carry-forward"-Abbildungen ($\alpha_3^{(k)} = 1$) mit einer Grauwertkontraktion ($| \alpha_3^{(k)} | < 1$) abzubilden. Die dekodierte Sequenz konvergiert dann bei beliebigem Anfangsbild gegen die kodierte Sequenz.

4.2 Algorithmus

Dieses Teilkapitel der Lösung faßt den implementierten Algorithmus zur Kodierung von Transformationsframes zusammen. Die Hauptkomponenten des Algorithmus wurden in den vorrangegangenen Abschnitten behandelt.

Zur Kodierung eines Frames f_i in einer Sequenz sei vorrausgesetzt, daß der vorrangegangene Frame f_{i-1} bereits als f'_{i-1} kodiert ist. Um eine Fehlerfortpflanzung zu vermeiden werden Blocktransformationen nicht von f_{i-1} nach f_i definiert, sondern von f'_{i-1}.

Die Parameter des Algorithmus sind: eine Seitenlänge B der Startblöcke für die Quadtreepartitionierung, die kleinste zugelassene Blockgröße B_{min} und eine Toleranzgrenze r der RMS-Metrik. Die Eingaben sind die Frames f'_{i-1} und f_i.

Der Algorithmus kann durch den Pseudocode in Abbildung 4.2 beschrieben werden. Das Verfahren der Quadtreezerlegung ist an sich ein rekursives. Es wird hier durch einen Stack von Bildblöcken realisiert, der die Operationen `push(`b`)` und `pop()` bietet. Die Funktion `ausgabe()` gibt für den aktuellen Block die Koeffizienten der Blocktransformation und dessen Struktur in der Quadtreezerlegung aus. Konzepte zur Implementierung der Funktion `ausgabe()` werden in den Abschnitten 4.3.2 und 4.3.3 behandelt. Die vier quadratischen Teilblöcke eines Blocks b seien mit $b^{(1)}, \ldots, b^{(4)}$ bezeichnet. Ein Detail der Implementierung ist die Behandlung von Blöcken der Größe 2×2 Pixel. Statt der 4 Koeffizienten der Transformation auszugeben, werden die Grauwerte der 4 Pixel des Blocks ausgegeben.

Der Algorithmus hat zwei Effekte: die Struktur der Quadtreepartitionierung und die Koeffizienten der Grauwerttransformationen werden ausgegeben; und die berechneten Transformationen werden angewandt, damit für die Kodierung des nachfolgenden Frames wieder ein approximierter Frame als Eingabe zur Verfügung steht.

Der Algorithmus liefert eine Approximation des zu kodierenden Frames und den fraktalen Code, mit dem die Approximation erfolgt. Der Algorithmus terminiert immer, weil bei der Quadtreezerteilung die Seitenlänge der Blöcke halbiert wird und eine minimale Seitenlänge als Abbruchkriterium vorgegeben ist.

Ein bereits angesprochenes Problem ist noch offen: das Problem der nichtlösbaren Gleichungssysteme bei der Least-Squares Approximation. Die Implementierung verwendet folgende ad hoc Lösung: Falls für einen Block b das Gleichungssystem unlösbar ist, wird der Block verworfen und durch einen Block ersetzt, der im Kodierer und im Dekodierer bekannt ist und für den das Gleichungssystem beweisbar immer lösbar ist (in der Implementierung ein Block mit konstanter Graustufe).

Der Algorithmus zur Kodierung eines Referenzframes ist völlig analog möglich, mit dem einzigen Unterschied, daß die Ursprungsblöcke für die Least-Squares Approximation aus dem selben Frame genommen werden. Eine andere Möglichkeit ist es, von einem festen Bild ausgehend, das Kodierer und Dekodierer bekannt

- `Zerteile den Frame` f_i `in quadratische Startblöcke` b_j `der Größe` $B \times B$.
- **for all** b_j **do**
  - `push(`b_j`)`
  - **while** b `= pop()` $\neq \emptyset$ **do**
    - **if** `Größe von` b `<` B_{min} **then** **continue while**.
    - `Berechne` $d_{rms}(b, b')$ `(`b' `ist der Block in` f'_{i-1} `mit gleicher Geometrie wie` b`.)`
    - **if** $d_{rms}(b, b') > r$ **then**
      - `Berechne Least-Squares Approximation` $\tilde{b}$ `von` b' `nach` b.
      - `Berechne` $d^{(k)} := d_{rms}(\tilde{b}^{(k)}, b^{(k)})$ $k = 1, \ldots, 4$; `setze` $n :=$ `Anzahl der Teilblöcke mit` $(d^{(k)} > r)$
      - **if** $n \in \{0, 1, 2\}$ **then** `ausgabe(`b`)`
if $d^{(k)} < r$ **then** `ausgabe(`$b^{(k)}$`)`
else `push(`$b^{(k)}$`)`
      - **if** $n \in \{3, 4\}$ **then** `push(`$b^{(k)}$`)` $\forall\, k$.
  - `Führe die Blocktransformation von` b'_j `auf` b_j `durch`

Abbildung 4: Pseudocode zur Berechnung einer Transformation $f'_{i-1} \rightarrow f_i$

ist, den Referenzframe als Transformationsframe zu kodieren. In diesem Fall ist der Algorithmus sogar exakt gleich.

4.3 Implementierung

Dieses Teilkapitel der Lösung behandelt Details der prototypischen Implementierung des angegebenen Algorithmus'. Das implementierte System zur fraktalen Videokompression umfaßt zwei Teilprogramme: einen Kodierer (`kodi`), der Einzelbilder liest und einen fraktalen Code schreibt; und einen Dekodierer (`deko`), der den fraktalen Code liest und die Approximation des ursprünglichen Filmes auf einem X-Window System (X11) anzeigen oder als Einzelbilder auf Dateien ausgeben kann. Eine zusätzliche Funktionalität des Dekodierers erlaubt die Ausgabe der Partitionsstruktur.

Die Programme bestehen aus mehreren in ANSI-C geschriebenen Modulen. Der folgende Abschnitt zeigt die Struktur der Programme. Weiterere Abschnitte behandeln zwei Verfahren zur Quantisierung der Transformationskoeffizienten, von denen nur das triviale implementiert ist und die effiziente Ein- und Ausgabe von Partitionsinformationen. Der letzte Abschnitt dieses Teilkapitels versucht die Frage nach der Parallelisierbarkeit des Codes zu beantworten.

4.3.1 Modulstruktur

Einen Überblick über die Modulstruktur des Kodierers `kodi` zeigt Abbildung 5, die Struktur des Dekodierers zeigt Abbildung 6. Beide Programme teilen eine Reihe von Modulen, die einen Rahmen für Ein- und Ausgabe und für globale Methoden bilden:

1. `ppm_io` - Methoden für das Lesen und Schreiben von (binären) PPM[2] Bildern der Formate P5 und P6. Diese Komponente besteht aus folgenden Dateien:

   ```
   ppm_io.h  rppm_in.c  rppm_out.c
   ```

2. `kode_io` - Methoden für das Lesen und Schreiben von binären Codes zur Repräsentation von Quadtreestrukturen oder fraktaler Transformationskoeffizienten. Dazu gehören folgende Dateien:

   ```
   kode_io.h   kode_io.c  kio_stub.c  kode_in.c
   kode_out.c
   ```

3. `global` - Definition globaler Datenstrukturen, Definition von globalen Rückgabewerten und Methoden zur Ausgabe von Fehlermeldungen und Methoden zur Ausgabe von Informationen über den Kodier- bzw Dekodierprozeß. Diese Komponente umfaßt die Dateien:

   ```
   bildtyp.h    global.h  matrix.h   rueckgabe.c
   rueckgabe.h  typen.h   verbose.c  verbose.h
   version.h
   ```

4. `args` - Methoden zur Verarbeitung der Kommandozeilen Optionen und Argumente. Für Kodierer und Dekodierer sind verschiede, aber strukturell gleiche Dateien vorhanden. Diese Komponente umfaßt:

   ```
   getopt.c    getopt.h    kodi_args.c  kodi_args.h
   deko_args.c  deko_args.h
   ```

[2]PPM steht für "portable pixmap file format", ein rechnerunabhängiges Bildformat von Jef Poskanzer. Für sehr viele andere Bildformate existieren Konversionsprogramme von und nach PPM

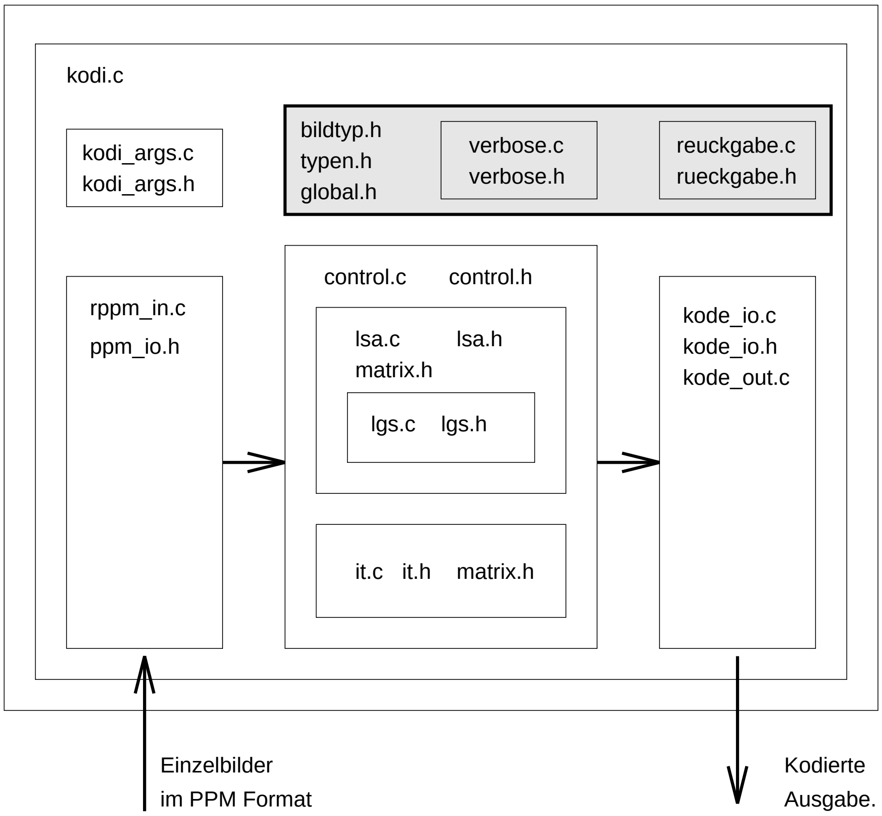

Abbildung 5: Modulstruktur des Kodierers

Der Kodierer enthält neben den Rahmenkomponenten Module, die die Quadtreepartitionierung und die Berechnung der Transformationskoeffizienten vornehmen. Alle Module (außer `kodi.c`) bestehen aus je einer Header-Datei mit Endung "`.h`", die die Deklaration von Datenstrukturen und Prototypen für die exportierten Funktionen enthält und einer Implementationsdatei mit Endung "`.c`", die die Funktionalität erbringt. Der Kodierer umfaßt folgende Module:

`kodi.c`	Hauptmodul. Verantwortlich für die Initialisierung und den Ablauf der Kodierung. Operiert auf Bildern.
`control.*`	Kontrolliert die Quadtreepartitionierung. Operiert auf Blöcken.
`lsa.*`	Berechnet die Least-Squares Approximation mit 4 Koeffizienten. Operiert auf Blöcken.
`lgs.*`	Berechnet die Lösung eines Linaren Gleichungssystems mit 4 Gleichungen für 4 Vektoren simultan.

Der Kodierer benutzt das Modul "`it`" des Dekodierers, um die berechnete Transformation tatsächlich anzuwenden; das beinhaltet insbesondere, den geschriebenen fraktalen Code nocheinmal zu lesen und exakt so zu interpretieren, wie es auch der Dekodierer später wird. Dadurch ist sichergestellt, daß Kodierer und Dekodierer intern immer auf denselben aproximierten Bildern arbeiten, um eine

Fehlerpropagation zu vermeiden, die als Folge von Differenzen auftreten kann.

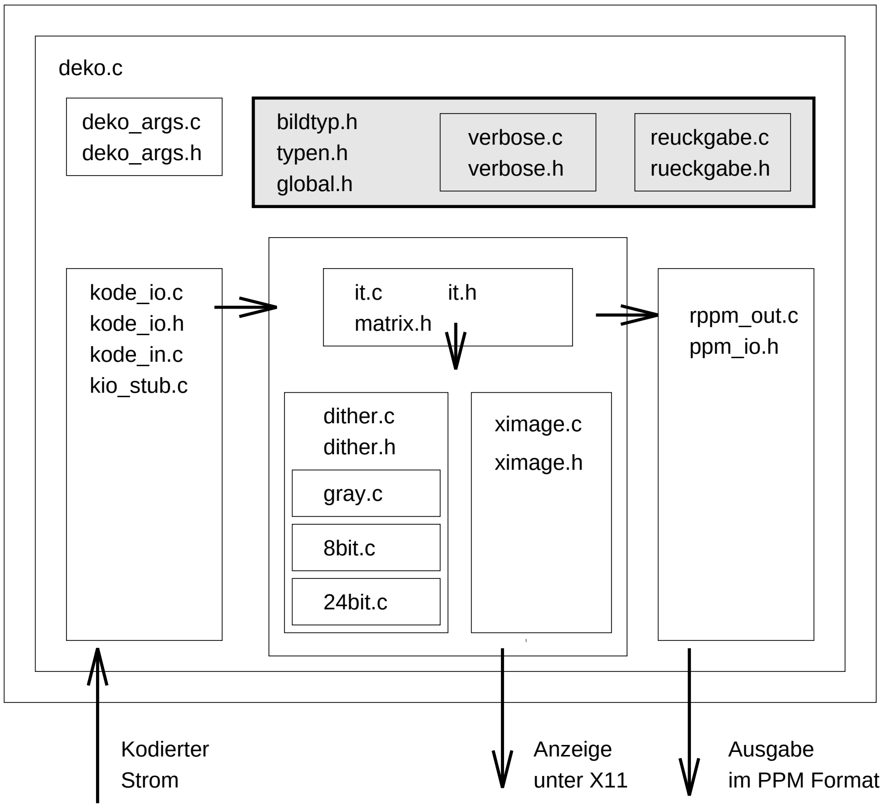

Abbildung 6: Modulstruktor des Dekodierers

Der Dekodierer besteht aus Komponenten, die die fraktalen Codes auf ein gegebenes Bild anwenden und die die Bilder für die Ausgabe unter X11 aufbereiten (Dithering). Er umfaßt folgende Module:

`deko.c`	Hauptmodul. Verantwortlich für die Initialisierung und den Ablauf der Dekodierung. Operiert auf fraktalen Codes.
`it.*`	Iteriert die fraktalen Codes auf den im Speicher befindlichen Bildern.
`ximage.*`	X11 Interface zur Ausgabe der Frames auf einem Display.
`dither.*`	Hauptkomponente des Dithering. Umfaßt die Module `gray.c`, `8bit.c` und `24bit.c`
`gray.c`	Ausgabe des Luminanzkanals für 8-Bit Graustufendisplays.
`8bit.c`	4×4 Ordered Dithering Algorithmus, der im $YCrYb$-Raum arbeitet. Der Algorithmus ist aus einer MPEG Software Implementierung übernommen.
`24bit.c`	Dithering für 24-Bit Truecolor Displays.

Die Hauptdatenstrukturen der Programme sind eine Bild-Struktur und eine Code-Struktur.

Die Bildstruktur (Datentyp `bild_t`) enthält alle Informationen über ein im Speicher befindliches Bild, insbesondere die Daten der $YCrCb$-Komponenten des Bildes selbst sowie die Repräsentation für den X-Server im Dekodierer.

Die Codestruktur (Datentyp `kode_t`) enthält die Informationen über den fraktalen Code. Sie verwaltet Puffer für die Daten der Quadtreestruktur und für die Transformationskoeffizienten. Der fraktale Code eines Frames wird erst nach dessen vollständiger Kodierung auf Datei geschrieben bzw. wird zur Dekodierung Frameweise in den Puffer gelesen.

4.3.2 Quantisierung

Dieser Abschnitt behandelt zwei Verfahren zur Quantisierung der Transformationskoeffizienten. Das triviale Verfahren, das im Prototypen implementiert ist, ist zwar einfach und effizient, besitzt aber spezifische Probleme.

Das einfache Verfahren besteht darin, den durchschnittlichen Wertebereich der Koeffizienten für eine Klasse von Bildern empirisch zu bestimmen. Die erhaltenen Bereiche werden linear auf einen Zahlenbereich abgebildet, der mit einer vorgegebenen Anzahl von Bits kodierbar ist. Die Implementierung verwendet 8 Bit pro Koeffizient.

Folgende Tabelle zeigt die empirisch festgestellten Bereiche für die 4 Koeffizienten. Daraus ergeben sich Werte a, b, die das erhaltene Intervall mit der Rechenvorschrift $x' = ax + b$ auf das Intervall $[0, 255]$ abbilden.

	α_1	α_2	α_3	α_4
Bereich	$[-52, 86]$	$[-56, 69]$	$[-6.05, 4.6]$	$[-295, 1005]$
a	1.8478261	2.04	23.943662	0.19615385
b	52.0	56.0	6.05	295.0

Probleme gibt es mit diesem einfachen Verfahren, ersten wenn Koeffizienten nicht innerhalb des erwarteten Bereichs liegen und zweitens aufgrund der Ungenauigkeit durch die sehr großen vorgegebenen Intervalle. Im ersten Fall fallen die quantisierten Werte aus dem Intervall $[0, 255]$ und werden abgeschnitten. Die adaptive Partitionierung ist in gewissen Grenzen tolerant für solche Fehler, denn wenn die Approximation nicht den Anforderungen entspricht, wird die Blockgröße verringert. Auf diese Weise erhält man immer akzeptable Bildqualität auf Kosten der Kompressionsrate.

Das folgende Verfahren Monros und Dudbridges[Monro et al. 92b] versucht die beiden Probleme zu beseitigen:

Zur Quantisierung von 4 Koeffizienten einer Blocktransformation werden aus dem Domainblock die Grauwerte von 4 festgelegten Punkten

$$f_1 := f(x_1, y_1), \ldots, f_4 := f(x_4, y_4)$$

mit den Koeffizienten der Blocktransformation abgebildet.

$$f'_i = \alpha_1 x_i + \alpha_2 y_i + \alpha_3 f_i + \alpha_4 \tag{35}$$

Erwartungsgemäß liegen die transformierten Grauwerte innerhalb des Intervalls $[0, 255]$, denn sie sind die Approximation von Grauwerten aus diesem Bereich. Sie können daher besser quantiert werden als die eigentlichen Koeffizienten, die theoretischen unbegrenzt sein können. Monro und Dudbridge quantisieren die erhaltenen Werte mit 6 Bit und erzielen damit gute Ergebnisse.

Da im Dekodierer die Grauwerte $f(x_i, y_i)$ des Ursprungblocks bekannt sind, kann dieser die Koeffizienten durch Lösen eines Gleichungssystems der Gleichungen aus (35) zurückberechnen.

Das Gleichungssystem kann für einen festen Satz von Koordinaten statisch gelöst werden. Sinnvoll ist es, möglichst weit auseinanderliegende Punkte zu wählen, am besten die Eckpunkte der Blöcke, damit die einzelnen Gleichungen stärker voneinander unabhängig werden. Eine Lösung für die Eckpunkte eines Blocks (jeweils das Zentrum des Pixels, also die x- und y-Koordinaten 0.5 und $0.5 + \delta$) liefert folgender Ansatz für das Gleichungssystem:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & f_1 \\ 1 & 0.5+\delta & 0.5 & f_3 \\ 1 & 0.5 & 0.5+\delta & f_2 \\ 1 & 0.5+\delta & 0.5+\delta & f_4 \end{bmatrix} \begin{bmatrix} \alpha_4 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} f'_1 \\ f'_3 \\ f'_2 \\ f'_4 \end{bmatrix} \tag{36}$$

Aus der folgenden Lösung des Gleichungssystems können die Koeffizienten abgelesen werden:

$$\begin{bmatrix} 1 & 0.5 & 0.5 & f_1 \\ 0 & \delta & 0 & f_3 - f_1 \\ 0 & 0 & \delta & f_2 - f_1 \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} \alpha_4 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} f'_1 \\ f'_3 - f'_1 \\ f'_2 - f'_1 \\ Z \end{bmatrix} \tag{37}$$

wobei $z = (f_1 - f_2 - f_3 + f_4)$ und $Z = (f'_1 - f'_2 - f'_3 + f'_4)$ ist.

Damit ergeben sich die Koeffizienten als:

$$\begin{aligned} \alpha_3 &= Z/z \\ \alpha_2 &= \tfrac{1}{\delta}((f'_2 - f'_1) - \alpha_3(f_2 - f_1)) \\ \alpha_1 &= \tfrac{1}{\delta}((f'_3 - f'_1) - \alpha_3(f_3 - f_1)) \\ \alpha_4 &= f'_1 - 0.5\alpha_1 - 0.5\alpha_2 - f_1\alpha_3 \end{aligned} \tag{38}$$

Probleme gibt es, wenn das Gleichungssystem nicht eindeutig lösbar ist. Das kann nur passieren, wenn $z = 0$ ist. Da z nur von Werten abhängt, die im Kodierer und im Dekodierer bekannt sind, können beide eine gemeinsame Strategie zur Behandlung dieses Problems entwickeln. Beispielsweise können beide die Grauwerte f_i für die Quantisierung so ändern, daß $z \neq 0$ ist, welches immer möglich ist.

Anmerkung 6 *Das ursprüngliche Verfahren zur Quantisierung von Monro und Dudbridge für die Einzelbildkodierung besitzt eine Schwierigkeit, die für Bewegtbildkodierung wegfällt: bei der Kodierung mit selbstaffinen Transformationen hat der Dekodierer nicht die gleiche Information wie der Kodierer. Bei der Abbildung eines Domainblocks auf seine 4 Teilblöcke hat man jedoch 4 Fixpunkte in den Ecken des Blocks, aus denen man zusammen mit weiteren 12 Punkten die 16 Transformationskoeffizienten des Blocks wiedergewinnen kann.*

4.3.3 Variabel lange Codes

Dieser Abschnitt behandelt die Technik variabel langer Codes zur Beschreibung einer Quadtreepartition eines Bildes.

Variable lange Codes werden in der Huffman-Kodierung eingesetzt. Die Häufigkeit der zu kodierenden Zeichen bestimmt deren Codelänge - je häufiger ein Zeichen auftritt, desto kürzer ist sein Code. Das Ziel dieser Technik ist, eine verlußtfreie Kompression der Daten zu erreichen. Die Codes müssen so gewählt sein, daß sie eindeutig umkehrbar sind.

Die Umkehrbarkeit der Codes ist gesichert, wenn sie sich als binäre Übergangsdiagramme darstellen lassen. Abbildung 7 zeigt die Übergänge für die Quadtreezerlegung von Blöcken.

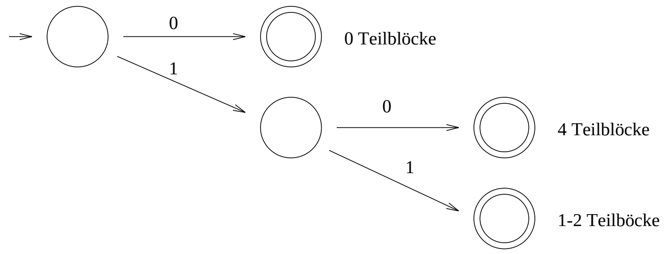

Abbildung 7: Übergangsdiagramm für Blockcodes

Falls der Code für 1-2 Teilblöcke übertragen wird, folgt noch ein 4 Bit Code, der die Lage dieser Blöcke angibt.

Nicht immer ist es nötig, alle 4 Koeffizienten der Grauwerttransformation zu übertragen. Beispielsweise ist für Blöcke ohne Bewegung keine Transformation nötig, ebenso für konstante Blöcke. Falls die Transformationskoeffizienten α_1 und

α_2 nahe bei 0 sind, brauchen diese nicht übertragen werden. Ein Spezialfall muß noch berücksichtigt werden: falls das Gleichungssystem der Least-Squares Approximation nicht lösbar ist, wird ein spezieller Koeffizientencode übermittelt. Abbildung 8 zeigt die variablen Codes für die verschiedenen Typen von Transformationskoeffizienten.

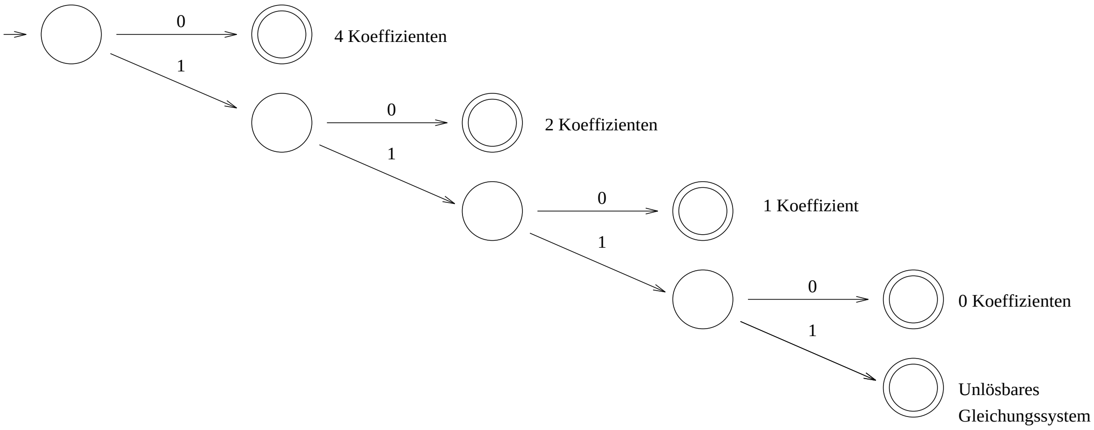

Abbildung 8: Übergangsdiagramm für Koeffizientencodes

Beispiel 4 *Codelängen für verschiedene Blocktypen*

Block ohne Bewegung	*5 Bit (*`01111`*)*
Block ohne Teilblöcke und 4 Koeffizienten	*2 Bit (*`00`*)*
	+ 32 Bit für Koeffizienten.
Block mit 4 Teilblöcken	*2 Bit (*`10`*)*
Block mit 1-2 Teilblöcken und 2 Koeffizienten	*8 Bit (*`11xxx10`*)*
	+ 16 Bit für Koeffizienten

Zum effizienten Lesen der Codes wird immer eine feste Anzahl von p Bits aus dem Puffer gelesen. Mit diesem Wert werden zwei Arrays der Größe 2^p adressiert, von denen das eine die Anzahl n der gültigen Bits enthält und das andere den tatsächlichen Code. Der Aufwand hängt bei diesem Verfahren nicht von der Länge eines Codes ab.

4.3.4 Parallelisierbarkeit

Der in dieser Arbeit entwickelte Algorithmus läßt sich in natürlicher Weise auf zwei Ebenen der Granularität parallelisieren, bedingt auch auf einer weiteren:

1. Auf der Ebene von Startblöcken für die Quadtreepartitionierung ist der Algorithmus sehr gut für SIMD Rechner geeignet, weil keine Daten zwischen den Prozessoren während der Berechnung ausgetauscht werden müssen. Jeder Prozessor erhält zu Beginn der Berechnung den zu kodierenden Bildblock als Eingabe. Alle Prozessoren führen dann dieselben Berechnungen synchron durch; Prozessoren, die mit ihren Blöcken fertig sind warten. Es kann nur bis zur Anzahl der vorhandenen Blöcke im Bild skaliert werden. Der Algorithmus eignet sich daher am besten für SIMD Rechner mit einer moderaten Anzahl (100 - 200) von Prozessoren.

   Der maximale Speedup wird erreicht, wenn sämtliche Teilblöcke der Quadtreepartitionierung erzeugt werden, weil dann der SIMD Recher mit n rechnenden Prozessoren genau n mal so schnell ist, wie ein Einzelprozessor. In diesem Fall ist auch die Prozessorauslastung am besten. Die schlechteste Prozessorauslastung tritt in folgenden Fall auf: genau ein Block erzeugt sämtliche Teilblöcke, während die anderen keine Teilblöcke besitzten, d.h. außer dem rechnenden Prozessor warten alle Prozessoren. Die Kodierzeit erreicht in beiden extremen Fällen ihr Maximum. Für diese Granularität der Parallelisierung ist die maximale Kodierzeit nahe der durchschnittlichen, bzw. sie kann ohne großen Effizienzverlust konstant gemacht werden.

2. Auf der Ebene von Frames läßt sich eine sehr grobe Parallelisierung für Multiprozessoren oder Multicomputer erreichen. Diese Art der Parallelisierung ist für fast alle denkbaren Kodierungverfahren möglich. Jeder Prozessor bearbeitet unabhängig k Frames $f_n, \ldots, f_{n+k}$ der zu kodierenden Bildfolge und erzeugt daraus einen Referenzframe f'_n und $k-1$ Transformationsframes. Diese Parallelisierung, die mit einer Pipeline-Technik vergleichbar ist, läßt sich beliebig skalieren, vorrausgesetzt, man kann ausreichend viele Bilder schnell genug zu den Prozessoren bringen. Jeder Prozessor muß Speicher für die k Teilframes bereitstellen, die er kodiert. Bei entsprechender Skalierung ist es immer möglich so viele kodierte Frames je Zeiteinheit auszugeben, wie unkodierte gelesen werden - jedoch mit einer Verzögerung von genau der Zeit, die ein einzelner Prozessor für die Kodierung seiner k Frames benötigt.

   Beispielsweise ergaben Tests des prototypischen Kodierers auf einem Intel Paragon Parallelrechner eine durchschnittliche Kodierzeit von 1.5 Sekunden pro (384x288 Pixel) Frame mit einem Prozessor. Um 25 Frames pro Sekunde zu kodieren, bräuchte man also ca. 38 - 40 Prozessoren eines solchen Rechners. Die zeitliche Verzögerung zwischen dem Anfallen der Bilder und der Ausgabe der kodierten Sequenz würde $k \times 1.5$ Sekunden betragen.

3. Eine Parallelisierung mit der Granularität von einem Bildpixel pro Prozessor auf massiv parallelen SIMD Maschinen ist für die Summationen zur Berechnung der Least-Squares Approximation und für die Berechnung des RMS-Fehlers denkbar. Der Algorithmus für die Quadtreepartitionierung ist jedoch für diese feine Granularität nicht gut geeignet.

   Auf einem SIMD-Rechner mit mehreren tausend Prozessoren ist daher eher eine Kombination aus den ersten beiden Parallelisierungsmöglichkeiten sinnvoll als die Parallelisierung auf Pixelebene. Das würde beinhalten, daß alle Prozessoren jeweils synchron einen Block bearbeiten, aber insgesamt mehrere Frames aus verschiedenen Teilen einer Sequenz gleichzeitig auf den Prozessoren kodiert werden. Diese Kombination ist auch deshalb interessant, weil sie eine nahezu konstante Kodierzeit mit mit einer hohen Skalierfähigkeit vereint.

5 Ergebnisse

Dieses Kapitel faßt die wichtigsten quantitativen Ergebnisse der Studienarbeit zusammen. Es beschreibt verschiedene Zusammenhänge zwischen Kodierqualität, Kompressionsfaktor und Kompressionszeit, die mit der prototypischen Implementierung des Verfahrens erreicht werden.

5.1 Testsequenz

Alle Werte beziehen sich auf Teile einer 101 Frames umfassende Testsequenz. Sie zeigt eine sich verbeugende und sich drehende Person vor einem unbeweglichen Hintergrund[3]. Abbildung 13 zeigt Beispielframes (Frame Nr. 8 und Nr. 20) aus diesem Film, zusammen mit einer fraktalen Kodierung und der zugehörigen Partitionierung. Die Sequenz ist leicht zu kodieren, denn Bewegung beschränkt sich auf ca. ein Drittel der Fläche. Die Ergebnisse sind nicht auf andere Filme übertragbar.

Die Testsequenz stand nach der Aufnahme als MPEG-Kodierung zur Verfügung. Die Sequenz besitzt folgende Charakteristika:

Anzahl der Frames	101
Bildfläche	384×288 Pixel
Farben	24 Bit RGB
Größe der Sequenz (unkodiert)	33 509 376 Byte
Größe pro Frame (unkodiert)	331 776 Byte
MPEG: Zwischendatei-Größe	4 835 648 Byte
MPEG: durchschnittliche Größe pro Frame	46 977 Byte
Aufnahme	17 - 18 Frames pro Sekunde

Die MPEG Kodierung ist von einer sehr guten Qualität; es sind keine störenden Artefakte erkennbar. Die Bewegung in der Sequenz ist verwaschen aufgrund einer langen Blendenöffnungszeit der Kamera, d. h. die Einzelbilder sind an den Stellen mit schneller Bewegung unscharf. Für vergleichende Ergebnisse werden nur die ersten 40 Frames betrachtet.

Die kodierten Sequenzen enthalten immer einen Referenzframe als ersten Frame und sonst nur Transformationsframes (wobei der Referenzframe als Transformationsframe gegebüber einem internen Bild kodiert ist). Bei verschiedenen Läufen der Kodierung wurde jeweils mit Blöcken der Seitenlänge 32 Pixel begonnen und die Quadtreepartitionierung über 5 Stufen hinweg durchgeführt.

Das folgende Beispiel enthält ein konkretes Kodierergebnis für die gesamte 101 Frames umfassende Sequenz:

[3] Der Verfasser der Arbeit im Roboterlabor des IPVR

Beispiel 5 *Bei einer vorgegebenen RMS-Toleranz von 7.5 läßt sich die gesamte Sequenz auf eine Größe von* 669 494 *Byte komprimieren (Faktor 1 : 50). Eine zusätzliche Huffmankodierung bringt eine weitere Kompression von 11 % auf* 595 683 *Byte (Faktor 1 : 56.25). Die Kodierzeit beträgt 115.17 Sekunden auf einer SUN SPARCstation 10 Doppelprozessormaschine mit 50 Mhz Prozessoren. Das ergibt eine durchschnittliche Kodierzeit von ca. 1.4 Sekunden pro Frame. Der durchschnittliche RMS-Fehler liegt für den Y-Kanal bei 6.907, für den Cr-Kanal bei 5.890 und für den Cb-Kanal bei 5.661.*

Der Dekodierer kann diese Sequenz auf der SPARCstation 10 (bei Verwendung eines 8-Bit Farbditheralgorithmus) mit ca. 15 Frames pro Sekunde anzeigen, während die reine Dekodierleistung (ohne Anzeige und Dithering) ca. 28 Frames pro Sekunde beträgt. Falls nur der Luminanzkanal dekodiert wird, erhöht sich dieser Wert auf 42 Frames pro Sekunde.

5.2 RMS-Toleranz als Parameter

Dieser Abschnitt vergleicht die Kodierergebnisse für die ersten 40 Frames der Testsequenz für verschiedene RMS-Toleranzgrenzen. Die Kodierläufe der 40 Frames erfolgten auf einem Sillicon-Graphics Indigo Rechner, der während der Kodierung eine mittlere Prozessorlast (4 Benutzer) aufwies.

Die folgende Tabelle enthält eine Übersicht über Durchschnittswerte der erzielten Ergebnisse. Die Tabelle vergleicht Dateigrößen der kodierten Dateien, Kompressionsfaktoren, Kodierzeiten und die erzielten RMS-Fehler für den Luminanzkanal Y und die Crominanzkanäle Cr und Cb. Bei den Kodierzeiten (in Sekunden) werden die tatsächlich verbrauchte Zeit und die Prozessorzeit unterschieden, um den tatsächlichen Aufwand unabhängig von der Prozessorlast besser abschätzen zu können.

Toleranz	Größe	Faktor	Zeit	CPU Zeit	$d_{rms}Y$	$d_{rms}Cr$	$d_{rms}Cb$
4.0	1034764	12.82	95.8	70.4	5.24	3.96	3.80
5.0	596748	22.26	83.3	54.4	5.45	4.45	4.26
6.0	413078	32.18	54.5	45.4	6.05	5.05	4.63
6.5	363281	36.59	59.0	44.5	6.30	5.37	4.87
7.0	327543	40.60	74.4	44.0	6.70	5.79	5.29
7.5	298089	44.62	54.3	39.3	6.99	5.99	5.54
8.0	274350	48.49	48.6	40.8	7.29	6.30	5.80
8.5	253293	52.53	58.2	40.7	7.65	6.35	6.06
9.0	235001	56.63	62.0	42.6	8.02	6.68	6.54
9.5	217984	61.07	44.4	37.9	8.42	6.99	6.95

Die durchschnittlichen Prozessorzeiten für die Kodierung eines Frames liegen zwischen 1.76 Sekunden und 0.95 Sekunden pro Frame. Der erkennbare Zusammenhang zwischen RMS-Toleranz und Kodierzeit ist auf den Algorithmus zur Quadtreepartitionierung zurückzuführen, dessen Aufwand direkt von der RMS-Toleranz abhängt.

In den folgenden 3 Diagrammem werden Ergebnisse für die 40 Einzelframes für jeweils die RMS-Toleranzen 5.0, 7.5 und 9.0 dargestellt.

Abbildung 9 zeigt die Länge der kodierten Frames. Die Kodierung des Referenzframes (Frame 0) ist um eine Größenordnung schlechter als die Kodierung der Transformationsframes (1-39). Die Abbildung zeigt auch, daß die Framegröße in einem direkten Zusammenhang zum vorgegebenen RMS-Toleranz steht, denn die 3 Kurven verlaufen ab den 5. Frame parallel. Die Kodierung mit RMS-Toleranz 5.0 fällt für die ersten 4 Frames aus diesem Zusammenhang, weil bei der Toleranzgrenze von 5.0 der Hintergrund nicht als unbeweglich erkannt wird. Das ist ein Problem dieses Verfahrens, das von bereits approximierten Frames ausgeht, um neue Frames zu kodieren.

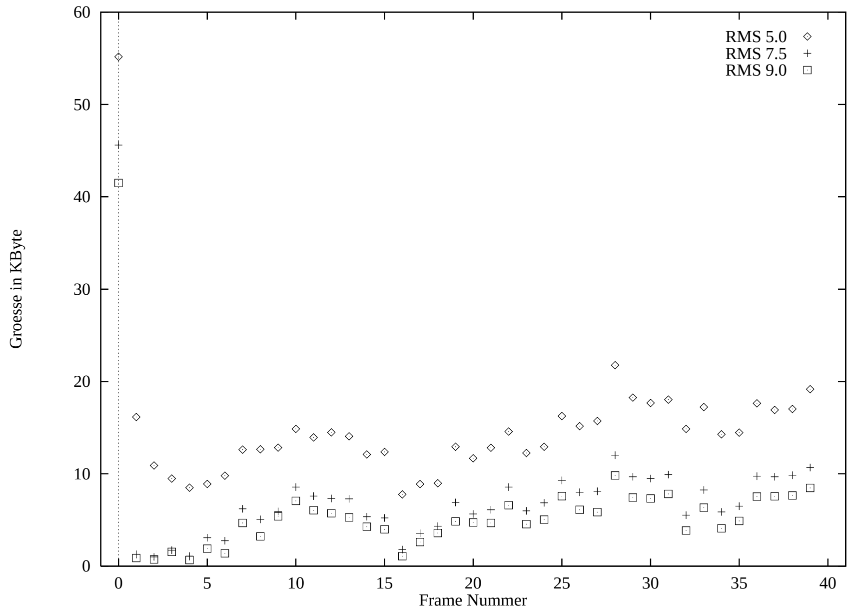

Abbildung 9: Größen der einzelnen Frames

Eine Übersicht über die Kodierzeiten für die einzelnen Frames gibt Abbildung 10. Ein direkter Zusammenhang zwischen der Kodierzeit der einzelnen Frames und der RMS-Toleranz wie in der Tabelle der Durschnittswerte ist nicht zu erkennen. Die Kodierzeiten für einzelne Frames werden zu stark durch die schwankende Prozessorlast beeinflußt. Die Graphik zeigt jedoch etwa den Bereich, in dem die Kodierzeit liegt: sie schwankt meist zwischen 1 - 2 Sekunden und liegt immer unter 5 Sekunden.

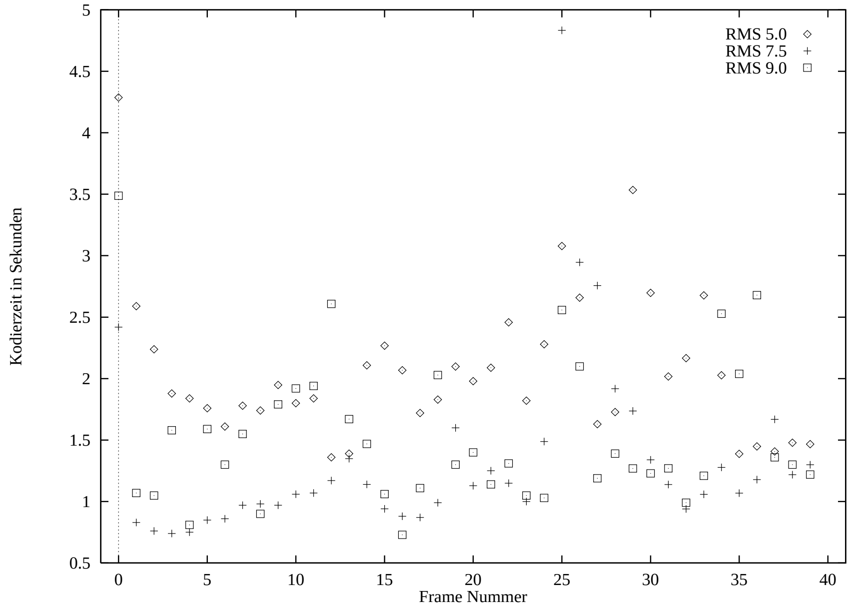

Abbildung 10: Kodierzeiten der einzelnen Frames

Die Übersicht über die erreichten RMS-Fehlerwerte in Abbildung 12 beschränkt sich auf die Luminanzkomponente Y. Das Diagramm zeigt einen signifikanten Zusammenhang zwischen der RMS-Toleranz und dem erreichten RMS-Fehler für nahezu alle Frames.

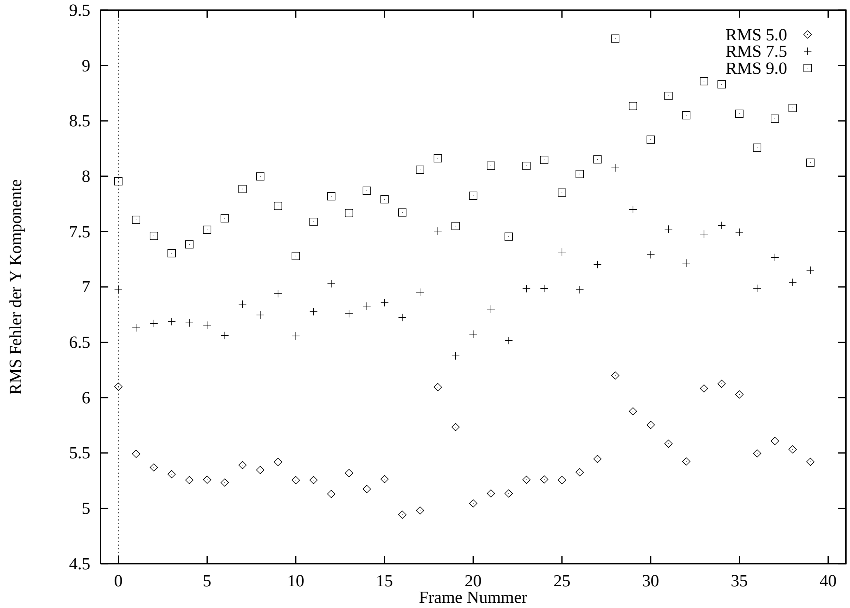

Abbildung 11: RMS-Fehler der einzelnen Frames

5.3 Kodierergebnisse

In diesem Abschnitt werden die erreichten RMS-Fehler in Zusammenhang mit der Kompression gesetzt. Um einen Vergleich mit JPEG zu bekommen, wurde der erste Frame der Sequenz mit verschiedenen Qualitätsparametern JPEG-kodiert. Unberücksichtigt bleibt dabei die Tatsache, daß die Sequenz bereits mit einem DCT-basiertem Verfahren kodiert wurde, die sich auf den gemessenen RMS-Fehler auswirken kann. Die JPEG-Kodierung ist in der Leistungsfähigkeit vergleichbar mit einer MPEG-Kodierung mit "I-Frames".

Abbildung 12 zeigt den Zusammenhang zwischen RMS-Fehler und den Gößen der kodierten Dateien für die Durchschnittswerte aus 40 Frames, für Referenzframes und für den gut zu kodierenden Frame Nr. 16 (jeweils die Y-Kanäle). Bei hohen Kompressionsraten ist Leistung des Verfahrens im Durchschnitt nahe bei der Leistung von JPEG.Einzelne Frames lassen sich zum Teil besser kodieren, aber die Kodierleistung für Referenzframes ist sehr schlecht.

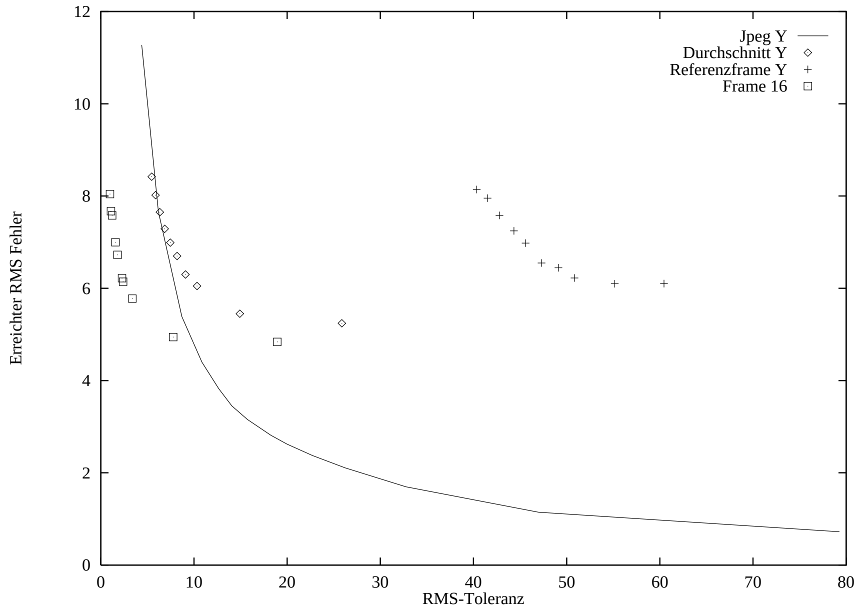

Abbildung 12: Verhältnis zwischen Kompressionsgrad und RMS-Fehler

5.4 Beispielframes

Abbildung 13 zeigt zwei Frames des Sequenz. Frame 8 ist ein scharfes Einzelbild, in dem sich nur eine langsame Bewegung gegenüber den benachbarten Frames abspielt. Frame 20 dagegen weist eine schnelle Bewegung auf und ist an den sich bewegenden Stellen unscharf. Als Beispiel für eine fraktale Kodierung dient die Kodierung mit RMS-Toleranz 8.0. Frame 8 ist auf eine Länge von 4 701 Byte (1:70.5) komprimiert bei einem RMS-Fehler des Y-Kanals von 7.21. Frame 20 ist auf eine Länge von 5 463 Byte (1:60.7) komprimiert bei einem RMS-Fehler von 6.72. In beiden Beispielen sind deutliche Blockartefakte in der fraktalen Kodierung erkennbar. Unter den kodierten Bildern ist die Partitionierung abgebildet. Die Abbildungen 14 und 15 zeigen die kodierten Frames nocheinmal in Vergrößerung.

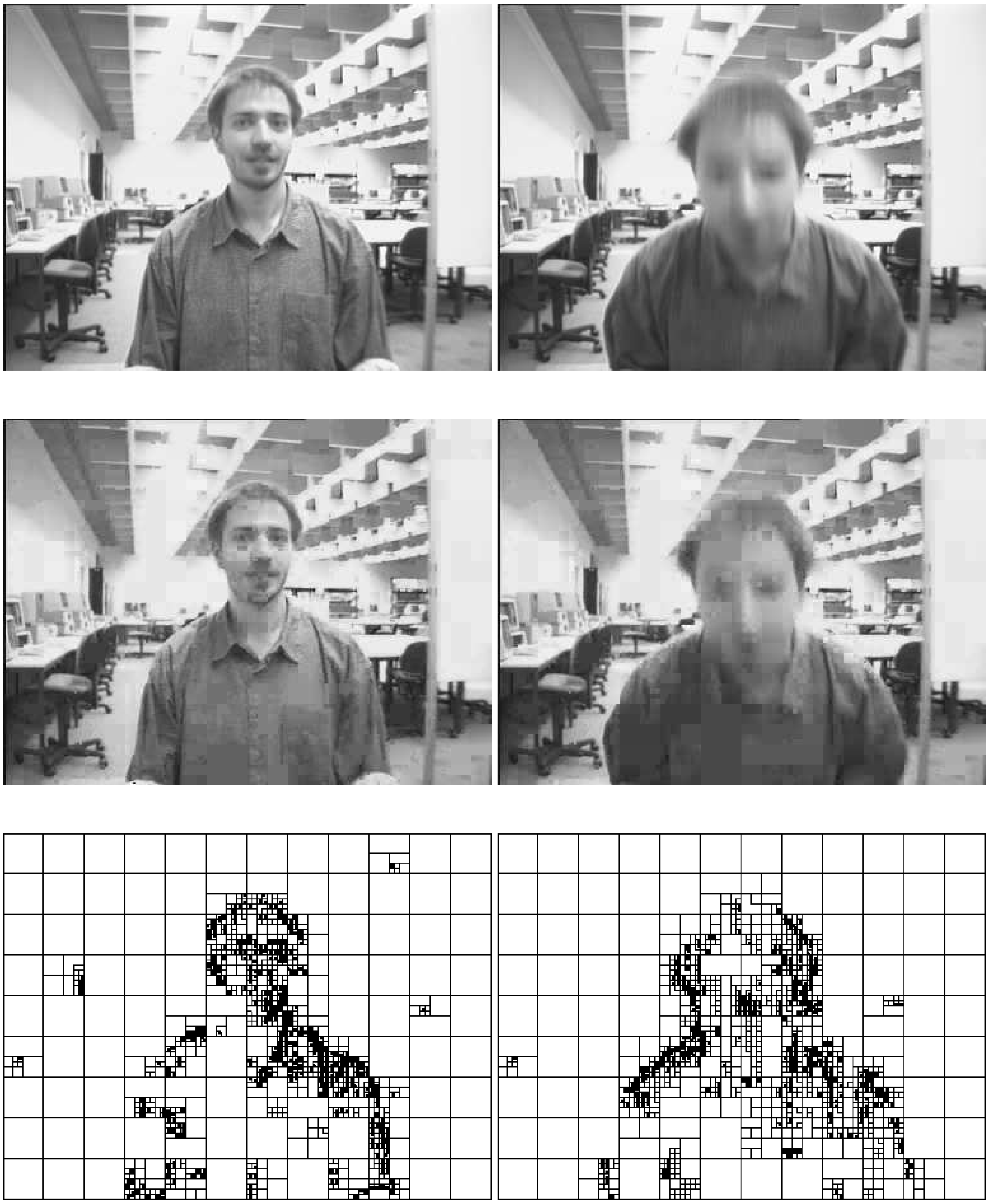

Abbildung 13: Orginalframes 8 und 20, fraktale Kodierungund Partitionierungen

Abbildung 14: Vergrößerung der fraktalen Kodierung für Frame 8

Abbildung 15: Vergrößerung der fraktalen Kodierung für Frame 20

A INTERNET Quellen

Das INTERNET bietet eine Reihe von aktuellen Informationen zum Thema "Fraktale Bildkompression". Die für diese Arbeit verwendeten Dienste im INTERNET sind "USENET News", "Anonymous FTP" (File Transfer Protocol) und "WWW" (World Wide Web).

1. USENET News ist ein Diskussionsforum, das eine baumartige Hierarchie verschiedener Themengruppen besitzt. Relevante Gruppen für diese Arbeit sind:

   `comp.compression`
   `comp.compression.research`
   `sci.fractals`
   `comp.multimedia`
   `comp.graphics`

   In mehreren Newsgruppen werden "FAQ's" (Frequently Asked Questions) geführt. "FAQ's" sind Texte mit Antworten zu den am meisten in der Newsgruppe gefragten Fragen. Im FAQ der Gruppe comp.compression befindet sich zum Beispiel ein umfangreicher Abschnitt zum Thema "Fraktale Bildkodierung".

2. Anonymous FTP ermöglicht es, Dateien von entfernten Rechnern zu übertragen ohne eine Benutzerkennung auf dem entfernten Rechner zu benötigen. Solche anonymen FTP-Server archivieren u.a. Programmcode, Dokumentationen, Bildmaterial und technische Berichte. Die folgenden FTP-Server bieten Material zur fraktalen Bildkodierung:

   - `legendre.ucsd.edu:/pub/Research/Fisher`
     enthält Arbeiten von Yuval Fisher. Dort befindet sich das vorab veröffentlichte Inhaltsverzeichnis und die Literaturliste aus Fishers neuem Buch zur fraktalen Kompression [Fisher 94].
   - `lyapunov.ucsd.edu:/pub/fractal`
     Programmcode von W. D. Young.
   - `schmance.uwaterloo.ca:/u/ftp/pub/Fractal`
     enthält eine umfangreiche Literaturliste; zusammengestellt von John Kominek (`fractal.biblio.ps`).
   - `fidji.informatik.uni-freiburg.de:/papers/fractal`
     bietet eine Zusammenstellung von Artikeln (im PostScript Format) zur fraktalen Bildkodierung, geführt von D. Saupe und R. Hamazaoui.

  - `kurosawa.unice.fr:/pub/compression/fractals`
    enthält ausführbaren Programmcode von Y. Fisher.
  - `vision.auk.dk:/pub/Limbo`
    bietet Programmcode zu [Frigaard et al. 94] an.

3. WWW ist ein globales Hypertext System, in das verschiedene Dokumenttypen integriert werden können. Ein interessanter Startpunkt für eine Reise durch einen fraktalen Hypertextraum ist das von Yuval Fisher angebotene Dokument:

   `http://legendre.ucsd.edu/Research/Fisher/fractal.html`

   Es enthält eine Beschreibung von Fishers neuem Buch, Material zu fraktaler Bildkompression (Bilder, Programmcode und Paper), und Verweise auf andere im INTERNET zugängliche Angebote zu fraktaler Bildkodierung.

B Installation der Software

Die für diese Studienarbeit entwickelte Software, bestehend aus einem fraktalen Kodierer (`kodi`) und einem Dekodierer (`deko`), ist auf Unix-kompatiblen Systeme mit X11 lauffähig. Der Quellcode, eine Kollektion von Modulen in der Programmiersprache ANSI-C, wird als komprimierte `tar`-Datei (tape archive) weitergegeben. Der Dateiname setzt sich aus einem Softwarenamen (`fractal_movie`), einer Versionsnummer und Suffixen für das Dateiformat zusammen. Die aktuelle Version bei Abgabe dieses Berichts ist `1.0`, so daß sich insgesamt folgender Name ergibt:

`fractal_movie-1.0.tar.gz`

Mit folgenden Befehlen kann die Datei ausgepackt werden:

```
> gzip -d fractal_movie-1.0.tar.gz
> tar -xvf fractal_movie-1.0.tar
> cd fractal_movie-1.0
```

Ergebnis dieser Befehle ist, daß sich die Quelldateien in einem neu angelegten Verzeichnis `fractal_movie-1.0` befinden.

Die Kompilation des Kodierers `kodi` erfolgt mit folgenden Befehlen:

```
> make -f Makefile.kodi
```

Der Dekodierer `deko`, für den ein `Imakefile` vorhanden ist, wird mit folgenden Befehlen erzeugt:

```
> xmkmf
> make
```

Alternativ kann der Dekodierer mit:

```
> make -f Makefile.deko
```

kompiliert werten, falls der Befehl `xmkmf` nicht verfügbar ist.

Kodierer und Dekodierer konnten erfolgreich auf folgenden Systemem kompiliert werden (jeweils mit X11R4 oder X11R5):

- SunOS 5.3 (Solaris, Sun SPARC 10)
- SunOS 4.1.3 (Sun SPARC 2)
- AIX 3.2.5 (IBM RS/6000)
- HPUX A.09.01 (Hewlet Packard HP 9000/730)
- IRIX 4.0.5H (Silicon Graphics Indigo 2)
- OSF/1 1.0.4 (Intel Paragon)
- Ultrix 4.3 (DEC 5000)
- Linux 1.1.17 (Intel 80386 Architektur)

C Verfügbare Optionen

C.1 Kodierer `kodi`

Das Programm `kodi` kann mit folgenden Optionen aufgerufen werden:

`-b`	`--maxblocksize`	Blockgröße für die Quadtreepartitionierung in Pixel. (Integer Argument).
`-h`	`--help`	Gibt eine Versionsnummer und eine Liste der verfügbaren Optionen aus. (Nicht alle sind implementiert).
`-l`	`--quadtreelevel`	Anzahl der Stufen der Quadtreepartitionierung. (Integer Argument).
`-o`	`--output`	Name der Ausgabe-Datei für die kodierte Sequenz. (String Argument).
`-r`	`--rms`	Grenzwert der Root Mean Square Metrik für die Partitionierung (Floating Point).
`-s`	`--synchron`	Kodierer wartet, bis Eingabe Bilder lesbar sind. Diese werden nach der Kodierung gelöscht. Nützlich, um die Ausgabe eines anderen Prozesses als Eingabe zu benutzen, ohne alle Bilder speichern zu müssen. Eingabefiles werden durch C-printf Format spezifiziert.
`-v`	`--verbose`	Umfang der auszugebenden Informationen zur Kodierung (Integer Argument). 0: keine Ausgabe, 1: Kodierzeit und Kompressionsrate werden angezeigt, 2: zusätzlich wird der Fehler ausgegeben.
`-V`	`--version`	Gibt die Versionsnummer aus.

Zur Kodierung einer Testsequenz, deren Einzelbilder unter

```
in000.ppm, in001.ppm ... in100.ppm
```

vorliegen, ist folgender Aufruf möglich:

```
> kodi -b32 -l5 -r8.0 -v2 -o out.lsa in*.ppm
```

Falls die Eingabedateien von einem weiteren Prozeß synchron erzeugt werden, wird der Kodierer durch folgenden Aufruf gestartet:

```
> kodi -b32 -l5 -r8.0 -v2 -s -o out.lsa in%3.3d.ppm
```

Bedeutung dieser Optionen: `-b32`: die Blockgröße für die Quadtreepartitionierung ist 32×32 Punkte, `-l5`: maximal sind 5 Stufen zulässig (bis zu 2×2

Blöcken), `-r8.0`: bei einem RMS-Fehler von 8.0 oder besser wird in der Partitionierung abgebrochen, `-v2`: maximale Ausgabe von Informationen zur Kodierung, `-s`: kodierer wartet, bis Eingabe lesbar, nach dem Lesen wird die Datei gelöscht, `-o out.lsa`: Name der Ausgabedatei.

Alternativ können auch lange Optionsname (oder eindeutige Prefixe davon, oder lange und kurze Optionen gemischt) angegeben werden, z.B.:

```
kodi --maxblock=32 --qaudt=5 --rms=8.0 -v2 -o out.lsa in*.ppm
```

C.2 Dekodierer `deko`

Das Programm `deko` akzepiert die folgenden Optionen:

`-d`	`--dither`	Art des Dithering-Algorithmus für die Darstellung der Sequenz. (String Argument). Verfügbar sind: grau, 8bit, 24bit. Es genügt die Angabe eines Prefixes.
`-h`	`--help`	Gibt eine Übersicht über die verfügbaren Optionen aus (sind aber nicht alle implementiert).
`-l`	`--loop`	Sequenz wird endlos angezeigt.
`-n`	`--nodisplay`	Kein Display. Nützlich für Zeitmessung der Dekodierung ohne Dithering und Display bzw. für die Ausgabe der Dekodierten Bilder in Dateien.
`-o`	`--output`	Dateiprefix für die Ausgabe der Einzelbilder (String). An das Dateiprefix wird eine 3 Stellige Sequenznummer beginnend mit 0 und ein Suffix (`.ppm` oder `.pgm`) angehängt. Ohne die Option wird keine Dateiausgabe gemacht.
`-s`	`--sleep`	Angabe der Zeit in Sekunden, die der Dekodierer zwischen jedem Bild pausieren soll. (Integer).
`-p`	`--partition`	Es soll nur die Struktur der Partitionierung angezeigt werden.
`-v`	`--verbose`	Ausgabe nützlicher Informationen (ohne Argument).
`-V`	`--version`	Gibt die Versionsnummer aus.

Der Minimalaufruf benötigt als Argument einen Dateinamen, z.B.:

```
> deko out.lsa
```

→ Sequenz in der Datei `out.lsa` wird in einem X11-Fenster angezeigt.

Verwendete Literatur

[Ali et al. 92] Ali, Maaruf, C. Papadopoulos und T. Clarkson: *The Use of Fractal Theory in a Video Compression System*. In: Storer, James, Hrsg.: *IEEE Data Compression Conference Proceedings '92*, S. 259–268, 1992.

[Ammeter et al. 93] Ammeter, Harald und K. Hazeghi: *Fraktale Bilddatenkompression macht Fortschritte*. Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätzwerke, Bd. 84, Nr. 7, S. 11 – 16, 1993.

[Anson 93] Anson, Louisa F.: *Fractal Image Compression*. Byte, Nr. 10, S. 195–202, 1993.

[Apiki 93] Apiki, Steve: *Compression with Fractals*. Byte's Essential Guide to Windows, Spring, S. 95–96, 1993.

[Baharav et al. 94] Baharav, Z., D. Malah und E. Karnin: *Hierarchical Interpretation of Fractal Image Compression and Its Application to Fast Decoding*. Technischer Bericht Technion – Israel Institute of Technology, Department of Electrical Engineering, Haifa 32000, Israel, 1994.

[Baran 93] Baran, Nicholas: *Fractal Compression Goes On-Line*. Byte, Nr. 9, S. 40, 1993.

[Barnsley 88a] Barnsley, Michael F.: *Fractal modelling of real world images*. In: Peitgen, Heinz Otto und D. Saupe, Hrsg.: *The science of fractal images*, S. 219–242. Springer, New York, 1988.

[Barnsley 88b] Barnsley, Michael F.: *Fractals Everywhere*. Academic Press, Boston (u.a.), 1988.

[Barnsley et al. 85] Barnsley, M. F. und S. Demko: *Iterated function systems and the global construction of fractals*. In: *Proceedings of the Royal Society, London*, Bd. A 399, S. 243–275. Royal Society, London, 1985.

[Barnsley et al. 86] Barnsley, M. F., V. Ervin, D. Hardin und J. Lancaster: *Solution of an inverse problem for fractals and other sets*. In: *Proceedings of the National Academie of Science, USA*, Bd. 83, S. 1975–1977. National Academie of Science, 1986.

[Barnsley et al. 88a] Barnsley, Michael F., A. Jacquin, F. Malassenet, L. Reuter und A. D. Sloan: *Harnessing Chaos for Image Synthesis*. In: *ACM SIGGRAPH, Computer Graphics*, Bd. 22, S. 131–140. ACM, 1988.

[Barnsley et al. 88b] Barnsley, Michael F. und A. D. Sloan: *A better way to compress images*. Byte, Nr. 1, S. 215–223, 1988.

[Barnsley et al. 89] Barnsley, Michael F., J. H. Elton und D. P. Hardin: *Recurrend iteraded function systems.* Constructive Approximation, Bd. 5, Nr. 1, S. 3–31, 1989.

[Barnsley et al. 92] Barnsley, Michael F. und L. P. Hurd: *Fractal Image Compression.* AK Peters, Wellesley, Massachusetts, 1992.

[Barthel et al. 93] Barthel, Kai Uwe, T. Voyé und P. Noll: *Improved Fractal Image Coding.* In: *Proceedings of PCS*, 1993.

[Barthel et al. 94] Barthel, Kai Uwe und T. Voyé: *Adaptive Fractal Image Coding in the Frequency Domain.* In: *Proceedings of International Workshop on Image Procession: Theory, Methodology, Systems and Applications*, Budapest, Ungarn, 1994.

[Beaumont 91] Beaumont, J. M.: *Image data compression using fractal techniques.* British Telecom Technological Journal, Bd. 9, Nr. 4, S. 93–109, 1991.

[Cochran et al. 94] Cochran, Wayne O., J. C. Hart und P. J. Flynn: *Fractal Volume Compression.* Technischer Bericht School of EECS, Washington State University, 1994.

[Davoine et al. 94a] Davoine, F., E. Bertin und J.-M. Chassery: *From rigidity to adaptive tessellations for fractal image compression: comparative studies.* Technischer Bericht TIMC–IMAC Laboratory, Grenoble Cedex, Frankreich, 1994.

[Davoine et al. 94b] Davoine, F. und J.-M. Chassery: *Adaptive Delaunay Triangulation for Attractor Image Coding.* Technischer Bericht TIMC–IMAC Laboratory, Joseph Fourier University, Grenoble Cedex, Frankreich, 1994.

[Dettmer 92] Dettmer, Roger: *Form and functions. Fractal based image compression.* IEE Review, Bd. 38, Nr. 9, S. 323–327, 1992.

[Dudbridge 92] Dudbridge, F.: *Image Approximation by self affine fractals.* Doktorarbeit, University of London, 1992.

[Falconer 90] Falconer, Kenneth: *Fractal Geometry. Mathematicals Foundations and Applications.* John Wiley & Sons, Chichester, 1990.

[Fisher 92] Fisher, Yuval: *Fractal image compression.* In: *Siggraph 92 Course Notes*, 1992.

[Fisher 94] Fisher, Yuval: *Fractal Compression. Theory and Application to Digital Images.* (erscheint Sept. 94 bei Springer, Verfügbar sind ein Vorabdruck des Inhaltsverzeichnisses und der Literaturliste), 1994.

[Fisher et al. 92] Fisher, Yuval und A. F. Lawrence: *Fractal image compression for mass storage applications.* In: *Proceedings SPIE. (Image Storage and Retrieval Systems)*, Bd. 1662, S. 244–255, San Jose. SPIE, 1992.

[Fisher et al. 94] Fisher, Y., D. Rogovin und T. P. Shen: *Fractal (Self-VQ) Encoding if Video Sequences.* In: *(erscheint in:) Proceedings of the Spie*, Visual communications and image processing '94, Chicago, Illinois, 1994.

[Frigaard et al. 94] Frigaard, Carsten, J. Gade, T. J. Hemmingsen und T. Sand: *Image Compression Based on a Fractal Theory.* Technischer Bericht Institute for Electronic Systems, Aalborg University, Denmark, 1994.

[Gipser 90a] Gipser, T.: *Fraktale – eine neue Sprache der Wissenschaft. Teil 1.* Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes der Schweizerischen Elektrizitätswerke, Bd. 81, Nr. 9, S. 43–48, 1990.

[Gipser 90b] Gipser, T.: *Fraktale – eine neue Sprache der Wissenschaft. Teil 2.* Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes der Schweizerischen Elektrizitätswerke, Bd. 81, Nr. 9, S. 49–54, 1990.

[Graf 92] Graf, S.: *Barnsley's scheme for the fractal encoding of images.* J. Comprexity, Bd. 8, Nr. 1, S. 72–78, 1992.

[Harrer 93] Harrer, Thomas: *IFS: Bildcodierung mit einfachen Transformationen.* Hauptseminar: Fraktale und Bildkodierung. Fakultät Informatik, Stuttgart, 1993.

[Horn 91] Horn, Alastair N.: *IFSs and the Interactive Design of Tiling Structures.* In: Crilly, A. J., R. A. Earnshaw und H. Jones, Hrsg.: *Fractals and Chaos*, Kap. 6, S. 119–144. Springer, New York, 1991.

[Hurd et al. 92] Hurd, L. P., M. A. Gustavus und M. F. Barnsley: *Fractal video compression.* In: *Digest Papers IEEE Computing Society International Conference. Presented at 37th annual IEEE International Computer Conference COMPCON Spring '92, San Francisco*, S. 41–42, Piscataway. IEEE Service Center, 1992.

[Hutchinson 81] Hutchinson, J.: *Fractals and self similarity.* Indiana University Journal of Mathematics, Bd. 30, S. 13–747, 1981.

[Jacobs et al. 92] Jacobs, E. W., Y. Fisher und R. D. Boss: *Image compression: A study of the iterated transform method.* Signal Processing, Bd. 29, S. 251–263, 1992.

[Jacquin 89] Jacquin, Arnaud: *A fractal theory of iterated markov operators with applications to digital image coding.* Doktorarbeit, Georgia Institute of Technology, 1989.

[Jacquin 90] Jacquin, A.: *Fractal Image Coding Based on a Theory of Iterated Contractive Image Transformations.* SPIE Visual Communications and Image Processing, Bd. 1360, S. 227–239, 1990.

[Jacquin 92] Jacquin, A.E.: *Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformation.* IEEE Transactions on Image Processing, Bd. 1, Nr. 1, 1992.

[Jacquin 93] Jacquin, A.: *Fractal image coding: a review.* Proceedings of the IEEE, Bd. 81, Nr. 10, S. 1451–1465, 1993.

[Kawamata et al. 92] Kawamata, M., H. Kanbara und T. Higuchi: *Determination of IFS codes using scale-space correlation functions.* In: *Workshop Notes. IEEE International Workshop on Intelligent Signal Processing and Communication Systems*, S. 219–33, Taipei, Taiwan, 1992.

[Kesy 93] Kesy, Oliver: *Bilder schrumpfen. Neue Methoden und Programme zur Bildkompression.* c't, Nr. 11, S. 120–126, 1993.

[Lewis et al. 92] Lewis, A.S. und G. Knowles: *Image Compression using the 2-D Wavelet Transform.* IEEE Transactions on image procession, Bd. 1, Nr. 2, S. 244–250, 1992.

[Meneghini et al. 91] Meneghini, R. und P. Stutz: *How pictures emerge from themselves. Fractal based image compression.* Bulletin des Schweiyerischen Elektrotechnischen Vereins, des Verbandes der Schweizerischen Elektrizitätswerke, Bd. 82, Nr. 21, S. 11–15, 1991.

[Meyer 93] Meyer, Peter: *Fraktale Bildkompression.* Hauptseminar: Fraktale und Bildcodierung. Fakultät Informatik, Stuttgart, 1993.

[Molisch 94] Molisch, Andreas: *Studienarbeit Fraktale Bildkodierung – Entwurf und Realisierung eines Encoders / Decoders zur fraktalen Bildkodierung.* Studienarbeit, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Abt. Praktische Informatik – Bildverstehen, 1994.

[Monro 93a] Monro, D. M.: *Generalized Fractal Transform: Complexity Issues.* In: Storer, James, Hrsg.: *IEEE Data Compression Conference Proceedings '93*, S. 254–261, 1993.

[Monro 93b] Monro, D.M.: *A hybrid fractal transform.* Proceedings ICASSP, Nr. V, S. 169–172, 1993.

[Monro et al. 92a] Monro, D.M. und F. Dudbridge: *Fractal approximation of image blocks.* Proceedings ICASSP, Nr. III, S. 485–488, 1992.

[Monro et al. 92b] Monro, D.M. und F. Dudbridge: *Fractal Block coding of images*. Electronic Letters, Bd. 28, Nr. 11, S. 1053–1055, 1992.

[Neuhauser et al. 92] Neuhauser, Michael A., W. G. Kropatsch und I. J. Leitgeb: *Iterierte Funktionensysteme. Eine Lösung des eindimensionalen inversen Problems*. Technischer Bericht Technische Universität Wien, 1992.

[Novak 92] Novak, Mirek: *Attractor Coding of Images*. Technischer Bericht Image Coding Group, Department of Electrical Engineering, 1992.

[Øien et al. 91] Øien, G.E., S. Lepsøy und T. Ramstad: *An inner product space approach to image coding by contractive transformations*. Proceedings ICASSP, S. 2773–2776, 1991.

[Øien et al. 94] Øien, G.E., Z. Baharav, S. Lepsøy, E. Karnin und D. Malah: *A new improved collage theorem with applications to multiresolution fractal image coding*. Elektronisch Publiziert, 1994.

[Patel et al. 199] Patel, Ketan, B. C. Smith und L. A. Rowe: *Performance of a Software MPEG Video Decoder*. Technischer Bericht Computer Science Division-EECS, University of California, ca. 1993.

[Peitgen et al. 92] Peitgen, Heinz-Otto, H. Jürgens und D. Saupe: *Bausteine des Chaos. Fraktale*. Springer, Cotta, New York, Berlin, Heidelberg, Stuttgart, 1992.

[Rabbani et al. 91] Rabbani, Majid und P. W. Jones: *Digital Image Compression Techniques*, Bd. TT7 d. Reihe *Tutorial Texts in Optical Engineering*. SPIE Optical Engineering Press, 1991.

[Ramamurthi et al. 86] Ramamurthi, B. und A. Gersho: *Classified vector quantisation of images*. IEEE Transactions on Communication, Bd. 34, 1986.

[Reddaway et al. 88] Reddaway, S.F., A. Wilson und A. Horn: *Fractal graphics and image compression on a SIMD processor*. In: *Proceedings. The 2nd Symposium on the Frontiers of Massively Parallel Computation*, S. 265–274, Fairfax, VA, USA. IEEE Comput. Soc. Press, 1988.

[Saupe 94] Saupe, Dietmar: *Breaking the Time Complexity of Fractal Image Compression*. Technischer Bericht Institut für Informatik, Universität Freiburg, Rheinstr. 10–12, 79104 Freiburg, 1994.

[Saupe et al. 94] Saupe, Dietmar und R. Hamazaoui: *A Guided Tour of the Fractal Image Compression Literature*. Elektronisch publiziert, 1994.

[Sedgewick 92] Sedgewick, Robert: *Algorithmen*. Addison-Wesley, 1992.

[Vences et al. 94] Vences, Lucia und I. Rudomin: *Fractal Compression of Single Images and Image Sequences using Genetic Algorithms*, 1994.

[Vrscay 91] Vrscay, E.R.: *Iterated Function Systems: Theory, Applications and the Inverse Problem.* In: Belair, J. und S. Dubuc, Hrsg.: *Fractal Geometry and Analysis*, S. 405–468. Kluwer Academic, 1991.

[Zaccarin et al. 93] Zaccarin, André und B. Liu: *A Novel Approach for Coding Color Quantized Images.* IEEE Transactions on Image Progession, Bd. 2, Nr. 4, S. 442–447, 1993.

[Zhao et al. 94] Zhao, Y. und B. Yuan: *Image compression using fractals and discrete cosine transform.* Electronic Letters, Bd. 30, Nr. 6, S. 474–475, 1994.

[Zorpette 88] Zorpette, G.: *Fractals: not just another pretty picture.* IEEE Spectrum, Bd. 25, Nr. 10, S. 29–31, 1988.

Weitere Literatur

[Barnsley et al. 90] Barnsley, M. F. und A. D. Sloan: *Methods and apparatus for image compression by iterated function systems.* United States Patent No. 4 941 193, 1990.

[Barnsley et al. 91] Barnsley, M. F. und A. D. Sloan: *Methods and apparatus for processing digital data.* United States Patent No. 5 065 447, 1991.

[Monro et al. 93] Monro, D.M., D. Wilson und J. Nicholls: *High speed image coding with the Bath Fractal Transform.* In: *IEEE International Symposium on Multimedia Technologies*, Southampton, 1993.

[Monro et al. 94] Monro, Donald und J. Nicholls: *Real Time Fractal Video for Personal Communications.* Fractals – An Interdisciplinary Journal On The Complex Geometry of Nature, Bd. 2, Nr. 3, S. 391–394, 1994.

[Wilson et al. 94] Wilson, David, J. Nicholls und D. Monro: *Rate Buffered Fractal Video*, 1994.