4 Implementierung

Basierend auf den Ausführungen des vorangegangenen Kapitels beschreibt dieses Kapitel verschiedene Varianten einer Implementierung und die damit erzielbaren Ergebnisse. Dieses Kapitel ist eine direkte Anwendung des diskreten Modells zur fraktalen Bildkodierung aus Kapitel 3 und stellt unterschiedliche Möglichkeiten für konkrete Implementierungen vor.

Zu Beginn des Kapitels befindet sich eine Diskussion über den Aufbau eines Codec's zur fraktalen Bildkodierung. Diese Diskussion stellt die Grundlage für eine Einordnung der nachfolgenden Implementierungsvarianten in ein Gesamtsystem zur fraktalen Bildkodierung dar und ermöglicht so eine genaue Beurteilung der Verfahren. Aus diesem Grund werden im Anschluß an diese Diskussion verschiedene Verfahren in ihrer konkreten Implementierung beschrieben sowie deren Einsatz motiviert und diskutiert.

Das folgende Kapitel 4 hängt noch nicht von einer speziellen Implementierung ab und die Aussagen beziehen sich auf die gesamte Klasse von Systemen zur fraktalen Bildkodierung, die sich mit dem Modell aus Kapitel 3 beschreiben und charakterisieren lassen. Ein Vergleich der Performance verschiedener Verfahren sowie eine Diskussion ihrer Ergebnisse wird im abschließenden Kapitel 5 gegeben.

Den Abschluß des Kapitels bildet eine Diskussion über eine Leistungsbewertung von Kompressionsverfahren durch die Angabe der Bildqualität in Abhängigkeit von der Kompressionsrate, wobei im speziellen auch die Problematik eines Maßes für die Bildqualität im Kontext der fraktalen Bildkodierung diskutiert wird.

1. Codec zur fraktalen Bildkodierung: Kodierer

Der Aufbau des Codec's ist eine Folgerung der Modellbildung aus Kapitel 2 und 3. Teile des allgemeinen Aufbaus, wie z.B. die Partitionierung, ergeben sich mit Sicherheit schon aus dem Modell von Barnsley, während sich die technischen Details, wie z.B. die Wahl der Transformationen, eher aus dem diskreten Modell ergeben.

Anhand der Struktur des Codec's sind dann schon qualitative Aussagen über die Leistungsfähigkeit von Verfahren zur fraktalen Bildkodierung möglich. Allein aufgrund der Struktur eines Codec's sind natürlich noch keine Aussagen in Bezug auf mögliche Kompressionsraten oder Bildqualitäten möglich. Dafür lassen sich allerdings Aussagen über den Leistungsumfang des Systems, wie z.B. Möglichkeiten der hierarchischen Codierung, schon treffen und beschreiben.

Bei der Entwicklung des Codec's auf der Basis des diskreten Modells finden sich natürlich dessen Bestandteile direkt im Codec wieder oder werden zu größeren Funktionseinheiten zusammengefaßt. So führt dann die Übertragung des diskreten Modells in einen Codec zu dem folgenden Kodierer:

Abb. 1: *Kodierer*

Als Eingabe für den Kodierer dient ein digitales Bild. Die Frage nach dem Format des Bildes, RGB oder YUV, ist für die Funktion des Kodierers nicht entscheidend, da keine besonderen Voraussetzungen an die Farb- oder Helligkeitsdarstellung gestellt werden. Bei der Farbdarstellung besitzt man somit die Wahl zwischen einem RGB- und einem YUV-Format mit den schon in Kapitel 1 (Siehe *Farbraum-Transformation* auf Seite 9) besprochenen Vor- und Nachteilen.

Entsprechend den Modellen aus Kapitel 2 und 3 muß das Eingabebild partitioniert werden. In der Art der Partitionierung ist man frei, solange man die Forderung einhält, daß die Partitionierung vollständig und disjunkt ist. Die in der Abbildung *Kodierer* auf Seite 80 gezeigte uniforme Partitionierung ist dabei nur ein erster Ansatz.

Adaptive Partitionierungen sind mit der eigentlichen Kodierung gekoppelt und versuchen die Anzahl der Teilmengen und somit auch die Anzahl der benötigten Transformationen möglichst klein zu halten, um eine möglichst kompakte Kodierung zu erreichen. Dieser Versuch, die Anzahl der benötigten Transformationen möglichst gering zu halten, führt zu der Rückkopplung zwischen der Partitionierung und Kodierung, da in Abhängigkeit vom Kodierungsfehler ε versucht wird, möglichst große Teilmengen mit einer Transformation zu kodieren.

Nach der Kodierung des Bildes durch kontrahierende Abbildungen werden die Daten einer Entropie-Kodierung unterzogen. Die Wahl der Entropie-Kodierung unterliegt dann vollständig dem Gesichtspunkt einer möglichst effektiven verlustlosen Komprimierung und wird nicht von den vorangegangenen Komponenten des Kodierers beeinflußt, so daß man die Wahl zwischen den gängigsten Verfahren, wie z.B. Huffman, einer LZ-Variante oder einer arithmetischen Kodierung besitzt.

Wie man dieser Diskussion der Komponenten des Kodierers entnehmen kann, sind im Kontext dieser Arbeit vor allem die schwarz eingerahmten Komponenten des Kodierers von Interesse und sollen deshalb auch im folgenden behandelt werden. Zuerst stellt sich allerdings noch die Frage, welche Eigenschaften sich direkt aus der Struktur des Kodierers ableiten lassen.

Progressives Senden

Der Begriff *progressives Senden* beschreibt die Eigenschaft einer Bildkodierung, mit der ein additives Versenden von Bilddaten die Auflösung des Bildes beim Empfänger erhöht. Für eine erhöhte Auflösung Q_2 beim Empfänger bedarf es also keines wiederholten Versendens der gesamten Bilddaten, sondern es wird nur die Differenz zur geringeren Auflösung ($Q_2 - Q_1$) zusätzlich übertragen.

Abb. 2: *Progressives Senden*

Diese Eigenschaft einer Kodierung ist nicht mit einer *interlaced Kodierung* zu verwechseln, bei welcher die Auflösung konstant bleibt und beim Empfang nur ein Zeilensprungverfahren eingesetzt wird. Für eine Erhöhung der Bildauflösung ist bei dieser Kodierung ein erneutes Versenden der gesamten Bilddaten notwendig. Eine *interlaced Kodierung* stellt keine besonderen Anforderungen an den Kodierer bzw. Dekodierer, im Gegensatz zu progressiven Senden, welches eine Ordnung der Bilddaten voraussetzt. Bilddaten für geringere Auflösungen müssen vor den Bilddaten für die höheren Auflösungen kodiert und versendet werden.

Diese Ordnung steht im Widerspruch zu obigem Kodierer, so daß mit Hilfe dieses Codec's zur fraktalen Bildkodierung kein progressives Senden möglich ist. Der Widerspruch liegt in der vollständig disjunkten Patitionierung des Eingabebildes und an der Rückkopplung zwischen Partitionierung und Kodierung. Die Partitionierung bestimmt die Teilmengen der Bilddaten, die zusammenhängend in der vorliegenden Auflösung kodiert werden. Da die Partitionierung vollständig ist, werden nacheinander alle Teilmengen einer Auflösung kodiert und so eine Kodierung des gesamten Bildes vorgenommen. Eine Erhöhung der Auflösung bedeutet eine neue Kodierung aller Teilmengen in der neuen Auflösung und somit auch des gesamten Bildes.

Hierarchische Kodierung

Eine weitere Art der Kodierung, die für Anwendungen in einer heterogenen Netzwerkumgebung mit Wiedergabegeräten unterschiedlicher Auflösung von sehr großem Interesse und auch mit dem progressiven Senden sehr eng verbunden ist, stellt die *hierarchische Kodierung* dar. Das Ziel der hierarchischen Kodierung ist ähnlich dem Ziel des progressiven Sendens, nämlich die Bereitstellung eines Bildes in verschiedenen Auflösungen, angepaßt an das Wiedergabegerät. Bei beiden Verfahren geschieht dies unter der Nebenbedingung der möglichst kompakten Übertragung bzw. Speicherung der Bilddaten. Doch während beim progressiven Senden der Aspekt des additiven Sendens und damit eine effektive Nutzung der zur Verfügung stehenden Bandbreite im Vordergrund steht, ist bei einer hierarchischen Kodierung der Aspekt der Darstellung eines Bildes in verschiedenen Auflösungen der Hauptaspekt.

Diese Unterscheidung zwischen diesen beiden Verfahren wird vor allem in der Literatur zu Signalverarbeitung nicht vorgenommen, da bei den meisten Verfahren aus der Signalverarbeitung diese beiden Eigenschaften einer Kodierung zusammenfallen. Bei der fraktalen Bildkodierung ist dies nicht der Fall, wie im folgenden gezeigt wird, weshalb in dieser Arbeit zwischen den beiden Begriffen unterschieden wird.

Der Begriff “hierarchische Kodierung” läßt im Kontext der fraktalen Bildkodierung verschiedene Ansätze und Interpretationen zu. Ein erster Ansatz ist sicherlich der Einsatz der in Kapitel 1 auf Seite 24 beschriebenen Pyramiden Schemata. So läßt sich die Laplace-Pyramide direkt auf die fraktale Kodierung anwenden und es können Pyramiden mit beliebig vielen Stufen konstruiert werden.

Abb. 3: *Laplace-Pyramide für fraktale Bildkodierung*

Diese Art der hierarchischen Kodierung besitzt somit dieselben Eigenschaften wie alle Kodierungsschemata, die nach dem Prinzip der Laplace-Pyramide arbeiten. Der Kodierer benötigt für jede Auflösung, die versendet oder gespeichert werden soll, einen Puffer. Der zusätzliche Speicherbedarf des Dekodierers ist ein zusätzlicher Bildpuffer, da sich die Dekodierung einer Auflösung immer auf die nächste kleinere Auflösung des Bildes bezieht. Zusätzlich zum erhöhten Speicherbedarf sind der zusätzliche Rechenaufwand

beim Kodieren und auch beim Dekodieren als Nachteil zu nennen. Vor allem bei einem Zugriff des Dekodierers auf die höchste Auflösungsstufe ist der Rechenaufwand wesentlich höher als bei einer nicht hierarchischen Kodierung.

Das Ergebnis der Kodierung gemäß der Laplace-Pyramide ist dann eine Folge von fraktal kodierten Bildern mit aufsteigenden Auflösungen, wobei ein Bild mit höherer Auflösung mit Hilfe der nächst geringeren Auflösung kodiert wird. Die Kodierung liefert somit eine Hierarchie von Auflösungen, wobei die Auflösungsstufen bei der Kodierung festgelegt werden.

Ein zweiter Ansatz wird in [Baharav93] vorgestellt und orientiert sich stark an Pyramiden-Schemata, wie sie in der Signalverarbeitung [Burt83] verwendet werden. Der Ansatz beschreibt eine Methode zur Berechnung von Fixpunkten für Bilder mit einer bestimmten Auflösung aus den Fixpunkten für Bilder mit einer verschiedenen Auflösung. Beim Dekodieren entsteht eine Pyramide von Fixpunkten bzw. Bildern mit verschiedenen Auflösungen, die sich auseinander berechnen, so daß man damit eine hierarchische Kodierung mit Bildern verschiedener Auflösungen besitzt. Dieser Ansatz ermöglicht durch seine Konstruktion auch gleichzeitig wieder das progressive Senden.

Die dritte Art, fraktale Bildkodierung als eine hierarchische Kodierung zu interpretieren, ist eng mit der Konstruktion des Dekodierers gekoppelt, weshalb diese Interpretation in den folgenden Abschnitt über den Dekoder integriert ist.

2. Codec zur fraktalen Bildkodierung: Dekodierer

Eine Betrachtung der Modelle aus Kapitel 2 und 3 legt die Erwartung nahe, daß Codec's zur fraktalen Bildkodierung stark asymmetrisch aufgebaut sind. Während bei der Kodierung aufwendige Partitionierungs- und Suchverfahren angewendet werden, stellt sich die Dekodierung als einfache iterierte Anwendung der bei der Kodierung gefundenen Transformationen dar. Diese Asymmetrie in der Komplexität und damit auch im Rechenaufwand spiegelt sich dann auch in der einfachen Struktur des Dekodierers wieder:

Abb. 4: *Dekodierer*

Der Dekodierer nimmt als Eingabe die Entropie-kodierten Daten und dekodiert diese so, daß in den nächsten Verarbeitungsschritten die Generierung des Ausgabebildes aus den gespeicherten Transformationen vorgenommen werden kann. Bei der Erzeugung des Ausgabebildes besitzt der Dekodierer nun zwei Rückkopplungen, die beide die Qualität des Ausgabebildes beeinflussen.

Die erste Rückkopplung dient der Anpassung der Auflösung des Ausgabebildes an die Auflösung des Ausgabegerätes. Durch die Kodierung des Eingabebildes mit Hilfe von Abbildungen und deren Fixpunkte besitzt das kodierte Bild keine feste Auflösung mehr. Die Auflösung des Ausgabebildes muß deshalb zu Beginn des Dekodierung festgelegt werden, so daß man im Prinzip in der Lage ist, jede gewünschte Auflösung zu produzieren. Die Produktion von Auflösungen, die wesentlich größer sind als die des Eingabebildes beim Kodieren, macht allerdings nur eingeschränkt Sinn, da der Dekodierer dann Pixel erzeugen muß, die im Eingabebild und damit bei der Bestimmung der Transformationen nicht vorhanden sind bzw. waren.

Ein ähnliches Problem besitzt man natürlich auch bei der Generierung von Ausgabebildern mit einer geringeren Auflösung als die des Eingabebildes. Der hier entstehende Fehler beruht allerdings eher auf einer Art “Downsampling”, da die Kodierung mit einer höheren Auflösung bzw. einer größeren Genauigkeit vorgenommen wurde, als für die Ausgabe notwendig ist. Während bei der Dekodierung von Auflösungen, die größer als die kodierte Auflösung sind, durch die Anwendung der Transformationen Pixel “erfunden” werden, kommt es bei kleineren Auflösungen zu einer Nichtberücksichtigung von Pixelwerten, die berechnet werden können.

Geht man nun von einer kodierten Auflösung aus und möchte diese auf Ausgabegeräten mit verschiedenen kleineren Auflösungen darstellen, so benötigt man eine Datei mit den Transformationen des kodierten Bildes. Aus dieser Datei lassen sich alle anderen Auflösungen generieren. In diesem Sinne ist eine fraktale Bildkodierung auch eine hierarchische Bildkodierung. Im Gegensatz zu den Ansätzen aus der Signalverarbeitung fehlt der fraktalen Bildkodierung allerdings die Eigenschaft des progressiven Sendens, da für die Generierung einer Auflösung immer alle Transformationen vorhanden sein müssen, was immer die Übertragung der ganzen Bilddaten bedeutet. So gesehen ist diese Art einer hierarchischen Bildkodierung sicherlich die schwächste Interpretation, dafür ist allerdings kein extra Aufwand zur Berechnung nötig, da diese Art der hierarchischen Kodierung schon durch die fraktale Bildkodierung gegeben ist.

3. Partitionierungsverfahren

Die beiden folgenden Unterkapitel über Partitionierungsverfahren und über die eingesetzten Transformationen sind eng miteinander verbunden. So ist die Notwendigkeit, ein

tung des Bildes als Element des $\mathbb{R}^M$ bedingt noch eine Umordnung der Bildelemente, führt aber zu einer eindeutigen und umkehrbaren Darstellung und ermöglicht eine einfachere und übersichtlichere Schreibweise. Bildbereiche R mit M_R Bildelementen sind so durch eine zugehörige Indexmenge $I = \{i_1, i_2, \ldots, i_{M_R}\} \subset \{1, 2, \ldots, M\}$ eindeutig bestimmt. Mit dieser Schreibweise kann man Bildbereiche immer noch als 2-dimensionale Bereiche eine Bildes betrachten, gleichzeitig erleichtert sie die Formulierung der nachfolgenden Verfahren. Für den hier betrachteten Fall eines Graustufenbildes bedeutet dies, daß $f \in \mathbb{R}^3$ ist, da neben den beiden räumlichen Dimensionen mit den Koordinaten (x, y) noch der Graustufenwert als dritte Dimension mit der Koordinate $z = f(x, y)$ hinzukommt.

Das Ziel der Kodierung ist die Bestimmung einer Menge von Abbildungen $T_1, T_2, \ldots, T_N$ mit $T_i : D_i \to R_i$ für $i = 1, 2, \ldots, N$ und die Festlegung der Definitions- und Wertebereiche D_i und R_i.

Abb. 5: *Abbildung* $T_i : D_i \to R_i$

Die in der Literatur oft vorkommende Bezeichnung R_i (R = *range* = Wertebereich) für die Definitionsbereiche, welche oft in Anlehnung an die Bezeichnung von Barnsley gewählt wird, wird hier nicht gewählt, sondern es wird die unter dem Gesichtspunkt der Kodierung richtige Bezeichnung verwendet. Barnsley hat die Vergabe der Bezeichnungen vermutlich unter dem Gesichtspunkt der Dekodierung gewählt, was allerdings die Beschreibung der Kodierungsverfahren unverständlicher macht.

Für eine vollständige Kodierung des Eingabebildes muß die Vereinigung der Wertebereiche $R = \bigcup_{i=1}^{N} R_i$ das Bild vollständig überdecken. An die Definitionsbereiche D_i muß eine solche Bedingung nicht gestellt werden. Die Definitionsbereiche können beliebige Teilmengen des zu kodierenden Bildes sein. Mit Hilfe dieser Mengen soll eine Abbildung $T = \bigcup_{i=1}^{N} T_i$ des Bildes auf sich selbst definiert werden, so daß der Abstand $d(f \cap (R_i \times \square), T_i(f))$ für $i = 1, 2, \ldots, N$ minimal ist. Das Ziel ist also eine Minimierung der Abstände zwischen dem Eingabebild und den Abbildern von D_i unter T_i über den entsprechenden Wertebereichen R_i:

Abb. 6: *Minimierung von* $d(f \cap (R_i \times \square), T_i(f))$

Gesucht werden somit Definitionsbereiche D_i und Abbildungen T_i, so daß die Anwendung von T_i auf den Bildbereich über D_i eine Menge ergibt, die dem Bildbereich über R_i sehr nahe ist. Geschieht dies für alle Wertebereiche R_i, so erhält man eine Überdekkung oder Collage $f' = T(f)$ des Eingabebildes f.

Die Collage f' ist somit ein Bild, welches unter der Anwendung von T einen möglichst geringen Abstand vom Eingabebild f besitzen soll:

$$d(f, f') < \varepsilon \text{ mit } \varepsilon > 0 \text{ möglichst klein}$$

Gemäß dem Collage Theorem impliziert dies einen Attraktor f_T von T, der das Eingabebild f ebenfalls gut approximiert:

$$d(f, f_T) < \frac{1}{1-s} \cdot \varepsilon$$

mit s als Lippschitz-Faktor von T und T strikt kontrahierend

Die Idee der fraktalen Bildkodierung liegt also in der Berechnung der Abbildung T mit Hilfe einer Collage f'. Gemäß dem Collage Theorem besitzt diese Abbildung T dann auch einen Fixpunkt f_T, der das Bild f approximiert, so daß

$$f \approx f' = T(f') \approx T(f)$$

gilt. Der erste Schritt zur Berechnung von T und der Collage f' ist somit die Festlegung der Definitions- und Wertebereiche für die partiellen Abbildungen T_i, also die Patitionierung des zu kodierenden Bildes.

Die Partitionierung des Eingabebildes ist prinzipiell unabhängig von den verwendeten Abbildungen und auch die Wahl der Abbildungen ist prinzipiell nur durch die Anforderungen aus Kapitel 2 und 3 beschränkt. Unter dem Aspekt der Anwendung der

Verfahren ändert sich diese Situation jedoch stark. Wie in Kapitel 3 schon angedeutet, führen die Forderung nach Berechenbarkeit und die Notwendigkeit zur Begrenzung des Rechenaufwandes zur Wahl von affinen Transformationen für die Abbildungen T_i.

Das Ziel einer möglichst kompakten Speicherung des Eingabebildes führt bei einer Implementierung zu der Nebenbedingung, daß man die Anzahl der Transformationen für die Beschreibung des Eingabebildes f möglichst gering halten möchte. Damit ist auch die Entwicklung von adaptiven Partitionierungsverfahren motiviert, deren Ziel es ist, möglichst große gleichartige Bildbereiche mit einer Transformation zu kodieren und damit weniger Transformationen für die Kodierung des Eingabebildes zu benötigen als die starren uniformen Partitionierungsverfahren.

Uniforme Partitionierung

Besteht eine Partitionierung aus sich nicht überlappenden gleichgroßen Teilmengen des Eingabebildes, so besitzt man eine uniforme Partitionierung, wie die folgende Abbildung am Beispiel einer uniformen Partitionierung mit Hilfe von Quadraten der Größe M_r^2 zeigt.

Abb. 7: *Uniforme Partitionierung*

Für ein Eingabebild der Größe $M \times M$ bedeutet dies in der Vektorrepräsentation eine Partitionierung der Indizes $1, 2, \ldots, M^2$ in $N = M^2 / M_r^2$ gleich große Teilmengen $\{J_n\}_{n=1}^{N} = J$, so daß eine vollständige uniforme disjunkte Partitionierung des Eingabebildes entsteht. Diese Teilmengen sind die *range*-Partitionierung oder Wertebereichs-Partitionierung des fraktalen Modells. Für ein bestimmtes Ausgangsmodell zur Partitionierung wird auch eine Menge von *domain*-Blöcken oder Definitionsbereichen gewählt, die sogenannte *library* $I = \{I_1, I_2, \ldots, I_Q\}$. Die Menge I enthält all jene Mengen, die als Definitionsbereich für ein T_i in Frage kommen können.

Gemäß Kapitel 3 sind die partiellen Abbildungen dann definiert als $T_n(x) = P_{J_n}(K_n(D(F_{I_{q(n)}}(x))))$ und deren Aufsummierung ergibt eine Abbildung des gesamten Eingabebildes:

$$T(x) = \sum_{n=1}^{N} T_n(x)$$

Mit einer Wertebereichs-Partitionierung und einer Menge von Definitionsbereichen läßt sich der Algorithmus zur fraktalen Bildkodierung nun folgendermaßen formulieren:

Algorithmus zur fraktalen Bildkodierung

Für jeden Indexblock $J_n \in J$ des Wertebereichs:
Wähle einen Indexblock $I_q \in I$ des Definitionsbereichs und eine *kernel*-Transformation $K_n \in K$, so daß $d(x_{J_n}, K_n(D(F_{I_{q(n)}}(x))))$ minimal ist.

Aufgrund der Darstellung des Eingabebildes gemäß dem Modell aus Kapitel 3 ist diese Formulierung des Algorithmus vollkommen unabhängig von der Struktur der Partitionierung und damit vom Aufbau der Mengen in I und J. Die Formulierung des Algorithmus und große Teile der Implementierung sind vollkommen unabhängig davon, ob die uniforme Partitionierung mit Hilfe von Quadraten, Dreiecken oder anderen geometrischen Formen durchgeführt wird.

In der Anwendung stellt sich der Algorithmus dann so dar, daß die Mengen I und J durch die Partitionierung gegeben sind und die *kernel*-Transformationen berechnet werden müssen. Die Operatoren D und $F_{I_{q(n)}}$ sind gemäß Kapitel 3 eher technischer Natur und spielen bei der Berechnung des Minimums nur eine untergeordnete bzw. keine Rolle. Die Auswirkungen dieser Operatoren auf die Kodierung wird bei der noch folgenden Diskussion der Ergebnisse behandelt.

Berechnung der Transformationen

Außer den allgemeinen Forderungen aus den vorangegangenen Kapiteln wurden sowohl die Menge K der kernel-Transformationen als auch einzelne Transformationen $K_n \in K$ nicht weiter spezifiziert, was für eine Implementierung natürlich nicht ausreicht. Aus Gründen der Berechenbarkeit wird in diesem Fall die Menge K immer als die Menge der affinen Transformationen angenommen, die den Voraussetzungen aus Kapitel 3 genügen. Für die uniforme Partitionierung und den Fall affiner Transformationen lassen sich die Abbildungen T_i dann folgendermaßen berechnen.

Zur Kennzeichnung, daß die Abbildungen T_i in diesem Fall affine Transformationen sind, werden sie mit ω_i bezeichnet. Gegeben sei also ein Graustufenbild mit 8 Bit/Pixel und einer Auflösung von $M \times M$ Pixeln. Für eine Implementierung ist eine Darstellung des Bildes in Form von 2 dimensionalen Pixelkoordinaten die zugänglichste, zumal eine Überführung der Darstellung aus und in eine Vektordarstellung gemäß Unterkapitel 1. auf Seite 62 jederzeit möglich ist. Diese Darstellung kommt auch dem Einsatz des euklidischen Abstandes für d zugute. Der euklidische Abstand wird einerseits deshalb gewählt, weil er sich einfach und schnell berechnen läßt und er andererseits gute Ergebnisse liefert.

$R_1, R_2, \ldots, R_N$ seien sich nicht überlappende quadratische Pixelblöcke der Größe $M_R \times M_R$ mit $M \equiv 0 \bmod M_R$. Die Forderung, daß M_R die Bildauflösung M teilt, gewährleistet, daß das Eingabebild sich ohne Rand aufteilen läßt. Bei einem Verzicht auf diese Forderung, was bei einem vorgegebenen Eingabebild und einer vorgegebenen Partitionierung durchaus vorkommt, muß das Bild gegebenenfalls als periodisch angesehen oder mit Füllpixeln aufgefüllt werden.

$D_1, D_2, \ldots, D_Q$ seien sich überlappende Pixelblöcke der Größe $M_D \times M_D$, wobei M_D ein ganzzahliges Vielfaches von M_R ist: $M_D = k \cdot M_R$ mit $k > 1$ beliebig aber fest. Diese Forderung an M_D erleichtert die Konstruktion und Implementierung des *decimation*-Operator D. Die Menge I besitzt somit $(M - M_D + 1)^2$ Elemente.

Für jedes R_i wird nun jenes D_i aus der Menge I gesucht, welches den Abstand $d(f \cap (R_i \times \square), \omega_i(f))$ minimiert. Die Durchführung dieser Suche ist sehr aufwendig. Durch die Verwendung von affinen Transformationen für die Abbildung besitzt man 8 Möglichkeiten, die beiden Quadrate aufeinander abzubilden, so daß bei einer erschöpfenden Suche $8(M - M_D + 1)^2$ Vergleiche für jedes der M_R R_i durchgeführt werden müssen. Insgesamt werden somit $8M_R(M - M_D + 1)^2$ Vergleiche durchgeführt. Schon bei einer Auflösung von 256×256 Pixeln für das Eingabebild, 8×8 Pixeln für die R_i und 16×16 Pixeln für die D_i bedeutet dies die Durchführung von 464.648 Vergleichen von zwei Quadraten für jedes der 1024 R_i, was insgesamt zur Durchführung von 475.799.552 Vergleichen führt.

Diese Suche beinhaltet somit die Durchführung der folgenden Operationen:

1. Einsatz des *decimation*-Operators:

Da $M_D = k \cdot M_R$ mit $k > 1$ ist, verringere die Auflösung der M_D so, daß sie die gleiche Auflösung wie die M_R besitzen.

2. Berechnung des euklidischen Abstandes:

Ziel ist die Bestimmung jenes D_i aus der Menge I, welches den geringsten euklidischen Abstand zu R_i besitzt.

3. Bestimmung der Abbildung ω_i:

Die Wahl eines D_i legt den räumlichen Anteil von ω_i schon fest, die Berechnung der Parameter für Helligkeit und Kontrast steht allerdings noch aus.

Mit den zwei räumlichen Dimensionen und den Grauwerten als dritte Dimension besitzen die affinen Transformationen ω_i die folgende Gestalt:

$$(1)\quad \omega_i\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

Der Parameter bestimmt den Kontrast und die Helligkeit des transformierten Wertes. Der im zweiten Schritt bestimmte räumliche Anteil von hat die Gestalt:

Nach der Durchführung des ersten Schrittes besitzt man zwei gleichgroße quadratische Pixelblöcke, deren euklidischer Abstand berechnet werden muß. Die beiden Pixelblöcke bestehen aus jeweils Pixel:

formation ω_i für den aktuellen Wertebereich R_i gemäß Formel (1) bestimmt. Die Anwendung für alle Wertebereiche R_i aus der Menge J bestimmt für alle diese Wertebereiche je ein ω_i, so daß das Eingabebild vollständig durch diese ω_i bestimmt werden kann.

Kodierungsschema

Auf der Basis einer uniformen Partitionierung läßt sich ein Kodierungsschema zur fraktalen Bildkodierung aufstellen, dessen Transformationsanzahl fest ist, da die Anzahl der R_i fest vorgegeben ist. Die Anzahl der D_i wird durch die Wahl der Menge I bestimmt und im folgenden Algorithmus mit Q bezeichnet.

```
divide(□) /* Partitioniere das vollst. Eingabebild */;

/* Für alle R_i führe aus... */
for (i = 1 to N) do begin

   /* Für aktuelles R_i durchsuche alle D_i ∈ I... */
   for (j = 1 to Q) do

      /* Bestimme D_i ∈ I mit minimalem Abstand. */
      if ( d(f ∩ (R_i × □), ω_i(f)) < d(f ∩ (R_i × □), Minimum_D(R_i)) )
         then begin
            Minimum_D(R_i) := D_i;
            /* Berechne ω_i aus R_i und D_i. */
            calculate_ω(R_i);
         end;

   /* Speichere ω_i für R_i. */
   save_ω(R_i);

end;
```

Abb. 8: *Kodierungsschema für uniforme Partitionierung*

Die Anzahl der Transformationen für die Kodierung eines Eingabebildes hängt ausschließlich von der Größe M_R der R_i ab. Die Kodierung des vollständigen Bildes erfolgt unabhängig von verschiedenen Bildbereichen, da M_R fest ist. Lassen sich Teile des Bildes mit der gewählten Größe der Partitionierung schlecht kodieren, so kann eine Verbesserung der Qualität der Kodierung nur durch eine vollständige neue Kodierung des Eingabebildes mit einem kleineren M_R erreicht werden. Eine kleineres M_R bedeutet allerdings eine größere Anzahl von ω_i für die Kodierung, was eine schlechtere Kompressionsrate zur Folge hat.

Wählt man M_R nicht mehr fest, sondern führt eine variable Anpassung der Größe an den Inhalt von einzelnen Bildbereichen durch, so kommt man zu adaptiven Partitionierungsverfahren, wie sie im folgenden noch diskutiert werden.

Die Einführung adaptiver Partitionierungsverfahren führt zu einem leicht veränderten Kodierungsschema, allerdings mit derselben Komplexität. Die Komplexität der Kodierung läßt sich aufgrund der beiden `for`-Schleifen und der Größe von N und Q aus obigen Schema ablesen und verhält sich insgesamt quadratisch zur Größe des Eingabebildes.

Weitere uniforme Partitionierungsverfahren

Ausgehend von dem diskreten Modell aus Kapitel 3 bedeutet die Einführung eines neuen Partitionierungsverfahren die Verwendung eines anderen Indexblocks und eine Anpassung von σ, d.h. es wird eine andere Nummerierung benötigt. Da die Struktur der Signalblöcke in die Modellbildung nicht eingeht, sind die durch einen Wechsel des Partitionierungsverfahren bedingten Änderungen des Kodierungsschemas für uniforme Partitionierungsverfahren (s. Seite 91) ausschließlich auf die implementations-spezifischen Operatoren beschränkt. Die Implementierung eines uniformen *triangular*-Partitionierungsverfahrens bedeutet daher eine Anpassung der bisher verwendeten *fetch*-, *decimation*- und *put*-Operatoren für Quadrate an die neue dreieckige Struktur der Werte- und Definitionsbereiche.

Das Verfahren zur triangulären Partitionierung eines Bildes beginnt mit der Zerlegung des Eingabebildes in zwei Dreiecke. Jedes dieser Dreiecke wird anschließend durch die Verbindung der Seitenhalbierenden in vier weitere Dreiecke unterteilt. Führt man dieses Verfahren rekursiv fort, so wird das Eingabebild rekursiv in gleichgroße Dreiecke zerlegt, so daß die Vereinigung der Dreiecke das Eingabebild vollständig überdeckt. Die Integration des Verfahrens in das Kodierungsschema für uniforme Partitionierungsverfahren (s. Seite 91) wird in der Prozedur `divide` vorgenommen.

Abb. 9: *Prinzip der triangular-Partitionierung*

Der Vergleich einer *triangular*-Partitionierung und einer uniformen Partitionierung mit Hilfe von Quadraten zeigt Gemeinsamkeiten auf. Eine uniforme Partitionierung mit Hilfe von Quadraten, deren Seitenlängen sich rekursiv durch Halbierung der Seitenlängen des nächst größeren Quadrates ergeben, führt in Bezug auf die Anzahl der Wertebereiche R_i zu einem ähnlichen Partitionierungsschema mit entsprechenden Kompressionsraten wie eine *triangular*-Partitionierung.

Der Hauptvorteil der *triangular*-Partitionierung liegt in der Eigenschaft, daß die Partitionsgrenzen und somit auch die damit verbundenen Artefakte nicht mehr ausschließlich horizontal und vertikal verlaufen. Diese Tatsache erhöht die subjektive Qualität des

kodierten Bildes, da das HVS horizontale und vertikale Artefakte vergleichsweise gut wahrnimmt, während schräge und unregelmäßige Artefakte für einen Betrachter als weniger störend empfunden werden [Kunt85] [Rama86] .

Die Konstruktion der *triangular*-Partitionierung zeigt desweiteren auch Erweiterungsmöglichkeiten für uniforme Partitionierungen mit Hilfe von Quadraten auf. Überträgt man die Idee der rekursiven Unterteilung auf Quadrate und gestaltet diese Unterteilung auch noch adaptiv, so erhält man eine Quadtree-Partitionierung, wie sie im folgenden Unterkapitel behandelt wird.

Die Verbesserung der subjektiven Qualität läßt sich natürlich auch durch die Verwendung anderer geometrischer Formen erhöhen. Allerdings stößt man bei der Auswahl der Formen sehr schnell auf ein Problem. Dieses Problem bei der Festlegung auf eine einfache geometrische Form zur vollständigen Überdeckung des Eingabebild läßt sich am Beispiel der hexagonalen Partitionierung verdeutlichen.

Unter dem Aspekt, möglichst viele Möglichkeiten zur Abbildung der verwendeten geometrischen Form mit Hilfe affiner Transformationen auf sich selbst zu besitzen, ist ein Hexagon einem Quadrat überlegen. Bei der Verwendung des Hexagons zur Kodierung und zur Partitionierung stößt man dann allerdings bei der Forderung nach einer vollständigen Überdeckung auf Schwierigkeiten, wie die folgenden Abbildungen zeigen:

Abb. 10: *Uniforme Hexagonale Partitionierung*

Eine vollständige Überdeckung ist nicht möglich. Um dennoch eine Kodierung zu ermöglichen, muß das Bild entweder mit Füllpixeln erweitert oder als periodisch angenommen werden, was am Bildrand zu Artefakten führt und zusätzlichen Speicherplatz benötigt, was sich nachteilig auf die Kompressionsrate auswirkt. Obwohl diese Partitionierung konform mit dem diskreten Modell aus Kapitel 3 ist, erweist sich diese Eigenschaft bei einer Implementierung als nachteilig, weshalb im folgenden auch nur Partitionierungsverfahren betrachtet werden, die eine vollständige Partitionierung problemlos ermöglichen.

4. Optimierungen des Verfahrens

Das oben beschriebene Kodierungsschema bietet nun verschiedene Ansätze für eine Optimierung an, wobei diese grundsätzlich in zwei verschiedene Richtungen durchgeführt werden kann. Zum einen ist eine Optimierung im Hinblick auf die Qualität des kodierten Bildes möglich, was die Einführung eines oberen Grenzwertes für den Fehler der Collage erfordert. Eine Optimierung mit dem Ziel, eine möglichst große Kompressionsrate zu erreichen, legt die Einführung eines oberen Grenzwertes für die Anzahl der Abbildungen ω_i nahe. Dieser Grenzwert beeinträchtigt die Qualität eines kodierten Bildes allerdings so stark, daß von einer Einführung abgesehen wird. Das Bestreben, die Anzahl der ω_i möglichst gering zu halten, bleibt jedoch bestehen.

Beide Ansätze führen in ihrer Konsequenz zu einer adaptiven Partitionierung mit einer variablen Größe M_R für die verschiedenen R_i, was folglich auch zu einer variablen Anzahl von Abbildungen ω_i führt.

Adaptive Partitionierungsverfahren

Die Notwendigkeit, sowohl die Qualität des kodierten Bildes als auch die Kompressionsrate zu optimieren, führt somit zu einer Ablösung der starren uniformen Partitionierungsschemata durch Schemata, welche die Bildeigenschaften bei der Partitionierung berücksichtigen. Das Ziel ist somit einerseits die Verringerung der Anzahl der ω_i zur Verbesserung der Kompressionsrate, zum anderen soll die Auflösung der Partitionierung an den Bildinhalt angepaßt werden. Beide Aspekte müssen in ein Kodierungsschema für eine adaptive Partitionierung integriert werden.

Das obige Kodierungsschema für uniforme Partitionierungen läßt sich in der vorgestellten Form nicht für eine adaptive Partitionierung anwenden, sondern bedarf einigen Anpassungen, wie z.B. die Einführung einer Markierung, ob für einen Bildbereich R_i schon eine Abbildung ω_i berechnet wurde oder ob der Bildbereich noch weiter unterteilt werden muß. Dies führt zu dem folgenden Kodierungsschema für adaptive Partitionierungen.

```
Wähle Toleranzwert ε_C für den max. Fehler der Abbildungen ω_i;

R_1 = □  /* Initialisiere R_1 mit dem vollst. Eingabebild */;
covered(R_1) :=0 /* Markiere R_1 als nicht abgebildet. */;

/* Solange nicht abgebildete R_i vorhanden sind... */
for (∃(R_i ∈ J) : covered(R_i) = 0 ) do begin

   /* Bestimme D_i ∈ I mit minimalem Abstand. */
   for (∀(D_i ∈ I) ) do
      if ( d(f ∩ (R_i × □ ), ω_i(f)) < d(f ∩ (R_i × □ ), Minimum_D(R_i)) )
         then Minimum_D(R_i) := D_i
```

```
      /* Markiere R_i als abgebildet, falls Fehler unter dem  */
      /* Toleranzwert oder falls keine weitere Unterteilung  */
      /* der R_i vorgenommen wird. Berechne ω_i .            */
      if ( d(f ∩ (R_i × □ ), Minimum_D(R_i)) < ε_C ) OR ( size(R_i) < r_min )
         then begin
            covered(R_i) := 1;
            calculate_ω(R_i);
            save_ω(R_i)
         end;

      /* Markiere R_i als abgebildet und teile R_i in kleinere */
      /* Partitionen.                                          */
      else begin
         covered(R_i) := 1;
         divide(R_i);
      end;
   end;
```

Abb. 11: *Kodierungsschema für adaptive Partitionierung*

Der Vergleich mit dem Kodierungsschema für uniforme Partitionierungen zeigt, daß die Anzahl der beiden Schleifendurchläufe nun nicht mehr konstant ist, sondern von dem Toleranzwert ε_C für den Fehler der Collage und von der Auflösung der Partitionierung abhängt. Auch wenn die Komplexität der Laufzeit sich im Vergleich zur uniformen Partitionierung nicht verändert, so variiert sie stärker bei einer adaptiven Partitionierung in Abhängigkeit vom Bildinhalt. Dies ist vor allem durch die Auflösung der Partitionierung bedingt, welche durch den Toleranzwert ε_C und die minimale Größe r_{min} der R_i festgelegt wird.

Analog zum Fall der uniformen Partitionierungen geht das konkrete Partitionierungsverfahren, wie es in der Prozedur `divide` implementiert wird, weder in die Formulierung des Kodierungsschemas noch in die Komplexität des Laufzeitverhaltens für das obige Kodierungsschema ein.

Das Partitionierungsverfahren ist vollständig in der Prozedur `divide` gekapselt und läßt sich somit in obigem Kodierungsschema leicht austauschen. Das Ergebnis der Prozedur sind Teilmengen des Eingabebildes, die sowohl als Wertebereiche R_i in die Menge J als auch als Definitionsbereiche D_i in die Menge I aufgenommen werden. Ein Austausch der Prozedur `divide` bewirkt also, daß die Elemente der Mengen J und I sich in der Struktur ändern. Diese Änderung muß bei der Implementierung berücksichtigt werden, wie z.B. bei der Berechnung des Abstandes $d(f \cap (R_i \times \square), \omega_i(f))$. Die betroffenen Operatoren wurde in den vorangegangen Kapiteln in ihrer allgemeinen Funktion schon beschrieben, weshalb im folgenden die Vorgehensweisen bei verschiedenen adaptiven Partitionierungsverfahren und die Implementierung der Operatoren beschrieben werden soll.

Quadtree-Partitionierung

Bei der Quadtree-Partitionierung wird das Bild in vier gleichgroße Rechtecke aufgeteilt. Gemäß dem Kodierungsschema für adaptive Partitionierungen auf Seite 95 wird nun jedes dieser vier Rechtecke rekursiv in vier gleichgroße Rechtecke unterteilt, bis jedes Rechteck mit einem vorgegebenen Fehler abgebildet oder nicht mehr unterteilt werden kann, was sich schematisch folgendermaßen darstellt:

Abb. 12: *Quadtree-Partitionierung*

Die Mengen J und I bestehen somit aus Quadraten verschiedener Größen. Die Wertebereiche $R_i \in J$ sollen nun aus Pixelquadraten der Größe $2^{b_1} \times 2^{b_1}, \ldots, 2^{b_q} \times 2^{b_q}$ und die entsprechenden Definitionsbereiche $D_i \in I$ aus Pixelquadraten der Größe $2^{d_1} \times 2^{d_1}, \ldots, 2^{d_q} \times 2^{d_q}$ bestehen. Unter Anwendung des Kodierungsschemas für adaptive Partitionierungsverfahren auf Seite 95 zeigt sich aufgrund des diskreten Modells aus Kapitel 3, daß die Änderungen im Vergleich zu einer uniformen Partitionierung auf der Basis von Quadraten, die bei einer Implementierung vorgenommen werden müssen, minimal sind, da nur ein einziger Operator angepaßt werden muß. Während man bei der entsprechenden uniformen Partitionierung nur einen festen *decimation*-Operator benötigt, da es nur Quadrate mit zwei verschiedenen Größen gibt, ist bei der Quadtree-Partitionierung ein variabler *decimation*-Operator mit q Faktoren $2^{d_1 - b_1}, \ldots, 2^{d_q - b_q}$ mit $d_i > b_i$ für $i = 1, 2, \ldots, q$ nötig.

Abb. 13: *Quadtree-Partitionierung von Lena*

Bei allen anderen Berechnungen, wie z.B die Berechnung der Transformationen oder des euklidischen Abstandes, sind eventuelle Änderungen rein technischer Natur, die durch die nun verschiedenen neuen Größen der Pixelblöcke bedingt sein können.

Trianguläre Partitionierung

Analog zum Fall der uniformen Partitionierungen lassen sich die Quadrate der Quadtree-Partitionierung auch bei einem adaptiven Partitionierungsverfahren durch Dreiecke ersetzen. Die Motivation zum Einsatz und die Änderungen im Bezug auf den quadratischen Fall entsprechen der Diskussion für die uniforme Partitionierung, so daß sie an dieser Stelle nicht noch einmal wiederholt werden soll. Das Verfahren zur Partitionierung des Bildes beginnt mit der Zerlegung des Eingabebildes in zwei Dreiecke. Jedes dieser Dreiecke wird anschließend durch die Verbindung der Seitenhalbierenden in vier weitere Dreiecke unterteilt. Dieses Verfahren wird dann in das Kodierungsschema für adaptive Partitionierungsverfahren auf Seite 95 eingebettet, so daß eine rekursive Zerlegung des Eingabebildes erfolgt, welche in der Prozedur `divide` implementiert wird.

Abb. 14: *Triangular Partitionierung von Lena*

HV-Partitionierung

Ein Nachteil der Quadtree-Partitionierung ist, daß die Definitionsbereiche D_i nicht kontext-sensitiv aufgebaut werden. Die HV-Partitionierung (Horizontale-Vertikale-Partitionierung) arbeitet wie die Quadtree-Partitionierung, nur mit dem Unterschied, daß die Position der Partitionierung innerhalb eines Rechtecks variabel ist, wobei die entstehenden Teilmengen des Eingabebildes eine selbstähnliche Struktur haben sollten. So kann man versuchen, die Partitionierung so vorzunehmen, daß Kanten immer diagonal durch eine Partition laufen:

Abb. 15: *Prinzip der HV-Partitionierung*

Innerhalb eines Pixelblocks der Größe $N \times M$ soll dann eine Partitionierung vorgenommen werden, wenn eine Kante den Pixelblock durchläuft. Dafür berechnet man zeilen- bzw. spaltenweise sukzessiv die horizontalen und vertikalen Summen und daraus die Mittelwerte der Pixelwerte. Anschließend werden die Differenzen von zwei aufeinanderfolgenden Mittelwerten mit dem kürzesten Abstand zum Rand des Pixelblocks multipliziert. Der Pixelblock besitze die Pixelwerte $r_{i,j}$ für $0 \le i < N$ und $0 \le j < M$, dann ergeben sich die folgenden Berechnungen:

$$h_j = \frac{\min(j, M-j-1)}{M-1} \cdot \left(\sum_i r_{i,j} - \sum_i r_{i,j+1} \right)$$

$$v_i = \frac{\min(i, N-i-1)}{N-1} \cdot \left(\sum_j r_{i,j} - \sum_j r_{i+1,j} \right)$$

Die Partitionierung wird an der horizontalen Position j oder der vertikalen Position i durchgeführt, je nachdem welches Maximum der Werte $|h_j|$ und $|v_i|$ größer ist.

Die Mengen J und I bestehen somit aus Rechtecken verschiedener Größen, was eine Anpassung der *fetch*-, *decimation*- und *put*-Operatoren bedingt. Besonders der *decimation*-Operator muß bei diesen Verfahren nochmal modifiziert werden, da er in horizontaler und in vertikaler Richtung verschiedene Faktoren verwenden muß. Die Berechnungen, wie z.B die Berechnung der Transformationen oder des euklidischen Abstandes, sind Änderungen rein technischer Natur, die durch die nun verschiedenen neuen Größen der Pixelblöcke bedingt sind.

Abb. 16: *HV-Partitionierung von Lena*

Verringerung des Suchaufwands

Obwohl der Einsatz von adaptiven Partitionierungsverfahren die Anzahl der notwendigen Transformationen für die Kodierung eines Bildes senkt und damit auch die Größe der Menge J und I verringert, was neben besseren Kompressionsraten auch eine Laufzeitverbesserung zur Folge hat, bleibt immer noch ein enormer Rechenaufwand für die Suche des optimalen D_i für ein gegebenes R_i bestehen. Zur Senkung dieses Rechenaufwandes werden die R_i und D_i bei ihrer Berechnung mit Hilfe eines Klassifikationsschema in verschiedene disjunkte Klassen eingeteilt. Die Suche nach einem optimalen D_i für ein gegebenes R_i findet dann anschließend nur noch anhand jener D_i statt, die sich in

derselben Klasse wie das gegebene R_i befinden. In Abhängigkeit von dem verwendeten Klassifikationsschema verringert dieses Vorgehen den Aufwand für die Suche drastisch. Als Klassifikationsschema können Verfahren verwendet werden, bei denen man nach der Einteilung der R_i und D_i in die verschiedenen Klassen davon ausgehen kann, daß sich alle Klassenmitglieder gut mit Hilfe der verwendeten affinen Transformationen aufeinander abbilden lassen. Ein Beispiel für ein solches Verfahren ist eine Klassifikation anhand des Mittelwertes und der Varianz der Pixelwerte [Fisher92b]. Weitere Verfahren sind in [Boss89] und [Jacobs92] zu finden.

Die Klassifikation der Definitions- und Wertebereiche dient ausschließlich der Verbesserung des Laufzeitverhaltens und dies teilweise zu Lasten der Qualität, da eine ungenaue Klassifikation das Finden der optimalen Kombination von R_i und D_i verhindert, weshalb auf diese Verfahren im Rahmen dieser Arbeit auch nicht weiter eingegangen werden soll.

5. Kompressionsrate und Maße für die Bildqualität

Den vorausgegangenen Ausführungen ist zu entnehmen, daß eine fraktale Bildkodierung i.a. eine Kodierung mit Verlust ist. Das Ziel der Kodierung ist zwar die Bildung eines Modells für ein Eingabebild mit Hilfe affiner Transformationen, doch unterscheidet sich eine fraktale Bildkodierung in diesem Punkt entscheidend von anderen modellbasierten Verfahren, da das hier erzeugte Bildmodell nicht exakt das Eingabebild beschreibt. Für einzelne Sonderfälle von Eingabebildern ist dies zwar möglich, doch gilt dies nicht für einen allgemeinen Kodierungsansatz, wie er hier diskutiert wird.

Bei der Diskussion von experimentellen Ergebnissen müssen die erzielten Kompressionsraten somit in ein Verhältnis zu den erzielten Bildqualitäten gesetzt werden. Während die Kompressionsrate exakt gemessen werden kann, steht man bei der Bildqualität vor dem Problem, daß für eine objektive Messung der Qualität eines Bildes ein entsprechendes Modell für das HVS existieren müßte. Da allerdings kein solches Modell für das HVS existiert, finden Maße Verwendung, die dem menschlichen Empfinden von guter und schlechter Bildqualität möglichst nahe kommen sollen. Diese Verfahren sind jedoch immer nur eine Annäherung, so daß neben berechenbaren objektiven Maßen zur Bildqualität auch Verfahren zur Messung subjektiver Bildqualität existieren.

Der subjektiven Bildqualität wird bei der Anwendung einer fraktalen Bildkodierung in mehrfacher Hinsicht Rechnung getragen. Durch fraktale Bildkodierung hervorgerufene Fehler werden zum einen durch das HVS als nicht so störend empfunden, wie Fehler, die bei anderen Kompressionsverfahren auftreten [Kunt85] [Rama86]. Zum anderen wird auch der Einsatz von adaptiven Partitionierungsverfahren und von Partitionierungsverfahren mit Teilmengen motiviert, deren Begrenzungslinien möglichst unregelmäßig und nicht gleichmäßig horizontal und vertikal verlaufen. Die Trennung zwischen einer objektiven und einer subjektiven Bildqualität ist im Kontext der fraktalen Bildkodierung somit sinnvoll. In dieser Arbeit wird ein objektiver Qualitätsbegriff zur Einordnung der Verfahren untereinander und zum Vergleich mit anderen

Bildkompressionsverfahren herangezogen. Ein subjektiver Qualitätsbegriff wird für die Verfeinerung einzelner fraktaler Bildkodierungsverfahren und deren Motivation verwendet.

Als objektiver Qualitätsbegriff dient in dieser Arbeit der *peak signal-to-noise ratio* (PSNR) zur Messung der Differenz zwischen dem Originalbild f und dem komprimierten Bild f':

$$PSNR = 20 \log_{10}\left(\frac{P}{d(f, f')}\right)$$

Der Wert P ist der *peak*-Wert des Signals, also der größte mögliche Wert des Signals, der bei digitalen Graustufenbildern typischerweise 255 beträgt. Der euklidische Abstand $d(f, f')$ gibt den mittleren quadratischen Fehler zwischen den beiden Bildern an, den man auch als *noise* oder Rauschen des Originalbilds f bezeichnen kann, wenn man das komprimierte Bild f' in der Form $f' = f + d(f, f')$ beschreibt. Die Einheit ist dB und gibt das Verhältnis des *peaks* des Signals zur Differenz der beiden Bilder an. Eine Zunahme des PSNR von 20 dB entspricht der zehnfachen Abnahme des euklidische Abstand der beiden Bilder.

Eine Bestimmung der subjektiven Qualität läßt sich mit Hilfe des *bubble sort* Algorithmus vornehmen. Die zu beurteilenden Bilder werden nacheinander von einem Gutachter bildweise anhand der subjektiv empfundenen Bildqualität mit dem *bubble sort* Algorithmus in eine Liste der schon beurteilten Bilder eingeordnet [Jain89]. Das Ergebnis ist eine nach der subjektiven Bildqualität geordnete Liste der zu beurteilenden Bilder, in welcher die Empfindlichkeit des HVS gegenüber verschiedenen Kodierungsfehlern direkt eingegangen ist. Durch die Verwendung des *bubble sort* Algorithmus vergleicht ein Gutachter immer nur zwei Bilder in einem direkten Vergleich und ordnet diese beiden Bilder dann gemäß ihrer Qualität. Dieses Verfahren basiert also auf der Tatsache, daß ein Gutachter immer nur zwei Bilder gleichzeitig beurteilen muß und liefert nach [Frenden60] bessere Ergebnisse, als Verfahren, bei denen ein Gutachter mehrere Bilder gleichzeitig begutachtet.

Analog zu anderen Bildkompressionsverfahren existieren auch für die fraktale Bildkodierung Verfahren zur Verbesserung der subjektiven Qualität. Zu dieser Klasse von Verfahren gehören Verfahren, die eine Nachbearbeitung im kodierten Bild vornehmen, wie z.B. eine Glättung der Partitionierungsgrenzen [Kunt85] [Rama86] oder auch Verfahren, welche die Helligkeitsverteilung des Eingabebildes besonders speichern und diese anschließend in das kodierte Bild eingehen lassen, wie es auch in Barnsley's Patent beschrieben wird [Barnsley93]. Da diese Verfahren allerdings ausschließlich die Verbesserung der subjektiven Bildqualität des kodierten Bildes zum Ziel haben und unabhängig von der eigentlichen Kodierung sind, werden sie in dieser Arbeit nicht weiter betrachtet.

6. Zusammenfassung, Bewertung und Schlußfolgerungen

Anhand des vorliegenden Kapitels 4 läßt sich das Leistungsspektrum für eine fraktale Bildkodierung bestimmen, welches im nachfolgenden Kapitel 5 durch Ergebnisse aus experimentellen Untersuchungen ergänzt wird. Die beschriebenen Leistungscharakteristika sind aufgrund ihrer Ableitung aus dem Modell weitgehend von funktionalem Charakter und beziehen sich deshalb auch noch nicht auf konkrete Angaben zu Kompressionsraten und Bildqualität. Dies erlaubt die Trennung von Ergebnissen allgemeinen Typs, die für die ganze Klasse von Systemen zur fraktalen Bildkodierung gelten, und von Ergebnissen, die auf der Basis einer speziellen Implementierung gewonnen wurden.

So lassen sich die in diesem Kapitel gewonnenen Erkenntnisse als allgemein gültige Eigenschaften für fraktale Bildkodierungssysteme interpretieren und dienen so einer Charakterisierung, Bewertung und Einordnung dieser Systeme im Verhältnis zu anderen Kompressionsverfahren, wie sie im zweiten Teil von Kapitel 5 vorgenommen wird.

Anhand des Codec's und des Modells aus Kapitel 3 läßt sich so nachweisen, daß eine fraktale Bildkodierung in Bezug auf den Rechenaufwand für die Kodierung bzw. Dekodierung stark asymmetrisch ist. Diese Asymmetrie hat natürlich auch Einfluß auf das Einsatzgebiet eines solchen Kodierungssystems, da für die Kodierung leistungsfähigere Systeme als für die Dekodierung benötigt werden. Aufgrund der Eigenschaft, eine auflösungsunabhängige Kodierung zu sein wie in [*Codec zur fraktalen Bildkodierung: Dekodierer* auf Seite 83] beschrieben, werden an das System zur Dekodierung nur minimale Anforderungen in Bezug auf Rechenleistung und Bildschirmauflösung gestellt.

Diese positive Eigenschaft für die Darstellung von Bilddaten in einem heterogenen Rechnerumfeld, wie es z.B. bei einem Online-Katalog der Fall ist, wird leider nicht von der Fähigkeit zum progressiven Senden unterstützt, da diese Fähigkeit nach [*Progressives Senden* auf Seite 81] der fraktalen Bildkodierung fehlt. Zur Darstellung und Dekodierung eines Bildes müssen immer alle komprimierten Bilddaten übertragen werden, was besonders bei großen Datenmengen und schmalbandigen Netzwerkverbindungen ein Nachteil ist, vor allem im Vergleich zu Verfahren mit der Eigenschaft zum progressiven Senden, wie es z.B. bei einer Kodierung mit der Hilfe von Wavelets [*Wavelet Analyse (wavelet analysis)* auf Seite 28] möglich ist.

Die Definition und Diskussion des Codec's liefert in Verbindung mit Kapitel 3 auch weitere Anhaltspunkte für mögliche Verbesserungen und Optimierungen innerhalb des Gesamtsystems. Basierend auf einem Kodierungsschema für uniforme und einem für adaptive Partitionierungsverfahren führen diese Betrachtungen zur Definition und Diskussion verschiedener Partitionierungsverfahren. Die Anwendung der verschiedenen Partitionierungsverfahren ändert nichts an den zugrundeliegenden Kodierungsschemata, so daß durch diese Anwendung keine Änderungen am Funktionsumfang des Gesamtsystems auftreten.

Die Diskussion der verschiedenen Partitionierungsverfahren ist dann aus mehreren Gründen ein weiteres zentrales Thema dieses Kapitels. Einerseits beeinflußt die Wahl des Partitionierungsverfahrens die Kompressionsrate und die Laufzeit des Verfahrens durch die Wahl und die Anzahl der Definitions- und Wertebereiche und andererseits hat

diese Wahl natürlich auch Auswirkungen auf die Qualität des kodierten Bildes. Aus diesem Grund bildet ein Abschnitt über den Zusammenhang zwischen der Kompressionsrate und der Qualität des kodierten Bildes den Abschluß dieses Kapitels. Dieser Abschnitt findet anschließend in Kapitel 5 seine Fortsetzung, wenn anhand einer konkreten Implementierung die Performance von Verfahren zur fraktalen Bildkodierung anhand verschiedener Implementierungen gemessen wird. Die Basis der experimentellen Untersuchungen bilden die Verfahren und deren Diskussion aus diesem Kapitel, so daß Kapitel 5 auch als Überprüfung und Bestätigung der bisher getroffenen Aussagen betrachtet werden kann.