Berichte zur Rechnerarchitektur
Band 3 Nr. 11 (1997)

Technischer Bericht

Wavelets: Grundlagen und Anwendung in der Bildkompression

Michael M. Gutzmann René Scholz

Friedrich-Schiller-Universität Jena
Institut für Informatik
Lehrstuhl für Rechnerarchitektur und -kommunikation

Hrsg.: Prof. Dr. Ing. W. Erhard

FSU

Wavelets: Grundlagen und Anwendung in der Bildkompression

Michael M. Gutzmann **René Scholz**

Lehrstuhl für Rechnerarchitektur und -kommunikation, Universität Jena
D-07740 Jena, Germany
{gutzmann,mrz}@informatik.uni-jena.de
☎: +49/3641/946375 fax: +49/3641/946372

Dezember 1997

Kurzfassung

Dieser Bericht beschreibt theoretische Grundlagen der Bildkompression und geht dabei speziell auf den Einsatz von Wavelets ein. Die bei den wavelet-basierten Verfahren durchzuführenden Schritte werden anhand verschiedener Alternativen demonstriert. Ein Vergleich mit anderen bekannten Bildkompressionsverfahren zeigt die hohe Leistungsfähigkeit der wavelet-basierten Verfahren auf.

1 Einleitung

Die Entwicklung effektiver Verfahren zur Datenkompression von Bildern hat aufgrund der ständig zunehmenden Nutzung grafikorientierter Netzwerkdienste, wie z. B. dem *World Wide Web* (WWW) oder Video-Conferencing, eine große Bedeutung gewonnen.
Obwohl gegenwärtig große Anstrengungen unternommen werden, die Übertragungsleistungen im Internet mittels moderner Kommunikationstechniken wie z. B. dem *Asynchronous Transfer Mode* (ATM) zu erhöhen, ist es auch in naher Zukunft schwierig, akzeptable Antwortzeiten für die Übertragung von Bildern, Videos und Audiodaten zu ermöglichen.
Aber auch für viele Anwendungen aus der Bildverarbeitung, wie der Archivierung von Fingerabdrücken beim amerikanischen FBI [BBH93, Bri96] oder bei der Entwicklung neuer und verbesserter Fernsehnormen, wie dem *High Definition Television* (HDTV) [Dup95, Cri97], ist eine Bildkompression unumgänglich geworden.
Zu diesem Zweck wurden in der Vergangenheit bereits viele verschiedene Kompressionsverfahren entwickelt, insbesondere die Verfahren, die den bekannten Bildformaten `GIF` und `JPEG` zugrunde liegen [Inc87, Wal91]. Trotz aller Fortschritte bei der Optimierung dieser Verfahren, sind sie heute, im multi-medialen Zeitalter, nicht mehr ausreichend.
Diese klassischen Bildkompressionsverfahren erreichen lediglich Kompressionsraten von etwa 2 : 1 bis 4 : 1 [How93], was für die aktuellen Anforderungen nicht mehr ausreichend ist. Deshalb wird derzeit sehr intensiv an besseren Kompressionsverfahren für Bilder gearbeitet, die noch höhere Kompressionsraten bei einer ausreichend guten Bildqualität ermöglichen. Es ist klar, daß es sich dabei um eine verlustbehaftete (*lossy*) Kompression handeln muß,

denn herkömmliche, verlustfreie Verfahren wie etwa das *Lempel-Ziv-Welch* (LZW) Verfahren [Wel84], das im `GIF`-Format angewendet wird, sind schon recht weit ausgereizt und können nicht mehr die in Zukunft benötigten hohen Kompressionsraten erzielen.
Für die meisten Realweltbilder (Fotografien) bietet sich eine mit Verlusten behaftete Kompression geradezu an, da das menschliche Auge geringe Unterschiede, z. B. im Farb- oder Helligkeitsverlauf zwischen dem Originalbild und dem komprimierten Bild, kaum wahrnehmen kann.
Neben der fraktalen Bildkompression sind in letzter Zeit besonders die Wavelets ein Schwerpunkt der Forschung geworden, da sich mit ihnen sehr erfolgversprechende Ergebnisse erzielen lassen [HJS95]. Auch für *JPEG 2000*, dem Nachfolger des JPEG-Standards, ist der Einsatz von Wavelets beabsichtigt [Com97].

2 Grundlagen der Bildkompression

Die (Bild-)Kompression wurzelt in der Informations- und Kodierungstheorie, deren mathematische Grundlagen 1949 durch Shannon beschrieben wurden [SW49, SW76].

In diesem Zusammenhang betrachten wir ein Bild als eine diskrete Zeichenquelle voneinander unabhängiger Zeichen, wobei der Zeichenvorrat die Werte 0 bis 255 umfaßt (es werden nur Graustufenbilder eingesetzt).
Nach Shannon kann man die mittlere Kodierungslänge in Bit's, also die Anzahl von Bit's die pro Zeichen durchschnittlich benötigt werden, um die Nachricht zu kodieren, nach unten durch die (*zero-order*) Entropie H abschätzen [SW76, HQ83, Lu97].
Die Entropie H einer Nachricht mit N Zeichen aus einer diskreten Nachrichtenquelle ist:

$$H := -\sum_{i=1}^{N} p_i \log_2 p_i \text{ bit}, \qquad \text{mit } \sum_{i=1}^{N} p_i = 1, \tag{1}$$

wobei p_i die Wahrscheinlichkeit des Auftretens des Zeichens i und das **bit** die Maßeinheit der Entropie bzw. Information ist.[1]
Die Wahrscheinlichkeit eines Zeichens bzw. Bildpunktes wird dabei durch dessen relative Häufigkeit ersetzt. Je kleiner die Wahrscheinlichkeit p_c eines Zeichens c ist, um so größer ist die damit zusammenhängende Informationsmenge $-\log_2 p_c$ bit.

Das Verhältnis der Entropie einer Nachrichtenquelle zu ihrem theoretischen Maximalwert wird *relative Entropie* genannt [SW76]. Dieses ist die größtmögliche Informationsdichte bei Verwendung des gleichen Zeichenvorrats (Alphabets). Die Differenz (1 − relative Entropie) ist die *Redundanz*.
Je größer die Redundanz ist, um so kleiner wird die benötigte Kodierungslänge. Die Entropie des Testbildes `lena.pgm` (Abb. 1) beträgt 7.57 bit.
Um höhere Kompressionsraten als bei den verlustfreien Verfahren zu ermöglichen, muß die Redundanz des betrachteten Bildes vergrößert werden.
Zu diesem Zweck kann man die Wavelet-Transformation benutzen, womit man eine Dekorrelation der Bildpunkte erreicht. In [HJS95] wird angegeben, das dadurch bereits die Entropie des Bildes `lena.pgm` auf 4.24 bit sinkt.

[1] Bei Verwendung anderer Logarithmen ergeben sich andere Maßeinheiten, für $\log_{10}$ z. B. das **hartley**.

2.1 Messung der Bildqualität

Um den bei verlustbehafteten Verfahren entstehenden Bildfehler quantitativ zu erfassen, werden in der Bildverarbeitung die beiden Maße RMSE (Root Mean-Squared Error) und PSNR (Peak Signal-to-Noise Ratio, in dB) benutzt [HJS95, Fou95]:

$$\begin{aligned} \text{RMSE} =& \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - \hat{p}_i)^2} \\ \text{PSNR} =& 20\log_{10}\frac{255}{\text{RMSE}}, \end{aligned} \tag{2}$$

wobei p_i ein Bildpunkt des Originalbildes ist und $\hat{p}_i$ der zugehörige Punkt des komprimierten Bildes.
Allerdings erfassen diese Werte nicht den visuellen Eindruck, den ein menschlicher Betrachter vom komprimierten Bild hat. Hierzu sind wesentlich kompliziertere Fehlermaße nötig (s. a. [Lu97]).

2.2 Die Kompressionsrate

Die Kompressionsrate eines komprimierten Bildes ist meist wie folgt definiert:

$$k = \frac{\text{Dateigröße des Originalbildes in Byte}}{\text{Dateigröße des komprimierten Bildes in Byte}} .$$

Eine andere oft gebrauchte Angabe sind die *bits per pixel* (bpp), die für das komprimierte Bild[2] benötigt werden: bpp= $8/k$.

2.3 Testbilder

In der Literatur werden verschiedene Bilder zum Vergleich der entwickelten Bildkompressionsverfahren angegeben, das häufigste dabei eingesetzte Testbild ist `lena.pgm` in Abb. 1, welches auch hier verwendet wurde.

Abbildung 1: Testbild `lena.pgm` (256×256 Bildpunkte, Grauwerte)

[2] Ein unkomprimiertes Graustufenbild im Format `Raw PGM5` benötigt 8 bpp sowie einen Header.

2.4 Blockschema einer verlustbehafteten Bildkompression

Bei verlustbehafteten Kompressionsverfahren wird die im Ausgangsbild enthaltene Information einer bestimmten Transformation unterzogen. Durch die dabei stattfindende Dekorrelation der Bildpunkte wird die wesentliche Bildinformation in wenigen Transformationskoeffizienten konzentriert, s. a. Abb. 2, welches die Energiekonzentrierung durch die Wavelet-Transformation darstellt.

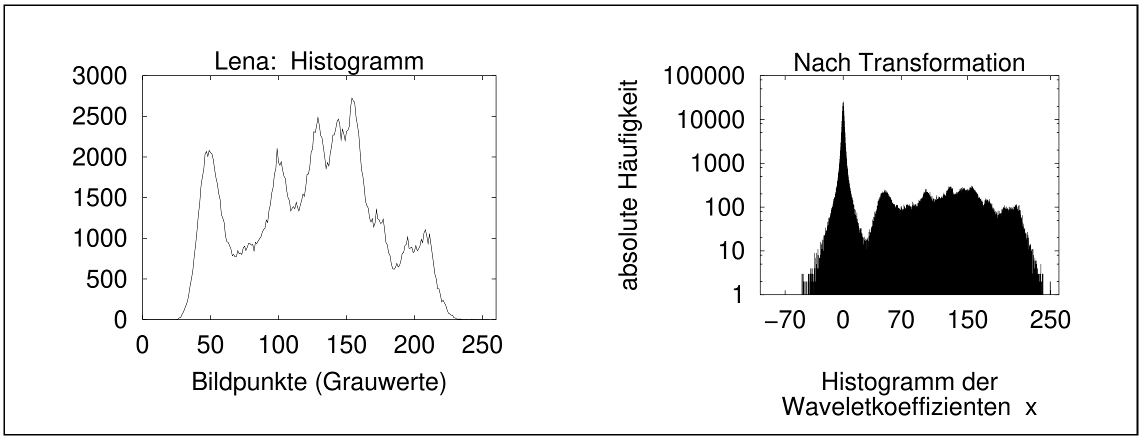

Abbildung 2: Energiekonzentrierung der Wavelet-Transformation

Die reellwertigen Koeffizienten werden anschließend quantisiert (hierbei findet der Informationsverlust statt) und die entstehenden ganzzahligen Werte mit einem normalen und verlustfreien Verfahren kodiert.
Dabei können die beiden Schritte Quantisierung und Kodierung durch entsprechende Algorithmen auch zusammengefaßt werden, wie dies z. B. beim EZW-Algorithmus[3] von Shapiro der Fall ist [Sha93]. Abb. 3 stellt dieses Vorgehen dar.

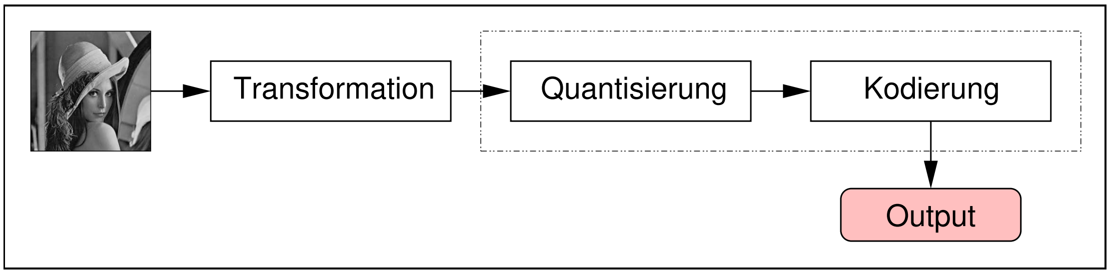

Abbildung 3: Blockschema einer verlustbehafteten Bildkompression

Dabei sind in jedem dieser drei Kompressionsschritte verschiedene Alternativen möglich. Beispiele für den Transformationsschritt sind u. a. die Fourier-Transformation, die Diskrete Kosinus Transformation (DCT), die Walsh-Hadamard Transformation (WHT) und die Wavelet-Transformation [Lu97].

[3]EZW: Embedded Zerotree Wavelet Algorithm

Bei Verwendung der Wavelet-Transformation für die Bildkompression ergibt sich ein Blockschema, siehe Abb. 4:

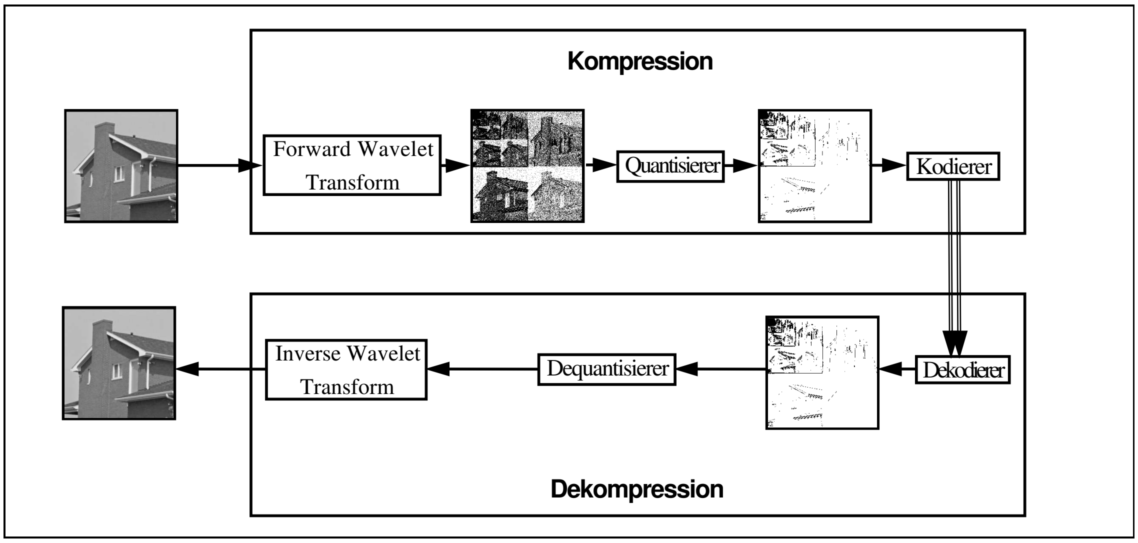

Abbildung 4: Bildkompression mittels Wavelet-Transformation

3 Grundlagen zur Theorie der Wavelets

Die Theorie der Wavelets ist ein noch recht junger Gegenstand der Forschung und wurde Ende der 80ger Jahre von mehreren Wissenschaftlern entwickelt, vor allem von Morlet, Meyer, Mallat und Daubechies [JS93, Gra95].
Als Ergebnis dieser Arbeiten liegt heute mit den Wavelets ein Funktionensystem vor, in dem Funktionen mit lokalen Besonderheiten gut approximiert werden können.
Dabei erlaubt die Wavelet-Transformation ähnlich wie bei der *Windowed Fourier Transform* (Gefensterte Fourier-Transformation, auch Short Time Fourier Transform, STFT) eine gleichzeitige Analyse des Zeit- und Frequenzverhaltens. Der große Vorteil der Wavelet-Transformation liegt in der automatischen Anpassung des Zeit-Frequenz-Fensters, so daß bei hohen Frequenzen eine feine Auflösung und bei kleinen Frequenzen ein entsprechend breites Fenster gewählt wird.

3.1 Die Kontinuierliche Wavelet-Transformation (CWT)

Eine Funktion $\psi \in L^2(\mathbb{R})$, welche die Zulässigkeitsbedingung

$$0 < C_\psi := \int_{\mathbb{R}} \frac{|\hat{\psi}(\xi)|^2}{|\xi|} \, d\xi < \infty, \tag{3}$$

erfüllt, heißt *Wavelet* (C_ψ heißt *Zulässigkeitskonstante*).
Hierbei ist $\hat{\psi}(\xi)$ die Fouriertransformierte von $\psi(t)$:

$$\hat{\psi}(\xi) = \int_{\mathbb{R}} \psi(t) \mathrm{e}^{-\mathrm{i}\xi t} \, dt \, .$$

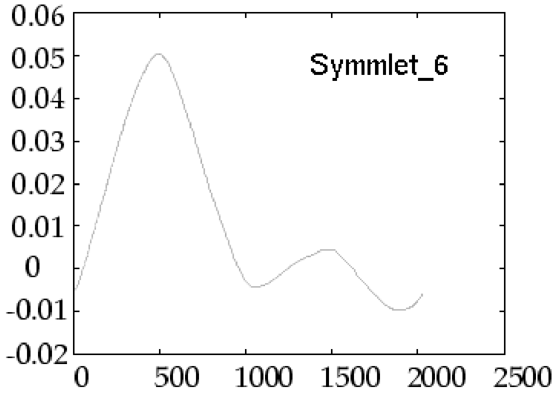

Abbildung 5: Die Waveletfunktion *Symmlet_ 6* [Gra95]

Ein Beispiel für eine Waveletfunktion ist in Abb. 5 zu sehen.

Durch Translation und Verschiebung werden aus einem *Mother-Wavelet* $\psi(t)$ weitere Wavelets $\psi_{\tau,s}(t)$ abgeleitet, welche eine Funktionenfamilie $\{\Psi_{\tau,s}\}$ bilden:

$$\psi_{\tau,s}(t) := \frac{1}{\sqrt{|s|}}\,\psi\left(\frac{t-\tau}{s}\right), \quad \tau, s \in \mathbb{R}, s \neq 0 \tag{4}$$

Interessant sind vor allem orthogonale Wavelets, da die Familie $\{\Psi_{\tau,s}\}$ dann eine Orthonormalbasis des $L^2(\mathbb{R})$ bildet.

Bei der CWT werden für die Parameter τ und s der Wavelets $\psi_{\tau,s}$ reelle Zahlen verwendet. Die Wavelet-Transformation $\mathcal{W}(f)$ einer Funktion $f \in L^2(\mathbb{R})$ mit dem Wavelet ψ ist dann die folgende Integraltransformation:

$$\mathcal{W}_{\tau,s}(f) = \langle f, \psi_{\tau,s}\rangle = \int_{\mathbb{R}} f(t)\,\psi_{\tau,s}(t)\,dt \qquad \tau, s \in \mathbb{R},\ s \neq 0 \tag{5}$$

Es wird also die Ausgangsfunktion f mit einer Wavelet-Funktion einer bestimmten “Breite” (durch den Parameter s bestimmt) multipliziert und das Ergebnis über die reelle Achse aufsummiert ($-\infty < \tau < +\infty$).

Wenn die Zulässigkeitsbedingung (3) erfüllt ist, dann besitzt die CWT eine Umkehrung [JS93] und man kann die Originalfunktion f wieder aus der Wavelet-Transformierten $\mathcal{W}$ berechnen:

$$f(t) = \frac{1}{C_\psi}\int_{\mathbb{R}^2} \mathcal{W}_{\tau,s}(f)\,\psi_{\tau,s}(t)\,\frac{d\tau\,ds}{s^2} \tag{6}$$

3.2 Eigenschaften der Wavelet-Transformation

Aufgrund des Skalarprodukts in (5) folgt, daß die Wavelet-Transformation eine lineare Transformation ist, die bei Verwendung von orthonormalen Wavelets sogar zu einer Orthonormaltransformation wird.
Weiterhin ist die Wavelet-Transformation energieerhaltend [LMR94], d. h. es gilt:

$$\|f\|^2 = \|\mathcal{W}(f)\|^2 \,. \tag{7}$$

Ein weiterer großer Vorteil ist das bessere Zeit-Frequenzverhalten der Wavelet-Transformation gegenüber der häufig eingesetzen Fourier-Transformation, da die Wavelet-Transformation *gleichzeitig* Informationen über das Zeit- und Frequenzverhalten des Signals liefert.
Allerdings kann man aufgrund der Zeit-Frequenz-Unschärferelation die Auflösung $\Delta\omega$ in der Frequenz und Δt in der Zeit nicht gleichzeitig beliebig klein werden lassen, denn das Produkt ist nach unten beschränkt [Chu92]:

$$\Delta\omega \cdot \Delta t \geqq \frac{1}{2} \,.$$

Mit $\Delta\omega$ und Δt wird ein Zeit-Frequenz-Fenster gebildet. Bei der gefensterten Fourier-Transformation, bei der die Funktion f noch mit einer Fensterfunktion g, wie z. B. $g(t) = \pi^{1/4}\,\mathrm{e}^{-t^2/2}$, multipliziert wird, sieht die Transformation von f folgendermaßen aus:

$$\hat{f}(\xi,\tau) = \int_{\mathbb{R}} f(t)\,\mathrm{e}^{-\mathrm{i}\xi t}\,g(t-\tau)\,dt \,.$$

Dabei ist die Form dieser Fenster fest. Bei der Wavelet-Transformation hingegen variieren diese Fenster im Verhältnis ihrer Seitenlängen, siehe Abb. 6.

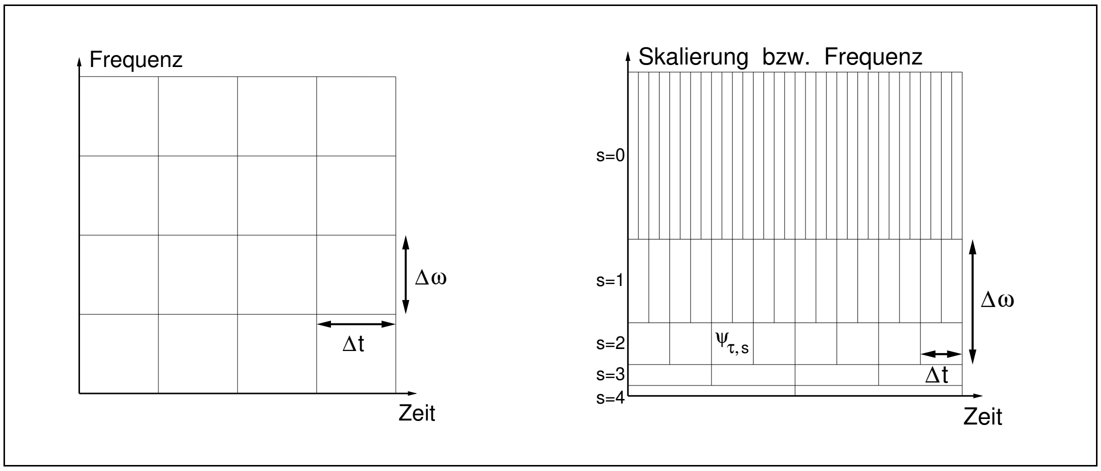

Abbildung 6: Parkettierung der Zeit-Frequenz-Ebene durch die STFT (links) und die Wavelet-Transformation (rechts)

Durch die Wavelets ist es so möglich, gezielt auf die Seitenlängen der Fenster Einfluß zu nehmen: für hohe Frequenzen wird das Fenster schmaler und die Auflösung in der Zeit besser, für tiefe Frequenzen wird das Fenster breiter und hat eine kleinere Höhe, wodurch eine bessere Frequenzauflösung erfolgt.

Dies ist besonders für die Bildkompression eine wertvolle Charakteristik: Bei vielen Realweltbildern sind oft große, gleichmäßig gefärbte Bildteile wie z. B. der Bildhintergrund vorhanden, die keine besonderen Strukturen aufweisen und daher einen tieffrequenten Signalcharakter haben. Hier ist eine bessere Frequenzauflösung wichtiger als eine genaue Ortsauflösung. Bei Kanten u. ä. Strukturen im Bild ist hingegen eine bessere Ortsauflösung wünschenswerter, da das menschliche Auge auf Veränderungen an solchen Strukturen viel empfindlicher reagiert als etwa beim Bildhintergrund.
Anhand der Abb. 6 wird auch deutlich, daß Wavelets mit einem kompakten[4] Träger[5] besonders geeignet sind, da mit diesen eine kleinere Zeitauflösung erreicht wird.
Auch die vom JPEG-Verfahren bekannten Blockeffekte treten bei Wavelets nicht auf, stattdessen erscheinen die aufgrund des Informationsverlustes während der Quantisierung auftretenden Bildstörungen als wellenförmige Artefakte, die von menschlichen Auge nicht so intensiv wahrgenommen werden, s. a. Abb. 13.

3.3 Die Schnelle Wavelet-Transformation (FWT)

Die praktische Realisierung der Wavelet-Transformation erfolgt durch ein fortgesetztes Filtern der Eingangsdaten mit je einem *Hochpaßfilter* g und *Tiefpaßfilter* h mit anschließendem *Downsampling* um 2.
Dieser Filtervorgang kann als diskrete Faltung des Vektors der Eingangsdaten mit dem Vektor der Filterkomponenten aufgefaßt werden (fortgesetzte Bildung von Skalarprodukten), woran sich aber noch das Downsampling anschließt, also jeder 2. Wert weggelassen wird.

Die FWT benutzt bestimmte endliche *Finite Impulse Response* (FIR) Filter, und zwar insbesondere die *Quadrature Mirror Filter*, für die Filter g und h.
Für diese Filter müssen folgende Bedingungen erfüllt sein:

- $g_k = (-1)^k \, h_{L-1-k}, \quad k = 0, \dots L-1$ (Die Filter bestehen aus L Komponenten)
- $\sum\limits_k h_k = \sqrt{2}$
- $\sum\limits_k g_k = 0$
- $\sum\limits_k h_k h_{k+2m} = \delta_{0,m}$

Diese Bedingungen sichern die Umkehrbarkeit und die Orthogonalität der Transformation durch die Filterprozesse.

Beispiel 1 *Der Daub2-Filter besitzt folgende 4 Komponenten:*

$$\begin{aligned} g &: (\ \ 0.48296, \quad 0.83652, \quad 0.22414, -0.12941) \\ h &: (-0.12941, -0.22414, \quad 0.83652, -0.48296) \end{aligned}$$

[4] Im $\mathbb{R}^n$ gleichbedeutend mit begrenzt und abgeschlossen, z. B. ist jedes endliche Intervall $[a, b]$ kompakt.
[5] Menge der Punkte, in denen eine Funktion (hier: Waveletfunktionen) von Null verschiedene Werte annimmt.

Der Vorgang des Filterns mit Downsampling wird durch die Operatoren G und H dargestellt, die Rekonstruktion der Ausgangsdaten (dann mit *Upsampling*) geschieht durch die adjungierten Operatoren G^* und H^*.
Die Operatoren G und H erzeugen in jedem Schritt i der FWT aus den Ausgangsdaten ein tieffrequentes Approximationssignal a_i und ein hochfrequentes Detailsignal d_i:

$$a_i = Ha_{i-1}, \qquad d_i = Ga_{i-1} \ .$$

Der Operator H wird dabei immer wieder auf die entstehenden Approximationssignale a_i angewendet, bis die Länge dieses Vektors a_i gleich 1 ist und daher kein Downsampling mehr durchgeführt werden kann (wir setzen der Einfachheit halber voraus, daß der Vektor der Ausgangsdaten eine Länge von 2^N Komponenten hat).
Nach N Transformationsschritten ist die FWT dann beendet.

Bei der Rekonstruktion werden die umgekehrten Filter g^* und h^* benutzt: $g_k^* = g_{L-1-k}$.
In jedem Schritt der Rekonstruktion wird aus je einem Detailsignal d_i und einem Approximationssignal a_i die nächst bessere Approximation a_{i-1} erzeugt, indem für a_i und d_i ein Upsampling stattfindet. Dabei wird zwischen je zwei Komponenten der Vektoren eine Null eingefügt, wodurch sich die Länge der Vektoren verdoppelt. Die entstehenden (temporären) Vektoren a_i' und d_i' werden dann mit den Filtern g^* und h^* gefiltert und anschließend addiert:

$$a_{i-1} = H^* a_i + G^* d_i \ .$$

Der Vektor $\mathbf{W}$ geht somit durch die Rekonstruktion (Reverse FWT, RFWT) wieder in den Originalvektor a über:

$$\mathbf{W} \xrightarrow{RFWT} \mathbf{a},$$

die Schnelle Wavelet-Transformation ist also *informationserhaltend.*

Die Zerlegung und Rekonstruktion eines Datenvektors durch die FWT ist in den Abbildungen 7 und 8 dargestellt.

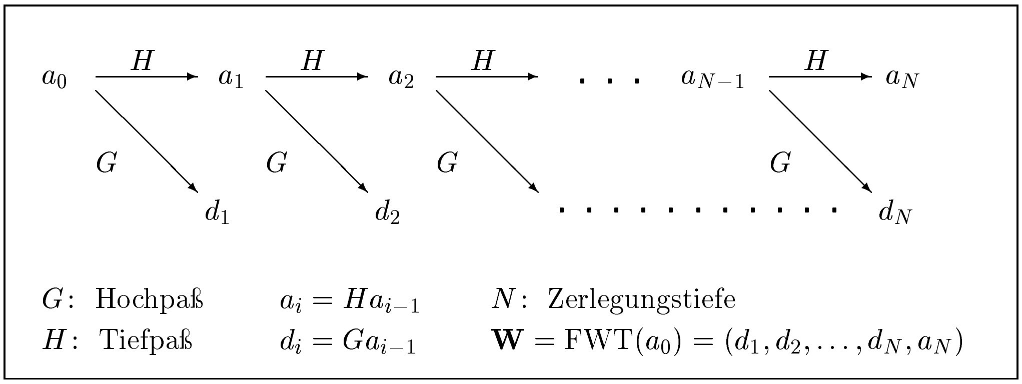

Abbildung 7: Zerlegung durch die FWT

$$
\begin{tikzcd}
a_N \arrow[r, "H^*"] & a_{N-1} \arrow[r, "H^*"] & a_{N-2} \arrow[r, "H^*"] & \cdots & a_1 \arrow[r, "H^*"] & a_0 \\
d_N \arrow[ur, "G^*"] & d_{N-1} \arrow[ur, "G^*"] & d_{N-2} \arrow[ur, "G^*"] & \cdots\cdots & d_1 \arrow[ur, "G^*"] &
\end{tikzcd}
$$

G^*: adj. Hochpaß $\qquad a_{i-1} = H^* a_i + G^* d_i$

H^*: adj. Tiefpaß $\qquad a_N, d_i \in \mathbf{W}$

Abbildung 8: Rekonstruktion

Der Ablauf der FWT soll in folgendem Beispiel nochmals verdeutlicht werden.

Beispiel 2 *Der einfachste mögliche Waveletfilter ist der Haarfilter, der nur aus 2 Komponenten besteht:*

$$h = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right), \quad g = \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right).$$

Der Vektor der Ausgangsdaten sei $a = a_0 = (1, -2,\ 3, -4)$.

Die Anwendung des Filters h *auf* a_0 *ergibt:*

$$\begin{aligned} a_1 &= \left(1 \cdot \frac{1}{\sqrt{2}} - 2 \cdot \frac{1}{\sqrt{2}}, \quad 3 \cdot \frac{1}{\sqrt{2}} - 4 \cdot \frac{1}{\sqrt{2}}\right) \\ &= \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right). \end{aligned}$$

Das Downsampling wird bereits erreicht, indem der Filter h *nicht um eine Position, sondern gleich um zwei Positionen weitergeschoben wird.*

In Abb. 9 ist der komplette Ablauf der Zerlegung dargestellt.

Da die FWT im wesentlichen auf der wiederholten Berechnung von Skalarprodukten zur Durchführung der diskreten Faltung beruht, liegt die Komplexität der FWT bei Transformation eines eindimensionalen Datenvektors der Länge 2^N in der Größenordnung von $\mathcal{O}(N)$, im Gegensatz zu den $\mathcal{O}(N \log N)$ Operationen, die bei der FFT benötigt werden [LMR94].

4 Anwendung der FWT auf 2D-Signale (Bilder)

Um die FWT zur Bildkompression nutzen zu können, muß diese zunächst auf den zweidimensionalen Fall verallgemeinert werden. Da in der Praxis meist nur seperable Wavelets

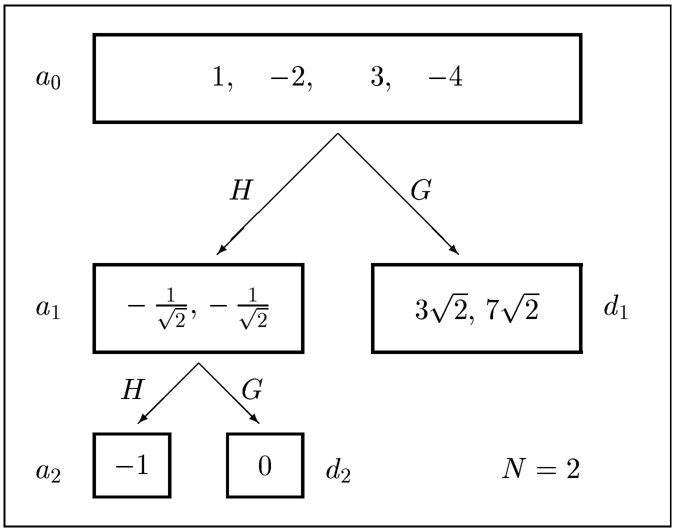

Abbildung 9: FWT-Zerlegung mit Haarfilter

verwendet werden, ist dazu lediglich eine Hintereinanderausführung von zwei eindimensionalen Wavelet-Transformationen notwendig. Dabei erfolgt zuerst eine Filterung der Bildzeilen, anschließend werden die Bildspalten der dabei entstehenden Zwischenbilder gefiltert. Als Ergebnis entstehen 4 Teilbilder oder Teilbänder, welche die tieffrequenten und hochfrequenten Frequenzanteile enthalten. Das zweimal tiefpaßgefilterte Teilband f_{LL} wird dann wieder weiterzerlegt, analog wie im eindimensionalen Fall in Abschnitt 3.3. Ein solcher Zerlegungsschritt ist in Abb. 10 demonstriert.

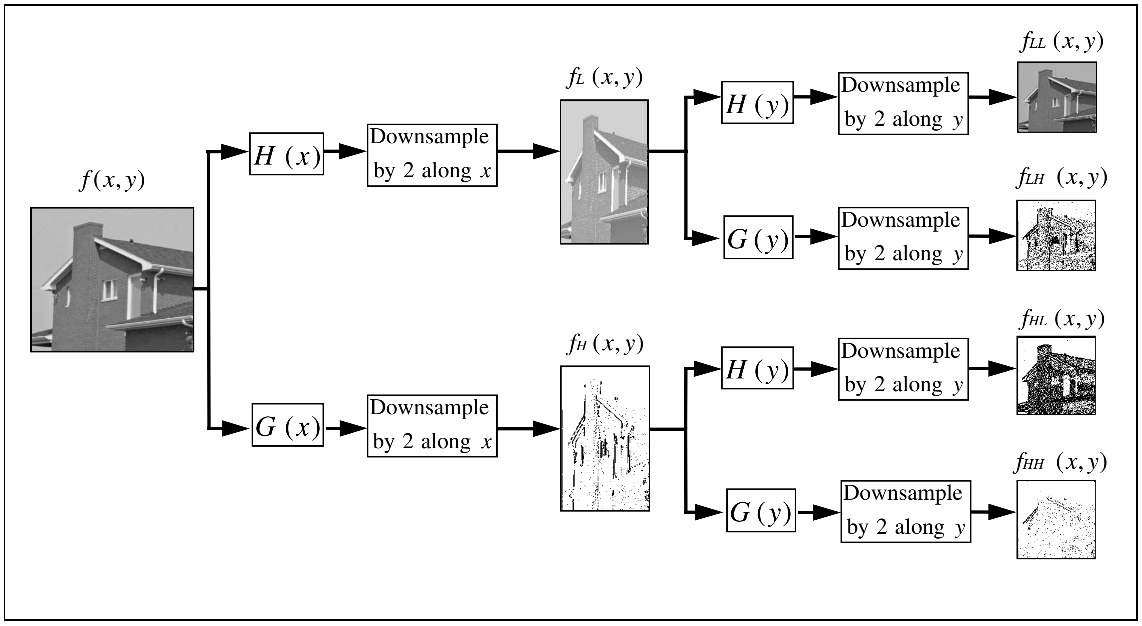

Abbildung 10: Ein Zerlegungsschritt der zweidimensionalen Wavelet-Transformation [JS93]

Nach Durchführung aller N Schritte (die Kantenlänge des Bildes sei wieder 2^N) liegt ein Baum von Teilbändern vor, aus dem sich das Originalbild ohne Verlust (abgesehen von Rundungsfehlern) wieder rekonstruieren läßt, siehe Abb. 11.

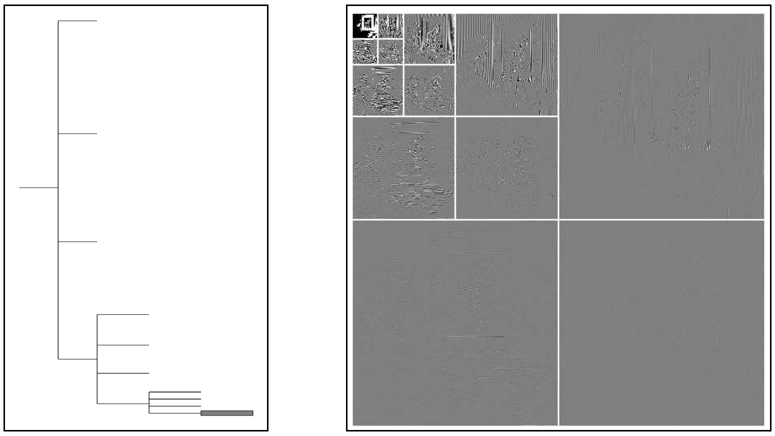

Abbildung 11: Standard Wavelet Zerlegung

4.1 Informationsverlust durch Abschneiden

Auf Grundlage der Eigenschaft der Energiekonzentrierung und der Eigenschaft (7) kann durch einfaches Abschneiden von betragsmäßig kleinen Waveletkoeffizienten x der erwünschte Informationsverlust zur Erhöhung der Kompressionsrate erreicht werden, vgl. Abb. 12:

$$\texttt{if } (|x| \leq cut) \texttt{ then } x = 0 \;.$$

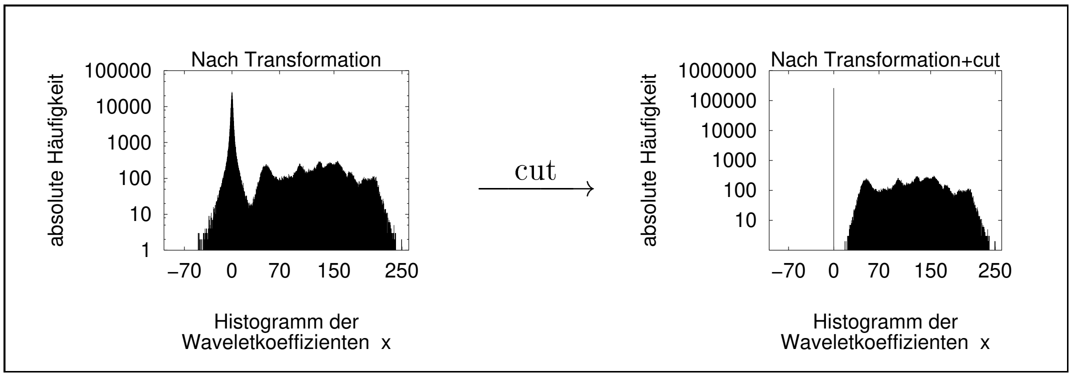

Abbildung 12: Abschneiden betragsmäßig kleiner Waveletkoeffizienten

Da sehr viele Koeffizienten durch dieses Abschneiden erfaßt werden, entstehen lange Folgen vieler Nullen, die z. B. durch *Run Length Encoding* (RLE) hervorragend komprimiert werden können.

Im tieffrequenten Teilband f_{LL} wird aufgrund der vergleichsweise gleichmäßigen Verteilung der Koeffizienten kein Abschneiden vorgenommen, da dieses die Bildqualität zu sehr beeinträchtigen und sich kaum auf die Kompressionsrate auswirken würde.
Die nach dem Abschneiden mit $cut = 70$ noch verbleibenden Koeffizienten stammen zum größten Teil aus dem Teilband f_{LL}, was in Abb. 12 rechts gut zu erkennen ist. In diesen Koeffizienten ist fast die gesamte Bildinformation konzentriert.
Ohne das Weglassen dieser Waveletkoeffizienten, die nur wenig Bildinformationen enthalten, würde man bei der anschließenden Quantisierung und Kodierung wesentlich schlechtere Ergebnisse für die Kompressionsrate erhalten.

Das Ergebnis des Abschneidens mit $cut = 70$ (ohne eine anschließende Quantisierung) ist in Abb. 13 dargestellt. Im Gegensatz zu einem JPEG-Bild sind die auftretenden Störungen nur lokaler Natur, wodurch die Blockstruktur des JPEG-Verfahrens vermieden werden kann.
Das wavelettransformierte Bild erreicht darüber hinaus auch eine größere Kompressionsrate als das JPEG-Verfahren, s. a. Abschnitt 7.

Originalbild	JPEG-Verfahren (PSNR=24.15)	WT nach $cut = 70$ (PSNR=24.11)

Abbildung 13: Informationsverlust durch Abschneiden von Waveletkoeffizienten und Vergleich mit dem JPEG-Verfahren

5 Verfahren zur Quantisierung

Anschließend an den ersten Schritt der Bildkompression, den Transformationsschritt, erfolgt als nächstes eine Quantisierung der entstandenen Transformationskoeffizienten. Der Quantisierungsschritt ermöglicht die verlustbehaftete Kompression von reellen Werten (auf einem Rechner liegen diese meist nur als Gleitkommazahlen vor und benötigten damit relativ viel Speicherplatz).

Unabhängig vom verwendeten Transformationsverfahren sind verschiedene Quantisierer entwickelt worden.
Die verwendeten Quantisierungsverfahren lassen sich dabei in zwei Hauptklassen unterteilen: Die skalare Quantisierung und die Vektorquantisierung.

Eine Vektorquantisierung erfordert einen wesentlich höheren Aufwand als die skalare Quantisierung und wird deswegen eher selten verwendet. Nähere Informationen zur Vektorquantisierung findet man in [CGV95].

5.1 Skalare Quantisierung

Bei diesem Verfahren wird jeder numerische Wert $x \in [\min, \max]$ einzeln quantisiert. Der Quantisierer ist dabei eine Treppenfunktion Q, s. a. Abb. 14:

$$Q(x) = r_i \text{ falls } x \in (d_i, d_{i+1}], \quad i = 0, 1, \ldots, N .$$

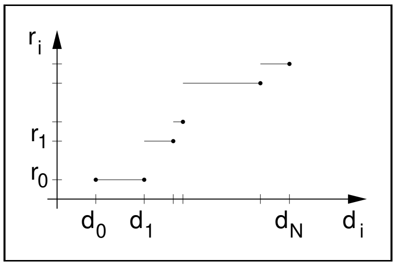

Abbildung 14: Quantisierungsfunktion Q

Die Rekonstruktionswerte r_i werden dabei als *reconstruction levels* bezeichnet und die Intervallgrenzen d_i als *decision levels.*
Durch den Quantisierer werden die numerischen Werte x auf ganzzahlige Indizes i abgebildet, welche bei der Dequantisierung dann auf Werte $\tilde{x} = r_i$ abgebildet werden, siehe Abb. 15:

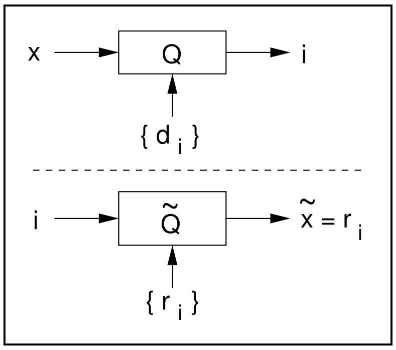

Abbildung 15: Skalarer Quantisierer

Folgende grundsätzliche Probleme ergeben sich bei Einsatz eines skalaren Quantisierers im Zusammenhang mit der Bildkompression:

- Die Vergrößerung der Anzahl N der Quantisierungsintervalle verbessert die Bildqualität, führt aber gleichzeitig zu einer Verschlechterung der Kompressionsrate.
- Die Berechnung von optimalen d_i und r_i bei vorgegebenem N erfordert die *gleichzeitige* Optimierung von Kompressionsrate und Bildqualität.

Erschwerend kommt dabei hinzu, daß die Kompressionsrate erst *nach* der Durchführung aller 3 Schritte des Kompressionsvorgangs (Transformation, Quantisierung, Kodierung) bekannt ist. Daher wird bei der Durchführung dieser Optimierung meist auf die Entropie als Approximation für die Kompressionsrate zurückgegriffen, um in akzeptabler Rechenzeit zu verwertbaren Ergebnissen zu gelangen.

In der Praxis verzichtet man oft auf die aufwendige Berechnung des optimalen Quantisierers, die aufgrund der Bildabhängigkeit der d_i und r_i für jedes Bild neu erfolgen müßte.
Es ist aber bereits mit relativ einfachen und schnellen Verfahren möglich, ausreichend gute Ergebnisse zu erzielen.

5.1.1 Uniforme Quantisierung

Die einfachste und oft genutzte Möglichkeit zur Quantisierung ist die Uniforme Quantisierung. Hierbei erfolgt eine äquidistante Einteilung des Intervalles $[\min, \max] = [d_0, d_N]$ in die N Quantisierungsintervalle $(d_i, d_{i+1}]$, vgl. auch Abb 16:

$$d_i = d_0 + i \cdot \frac{d_N - d_0}{N} \qquad \text{für } i = 0, \ldots, N$$

$$r_i = d_i + \frac{d_{i+1} - d_i}{2} = d_i + \frac{d_N - d_0}{2N} \qquad \text{für } i = 0, \ldots, N-1$$

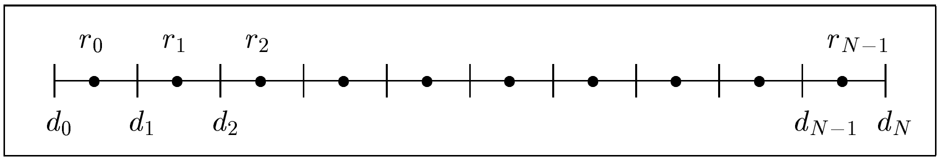

Abbildung 16: Uniforme Quantisierung

Diese Methode ist dann am besten geeignet, wenn die Waveletkoeffizienten relativ gleichmäßig über die x-Achse verteilt sind.
Vorteilhaft ist vor allem, daß keine längere Tabelle der r_i, $i = 0, 1, \ldots N$ abgespeichert werden muß, sondern nur die Werte von d_0, d_N und N.
Die nichtvorhandene Anpassung an die Verteilung der numerischen Werte x wirkt sich jedoch negativ auf die Bildqualität aus.

5.1.2 Uniforme Quantisierung mit Deadzone

Wenn man das Abschneiden der Waveletkoeffizienten in Abschnitt 4.1 berücksichtigt, kann man den Uniformen Quantisierer noch wesentlich verbessern, indem man eine sog. *Deadzone* $[-cut, cut]$ einführt, in der alle Transformationskoeffizienten auf den Wert 0 gesetzt werden. Der Abschneidevorgang wird damit durch den Quantisierungsschritt miterfaßt, siehe Abb 17.
Der Wert von cut muß noch zusätzlich mit abgespeichert werden.

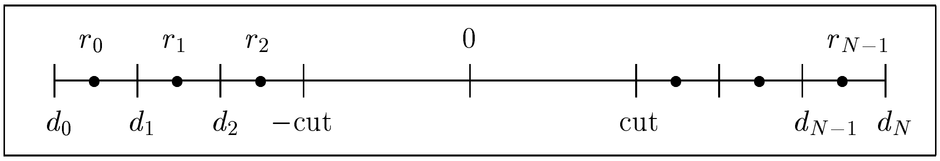

Abbildung 17: Uniforme Quantisierung mit *Deadzone*

Bei beiden Quantisierern ist es notwendig, das letzte, tieffrequente Teilband f^z_{LL} getrennt zu quantisieren, da die Transformationskoeffizienten dort eine relativ gleichmäßige Verteilung aufweisen (z sei die Anzahl der durchgeführten Zerlegungen während der Wavelet-Transformation).
Geeignet ist beispielsweise eine uniforme Quantisierung mit $N = 256$ Intervallen [SM95].

5.2 Lloyd-Max Quantisierer

Ein weiterer Quantisierer, der in der Literatur oft angesprochen wird, ist der Lloyd-Max Quantisierer [Max60, HJS95].

Der Lloyd-Max Quantisierer beruht auf dem Ansatz, die Intervallgrenzen d_i und die Rekonstruktionswerte r_i in Abhängigkeit von der Häufigkeitsverteilung $p(x)$ der Koeffizienten x in einem Intervall $(d_i, d_{i+1}]$ so zu bestimmen, daß der Rekonstruktions- oder Quantisierungsfehler minimal für eine feste, vorgegebene Anzahl von Quantisierungsintervallen N wird.
Der Einfluß auf die dadurch veränderte Entropie der Quantisierungsdaten wird dabei allerdings nicht berücksichtigt, was sich negativ auf das Verhältnis von Bildqualität und Kompressionsrate auswirkt.
Abhilfe schaffen hier die komplizierteren *Entropy Constrained* Quantisierer (Entropiebegrenzende Quantisierer), welche durch nichtlineare Optimierungsmethoden den Rekonstruktionsfehler für eine fest vorgegebene Entropie minimieren [Kar97].

Wenn die benutzte Fehlermetrik $D(x, y) = (x - y)^2$ ist, kann der Rekonstruktionsfehler

$$E(r_i) = \sum_{i=0}^{N-1} \left(\sum_{d_i < x \leq d_{i+1}} p(x)\, D(x, r_i) \right) = \sum_{i=0}^{N-1} \sum_{d_i < x \leq d_{i+1}} p(x)\, (x - r_i)^2$$

berechnet werden.

Um den Rekonstruktionsfehler für eine vorgegebene Verteilung der Waveletkoeffizienten $p(x)$ zu minimieren, müssen die r_i so gewählt werden, daß folgende Funktion minimal wird:

$$f(r_i) = \sum_{d_i < x \leq d_{i+1}} p(x)\, (x - r_i)^2 \, .$$

Mit der 1. Ableitung als notwendiger Bedingung ergibt sich:

$$\frac{df}{dr_i} = \sum_{d_i<x\le d_{i+1}} -2p(x)(x-r_i)$$

$$= 2\sum_{d_i<x\le d_{i+1}} -p(x)x + 2r_i \sum_{d_i<x\le d_{i+1}} p(x)$$

$$\frac{df}{dr_i} = 0 \quad \Longleftrightarrow \quad r_i = \frac{\sum\limits_{d_i<x\le d_{i+1}} p(x)x}{\sum\limits_{d_i<x\le d_{i+1}} p(x)}\,. \tag{8}$$

Die r_i bilden damit den Schwerpunkt im Intervall $(d_i, d_{i+1}]$.

Da jeder Koeffizient auf den nächstliegenden Rekonstruktionswert quantisiert werden sollte, ergibt sich:

$$d_i = \frac{r_{i-1}+r_i}{2}, \qquad d_0 = \min, \quad d_N = \max\,. \tag{9}$$

Die optimale Lage aller d_i und r_i ist erst dann gegeben, wenn die beiden Bedingungen (8) und (9) gleichzeitig erfüllt werden.
Zu diesem Zweck wird folgendes iteratives Verfahren durchgeführt:

1. Man startet den Algorithmus mit beliebigen Startwerten für die Intervallgrenzen d_i, z. B. indem die d_i zunächst äquidistant verteilt werden.
2. Aus den d_i werden mit (8) neue r_i berechnet.
3. Aus diesen r_i werden mit (9) neue d_i berechnet.
4. Weiter bei 2, bis ein vorgegebenes Konvergenzkriterium erfüllt wird.

Bei jedem Iterationsschritt wird der Rekonstruktionsfehler $E(r_i)$ weiter verringert. Man kann das Verfahren abbrechen, wenn z. B. eine vorgegebene Anzahl von Iterationsschritten erreicht wurde oder wenn die Änderung $\Delta d = |d_{i+1} - d_i|$ der d_i und $\Delta r = |r_{i+1} - r_i|$ der r_i ein bestimmtes $\epsilon > 0$ erreicht (bei Konvergenz ist $\Delta d = \Delta r = 0$).
Ein Beispiel für das Ergebnis dieses Verfahrens ist in Abb. 18 zu sehen.

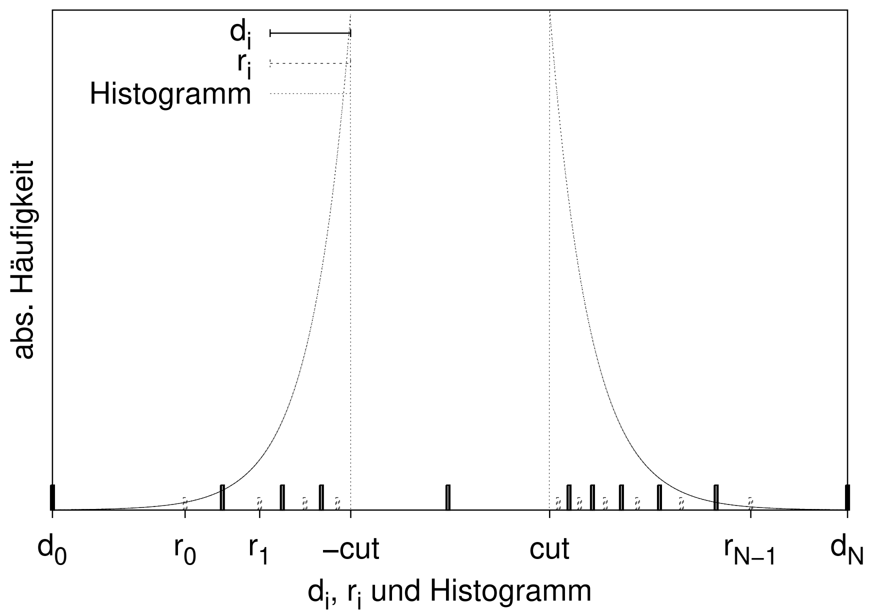

Abbildung 18: Beispiel eines Lloyd-Max Quantisierers

6 Verfahren zur Kodierung

Im Anschluß an die Quantisierung erfolgt durch eine binäre Kodierung der ganzzahligen Werte der Quantisierungsinformationen die eigentliche Kompression. Die Quantisierungsinformationen werden dabei verlustfrei mit möglichst wenigen Bits gespeichert. In der Regel wird dazu der *Huffman-Kodierer* oder ein vergleichbarer Algorithmus eingesetzt [PTVF92]. Vorher werden die langen Folgen von Nullen (engl. *runs*), die während des Abschneidevorgangs in Abschnitt 4.1 entstanden sind, mittels Run Length Encoding gepackt. Hauptsächlich dieser Schritt trägt damit zu der hohen Kompressionsrate bei.
Der Huffman-Kodierer kann zwar leicht implementiert werden, aber die Codesymbole werden dabei jeweils mit 1 oder mehreren Bits gespeichert. Bei der *Arithmetischen Kodierung* hingegen werden die Codesymbole mit Bruchteilen von Bits abgespeichert, wodurch beim Kodieren der Ausgangsdaten deren (zero-Order) Entropie erreicht werden kann [WNC87].

7 Leistungsvergleich

Neben wavelet-basierten Verfahren stehen insbesondere die fraktalen und die DCT-basierten Verfahren zur Bildkompression zur Verfügung.
Um die Leistungsfähigkeit der aktuellen Implementierungen dieser Verfahren zu vergleichen, wurden folgende typische Vertreter bzw. Programme herangezogen:

- Das Programm Wave (Version 0.3) von der Waveletgruppe des Dartmouth College [War98], Binaries: `encode/decode`.

- Das Programm `enc` als Referenzimplementation des fraktalen Verfahrens von Fisher [FBB+95], Binaries: `enc/dec`.

- Das Programm `cjpeg` (Version 6) als Referenzimplementation des JPEG-Verfahrens der *Independent JPEG Group* [Wal91], Binaries: `cjpeg/djpeg`.

Alle drei Verfahren wurden auf das Testbild `lena512`[6] angewendet. Dabei wurden konkret folgende Parameter benutzt:

Verfahren	Aufruf bzw. Parameter
JPEG	`$ cjpeg -grayscale -optimize -quality` Q `lena512.pgm`, wobei $Q = \{1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 50, 60, 70, 80, 85, 89, 94, 96\}$
Wave	`$ encode lena512.pgm lena512.wave` *ratio*, wobei $ratio = \{16, 20, 26, 34, 45, 60, 83, 115, 150, 200, 300, 400, 500, 700\}$
Fisher	`$ enc -D 2 -d` d `-t` t `-f -w 512 -h 512 lena512.raw lena512.trn`, wobei für d und t folgende Parameterwerte benutzt wurden: $d = 4,\ t = 17$ $d = 4,\ t = 16$ $d = 2,\ t = 16$ $d = 4,\ t = 15$ $d = 2,\ t = 15$ $d = 2,\ t = 14$ $d = 4,\ t = 13$ $d = 4,\ t = 11$ $d = 2,\ t = 11$ $d = 2,\ t = 10$ $d = 2,\ t = 9$ $d = 2,\ t = 8$ $d = 2,\ t = 7$ $d = 2,\ t = 6$ $d = 2,\ t = 5$ $d = 2,\ t = 3$ $d = 2,\ t = 1$

Das Ergebnis in Abb. 19 macht deutlich, daß das JPEG-Verfahren vor allem bei großen Kompressionsraten hinter den moderneren Verfahren zurückfällt. Die Differenz in der erreichten Bildqualität erstreckt sich (bei diesem Testbild) von 1.5 dB bis über 6 dB.

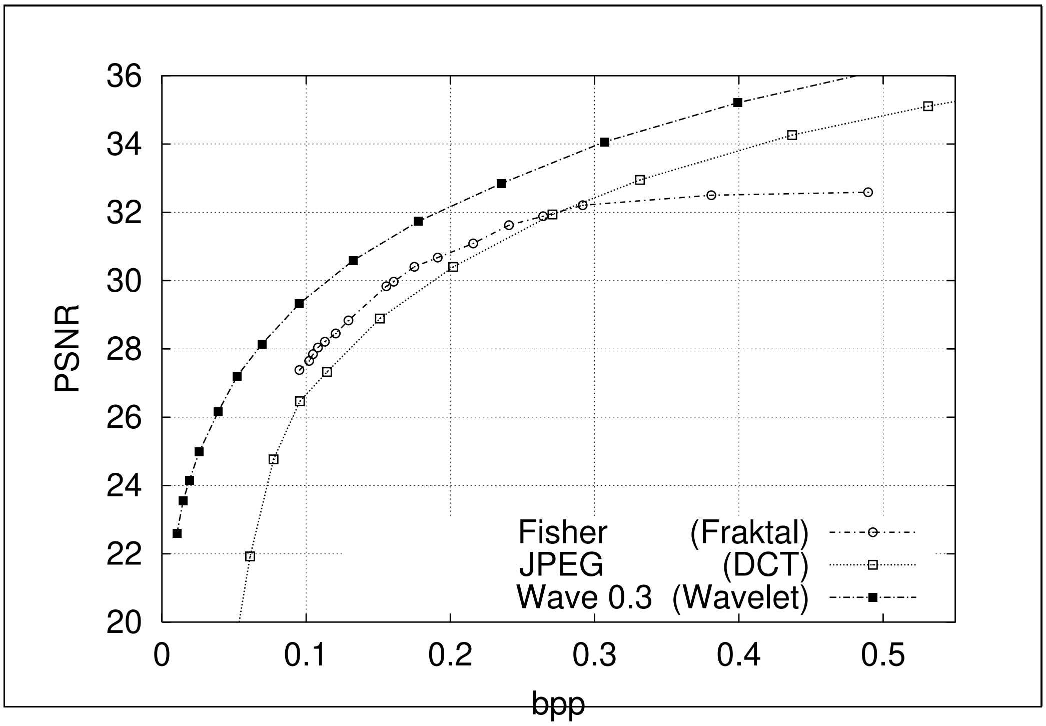

Abbildung 19: Numerischer Vergleich der drei Bildkompressionsverfahren

[6] Dies ist das selbe Bild wie in Abb. 1, hier allerdings in der Größe 512×512 Bildpunkte.

Mit dem Verfahren von Fisher erhält man zwar prinzipiell bessere Ergebnisse als beim JPEG-Verfahren, allerdings nicht bei kleinen Kompressionsraten. Aufgrund der intensiven Forschung an den fraktalen Kompressionsverfahren können sich diese jedoch zu einer ernsthaften Konkurrenz für die Wavelet-Verfahren entwickeln. Erste Anhaltspunkte dafür findet man z. B. in [CSW97].

Um einen optischen Vergleich der oben genannten Verfahren zu ermöglichen, wurde das Testbild `lena512` mit jedem Verfahren mit einer einheitlichen Kompressionsrate von 1 : 84 bzw. 0.095 bbp komprimiert und anschließend die PSNR als Maß für die Bildqualität ermittelt, wobei folgende Parametereinstellungen benutzt wurden:

Verfahren	Aufruf bzw. Parameter
JPEG	`$ cjpeg -grayscale -optimize -quality 4 lena512.pgm`
Wave	`$ encode lena512.pgm lena512.wave 83`
Fisher	`$ enc -D 2 -d 4 -f -t 17 -w 512 -h 512 lena512.raw lena512.trn`

In Abb. 20 kann man sehr gut erkennen, wie sehr sich die erreichte Bildqualität voneinander unterscheidet.

Abbildung 20: Optischer Vergleich der drei Bildkompressionsverfahren

Ein weiterer Aspekt dieser Kompressionsverfahren besteht in der benötigten Rechenzeit. Die genannten Programme wurden unter dem Betriebssystem Solaris 2.5.1 übersetzt und auf einer Workstation vom Typ *SUN Ultra1, Model 140* ausgeführt.
Dabei wurden folgende Rechenzeiten bei der Kompression des Testbildes `lena512.pgm` mittels des Unixbefehls `time` festgestellt (als Meßwert wurde die *Usertime* benutzt):

Programm	benötigte Rechenzeit
`cjpeg`	0.17 sec (mit `-quality 1`) bis 0.34 sec (mit `-quality 96`)
`encode`	19.45 sec (mit $ratio = 700$) bis 24.56 sec (mit $ratio = 16$)
`enc`	923 sec (bei $d = 4$ und $t = 17$) bis 9293 sec (bei $d = 2$ und $t = 1$)

Die Zeiten für die Dekompression der gepackten Bilder liegt beim JPEG-Verfahren und dem Wavelet-Verfahren in der gleichen Größenordnung wie bei der Kompression, das fraktale Verfahren hingegen läuft bei der Dekompression wesentlich schneller ab:

Programm	benötigte Rechenzeit
`djpeg`	0.04 sec (`-quality 1`) bis 0.20 sec (`-quality 96`)
`decode`	3.86 sec (mit $ratio = 700$) bis 6.23 sec (mit $ratio = 16$)
`dec`	1.73 sec

Die kurzen Rechenzeiten beim JPEG-Verfahren und beim Wavelet-Verfahren ermöglichen die Verwendung dieser Kompressionsverfahren auch bei Echtzeitanforderungen. Die langen Rechenzeiten beim Kompressionsvorgang der fraktalen Verfahren machen deren Einsatz derzeit nur im Bereich der Bildarchivierung sinnvoll.

8 Zusammenfassung

Bildkompressionsverfahren lassen sich in zwei Klassen einteilen: verlustfreie und verlustbehaftete Verfahren.
Die verlustfreien Bildkompressionsverfahren arbeiten unter der Rahmenbedingung, daß durch ihre Anwendung keine Information verloren gehen darf und das Bild nach der Kompression und Dekompression wieder völlig identisch mit dem Originalbild sein muß.
Die verlustbehafteten Bildkompressionsverfahren unterliegen dieser Rahmenbedingung nicht. Sie sind daher in der Lage auszunutzen, daß sich mit oftmals nur geringem Informationsverlust sehr hohe Kompressionsraten erreichen lassen, wobei entscheidend ist, daß die Abhängigkeit der Kompressionsrate vom Informationsverlust nicht proportional ist.
Die Herausforderung der Bildkompressionsverfahren besteht also darin, einen guten Kompromiß zwischen Informationsverlust und Kompressionsrate zu erzielen.

Dieser Bericht behandelt die Grundlagen der Bildkompression sowie den prinzipiellen Ablauf von verlustbehafteten Bildkompressionsverfahren, insbesondere die Schritte Transformation und Quantisierung.
Im Vordergrund der Transformation steht die FWT, die man durch Diskretisierung der Wavelet-Transformation erhält. Die besonderen Eigenschaften der Wavelet-Transformation bewirken eine Konzentrierung der Bildinformation in wenigen Transformationskoeffizienten und ermöglichen dadurch eine vergleichsweise hohe Kompressionsrate.
Im Vordergrund der Quantisierung stehen die Verfahren der skalaren Quantisierung, deren Aufgabe darin besteht, Intervalle von reellen Waveletkoeffizienten auf einzelne ganze Zahlen abzubilden.

Im Rahmen unserer Untersuchungen haben wir drei aktuelle Bildkompressionsverfahren auf einer modernen Workstation miteinander verglichen. Betrachtet wurden ein wavelet-basiertes Verfahren, ein JPEG-Verfahren und ein Verfahren der Fraktalen Bildkompression. Berücksichtigt wurden die Kompressionsraten, die Rechenzeiten und die Informationsverluste bezüglich der Bildqualität.
Dieser Vergleich hat die prinzipielle Leistungsfähigkeit der wavelet-basierten Verfahren auf praktischer Ebene bestätigt.

Literatur

[BBH93] J. Bradley, C. Brislawn, and T. Hopper. The FBI Wavelet/Scalar Quantization Standard for Gray-scale Fingerprint Image Compression. Technical report, Los Alamos Nat'l Lab, 1993. Tech. Report LA-UR-93-1659.

[Bri96] Chris Brislawn. The FBI Fingerprint Image Compression Standard, 1996. URL: `http://www.c3.lanl.gov/~brislawn/FBI/FBI.html`.

[CGV95] P. C. Cosman, R. M. Gray, and M. Vetterli. Vector Quantization of Image Subbands: A Survey. Technical report, June 1995.

[Chu92] C. K. Chui. *An Introduction to Wavelets.* Academic Press, San Diego, CA, 1992.

[Com97] JPEG Committee. Call for contributions for JPEG 2000: Image Coding System, March 1997. URL: `http://www.disc.org.uk/public/jpegnew.htm`.

[Cri97] Dale Cripps. The HDTV Newsletter, 1997. URL: `http://web-star.com/hdtv/hdtvnews1.html`.

[CSW97] Yung-Ching Chang, Bin-Kai Shyu, and Jia-Shung Wang. Region-based fractal image compression with quadtree segmentation. In *Proc. ICASSP'97*, Munich, 1997.

[Dup95] Michel Dupagne. High-Definition Television. In August E. Grant, editor, *Communication Technology Update*, pages 101–117. Focal Press Boston, 1995.

[FBB+95] Yuval Fisher, Izhak Baharav, Ben Bielefeld, Roger D. Boss, Karel Culik II, Frank Dudbridge, Bill Jacobs, Jarkko Kari, Ehud D. Karnin, Skjalg Lepsøy, Lars M. Lundheim, David Malah, Spencer Menlove, Geir Egil Øien, Dietmar Saupe, and Greg Vines. *Fractal Image Compression.* Springer, 1995.

[Fou95] Alain Fournier. Wavelets and their applications in computer graphics. SIGGRAPH '95, Course Notes, 1995. URL: `http://www.cs.ubc.ca/nest/imager/contributions/bobl/wvlt/download/notes.ps.Z.saveme`.

[Gra95] Amara Graps. An introduction to Wavelets. *IEEE Computational Science and Engineering*, 2(2):50–61, 1995. URL: `http://www.amara.com/IEEEwave/IEEEwavelet.html`.

[HJS95] M. L. Hilton, B. D. Jawerth, and A. Sengupta. Compressing still and moving images with wavelets. *Multimedia Systems*, 3(2), 1995. URL: `ftp://ftp.math.scarolina.edu/pub/wavelets/papers/varia/tutorial`.

[How93] Paul G. Howard. *The Design and Analysis of Efficient Lossless Data Compression Systems.* PhD thesis, Brown University, Department of Computer Science, June 1993. CS-93-28.

[HQ83] Werner Heise und Pasquale Quattrocchi. *Informations- und Kodierungstheorie.* Springer-Verlag Berlin Heidelberg, 1983.

[Inc87] CompuServe Incorporated. Graphics Interchange Format GIF87a, 1987. URL: `http://www.mediatel.lu/workshop/graphic/2D_fileformat/h_gif87a.html`.

[JS93] Bjorn Jawerth and Wim Sweldens. An overview of wavelet based multiresolution analyses. Technical report, University of South Carolina, 1993. URL: `ftp://maxwell.math.scarolina.edu/pub/wavelet/papers/varia/sirev-36-3.tex`.

[Kar97] Jarkko Kari. Scalar Quantization. 1997. URL: `http://www.cs.uiowa.edu/~jjkari`.

[LMR94] Alfred Karl Louis, Peter Maß und Andreas Rieder. *Wavelets.* Teubner Studienbücher: Mathematik. B. G. Teubner Stuttgart, 1994.

[Lu97] Ning Lu. *Fractal Imaging.* Academic Press, 1997.

[Max60] J. Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, (IT-6(1)):7–12, 1960.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C: the art of scientific computing.* Cambridge University Press, second edition, 1992. URL: `http://cfatab.harvard.edu/nr/bookc.html`.

[Sha93] Jerome M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Trans. SP*, 41:3445–3462, 1993.

[SM95] Tilo Strutz und Erika Müller. Bilddatenkompression mit Wavelet-Transformation. In Bernhard Lange, Hrsg., *8. Symposium Maritime Elektronik*, Seiten 75–78, Universität Rostock, 1995. URL: `http://www-nt.e-technik.uni-rostock.de/~ts/Papers/`.

[SW49] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, 1949.

[SW76] Claude E. Shannon und Warren Weaver. *Mathematische Grundlagen der Informationstheorie*, Seiten 41–143. R. Oldenbourg Verlag München Wien, 1976. Reihe 'Scienta nova'.

[Wal91] Gregory K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[War98] Dartmouth College Wavelet Warriors. Wavelet Image Compression Construction Kit, May 1998. URL: `http://www.cs.dartmouth.edu/~gdavis/wavelet/wavelet.html`.

[Wel84] Terry A. Welch. A Technique for High Performance Data Compression. *IEEE Computer*, 17(6):8–19, 1984.

[WNC87] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.