Fachbereich
Allgemeinwissenschaften
und
Informatik

GEORG-SIMON-OHM
FACHHOCHSCHULE
NÜRNBERG

DIPLOMARBEIT

Wavelet-Bildkompression

Entwurf, Realisierung und Vergleich von Methoden der Bildcodierung und -decodierung

Jochen Schmidt

G e o r g - S i m o n - O h m -
Fachhochschule Nürnberg
Fachbereich Allgemeinwissen-
schaften u n d Informatik

D i p l o m a r b e i t

Bearbeiter: Jochen Schmidt

Semester: IF 8 TK

Betreuende Dozenten: Prof. Dr. Rieckeheer
Prof. Dr. Eck

Ausgabetag: 07.11.1996 Abgabetag: 07.04.1997

Thema: Wavelet-Bildkompression
Entwurf, Realisierung und Vergleich von
Methoden der Bildcodierung und -decodierung

Zusammenfassung:

In dieser Diplomarbeit wird die Verwendung von Wavelets zur Bildkompression behandelt. Hierzu erfolgt zunächst eine Einführung in die Wavelet-Theorie, wo die mathematischen Grundlagen der Wavelet-Transformation erläutert werden. Besonderer Wert wurde auf die schnelle diskrete Wavelet-Transformation in Zusammenhang mit dem Algorithmus von Mallat gelegt.

Anschließend wird speziell auf die Wavelet-Bildkompression eingegangen, welche auch implementiert wurde. Das entstandene Programm wird im folgenden Kapitel beschrieben.

Es folgen einige Auswertungen, die mit Hilfe des erstellten Programms durchgeführt wurden. Es soll die Leistungsfähigkeit, aber auch die Grenzen, der Bildkompression mit Wavelets gezeigt werden. Hier ist auch ein kurzer Vergleich mit JPEG enthalten.

Vorwort

Verfahren zur Bildkompression werden heute immer wichtiger. Dies liegt nicht zuletzt an der stetig zunehmenden Verbreitung von Netzen (Internet) und dem Aufkommen des digitalen Fernsehens. Hierbei müssen oft große Datenmengen, insbesondere auch Bilder, übertragen werden, wobei die verfügbare Bandbreite begrenzt ist. Aber auch die platzsparende Speicherung von Bildmaterial auf Datenträgern darf nicht vergessen werden.

Für diese Zwecke existieren zwar bereits standardisierte Verfahren, wie JPEG oder MPEG, die aber einige grundlegende Mängel aufweisen. Die Bildkompression mit Wavelets ist eine Alternative zu den gängigen Verfahren, die allerdings sehr neu ist, weshalb auch noch kein normiertes Dateiformat zur Speicherung von Bildern, geschweige denn von Filmen, verfügbar ist. Es zeichnet sich aber eine zunehmende Verwendung von Wavelets zur Kompression von Bildern ab, daher ist es wohl nur eine Frage der Zeit, bis ein Standard verabschiedet wird.

Ich möchte den Leser aber darauf hinweisen, daß die Bildkompression nur ein kleiner Ausschnitt aus den vielfältigen Möglichkeiten ist, die sich durch die Anwendung von Wavelets ergeben. Auf einige andere Anwendungen wird im Verlauf der Arbeit hingewiesen.

Mein besonderer Dank gilt Herrn Prof. Dr. Rieckeheer für die vielen anregenden Diskussionen, die mir beim Verfassen dieser Diplomarbeit sehr geholfen haben. Außerdem möchte ich mich bei Oliver Anders, Martin Groh, Markus Schmidt und Yvonne Voll für das Korrekturlesen bedanken.

Diese Diplomarbeit wurde mit LaTeX unter Linux erstellt.

Nürnberg, im April 1997 Jochen Schmidt

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die Idee der Kompression von Bildern unter Verwendung der Wavelet-Transformation ist noch recht neu. Und obwohl das erste Wavelet bereits zu Beginn dieses Jahrhunderts (1910) von Haar entdeckt wurde[1] und in der sogenannten Haar-Transformation verwendet wird, wurde die Wavelet-Transformation in ihrer heute bekannten allgemeinen Form doch erst Anfang der 80er Jahre entwickelt. Seitdem hat das Interesse von Ingenieuren und Naturwissenschaftlern an der Wavelet-Transformation stetig zugenommen, was vor allem an den breit gefächerten Einsatzmöglichkeiten liegt. Diese reichen vom Lösen von partiellen Differentialgleichungen über die hier vorgestelle verlustbehaftete Kompression von Bildern bis hin zur Analyse und Nachbearbeitung von Bildern (Kantendetektion, Entfernung von Rauschen) bzw. generell der Signalverarbeitung.

Möglich wurde dies jedoch erst durch die Verfügbarkeit eines schnellen Algorithmus, welcher für die Durchführung der Transformation auf Computern essentiell ist. Dieser wird daher in dieser Arbeit auch eine zentrale Rolle spielen.

Bezüglich der Bildkompression muß angemerkt werden, daß noch kein standardisiertes Dateiformat zur Speicherung von mit Wavelets komprimierten Bildern existiert, wie es mit JPEG[2] für die Kompression mit Hilfe der diskreten Cosinus-Transformation[3] der Fall ist. Dennoch wird das Verfahren bereits von einigen Firmen vermarktet und das amerikanische FBI nutzt es zur Kompression von digitalisierten Fingerabdrücken.

1.1 Ziel dieser Diplomarbeit

Mit dieser Diplomarbeit ist folgende Zielsetzung verbunden:

[1] daher heute auch als Haar-Wavelet bezeichnet; damals gab es den Begriff Wavelet allerdings noch nicht

[2] **J**oint **P**hotographics **E**xperts **G**roup

[3] DCT

- Der Leser soll die theoretischen Grundlagen einer Wavelet-Transformation verstehen: Was ist ein Wavelet, wie wird eine Wavelet-Transformation durchgeführt, welche Eigenschaften hat sie?
- Ein weiterer Schwerpunkt liegt auf dem Verständnis der Funktionsweise des Algorithmus zur schnellen diskreten Wavelet-Transformation und dessen Durchführung in ein und zwei Dimensionen.
- Es wird ein Programm vorgestellt, in dem der beschriebene Algorithmus zur Kompression von Graustufenbildern implementiert wurde.
- Mit Hilfe dieses Programms soll die Leistungsfähigkeit der Wavelet-Kompression von Bildern bezüglich Kompressionsrate und Bildqualität erörtert werden.

1.2 Anmerkungen zum Aufbau

Zunächst möchte ich dem Leser einen kurzen Überblick über den Inhalt der folgenden Kapitel geben:

Kapitel 2 bildet die Basis der Arbeit. Hier werden die mathematischen Grundlagen der Wavelet-Transformation dargelegt, ausgehend von der Fourier-Transformation wird die kontinuierliche Wavelet-Transformation sowie die Wavelet-Reihe eingeführt. Gegen Ende des Kapitels werden einige Standardwavelets vorgestellt.

Kapitel 3 behandelt die diskrete Wavelet-Transformation. Hierbei wurde besonderer Wert auf eine Herleitung des Algorithmus zur schnellen Wavelet-Transformation, ausgehend von der Multiskalen-Analyse, gelegt. Im Anschluß an die Herleitung wird dieser Algorithmus zunächst für den eindimensionalen Fall vorgestellt. Es folgt der Übergang auf zwei Dimensionen, speziell im Zusammenhang mit der Transformation von Bildern. Das Kapitel endet mit einer kurzen Komplexitätsbetrachtung und der Erweiterung des Algorithmus auf sogenannte biorthogonale Wavelets.

Kapitel 4 Nachdem in den vorangegangenen Kapiteln die Grundlagen zur Wavelet-Transformation beschrieben wurden, wird nun zunächst auf die prinzipielle Vorgehensweise bei der Anwendung von Transformationen in der Bildkompression eingegangen. Im nächsten Abschnitt werden einige Eigenschaften vorgestellt, die Wavelets besitzen sollten, damit sie für die gewünschte Anwendung auch geeignet sind. Es schließt sich ein Kapitel an, in welchem über den Anwendungsbereich von Wavelets in der Praxis auch über die Bildkompression hinaus berichtet wird. Der letzte Teil des vierten Kapitels befaßt sich mit der Kompression von Farbbildern.

Kapitel 5 beschreibt ein Programm zur Wavelet-Bildkompression von Graustufenbildern mit orthogonalen Wavelets. Der erste Teil des Kapitels stellt eine Bedienungsanleitung dar, im zweiten dagegen wird auf einige Punkte bei der Implementierung eingegangen.

Kapitel 6 Hier sind Auswertungen enthalten, die mit Hilfe des in Kapitel 5 beschriebenen Programms durchgeführt wurden. Es wurde der Einfluß diverser Parameter auf die Kompressionsrate und Bildqualität untersucht. Abschließend folgt ein Abschnitt über die Leistungsfähigkeit der Wavelet-Bildkompression im Vergleich zu JPEG.

Kapitel 7 Dieses letzte Kapitel gibt einen kurzen Überblick über die Möglichkeiten, die noch zur Verbesserung des vorgestellten Verfahrens bleiben und die in dieser Arbeit nicht behandelt werden konnten.

Anhänge In den Anhängen befinden sich Abbildungen aller in dieser Arbeit verwendeten Bilder im Original, eine Beschreibung des vom erstellten Programm verwendeten Dateiformats zur Speicherung komprimierter Bilder sowie Tabellen mit Wavelets für die schnelle Wavelet-Transformation. Bei jeder Tabelle sind weiterhin die Graphen des Wavelets und der damit verbundenen Skalierungsfunktion abgebildet.

Vom Leser wird ein solides Grundwissen in Mathematik, insbesondere im Umgang mit der Integralrechnung, sowie Grundkenntnisse der Fourier-Transformation vorausgesetzt. Weiterhin sind Kenntnisse über die Vorgehensweise bei der Filterung von Bildern von Vorteil.

Kapitel 2

Wavelets – Eine Einführung

Dieses Kapitel soll eine Übersicht über die Wavelet-Transformation geben. Aufbauend auf der Fourier-Transformation wird zunächst die Funktionsweise der kontinuierlichen Wavelet-Transformation (manchmal auch als integrale Wavelet-Transformation bezeichnet) beschrieben. Es folgt die Herleitung der Wavelet-Reihe, auf welche die in Kapitel 3 erläuterte schnelle diskrete Wavelet-Transformation aufbaut.

Alle Betrachtungen beziehen sich der Einfachheit halber zunächst auf eindimensionale Signale. Im Anschluß daran wird eine Erweiterung auf zwei Dimensionen vorgestellt, wie sie für die Bildkompression benötigt wird. Diese bereitet aber keine prinzipiellen Probleme. Alle betrachteten Funktionen sollen reell sein, das heißt es wird nicht auf die Verwendung von komplexen Wavelets eingegangen; an einigen Stellen sind aber dennoch Hinweise enthalten, welche Änderungen in den jeweiligen Gleichungen dafür nötig wären.

Das hier Beschriebene findet man in der einen oder anderen Form ebenfalls in der einschlägigen Literatur. Hier sei zur Einführung auf *Digital Image Processing* von Castleman [2] sowie die Siggraph'95 Kursunterlagen *Wavelets and their Applications in Computer Graphics* [7] verwiesen. Ein Standardwerk von Daubechies ist *Ten Lectures on Wavelets* [4], wobei aber, genau wie in *An Introduction to Wavelets* [3] von Chui einiges an mathematischen Vorkenntnissen verlangt wird. Ein weiteres sehr gutes und ausführliches Buch ist *Wavelets and Subband Coding* [19]; aber auch hier ist die verwendete Mathematik recht anspruchsvoll.

Ich möchte außerdem darauf hinweisen, daß die Verwendung von Wavelets zur Bildkompression zwar praktisch überall wenigstens in groben Zügen beschrieben wird. Mit ist allerdings keine Literatur bekannt, die sich hauptsächlich und ausführlich (einige kurze Aufsätze sind schon vorhanden) mit diesem Thema beschäftigt, geschweige denn, daß die Vorgehensweise speziell in Bezug auf eine Implementierung genau beschrieben wird.

2.1 Von der Fourier- zur Wavelet-Transformation

Zu Beginn möchte ich kurz der Frage nachgehen, weshalb man anstelle der gerade in der Signalverarbeitung sehr verbreiteten Fourier-Transformation überhaupt eine andere Art von Transformation verwenden will. Zum besseren Verständnis soll an dieser Stelle auf einige Nachteile der Fourier-Transformation bei der Verarbeitung von Signalen, wie sie in der Praxis auftreten, eingegangen werden.

Die Fourier-Transformation verwendet als Basisfunktion für die Transformation die Funktion (mit j als imaginäre Einheit $j^2 = -1$)

$$g(t) = e^{jt} = \cos t + j \sin t \tag{2.1}$$

also eine sinusähnliche Funktion (*Welle*[1]). Hieraus ergibt sich eine orthonormale Basis $\{g_\omega\}$ des $L^2(0, 2\pi)$ mit $g_\omega(t) = g(\omega t)$, die durch Skalierung (Dilatation) der Basisfunktion $g(t)$ entsteht [3]. Man verändert also die Frequenz der Sinuswelle. Als Konsequenz ergibt sich, daß jede 2π-periodische Funktion $f(t)$ durch Überlagerung von Basisfunktionen mit verschiedenen Frequenzen aus der Menge $\{g_\omega\}$ erzeugt werden kann. Anzumerken ist, daß die Basisfunktionen $g_\omega(t) = e^{j\omega t}$ nicht zum Raum $L^2(\mathbb{R})$ gehören, auf welchem die Wavelet-Transformationen im allgemeinen operieren[2].

Hier die Formeln für die kontinuierliche Fourier-Transformation sowie ihre Inverse:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt \tag{2.2}$$

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t}\, d\omega \tag{2.3}$$

Welche Eigenschaften hat nun die Fourier-Transformation sowie deren Basisfunktion $g(t)$? Zunächst einmal fällt bei der Betrachtung der Basis $\{g_\omega\}$ auf, daß diese aus Sinus- und Cosinusfunktionen besteht. Betrachtet man den Graph einer Sinusfunktion, so sieht man, daß sich dieser auf der reellen Achse nach links und rechts bis ins Unendliche fortsetzt, er geht also nicht asymptotisch gegen null[3].

Daraus ergibt sich ein Problem, welches bei der Verarbeitung von Signalen nicht zu vernachlässigen ist: Nach der Fourier-Transformation eines Dirac-Stoßes, also eines im Ortsbereich sehr stark lokalisierten Signals, sieht man, daß sich dieser eine Impuls an irgendeiner Stelle im Ortsbereich auf das gesamte Spektrum

[1] engl. Wave

[2] siehe hierzu auch Definition 1 auf Seite 8

[3] es wird sich zeigen, daß ein Wavelet gerade diese Eigenschaft besitzt

auswirkt. Das bedeutet, ein im Ortsbereich stark lokales Signal ist im Frequenzbereich überhaupt nicht mehr lokal, sondern es breitet sich über alle Frequenzen hinweg mehr oder weniger stark aus. Ein Impuls beispielsweise muß nämlich durch Überlagerung von Sinusschwingungen aller möglichen Frequenzen erzeugt werden, da er keinerlei Ähnlichkeit mit einer der Basisfunktionen der Fourier-Transformation hat. Dies ist in Bildern an Stellen der Fall, an denen im Bild Kanten vorhanden sind, das heißt es tritt eine große Änderung im Ortsbereich auf, ähnlich einem Dirac-Stoß. Damit breitet sich eine Kante im Frequenzraum sehr weit aus. Man sähe es aber eigentlich lieber, wenn dies nicht so wäre; dann nämlich könnte man z. B. bei Anwendungen in der Mustererkennung versuchen, die Kanten mit geeigneten Algorithmen im Frequenzraum aufzuspüren.

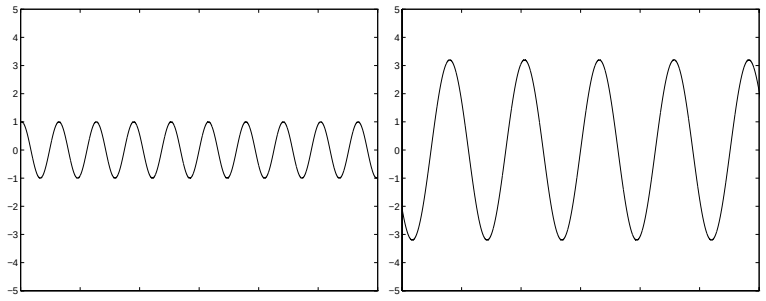
Ähnliches gilt auch für die Verarbeitung von Sprachsignalen (z. B. zur Erkennung von natürlicher Sprache). Es wäre doch wesentlich schöner, wenn sich ein plötzlich auftretendes Störgeräusch nicht im gesamten Frequenzraum ausbreiten würde, sondern vielmehr auch dort lokal aufträte; es könnte dann ohne größere Probleme entfernt werden.

Um diesen Nachteil der Fourier-Transformation zu umgehen, wurden schon bald verschiedene neue Ansätze entwickelt, teils in Form von Abwandlungen der Fourier-Transformation, aber auch in Form von neuen Transformationsarten. Als Abwandlungen der Fourier-Transformation sind die *gefensterte Fourier-Transformation*, die von Gabor[4] entwickelt wurde, sowie die *short-time Fourier-Transformation* (STFT) zu nennen. Diese Entwicklungen führten dann weiter hin zur Teilbandcodierung[5] von Signalen, ein Verfahren, welches in engem Zusammenhang mit der Berechnung der schnellen Wavelet-Transformation mit dem Algorithmus von Mallat steht (siehe Kapitel 3). Es zeigte sich aber, daß auch die abgewandelten Arten der Fourier-Transformation immer noch Nachteile hatten, die einfach durch die Wahl der Basisfunktionen entstehen. Man konnte das Problem mit der Lokalität zwar abschwächen, nicht jedoch völlig beseitigen. An dieser Stelle soll nun aber nicht näher auf diese Verfahren eingegangen werden; vielmehr möchte ich mich jetzt den Wavelets zuwenden.

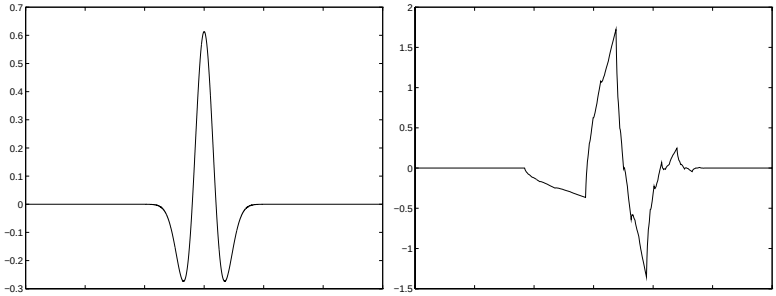
Die Wavelet-Transformation ist ein neues Verfahren, welches entwickelt wurde, um gerade den genannten Nachteil der Fourier-Transformation auszugleichen. Man verwendet als Basisfunktionen nicht mehr Wellen, sondern *Wavelets*[6]. Der Unterschied zwischen den beiden Arten von Funktionen ist in Abbildung 2.1 gezeigt.

Genau wie bei der Fourier-Transformation existieren verschiedene Transformationsarten; dies sind die kontinuierliche (oder integrale) Wavelet-Transfor-

[4] daher oft auch als *Gabor-Transformation* bezeichnet
[5] engl.: *subband coding*; siehe [19]
[6] engl. für *kleine Wellen*



(a) Wellenfunktionen



(b) Wavelets

Abbildung 2.1: Unterschied zwischen Wavelet-Funktionen und Wellenfunktionen

mation (CWT[7]), die Wavelet-Reihe[8] sowie die diskrete Wavelet-Transformation (DWT[9]). Auf diese drei Arten wird in den nächsten Kapiteln noch eingegangen. Zunächst möchte ich aber noch ein paar Worte über die Voraussetzungen verlieren, die zur Berechnung einer solchen Transformation nötig sind.

Die Funktionen, die transformiert werden, sollen quadratisch-integrierbar über den reellen Zahlen $\mathbb{R}$ sein, also $f(x) \in L^2(\mathbb{R})$ [3]:

[7] **C**ontinuous **W**avelet **T**ransform
[8] engl. Wavelet Series Expansion
[9] **D**iscrete **W**avelet **T**ransform

DEFINITION 1
Eine reelle Funktion $f(x)$ heißt quadratisch-integrierbar über den reellen Zahlen $\mathbb{R}$, wenn gilt

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx \; < \; \infty \tag{2.4}$$

man sagt dann $f(x) \in L^2(\mathbb{R})$.

Dies bedeutet also insbesondere, daß z. B. die Funktion $g(x) = \sin x$ nicht mit der hier vorliegenden Definition einer Wavelet-Transformation transformiert werden kann! Beschränkt man $g(x)$ hingegen auf ein (in diesem Fall beliebig großes, aber nicht unendlich großes) Intervall, so ist eine Transformation möglich.

Für den diskreten Fall bedeutet dies analog zu 2.4, daß $\{c_n\} \in l^2$, wenn gilt [3]:

DEFINITION 2
Eine Menge von diskreten Werten $\{c_n\}$ ($n \in \mathbb{Z}$) heißt quadratisch-summierbar, wenn gilt

$$\sum_{n=-\infty}^{\infty} |c_n|^2 \; < \; \infty \tag{2.5}$$

man sagt dann $\{c_n\} \in l^2$.

Für die Wavelet-Transformation wird eine Funktion $\psi(x)$ definiert, die als Basiswavelet bezeichnet wird. Durch Translation und Skalierung dieser Funktion wird eine Basis für die Transformation erzeugt. $\psi(x)$ ist eine Funktion, die für $|x| \to \infty$ sehr schnell null wird, d.h. es gilt

$$\psi(x) \in L^2(\mathbb{R}) \tag{2.6}$$

2.2 Die kontinuierliche Wavelet-Transformation

Die Funktion $\psi(x)$ wird als *Basiswavelet*[10] bezeichnet, wenn für ihre Fouriertransformierte $\Psi(s)$ die *Zulässigkeitsbedingung*[11] erfüllt ist:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(s)|^2}{|s|} \, ds \; < \; \infty \tag{2.7}$$

Damit diese Bedingung erfüllt ist, muß aufgrund des s im Nenner für $\Psi(0)$ gelten

$$\Psi(0) = 0 \Rightarrow \int_{-\infty}^{\infty} \psi(x) \, dx \; = \; 0 \tag{2.8}$$

[10] auch: *Mutter-Wavelet*
[11] engl.: admissibility condition

In der Praxis ist es meist hinreichend, wenn statt 2.7 nur das Kriterium 2.8 erfüllt ist [19, Seite 302].
Aus den Formeln 2.7, 2.8 sowie der Eigenschaft

$$\lim_{s\to\infty} \Psi(s) = 0 \tag{2.9}$$

ergibt sich, daß als Basiswavelet $\psi(x)$ die Impulsantwort eines jeden Bandpaßfilters verwendet werden kann, der die vorstehenden Bedingungen erfüllt: das Spektrum eines solchen Wavelets entspricht nämlich der Übertragungsfunktion eines Bandpasses [2, Seite 308ff.].

Die Basis-Funktionen für die Transformation werden nun durch Translation und Skalierung von $\psi(x)$ erzeugt [19, Seite 301ff.]:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{|a|}}\,\psi\left(\frac{x-b}{a}\right) \tag{2.10}$$

Hierbei sind a und b reelle Zahlen. Eine Änderung von b bewirkt eine Verschiebung des Wavelets entlang der x-Achse, während a der Skalierungsfaktor ist und die Breite des Wavelets beeinflußt. Die Translation entspricht dem Bewegen eines Fensters über die zu transformierende Funktion im Ortsbereich, ähnlich wie bei der gefensterten Fourier-Transformation oder STFT. Die Skalierung wird in diesem Zusammenhang oft auch als Dilatation bezeichnet und entspricht der Skalierung der Bandbreite eines Filters bzw. der Anpassung der Fenstergröße, was ja z. B. bei der STFT nicht möglich ist [7, Seite 12]. Der Faktor $\frac{1}{\sqrt{|a|}}$ dient der Energie-Normalisierung, das heißt er stellt sicher, daß die Normen der Basisfunktionen alle gleich sind. Diese ändert sich nämlich durch die Skalierung, wobei für die Berechnung der L^2-Norm einer Funktion gilt:

$$\left\|\psi\left(\frac{x-b}{a}\right)\right\| = \sqrt{\int_{-\infty}^{\infty}\left|\psi\left(\frac{x-b}{a}\right)\right|^2 dx} = \sqrt{a}\,\|\psi(x)\| \tag{2.11}$$

Das Basiswavelet $\psi(x)$ liegt dabei normalerweise bei $a = 1$ und $b = 0$.

Die kontinuierliche Wavelet-Transformation einer Funktion $f(x)$ ist dann definiert durch[12]

$$W_f(a,b) = \int_{-\infty}^{\infty} f(x)\,\psi_{a,b}(x)\,dx \;=\; \langle f, \psi_{a,b}\rangle \tag{2.12}$$

[12] werden statt reellen komplexe Wavelets verwendet, so ist in der Transformation $\psi_{a,b}(x)$ durch das konjugiert-komplexe Wavelet $\psi^*_{a,b}(x)$ zu ersetzen [19, Seite 302]

Die Transformationskoeffizienten $W_f(a,b)$ werden also als das Skalarprodukt der Funktion $f(x)$ mit jeder der Basisfunktionen $\psi_{a,b}(x)$ berechnet. Hierzu ist anzumerken, daß die kontinuierliche Wavelet-Transformation überbestimmt ist, das heißt es entstehen mehr Koeffizienten als für eine verlustfreie Durchführung der inversen Transformation nötig wären [2, Seite 310]. Dieses Problem[13] erledigt sich bei der diskreten schnellen Wavelet-Transformation.

Die inverse kontinuierliche Wavelet-Transformation lautet schließlich:

$$f(x) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(a,b)\, \psi_{a,b}(x)\, db \frac{da}{a^2} \tag{2.13}$$

Dabei ist C_ψ eine Konstante, die vom Basiswavelet $\psi(x)$ abhängig ist. Sie ergibt sich aus Formel 2.7.

Beschränkt man sich bei der Wahl des Skalierungsfaktors a auf positive reelle Zahlen, also $a \in \mathbb{R}^+$, so ergibt sich an Stelle von 2.13 eine vereinfachte Formel für die inverse Transformation:

$$f(x) = \frac{1}{C_\psi} \int_{0}^{\infty} \int_{-\infty}^{\infty} W_f(a,b)\, \psi_{a,b}(x)\, db \frac{da}{a^2} \tag{2.14}$$

Wobei sich die Konstante C_ψ aus einer abgewandelten Zulässigkeitsbedingung (anstelle von 2.7) ergibt:

$$C_\psi = \int_{0}^{\infty} \frac{|\Psi(s)|^2}{|s|}\, ds = \int_{-\infty}^{0} \frac{|\Psi(s)|^2}{|s|}\, ds \; < \; \infty \tag{2.15}$$

Hierbei gilt: Ist die Bedingung aus 2.7 erfüllt, so gilt auch 2.15.

2.2.1 Eigenschaften der Wavelet-Transformation

Dieses Kapitel beschreibt einige wichtige Eigenschaften der Wavelet-Transformation, wobei man wohl einiges von der Fourier-Transformation her wiedererkennen wird. Die Beschreibung dieser Eigenschaften erfolgt sinngemäß nach [19, Seite 304ff.]. Auch die Beweise finden sich dort.

[13] wir wollen ja letztendlich Daten (Bilder) komprimieren, wobei es ist nicht gerade hilfreich ist, wenn man nach der Transformation mehr Daten zur Verfügung hat als vorher; trotzdem kann auch hier eine bessere Kompression möglich sein, da oft sehr viele der entstehenden Koeffizienten null sein werden

2.2.1.1 Linearität

Die Linearität der kontinuierlichen Wavelet-Transformation ergibt sich direkt aus der Linearität des Skalarprodukts. Es gilt also für die Transformation der Funktion $f(x)$, welche sich aus der Linearkombination der beiden Funktionen $f_1(x)$ und $f_2(x)$ ergibt, sowie deren Wavelettransformierten $W_{f_1}(a,b)$ und $W_{f_2}(a,b)$ folgender Zusammenhang:

$$\begin{aligned} f(x) &= c_1 f_1(x) + c_2 f_2(x) \qquad \Leftrightarrow \\ W_f(a,b) &= c_1 W_{f_1}(a,b) + c_2 W_{f_2}(a,b) \qquad (c_1, c_2 \in \mathbb{C}) \end{aligned} \tag{2.16}$$

2.2.1.2 Verschiebungseigenschaft

Gegeben ist eine Funktion $f(x)$ sowie ihre Wavelettransformierte $W_f(a,b)$. Verschiebt man nun $f(x)$ auf der x-Achse um den Wert b', so ergibt sich die Funktion $\tilde{f}(x) = f(x-b')$. Die Transformierte von $\tilde{f}(x)$ ergibt sich dann zu:

$$W_{\tilde{f}}(a,b) = W_f(a, b-b') \tag{2.17}$$

2.2.1.3 Skalierungseigenschaft

Gegeben ist eine Funktion $f(x)$ sowie ihre Wavelettransformierte $W_f(a,b)$. Skaliert man nun $f(x)$ mit dem Faktor s, so ergibt sich die Funktion $\tilde{f}(x) = \frac{1}{\sqrt{|s|}} f\left(\frac{x}{s}\right)$; es wurde hierbei eine Energienormalisierung durchgeführt. Die Transformierte von $\tilde{f}(x)$ lautet dann:

$$W_{\tilde{f}}(a,b) = W_f\left(\frac{a}{s}, \frac{b}{s}\right) \tag{2.18}$$

2.2.1.4 Energieerhaltung

Für eine Funktion $f(x)$ und die zugehörige Wavelettransformierte $W_f(a,b)$ gilt die Energieerhaltung bei der Transformation ähnlich der Parsevalschen Formel für die Fourier-Transformation; die Konstante C_ψ ergibt sich aus Formel 2.7:

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |W_f(a,b)|^2 \, db \frac{da}{a^2} \tag{2.19}$$

2.2.1.5 Lokalitätseigenschaften

Wie bereits erwähnt, gibt es bei der kontinuierlichen Wavelet-Transformation eine Eigenschaft, die sie als Ersatz für die Fourier-Transformation sehr attraktiv macht. Diese Eigenschaft ist die sehr gute Lokalität, sowohl im Ortsbereich als

auch im Frequenzbereich, wobei verschiedene Wavelets auch verschieden gute Lokalitätseigenschaften besitzen. Das Haar-Wavelet beispielsweise besitzt eine exzellente Lokalität im Ortsbereich, während seine Lokalität im Frequenzbereich recht schlecht ist. Das Sinc-Wavelet zeigt genau umgekehrtes Verhalten; alle anderen liegen irgendwo zwischen diesen beiden Extremen. Näheres zu den genannten Wavelets folgt im Kapitel 2.4.

Gute Lokalität im Frequenzbereich bedeutet hierbei, daß sich eine bestimmte Stelle im Ortsbereich auf nur sehr wenige[14] Frequenzen auswirkt. Gute Lokalität im Ortsbereich bedeutet, eine Frequenz wirkt sich auf wenige Stellen im Ortsbereich aus. Mit Frequenzbereich ist dabei der Waveletraum gemeint, in den die Transformation erfolgt, nicht etwa der Frequenzbereich der Fourier-Transformation.

Nun wird zur Veranschaulichung des eben Gesagten die Transformation eines Dirac-Stoßes zum Zeitpunkt t_0 betrachtet:

$$W_\delta(a,b) = \int_{-\infty}^{\infty} \delta(t-t_0)\,\psi_{a,b}(t)\,dt \;=\; \psi_{a,b}(t_0) \tag{2.20}$$

Man kann sagen, das Wavelet paßt sich dem zu untersuchenden Signal an — bei hohen Frequenzen wird das Wavelet schmaler, bei niedrigen Frequenzen breiter[15].

2.2.2 Die zweidimensionale kontinuierliche Wavelet-Transformation

An dieser Stelle möchte ich die Erweiterung der kontinuierlichen Wavelet-Transformation auf zwei Dimensionen vorstellen. Diese ist eine recht geradlinige Fortführung der eindimensionalen Transformation [2, Seite 310], [6, Seite 454ff.].

Im Folgenden wird von einer Funktion $f(x,y) \in L^2(\mathbb{R}^2)$ ausgegangen. Die für die Transformation verwendete, natürlich ebenfalls zweidimensionale, Menge an Basisfunktionen ist definiert durch die Wavelets[16]:

$$\psi_{a,b_x,b_y}(x,y) = \frac{1}{a}\,\psi\left(\frac{x-b_x}{a}, \frac{y-b_y}{a}\right) \tag{2.21}$$

mit b_x und b_y als Verschiebungsfaktoren in x- bzw. y-Richtung und $a > 0$ als Skalierungsfaktor.

[14] im Idealfall auf eine einzige

[15] d.h. eine schmalere bzw. breitere Funktion der Basis $\{\psi_{a,b}(x)\}$ ist dem Signal „ähnlich“; dies wiederum bedeutet, daß nur bezüglich dieser ähnlichen Funktionen Transformationskoeffizienten entstehen, die wesentlich von null verschieden sind

[16] entsprechend Formel 2.10 im eindimensionalen Fall

Die zugehörige zweidimensionale kontinuierliche Wavelet-Transformation lautet dann[17]:

$$W_f(a, b_x, b_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, \psi_{a,b_x,b_y}(x, y)\, dxdy \tag{2.22}$$

Die inverse Transformation hierzu[18]:

$$f(x, y) = \frac{1}{C_\psi} \int_0^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(a, b_x, b_y)\, \psi_{a,b_x,b_y}(x, y)\, db_x db_y \frac{da}{a^3} \tag{2.23}$$

Die Erweiterung auf mehr als zwei Dimensionen ist ohne weitere Probleme möglich und wird in [6, Seite 454ff.] beschrieben.

2.3 Die Wavelet-Reihe

Die Wavelet-Reihe entspricht im Prinzip der kontinuierlichen Wavelet-Transformation, mit dem Unterschied, daß keine Skalierung bzw. Translation um reelle Zahlen mehr verwendet wird. Es ist jetzt vielmehr möglich, diese durch ganzzahlige Werte zu beschreiben. Die später in Kapitel 3 vorgestellte diskrete Wavelet-Transformation ist eine direkte Fortführung der Wavelet-Reihe. Näheres zur Wavelet-Reihe findet sich in der Literatur [3, Seite 4ff.], [2, Seite 312ff.].

Hier wird nur die Verwendung von sogenannten *dyadischen Wavelets* erläutert, da diese für die schnelle Wavelet-Transformation von Bedeutung sind. Ein dyadisches Wavelet entsteht durch die Verwendung von binären Skalierungsfaktoren $a = 2^{-j}$ sowie der gleichzeitigen Verschiebung des Wavelets um den Faktor $b = \frac{k}{2^j}$ (mit $j, k \in \mathbb{Z}$). Man erhält also als Verschiebungsfaktor ein ganzzahliges Vielfaches des Skalierungsfaktors.

Die Menge der Wavelets, die daraus entstehen, ist dann zunächst[19]:

$$\tilde{\psi}_{j,k}(x) = \psi\left(\frac{x - \frac{k}{2^j}}{2^{-j}}\right) = \psi(2^j x - k) \tag{2.24}$$

Auch bei diesen Wavelets soll eine Energienormalisierung durchgeführt werden; es ergibt sich dabei nach Formel 2.11:

$$\|\psi(2^j x - k)\| = \frac{1}{\sqrt{2^j}} \|\psi(x)\| \tag{2.25}$$

[17] entsprechend Formel 2.12
[18] entsprechend 2.14
[19] siehe auch [3]

Um die Normalisierung zu erreichen muß also jedes so skalierte und verschobene Wavelet aus Formel 2.24 mit dem Faktor $\sqrt{2^j}$ multipliziert werden.

Als Basisfunktionen für die Wavelet-Reihe werden demnach die folgenden Funktionen verwendet:

$$\psi_{j,k}(x) = \sqrt{2^j}\,\psi(2^j x - k) \tag{2.26}$$

Eine Funktion $\psi(x) \in L^2(\mathbb{R})$ wird als *orthogonales* (oder *orthonormales*[20]) Wavelet bezeichnet, wenn die Menge der Basisfunktionen $\{\psi_{j,k}(x)\}$ eine orthogonale Basis des $L^2(\mathbb{R})$ bilden.
Dies gilt, wenn (a)

$$\langle \psi_{j,k}, \psi_{l,m} \rangle \;=\; \delta_{j,l} \cdot \delta_{k,m} \qquad j,k,l,m \in \mathbb{Z} \tag{2.27}$$

und (b) eine Funktion $f(x) \in L^2(\mathbb{R})$ als *Wavelet-Reihe* entwickelt werden kann

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} c_{j,k}\,\psi_{j,k}(x) \tag{2.28}$$

Hierbei ist $\delta_{j,k}$ das Kronecker Symbol, welches definiert ist durch:

$$\delta_{j,k} := \begin{cases} 1 & \text{für} \qquad j = k, \\ 0 & \text{für} \qquad j \neq k \end{cases} \tag{2.29}$$

Analog zur Fourier-Reihe erhält man die Waveletkoeffizienten $c_{j,k}$ durch das Skalarprodukt

$$c_{j,k} = \langle f(x), \psi_{j,k}(x) \rangle = \sqrt{2^j} \int_{-\infty}^{\infty} f(x)\,\psi(2^j x - k)\,dx \tag{2.30}$$

Genau wie vorher die kontinuierliche Wavelet-Transformation, ist auch diese Transformation überbestimmt [2, Seite 313].

Nimmt man für $f(x)$ und $\psi(x)$ die Beschränkung vor, daß sie außerhalb eines gegebenen endlichen Intervalls[21] den Wert Null haben[22], so kann man statt der

[20] die Begriffe *orthogonal* und *orthonormal* werden in der Literatur meist gleichbedeutend verwendet (siehe [19]); an Stellen, an denen Wert darauf gelegt wird, daß die Norm gleich eins ist, wird explizit darauf hingewiesen

[21] hier wird in der Literatur zur Veranschaulichung oft das Intervall $[0, 1]$ verwendet (z. B. in [2])

[22] man sagt dann, das Wavelet hat einen *kompakten Träger* [20, Seite 194] (engl. *compact support*)

Indizes j und k einen einzigen Index n verwenden, der sich wie folgt ergibt [2, Seite 313f.]:

$$n = 2^j + k, \quad j = 0, 1, \ldots ; \; k = 0, 1, \ldots, 2^j - 1 \tag{2.31}$$

Für jedes n wird das zu verwendende j als größte ganze Zahl ermittelt, so daß gilt $2^j \leq n$; daraus errechnet sich dann das zugehörige k wie folgt: $k = n - 2^j$.

Bei dyadischen Wavelets mit kompaktem Träger erhält man so für jedes j eine bestimmte Anzahl an Wavelets, die gerade so groß ist, daß durch diese Wavelets das gesamte Intervall abgedeckt wird. Mit zunehmendem j werden die Wavelets immer schmaler, weshalb man auch immer mehr benötigt um das Intervall abzudecken, das heißt die Werte, welche k annehmen kann werden mehr. Diese Eigenschaft macht man sich beim Algorithmus von Mallat zunutze, der in Kapitel 3.5.1 über die (eindimensionale) diskrete Wavelet-Transformation beschrieben wird. Eine Illustration zum eben Gesagten befindet sich in Abbildung 2.2 sowie in [2, Seite 312ff.].

Mit dem neuen Index ergibt sich als Wavelet-Reihe für $f(x)$:

$$f(x) = \sum_{n=1}^{\infty} c_n \psi_n(x) \tag{2.32}$$

Die Koeffizienten c_n erhält man dann durch das Skalarprodukt

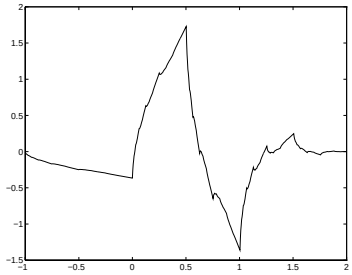
$$c_n = \langle f(x), \psi_n(x) \rangle = \sqrt{2^j} \int_{-\infty}^{\infty} f(x)\, \psi(2^j x - k)\, dx \tag{2.33}$$

wobei

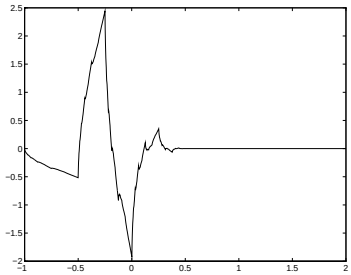
$$\psi_n(x) = \sqrt{2^j}\, \psi(2^j x - k) \tag{2.34}$$

mit den Formeln für j und k aus 2.31 berechnet werden.

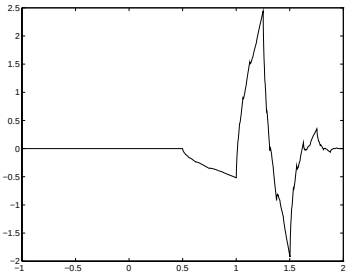
Als Beispiel für das einfachste orthonormale dyadische Wavelet mit kompaktem Träger soll an dieser Stelle das Haar-Wavelet genannt werden, welches in Kapitel 2.4.1 genauer betrachtet wird.



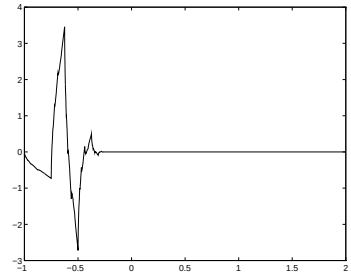
(a) j=1, k=0



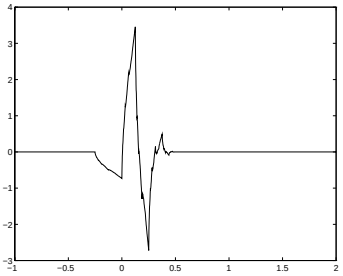
(b) j=2, k=0



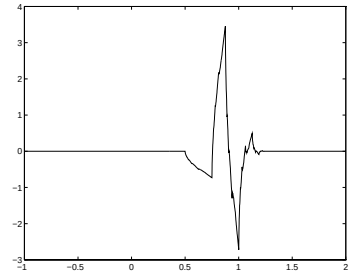
(c) j=2, k=1



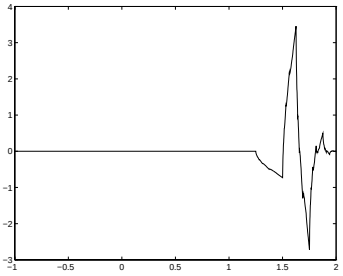
(d) j=3, k=0



(e) j=3, k=1



(f) j=3, k=2



(g) j=3, k=3

Abbildung 2.2: Dyadisches Wavelet mit kompaktem Träger bei verschiedenen Dilatationsfaktoren

2.4 Orthogonale Wavelets

In diesem Abschnitt möchte ich einige bekannte orthogonale Wavelets vorstellen, die in der Praxis immer wieder verwendet werden. Die meisten von ihnen besitzen einen kompakten Träger; diese Wavelets werden auch oft für die diskrete Wavelet-Transformation benutzt. Dasjenige, das keinen kompakten Träger hat, besitzt dafür andere erwähnenswerte Eigenschaften.

Den Anfang macht, schon aus historischen Gründen, das Haar-Wavelet. Es folgt eine kurze Beschreibung des zugehörigen dualen Wavelets, nämlich dem Sinc-Wavelet. Beide sind Extremfälle, wobei das eine einen kompakten Träger hat, das andere hingegen nicht. Auch die Daubechies-Wavelets dürfen bei einer solchen Betrachtung natürlich nicht fehlen.

Vorher noch eine Bemerkung zur Verwendung des Begriffs *Frequenzbereich*: Einige der folgenden Betrachtungen beziehen sich auf die Fouriertransformierte des jeweiligen Wavelets; dies ist meist dann der Fall, wenn von der Lokalitätseigenschaft eines Wavelets (im Orts- oder Frequenzbereich) die Rede ist. Die Lokalität eines Wavelets im Frequenzbereich (nach der Fourier-Transformation) hat direkte Auswirkungen auf die Lokalität der durch eine Wavelet-Transformation transformierten Signale im zugehörigen Waveletraum [19, Seite 206ff.].

Gute *Lokalität im Ortsbereich* bedeutet: eine Stelle im zu transformierenden Signal wirkt sich nur auf wenige (idealerweise einen einzigen) Transformationskoeffizienten aus. Je schneller also ein Basiswavelet gegen null geht (oder identisch null wird), desto besser ist die Lokalität im Ortsbereich.

Gute *Lokalität im Frequenzbereich* bedeutet: ein Transformationskoeffizient wirkt sich nur auf wenige (ideal: eine einzige) Stellen im Ortsbereich aus. Das bedeutet, je schneller die Fouriertransformierte eines Basiswavelets gegen null geht (oder identisch null wird), desto besser ist die Lokalität im Frequenzbereich.

Erwünscht ist nun natürlich sowohl eine sehr gute Lokalität im Orts- als auch im Frequenzbereich[23]; hierbei müssen aber Kompromisse geschlossen werden, wie man im Folgenden sehen wird.

2.4.1 Das Haar-Wavelet

Das Haar-Wavelet ist das einfachste orthonormale dyadische Wavelet. Es wurde bereits in der Haar-Transformation[24] verwendet und ist auch unter dem Namen Daubechies-2-Wavelet ($_1\psi$) bekannt[25]. Auch in der Literatur wird das Haar-Wavelet häufig als Beispiel verwendet [19, Seite 98ff., Seite 208ff.].

[23] die Fourier-Transformation beispielsweise ist überhaupt nicht lokal, das heißt eine Stelle im Ortsbereich wirkt sich auf alle Frequenzen mehr oder weniger stark aus

[24] die Haar-Transformation wird kurz erläutert in [2, Seite 292ff.]

[25] siehe hierzu Kapitel 2.4.3

Das Haar-Wavelet ist definiert durch

$$\psi(x) = \begin{cases} 1 & \text{für} \quad 0 \leq x < \frac{1}{2}, \\ -1 & \text{für} \quad \frac{1}{2} \leq x < 1, \\ 0 & \text{sonst} \end{cases} \tag{2.35}$$

Durch dyadische Skalierung und Translation entsteht die Menge der Basisfunktionen $\{\psi_{j,k}(x)\}$, wie in Formel 2.26 beschrieben. Der Graph von $\psi(x)$ ist in Abbildung 2.3 dargestellt.

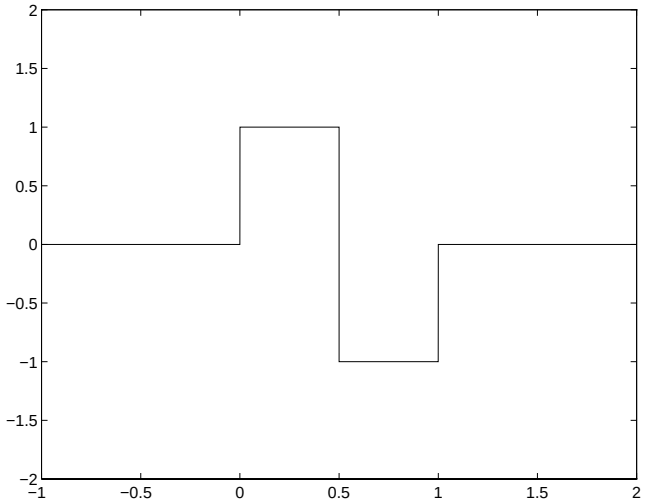


Abbildung 2.3: Das Haar-Wavelet

Das Haar-Wavelet hat im Ortsbereich einen kompakten Träger, da für $x \notin [0; 1]$ die Funktion identisch null ist. Betrachtet man das Amplitudenspektrum dieses Wavelets nach einer Fourier-Transformation[26], so stellt man fest, daß zwar $\lim_{\omega \to \infty} \Psi(\omega) = 0$ gilt, aber dieses Unendlichkeitsverhalten von $\Psi(\omega)$ entspricht dem der Funktion $\frac{1}{\omega}$, das heißt im Frequenzbereich hat das Wavelet keinen kompakten Träger und es geht (für ein Wavelet) relativ langsam gegen null.

Was kann man nun aus dieser Aussage schließen?

- Die Basisfunktionen des Haar-Wavelets werden im Ortsbereich für $j \to \infty$ immer schmaler, das heißt jede noch so große Änderung im Ortsbereich (beispielsweise ein Dirac-Stoß) kann beliebig genau mit den Basisfunktionen angenähert werden. Das Haar-Wavelet hat also eine sehr gute Lokalität im Ortsbereich. Bei zunehmender Auflösung im Ortsbereich geht auf der anderen Seite aber im Frequenzbereich Auflösung verloren.

[26] mit $\Psi(\omega)$ als Fouriertransformierte von $\psi(x)$

- Die Lokalität des Wavelets im Frequenzbereich ist dagegen nur sehr schlecht, da die Fouriertransformierte des Haar-Wavelets ein Unendlichkeitsverhalten wie $\frac{1}{\omega}$ hat.
- Für negative j mit einem sehr großen absoluten Wert ($j \to -\infty$), sind die Basisfunktionen im Ortsbereich sehr breit und verlieren dadurch an Auflösung, gewinnen aber dafür im Frequenzbereich an Auflösung hinzu.
- Die Basisfunktionen sind nicht glatt, da sie nicht einmal stetig sind [19, Seite 209].

Wie man in späteren Kapiteln sehen wird, ist das Haar-Wavelet ein extremes Wavelet: eine sehr gute Lokalität im Ortsbereich steht einer relativ schlechten Lokalität im Frequenzbereich gegenüber. Das Verhalten der anderen Wavelets (z. B. den Daubechies-Wavelets) liegt irgendwo zwischen dem Haar-Wavelet und dem sogenannten Sinc-Wavelet, dem Dualen zum hier vorgestellten Haar-Wavelet.

2.4.2 Das Sinc-Wavelet

Das Sinc-Wavelet, oft auch Littlewood-Paley-Wavelet genannt, ist dual zum Haar-Wavelet. Es ist definiert durch die Funktion[27]

$$\psi(t) = 2\,\mathrm{sinc}(2t) - \mathrm{sinc}(t) \tag{2.36}$$

mit

$$\mathrm{sinc}(t) = \frac{\sin \pi t}{\pi t} \tag{2.37}$$

es ergibt sich also für Gleichung 2.36

$$\psi(t) = 2\,\frac{\sin 2\pi t}{2\pi t} - \frac{\sin \pi t}{\pi t} = \frac{\sin \frac{\pi t}{2}}{\frac{\pi t}{2}}\,\cos\frac{3\pi t}{2} \tag{2.38}$$

Der Graph dieses Wavelets ist in Abbildung 2.4 dargestellt.

Das Sinc-Wavelet zeigt ein Verhalten umgekehrt zum Haar-Wavelet; es bietet einen kompakten Träger im Frequenzbereich, im Ortsbereich dagegen zeigt es ein Unendlichkeitsverhalten wie das Haar-Wavelet im Frequenzbereich, das heißt es geht für $t \to \infty$ genauso schnell gegen null wie die Funktion $\frac{1}{t}$.

2.4.3 Daubechies-Wavelets

Daubechies konstruierte eine ganze Familie von orthogonalen Wavelets, die im Folgenden mit ${}_N\psi$ ($N \in \mathbb{N}$) bezeichnet werden sollen. Diese besitzen einige bemerkenswerte Eigenschaften:

[27] siehe [19, Seite 221ff.] und [2, Seiten 322 und 333]

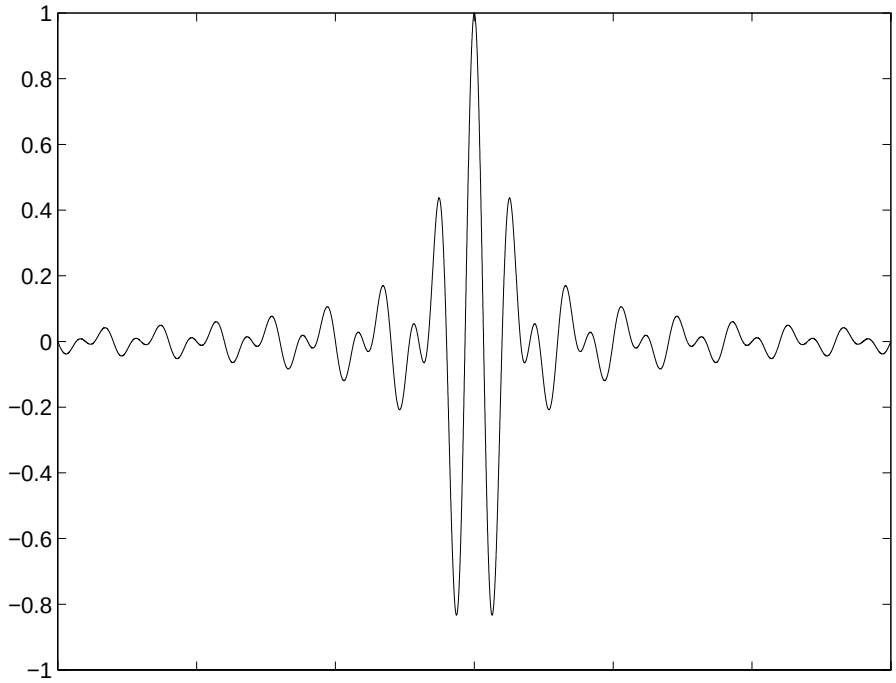


Abbildung 2.4: Das Sinc-Wavelet

- sie besitzen einen kompakten Träger, wobei die Größe des Trägerintervalls $2N - 1$ beträgt,

- für die jeweilige Trägergröße haben sie die maximal mögliche Anzahl verschwindender Momente, nämlich N,

- die Länge des bei der schnellen diskreten Wavelet-Transformation verwendeten Filters (siehe Kapitel 3) beträgt $2N$,

- die zum Wavelet gehörende Skalierungsfunktion (siehe ebenfalls Kapitel 3) besitzt maximale Glattheit für die gegebene Trägergröße.

Hierbei erhält man für $N = 1$ das vorher beschriebene Haar-Wavelet. Außer diesem ist keines der Daubechies-Wavelets symmetrisch, was übrigens allgemein für orthogonale Wavelets mit kompaktem Träger gilt [4].

Graphen von Daubechies-Wavelets für $N = 1, \ldots, 10$ befinden sich im Anhang C.

Anmerkung: Für die Anzahl der verschwindenden Momente liegt folgende Definition zugrunde:

DEFINITION 3 (MOMENT EINER FUNKTION)
Das m-te Moment einer Funktion f ist definiert als:

$$\int x^m f(x)\, dx \tag{2.39}$$

Hat ein Wavelet $\psi(x)$ N *verschwindende Momente*, so bedeutet dies also[28]:

$$\int x^m \psi(x)\, dx = 0 \qquad \text{für m = 0, ..., N-1} \tag{2.40}$$

[28] näheres siehe z. B. [7, Seite 612ff.]

Kapitel 3

Die diskrete Wavelet-Transformation

Die diskrete Wavelet-Transformation, welche sich direkt aus der Wavelet-Reihe ergibt, soll in diesem Kapitel erläutert werden, da mit Hilfe dieser Transformation schließlich die Bildkompression durchgeführt wird.

Man verwendet nun ein Sample einer kontinuierlichen Funktion $f(x)$, welches im Folgenden als $f[x]$ bezeichnet wird, um die abgetastete Funktion von der kontinuierlichen zu unterscheiden. Es wird angenommen, daß $f[x]$ ein Sample bestehend aus N Werten ist, wobei N eine Zweierpotenz ist, also $N = 2^n$, $n \in \mathbb{N}$. Damit ergeben sich für die diskrete Wavelet-Transformation an Stelle der Formeln 2.32 und 2.33 Summen, mit denen die Transformation dann folgendermaßen berechnet wird [2, Seite 332f.]:

$$c_{j,k} = \sum_{x=1}^{N} f[x]\, \psi_{j,k}(x) \tag{3.1}$$

Die zugehörige inverse Transformation erhält man durch:

$$f[x] = \sum_{j} \sum_{k} c_{j,k} \psi_{j,k}(x) \tag{3.2}$$

mit $j = 0, 1, \ldots, n-1$ und $k = 0, 1, \ldots, 2^j - 1$; $\psi_{j,k}(x)$ ist ein dyadisches Wavelet mit kompaktem Träger wie in 2.26:

$$\psi_{j,k}(x) = \sqrt{2^j}\, \psi(2^j x - k) \tag{3.3}$$

Entsprechend Gleichung 2.31 kann an Stelle von j und k auch nur ein Index verwendet werden.

Ich möchte an dieser Stelle nochmals explizit darauf hinweisen, daß die Einschränkung auf dyadische Wavelets zwar automatisch die Orthogonalität der Basis bewirkt, diese ist aber für die diskrete Wavelet-Transformation in der allgemeinsten Form nicht erforderlich. Es gibt also, genau wie bei der kontinuierlichen

Wavelet-Transformation, durchaus nichtorthogonale Wavelets, die als Basis verwendet werden können, auch wenn dies im Folgenden nicht der Fall ist.

Um nun die Funktionsweise der schnellen Wavelet-Transformation zu verstehen, müssen zunächst einige Grundlagen der dort verwendeten Techniken erläutert werden.

Die Entwicklung des Algorithmus der schnellen Wavelet-Transformation ergab sich historisch gesehen aus folgenden Gebieten:

- Multiskalen-Analyse[1],
- Teilbandcodierung und
- Filterbänke.

Den größten Einfluß hatten dabei wohl die Multiskalen-Analyse und die Teilbandcodierung. Aus jedem dieser beiden Verfahren läßt sich der Algorithmus herleiten. In dieser Arbeit soll dabei die Herleitung über die Multiskalen-Analyse erfolgen, da die dort verwendeten Techniken auch zur Konstruktion von orthonormalen Wavelets eingesetzt werden können.

Auf die Teilbandcodierung soll hier nicht näher eingegangen werden, obwohl sich der Algorithmus auch von dieser Seite aus sehr gut erläutern ließe. Ist eine genauere Betrachtung der Teilbandcodierung gewünscht, so möchte ich den Leser auf die Literatur verweisen [2, 19]. Nach [4, Seite 156] ist auch eine direkte Verknüpfung der Multiskalen-Analyse mit der Teilbandcodierung möglich.

Auf Filterbänke soll ebenfalls nur sehr kurz eingegangen werden, das heißt ich werde nur die Darstellung der kontinuierlichen Wavelet-Transformation als Faltungsintegral und damit als eine Bank von Bandpässen vorführen.

Die Herleitung der diskreten Wavelet-Transformation mit Hilfe der Multiskalen-Analyse läuft in etwa wie folgt ab:

- Einführung der Multiskalen-Analyse und einer dazugehörigen Skalierungsfunktion,
- Verknüpfung der Skalierungsfunktion mit Tiefpaßfiltern,
- Darstellung von Wavelets durch die Skalierungsfunktion,
- Verbindung der Wavelets mit Hochpaßfiltern.

[1] engl.: multiresolution-analysis

3.1 Darstellung der Wavelet-Transformation als Faltungsintegral

Im Folgenden wird gezeigt, wie sich die kontinuierliche Wavelet-Transformation (Gleichung 2.12 bzw. 2.14 für die inverse Transformation) so umformen läßt, daß eine Form entsteht, die bereits von der Fourier-Transformation her hinreichend bekannt ist: Die Faltung von zwei Funktionen.

Durch die Darstellung als Faltungintegral ist es nämlich möglich, eine Wavelet-Transformation unter Verwendung von linearen Faltungsfiltern[2] durchzuführen: Man erhält somit eine Filterbank bestehend aus Bandpässen [2, Seite 310f.].

Ausgehend vom Basiswavelet $\psi(x)$ wird die Menge der mit dem Faktor a skalierten und anschließend entsprechend Gleichung 2.11 normierten Wavelets $\psi_a(x)$ definiert[3]:

$$\psi_a(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right) \tag{3.4}$$

Weiterhin wird das Wavelet $\breve{\psi}_a(x)$ definiert; dieses ist das reflektierte[4] Wavelet zu $\psi_a(x)$:

$$\breve{\psi}_a(x) = \psi_a(-x) = \frac{1}{\sqrt{a}}\psi\left(-\frac{x}{a}\right) \tag{3.5}$$

Hieraus ergibt sich die Darstellung der kontinuierlichen Wavelet-Transformation als Faltungsintegral (statt Gleichung 2.12):

$$W_f(a,b) = \int_{-\infty}^{\infty} f(x)\,\breve{\psi}_a(b-x)\,dx \;=\; (f * \breve{\psi}_a)(b) \tag{3.6}$$

Aus dieser Darstellung erhält man eine Bank von Faltungsfiltern, wobei durch $\breve{\psi}_a(x)$ für jeden Wert von a ein anderer Bandpaßfilter definiert ist. Die inverse

[2] engl.: convolution filter

[3] der Unterschied zu Formel 2.10 besteht nur darin, daß hier keine Translation durchgeführt wird; diese folgt erst später

[4] dies gilt, falls $\psi_a(x)$ eine reelle Funktion ist, wovon hier generell ausgegangen wird; für den Fall, daß man komplexe Wavelets verwendet, muß man zusätzlich das konjugiert-komplexe nehmen, also $\breve{\psi}_a(x) = \psi_a^*(-x)$

Wavelet-Transformation lautet dann, äquivalent zu Gleichung 2.14:

$$\begin{aligned} f(x) &= \frac{1}{C_\psi} \int_0^\infty \int_{-\infty}^\infty W_f(a,b)\, \psi_a(b-x)\, db \frac{da}{a^2} \\ &= \frac{1}{C_\psi} \int_0^\infty \int_{-\infty}^\infty (f * \breve{\psi}_a)(b)\, \psi_a(b-x)\, db \frac{da}{a^2} \\ &= \frac{1}{C_\psi} \int_0^\infty (f * \breve{\psi}_a * \psi_a)(x)\, \frac{da}{a^2} \end{aligned} \tag{3.7}$$

Es bleibt noch anzumerken, daß sich diese Form der Darstellung der Wavelet-Transformation natürlich nicht auf den kontinuierlichen Fall beschränkt, sondern wie beschrieben auch für die diskrete Wavelet-Transformation durchführbar ist. Dann läßt sich die Faltung auch in Form einer *Faltungsmatrix* darstellen, was auch ausgenutzt wird. Zur Matrixdarstellung wird auf das nächste Kapitel verwiesen.

3.2 Faltungsfilter

Wegen der bereits in Kapitel 3.1 gewonnenen Darstellung einer Wavelet-Transformation als Faltung von zwei Funktionen, soll nun erläutert werden, wie sich ein diskreter *Faltungsfilter* aufbauen und in Matrixform darstellen läßt. Diese Art von Filtern wird auch im Kapitel 3.3, welches die Multiskalen-Analyse behandelt, verwendet, und man benötigt sie nicht zuletzt bei der Durchführung der schnellen Wavelet-Transformation.

Auf die theoretischen Grundlagen und Eigenschaften der Faltungsoperation soll an dieser Stelle nicht näher eingegangen werden, da diese bereits von der Fourier-Transformation her hinreichend bekannt ist. Es wird daher auf die Literatur zu diesem Thema verwiesen. Eine Einführung zur Faltung und speziell auch zur hier beschriebenen Matrizendarstellung bietet z. B. [2, Seite 148ff.] sowie [8, Seite 209ff.].

Da für die schnelle Wavelet-Transformation nur die diskrete Faltung von Bedeutung ist, möchte ich hier gleich mit dieser beginnen; betrachtet wird zunächst der eindimensionale Fall.

Die Faltung zweier diskreter Funktionen $f[i]$ und $g[i]$ wird wie folgt durchgeführt [2, Seite 152]:

$$h[i] = f[i] * g[i] = \sum_j f[j]\, g[i-j] \tag{3.8}$$

Die Anzahl der Samples von $f[i]$ soll mit m, die von $g[i]$ mit n bezeichnet werden. Das Ergebnis $h[i]$ der Faltungsoperation hat dann die Länge $N = m+n-1$. Damit

man nun die Faltung als Matrix darstellen kann, müssen noch einige Vorarbeiten erledigt werden. Zunächst wird die Funktion $f[i]$, die eine Länge $m < N$ hat, so erweitert, daß sie ebenfalls die Länge N hat. Dies geschieht durch Auffüllen der Funktion mit Nullen. Es entsteht die Funktion $f_p[i]$:

$$f_p[i] = \begin{cases} f[i] & \text{für} \quad 1 \leq i \leq m, \\ 0 & \text{für} \quad m < i \leq N \end{cases} \tag{3.9}$$

Mit $g[i]$ wird ebenso verfahren, womit man die diskrete Funktion $g_p[i]$ erhält.

Anschließend faßt man die $f_p[i]$ zu einem Vektor $\mathbf{f}$ bestehend aus N Elementen zusammen. Weiterhin entsteht die in der nachstehenden Gleichung verwendete $N \times N$ *Faltungsmatrix* $\mathbf{G}$, deren Zeilen von den $g_p[i]$ gebildet werden. Damit kann man die Faltung der beiden diskreten Funktionen nun wie folgt schreiben:

$$\mathbf{h} = \mathbf{G} \cdot \mathbf{f} = \begin{pmatrix} g_p[1] & g_p[N] & \dots & g_p[2] \\ g_p[2] & g_p[1] & \dots & g_p[3] \\ \vdots & \vdots & \vdots & \vdots \\ g_p[N] & g_p[N-1] & \dots & g_p[1] \end{pmatrix} \cdot \begin{pmatrix} f_p[1] \\ f_p[2] \\ \vdots \\ f_p[N] \end{pmatrix} \tag{3.10}$$

Man bezeichnet $\mathbf{G}$ als *zirkuläre* Matrix, da jede Zeile aus der vorhergehenden durch einen Rechtsshift entsteht.

Für die Darstellung der Faltung in zwei Dimensionen kann ähnlich verfahren werden; da diese aber in der vorliegenden Arbeit nicht verwendet wird, möchte ich den Leser auf die Literatur verweisen [2, Seite 153ff.].

3.3 Multiskalen-Analyse

Die Multiskalen-Analyse ist ein Verfahren, welches die Untersuchung von Signalen bei verschiedenen Auflösungen ermöglicht, wobei der Begriff Auflösung bei Bildern besonders anschaulich ist. Ziel dieses Verfahrens ist es, Strukturen in einem Signal zu erkennen; dabei wird davon ausgegangen, daß diese Strukturen im Signal ebenfalls in unterschiedlichen Größen vorkommen; der Rand eines Objekts in einem Bild kann beispielsweise sehr scharf umrissen sein, das heißt er ist nur wenige Pixel breit, er kann sich aber auch über ein großes Gebiet (viele Pixel) hinwegziehen. Mit Hilfe der Multiskalen-Analyse wird man bei der Untersuchung des Bildes den scharfen Rand bei einer großen Auflösung (dies entspricht einem kleinen Maßstab) finden, den fließenden Rand hingegen bei einer geringen Auflösung (großer Maßstab).

Der Begriff Maßstab wird an dieser Stelle übrigens nicht etwa zufällig verwendet, sondern er wurde in Analogie zum Maßstab einer Landkarte gewählt. Auch hier erkennt man bei großen Maßstäben nur die groben Strukturen, während man

bei kleinen Maßstäben auch Details sehen kann, die beim Herauszoomen aus der Karte (die Karte wird kleiner, die Auflösung geringer) verschwinden.

Die Verknüpfung der Wavelets mit der Multiskalen-Analyse besteht in der bei dyadischen[5] Wavelets durchgeführten Dilatation der Basisfunktion um den Faktor zwei. Statt aber nun das Wavelet größer zu machen, wird das Bild kleiner gemacht (Downsampling der Zeilen und Spalten jeweils um den Faktor Zwei, das Bild wird also um den Faktor Vier kleiner), was denselben Effekt hat, vom Rechenaufwand her aber erheblich besser ist[6]. Hierdurch erhält man ein Bild, das eine geringere Auflösung bzw. einen größeren Maßstab hat.

Das Verfahren, aus einem Bild durch Downsampling der Zeilen und Spalten jeweils ein neues, um den Faktor Vier verkleinertes Bild zu generieren und anschließend mit diesem weiterzuarbeiten (entweder mit dem Ziel, das Bild zu komprimieren oder auch Muster, wie z. B. Kanten zu erkennnen) ist auch unter dem Namen *Pyramiden-Algorithmus* bekannt und wird in einem Verfahren zur Bildkomprimierung verwendet, welches als *Laplace Pyramide* bezeichnet wird [2, Seite 321f.].

Zur Erinnerung nochmals das verwendetet dyadische Wavelet ($j, k \in \mathbb{Z}$):

$$\psi_{j,k}(x) = \sqrt{2^j}\,\psi(2^j x - k) \tag{3.11}$$

Wie man sieht, wird das Wavelet breiter (größer) für $j < 0$, was einem großen Maßstab entspricht (geringere Auflösung); für $j > 0$ wird das Wavelet schmaler, man erhält einen kleinen Maßstab (höhere Auflösung).

Die Multiskalen-Analyse ist nun ein Verfahren, welches zur Konstruktion von orthonormalen Wavelets verwendet werden kann. Es soll nun erläutert werden, wie man dies mit Hilfe der Multiskalen-Analyse und des Pyramidenverfahrens bewerkstelligen kann [7, Seite 43ff.].

Es zeigt sich, daß sich die *meisten* bekannten orthonormalen Waveletbasen durch dieses Verfahren erzeugen lassen, insbesondere *alle*, die einen kompakten Träger besitzen.

Für den Fall, daß eine genauere Abhandlung der Multiskalen-Analyse erwünscht ist als sie im Folgenden geboten wird, möchte ich den Leser auf die Literatur verweisen, speziell auf das Buch *Wavelets* [15] sowie den Aufsatz *An Overview of Wavelet based Multiresolution Analyses* [12].

Wie bereits erwähnt, wird bei der Multiskalen-Analyse die Menge der Signale (hier: Bilder) in ineinander verschachtelte Unterräume unterteilt, wobei jeder Unterraum einer anderen Skala (Maßstab) entspricht. In unserem speziellen Fall wird die Auflösung immer um den Faktor Zwei verringert. Es wird hier nochmals

[5] es sind hierbei natürlich nicht unbedingt dyadische Wavelets nötig; diese werden aber in Praxis meist verwendet, weshalb auch hier nur auf diese eingegangen wird. Bei anderen Wavelets werden einfach andere Skalierungsfaktoren (statt zwei) verwendet

[6] [2, Seite 321 oben]

darauf hingewiesen, daß der Signalraum, dem alle Signale entstammen müssen L^2 (im kontinuierlichen Fall) bzw. l^2 (im diskreten Fall) sein muß[7]! Die Unterräume werden im Folgenden mit V_i bezeichnet. Es folgt nun zuerst die Definition der Multiskalen-Analyse [4, 7, 20], [19, Seite 215] für den kontinuierlichen Fall; der diskrete wird hier nicht explizit definiert, da dieser dem kontinuierlichen Fall entspricht, wenn man die Integrale in Summen umwandelt und an Stelle von L^2 den Raum l^2 verwendet.

DEFINITION 4 (MULTISKALEN-ANALYSE)
Eine durch eine Skalierungsfunktion ϕ erzeugte orthonormale Multiskalen-Analyse von $L^2(\mathbb{R})$ ist eine Folge von abgeschlossenen Unterräumen V_i ($i \in \mathbb{Z}$), die folgenden Bedingungen genügt:

1. $\cdots \subset V_{i-1} \subset V_i \subset V_{i+1} \subset \cdots \subset L^2(\mathbb{R})$ *(Verkettung)*
2. $\lim_{i\to\infty} V_i = L^2(\mathbb{R})$, *also* $\overline{\bigcup V_i} = L^2(\mathbb{R})$ *(Ausschöpfung)*
3. $\lim_{i\to-\infty} V_i = 0$, *also* $\bigcap V_i = \{0\}$ *(Ausdünnung)*
4. $f(x) \in V_i \Leftrightarrow f(2x) \in V_{i+1}$ *(Skaleninvarianz)*
5. $f(x) \in V_i \Leftrightarrow f(x-k) \in V_i$, $k \in \mathbb{R}$ *(Translationsinvarianz)*
6. *Die Translationen $\phi_{i,k}$ der Skalierungsfunktion $\phi \in V_0$ mit*

$$\phi_{i,k} = \sqrt{2^i}\phi(2^i x - k)$$

bilden eine orthonormale Basis von V_i, das heißt es gilt

$$\langle \phi_{j,k}, \phi_{l,m} \rangle = \delta_{j,l} \cdot \delta_{k,m}$$

(Existenz einer Riesz-Basis)

Es folgen einige Erläuterungen zur obigen Definition:

zu 1: Die hier (und in [7]) verwendete Indizierung der Unterräume entspricht der Indizierung nach Mallat; in [20] wird die Indizierung nach Daubechies verwendet, die genau andersherum ist, also $V_i \subset V_{i-1}$. Die beiden Definitionen sind aber sonst äquivalent.

zu 2: Durch diese Bedingung wird ausgesagt, daß alle Funktionen aus $L^2(\mathbb{R})$ durch Funktionen aus den Multiskalen-Räumen beliebig genau angenähert werden können (siehe auch weiter unten).

[7] siehe Definitionen 1 und 2 (Gleichung 2.4 bzw. 2.5)

zu 4: Durch die Eigenschaft der Skaleninvarianz wird die Multiskalen-Analyse überhaupt erst möglich; sie besagt, daß die Auflösung der Räume V_i mit zunehmendem i immer größer wird; wenn also gilt $f(x) \in V_0$, dann liegt $f(2x)$ im Raum V_1 bzw. $f(2^i x)$ liegt in V_i.

zu 5: Die Translationsinvarianz besagt, daß die Räume V_i verschiebungsinvariant sind, das heißt durch Verschiebung einer Funktion $f(x) \in V_i$ auf der reellen Achse um den Wert k wird der Raum nicht verlassen.

zu 6: Bilden die Translationen $\phi_{i,k}$ eine Basis, die nicht orthonormal (orthogonal) ist, so liegt ebenfalls eine Multiskalen-Analyse vor, die aber eben nicht orthonormal (orthogonal) ist. Anmerkung: Die Skalierungsfunktion wird zur Projektion (Skalierung) von Funktionen in die Räume V_i verwendet. Sie ist selbst kein Wavelet, hängt mit diesem aber über eine einfache Beziehung zusammen (siehe weiter unten).

Als Skalierungsfunktion ϕ wird in der Praxis eine Glättungsfunktion[8] verwendet, das heißt es gilt[9]:

$$\int_{-\infty}^{\infty} \phi(x)\, dx \neq 0 \tag{3.12}$$

Weiterhin sollte sowohl die Skalierungsfunktion ϕ als auch ihre Fouriertransformierte Φ (und damit das zugehörige Wavelet) möglichst schnell auf null abfallen, damit eine gute Lokalität im Orts- und Frequenzbereich gewährleistet ist. Daher werden meist Skalierungsfunktionen (bzw. Wavelets) mit kompaktem Träger verwendet.

Um nun zum Zusammenhang zwischen der Multiskalen-Analyse und der Wavelet-Transformation zu kommen, betrachten wir im Folgenden die Approximation einer Funktion $f(x) \in L^2(\mathbb{R})$ durch die Multiskalen-Analyse. Die Approximation entspricht der orthonormalen Projektion $P_i(f)$ von $f(x)$ in den Raum V_i, es gilt also:

$$f = \lim_{i \to \infty} P_i(f) \tag{3.13}$$

Die Projektion $P_i(f)$ in den Raum V_i erfolgt durch Anwendung der Skalierungsfunktion $\phi_{i,k}$. $P_i(f)$ wird durch die sogenannten *Skalierungskoeffizienten* dargestellt, welche aufgrund der Orthogonalität durch Berechnung eines Skalarprodukts äquivalent zur Berechnung der Wavelet-Transformation ermittelt werden können:

$$s_{i,k}(f) = \langle f, \phi_{i,k} \rangle \tag{3.14}$$

[8] Tiefpaß

[9] im Gegensatz zum Wavelet, für das gelten muß: $\int_{-\infty}^{\infty} \psi(x)\, dx = 0$

Durch die Approximation entsteht aufgrund des Verlustes an Auflösung natürlich ein Fehler. Dieser soll als Fehler zwischen zwei Näherungen dargestellt werden, das heißt als Unterschied zwischen den Projektionen der Funktion in den Raum V_i und den Raum V_{i+1}. Der Fehler selbst liegt nun in einem Komplementärraum $W_i = V_{i+1} \ominus V_i$, der wegen der gewünschten Orthogonalität der Wavelets als orthogonales Komplement zu V_i dargestellt wird [7, Seite 44f.] [19, Seite 218f.].

Im Laufe dieses Kapitels wird schließlich gezeigt, daß der Projektionsfehler beim Übergang von einem Raum in den nächsten identisch ist mit einem Teil der Waveletkoeffizienten der Funktion.

DEFINITION 5 (WAVELETRÄUME)
Die Waveleträume W_i sind definiert durch die orthogonalen Komplemente von V_i im (übergeordneten) Raum V_{i+1}:

$$W_i = V_{i+1} \ominus V_i, \quad W_i \perp V_i$$

Es gilt also

$$V_{i+1} = V_i \oplus W_i \tag{3.15}$$

womit dann äquivalent zur Eigenschaft *Ausschöpfung* aus Definition 4 ebenfalls gilt [7, 15, 19, 20]:

$$L^2(\mathbb{R}) = \bigoplus_{i \in \mathbb{Z}} W_i \tag{3.16}$$

Dies bezeichnet man als *Waveletzerlegung von L^2*, da jeder Raum W_i entsprechend Gleichung 2.26 durch Translationen $\psi_{i,k}$ eines Basiswavelets ψ erzeugt werden kann, das heißt $\{\psi_{i,k}\}$ ist eine (orthonormale) Basis des Waveletraums W_i. Es kann bewiesen werden, daß dyadische Wavelets tatsächlich eine Basis dieser Räume bilden [13, Seite 153]; dies soll jedoch hier nicht vorgeführt werden, da es zum weiteren Verständnis nicht viel beiträgt. Statt dessen soll gezeigt werden, wie aus einer Skalierungsfunktion $\phi \in V_0$ ein Wavelet $\psi \in W_0 \subset V_1$ gewonnen werden kann, welches als Basiswavelet für $\psi_{i,k}$ fungiert.

Für ϕ gilt:

$$\phi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} h_k \phi(2x - k) \tag{3.17}$$

Diese Gleichung heißt *Dilatations-* oder *Verfeinerungs-Gleichung*[10], da durch sie die Skalierungsfunktion $\phi \in V_0$ im Raum V_1 dargestellt wird, der ja eine feinere Auflösung besitzt. Der Beweis für die Gültigkeit dieser Gleichung findet sich in [15, Seite 106], [19, Seite 216] und soll nun zum besseren Verständnis kurz skizziert werden.

Wie man sieht, ergibt sich die Skalierungsfunktion $\phi(2x - k)$ durch Dilatation und Translation aus $\phi(x)$. Aufgrund von Definition 4 können nun folgende Aussagen gemacht werden:

[10] engl.: *dilation* bzw. *refinement equation*

- Es gilt: $\phi(x) \in V_0$ und $V_0 \subset V_1 \Rightarrow$
- $\phi(x) \in V_1 \Rightarrow$
- $\phi(x)$ läßt sich als Linearkombination der Basisfunktionen von V_1 darstellen; die Basisfunktionen von V_1 aber sind bekannt und lauten

$$\{\sqrt{2}\phi(2x-k) \mid k \in \mathbb{Z}\}.$$

  Daraus folgt schließlich Gleichung 3.17, wobei h_k die Koeffizienten für die Linearkombination sind.

- Eine wichtige Folgerung aus 3.17 ist: *Allein durch die Koeffizienten* h_k ist die Skalierungsfunktion ϕ eindeutig bestimmt, sobald man einen Wert für die Norm von ϕ festgelegt hat (z. B. $\parallel \phi \parallel = 1$) [7, Seite 46]. Wie man noch sehen wird, ist damit auch ein Wavelet und damit natürlich eine Wavelet-Transformation bestimmt.

Die Koeffizienten h_k können als Impulsantwort eines diskreten Tiefpaßfilters $\mathbf{H}$ aufgefaßt werden. Zu dieser wichtigen Erkenntnis gelangt man nach der Durchführung einer Fourier-Transformation und diversen Umformungen; die genaue Vorgehensweise wird in [19, Seite 216] vorgeführt und es soll hier nur das Ergebnis präsentiert werden.

Nach [6, Seite 51ff.] erhält man die Koeffizienten h_k bei bekannter Skalierungsfunktion durch

$$h_k = \sqrt{2} \int_{-\infty}^{\infty} \phi(x)\phi(2x-k)\,dx \tag{3.18}$$

Weiterhin folgt aufgrund der Orthogonalität der Basis $\{\phi_{i,k}\}$ eines Raums V_i eine Orthogonalität der Koeffizienten h_k wie folgt[11] [2, 6, 7, 15]:

$$\sum_k h_k h_{k+2m} = \delta_{0m} \qquad \forall m \in \mathbb{Z} \tag{3.19}$$

Durch die Energienormalisierung von $\phi(x)$ erhält man eine l^2-Norm der Koeffizienten h_k mit dem Wert Eins [15, Seite 116]:

$$\sum_k |h_k|^2 = 1 \tag{3.20}$$

Ein weiteres Ergebnis ist[12]:

$$\sum_k h_k = \sqrt{2} \tag{3.21}$$

[11] Beweis siehe [6, Seite 51f.] oder [15, Seite 113]

[12] Herleitung siehe [15, Seite 114] oder [19, Seite 217]

Dabei werden für die Fouriertransformierte $\Phi(\omega)$ der Skalierungsfunktion $\phi(x)$ folgende Einschränkungen vorgenommen, auf die in der Praxis aber keine große Rücksicht genommen werden muß, da sie wohl immer erfüllt sein werden [19, Seite 217]:

- $\Phi(\omega)$ ist beschränkt,
- stetig in $\omega = 0$,
- $\Phi(0) \neq 0$

Nun soll aufgrund der vorherigen Schlußfolgerungen mit Hilfe der Skalierungsfunktion ϕ ein Basiswavelet $\psi \in W_0$ ermittelt werden, durch welches schließlich eine (orthonormale) Basis $\{\psi_{i,k}\}$ von W_i erzeugt wird.

Wegen $\psi \in W_0$ und $W_0 \subset V_1$ (und damit $\psi \in V_1$) gilt:

$$\psi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} g_k \phi(2x - k) \tag{3.22}$$

Dies gilt, da $\{\sqrt{2}\phi(2x - k) \,|\, k \in \mathbb{Z}\}$ eine orthonormale Basis von V_1 ist und somit natürlich auch alle Funktionen aus W_0 erzeugen kann. Die Koeffizienten der hierzu nötigen Linearkombination dieser Basisfunktionen sind mit g_k bezeichnet.

Zudem wird die Eigenschaft ausgenutzt, daß $\psi(x)$ eine Basis des Raumes W_0 sein soll, welcher ja orthogonal zu V_0 ist, das heißt es gilt:

$$\langle \phi(x - n), \psi(x) \rangle = 0 \qquad \forall n \tag{3.23}$$

Durch diese zusätzliche Bedingung entsteht ein Zusammenhang zwischen den Koeffizienten h_k der Skalierungsfunktion und den Koeffizienten g_k des Wavelets, der nach einigen Umformungen im Fourierraum hergeleitet werden kann[13]. Man erhält als Ergebnis einen Hochpaßfilter $\mathbf{G}$, der im Fourierraum wie folgt aussieht:

$$G\left(e^{j\omega}\right) = -e^{-j\omega}\, H^*\left(e^{j(\omega+\pi)}\right) \tag{3.24}$$

für die Darstellung im Ortsbereich ergibt sich somit:

$$g_k = (-1)^k h_{1-k} \tag{3.25}$$

Wie man sieht, ergeben sich die Filterkoeffizienten g_k aus denen von h_k, wobei die Reihenfolge genau entgegengesetzt ist und jeder zweite Koeffizient negiert wird. Aus diesem Grund wird der Hochpaßfilter $\mathbf{G}$ auch als *Spiegelfilter* von $\mathbf{H}$ bezeichnet.

[13] siehe [19, Seite 218ff.]

Außerdem gilt[14] wegen der oben genannten Bedingungen für h_k sowie Gleichung 3.25:

$$\sum_k g_k = 0 \tag{3.26}$$

Ich möchte hier nochmals darauf hinweisen, daß die Bedingungen, die ein Wavelet zu einem solchen machen, eigentlich sehr schwach sind. Es wird nur gefordert, daß ein Wavelet quadratisch-integrierbar über den reellen Zahlen sein muß und daß es die *Zulässigkeitsbedingung* (Gleichung 2.7) erfüllt. Deswegen sind auch in der vorangegangenen Darstellung keine weiteren Einschränkungen nötig.

Aufgrund von Gleichung 3.17, die jeweils für die Skalierungsfunktionen von zwei aufeinanderfolgenden Räumen V_i und V_{i+1} gilt, kann man die Basis $\{\phi_{i,j}\}$ des Raums V_i nun aus der Basis von V_{i+1} wie folgt berechnen [7, Seite 47]:

$$\begin{aligned} \phi_{i,j}(x) &= \sum_k h_k \, \phi_{i+1,2j+k}(x) \\ &= \sum_k h_{k-2j} \, \phi_{i+1,k}(x) \end{aligned} \tag{3.27}$$

Die Projektion einer Funktion $f(x)$ in den Raum V_i erfolgt nun wieder durch Berechnung der Skalierungskoeffizienten:

$$\begin{aligned} s_{i,j} &= \langle f, \phi_{i,j} \rangle \\ &= \sum_k h_k \, \langle f, \phi_{i+1,2j+k} \rangle \\ &= \sum_k h_{k-2j} \, \langle f, \phi_{i+1,k} \rangle \\ &= \sum_k h_{k-2j} \, s_{i+1,k} \\ &= (\mathbf{H}\mathbf{s}_{i+1})_j \end{aligned} \tag{3.28}$$

Wie man sieht, lassen sich die Skalierungskoeffizienten $s_{i,j}$, welche ja die Projektion von f in den Raum V_i darstellen, aus den Skalierungskoeffizienten $s_{i+1,j}$ durch Filterung mit dem Filter h_{k-2j} berechnen. Der Aufbau der Matrix $\mathbf{H}$ entspricht der Darstellung eines Faltungsfilters in Matrixform, wobei allerdings jede zweite Zeile fehlt. Die Elemente der Matrix sind:

$$\mathbf{H} = (h_{k-2j})_{j,k} \tag{3.29}$$

Durch das Fehlen jeder zweiten Zeile entstehen nach Anwendung des Filters mit der Faltungsmatrix $\mathbf{H}$ auf den Vektor $\mathbf{s}_{i+1}$, der die Skalierungskoeffizienten

[14] näheres in [15, Seite 114]

enthält, nur noch halb soviele Koeffizienten[15] wie vor dem Filtern; dies ist auch gut so, denn eine Funktion f hat ja im Raum V_{i+1} eine doppelt so hohe Auflösung wie in V_i.

Nachdem nun durch Gleichung 3.28 klar ist, wie man aus den Skalierungskoeffizienten $s_{i+1,j}$ die Koeffizienten $s_{i,j}$ berechnen kann, benötigt man noch den Beginn für die Iteration, nämlich die Skalierungskoeffizienten $s_{0,j}$. Hierzu wird angenommen, daß eine Funktion $f(x) \in V_0$ vorliegt[16]. Die Koeffizienten ergeben sich dann durch Darstellung der Funktion als Linearkombination der Basisfunktionen $\phi_{0,j}$ des Raums V_0:

$$f(x) = \sum_{j \in \mathbb{Z}} s_{0,j} \phi(x - j) \tag{3.30}$$

Für die tatsächliche Durchführung der schnellen Wavelet-Transformation wird dann angenommen, daß die vorliegenden diskreten Werte bereits die Skalierungskoeffizienten $s_{0,j}$ einer Funktion sind, die selbst allerdings gar nicht explizit angegeben wird. Sie wird für das Verfahren auch nicht weiter benötigt.

Für die Berechnung der Waveletkoeffizienten gilt eine ähnliche Beziehung: Man erhält die Waveletkoeffizienten $w_{i,j}$ für den Raum W_i durch Anwendung der Faltungsmatrix $\mathbf{G}$ auf die Skalierungskoeffizienten $s_{i+1,j}$ des übergeordneten Raums V_{i+1}:

$$\begin{aligned} w_{i,j} &= \langle f, \psi_{i,j} \rangle \\ &= \sum_k g_k \langle f, \phi_{i+1,2j+k} \rangle \\ &= \sum_k g_{k-2j}\, s_{i+1,k} \\ &= (\mathbf{G}\mathbf{s}_{i+1})_j \end{aligned} \tag{3.31}$$

hierbei sind die Matrixelemente gegeben durch:

$$\mathbf{G} = (g_{k-2j})_{j,k} \tag{3.32}$$

Auch diese Matrix enthält nur halb soviele Zeilen wie Spalten und wird zum Downsampling verwendet.

Das Ergebnis dieses Ausflugs in die Multiskalen-Analyse kann nun wie folgt beschrieben werden:

- An Stelle der aufwendigen direkten Berechnung der bei der Wavelet-Transformation auftretenden Skalarprodukte kann man die Waveletkoeffizienten durch Anwendung eines Filters auf die Skalierungskoeffizienten erhalten.

[15] dies entspricht dem Downsampling um den Faktor Zwei

[16] siehe auch [15, Seite 126]

- Durch die Wahl der Filterkoeffizienten h_k entsprechend den dafür angegebenen Bedingungen[17] ist eine Skalierungsfunktion und damit ein Wavelet festgelegt. Die Skalierungsfunktion bzw. das Wavelet werden dann gar nicht mehr explizit angegeben; sie werden für die Durchführung der schnellen Wavelet-Transformation auch nicht benötigt.

- Die Waveletkoeffizienten $w_{i,j} \in W_i$ geben den Fehler an, der bei der Projektion einer Funktion f vom Raum V_{i+1} in den Raum V_i gemacht wird.

- Kennt man die Skalierungskoeffizienten $\mathbf{s}_i$ einer Funktion $f \in V_m$ bei einer bestimmten Auflösung V_i und die bei der Projektion dieser Funktion in einen Raum V_m, der eine höhere Auflösung als V_i besitzt (also $m > i$), entstehenden Fehler $\mathbf{w}_i, \mathbf{w}_{i+1}, \ldots, \mathbf{w}_{m-1}$, so kann man die Funktion $f \in V_m$ exakt rekonstruieren.

- Die Vorgehensweise bei der Transformation entspricht damit der Berechnung der Koeffizienten durch ein Pyramidenschema, wobei eine exakte Wiederherstellung des Signals möglich ist.

- Die genannten Bedingungen für die diskreten Filter sind übrigens zum Teil bereits von der Teilbandcodierung her bekannt. Sie ergeben sich aus den Forderungen der exakten Rekonstruktion des Signals und der Vermeidung von Aliasing.

Anmerkung: Es würde an dieser Stelle zu weit führen, wenn man Methoden zum Finden von Filtern beschreiben wollte, die bestimmte Eigenschaften besitzen (z. B. Glattheit der Skalierungsfunktion, Anzahl der verschwindenden Momente, kompakter Träger, etc.). Daher möchte ich den interessierten Leser hier auf die Literatur verweisen, wo dies teilweise sehr ausführlich behandelt wird [4, 6, 15, 19].

Welche Eigenschaften eines Wavelets für die Bildkompression von Nutzen sind wird in Kapitel 4.2 noch erläutert.

3.4 Berechnung diskreter Werte für Wavelet und Skalierungsfunktion

Wie man im vorherigen Kapitel gesehen hat, kann man also durch geschickte Wahl eines Tiefpasses eine Skalierungsfunktion und ein dazugehörendes Wavelet festlegen. Nun soll gezeigt werden, wie man mit Hilfe dieser Filterkoeffizienten Wavelet und Skalierungsfunktion durch ein einfaches Verfahren approximieren kann[18]. Man erhält damit diskrete Werte für die beiden Funktionen und kann dann

[17] diese sind in Kapitel 3.5.1 nochmals zusammengefaßt

[18] siehe auch [6, Seite 54ff.] und [7, Seite 48]

z. B. ihre Graphen zeichnen. Für die im folgenden Kapitel beschriebene diskrete Wavelet-Transformation werden die Werte allerdings nicht benötigt.

Beide Funktionen werden durch fortlaufende Iteration gewonnen, wobei man mit einem Vektor startet, dessen Elemente Skalierungskoeffizienten darstellen. Dabei sind alle Elemente null, bis auf eines, welches gleich eins ist. Diese Werte stellen die Skalierungsfunktion bzw. das Wavelet in der niedrigstmöglichen Auflösung dar, wobei es vom anschließend angewandten Algorithmus abhängt, welche der beiden Funktionen man approximiert. Die Näherungen, die man dadurch erhält, sind recht gut, wobei das Verfahren auch schnell konvergiert. Die nun vorgestellten Formeln ergeben sich als direkte Konsequenz der Gleichungen 3.22, 3.28 sowie 3.31.

Will man die Skalierungsfunktion erhalten, so ist folgende Formel anzuwenden:

$$\begin{aligned} x_0 &= \begin{pmatrix} \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \\ x_{i+1} &= \sqrt{2}\mathbf{H}^{\mathrm{T}} x_i \end{aligned} \tag{3.33}$$

Man wendet also die Faltungsmatrix $\mathbf{H}$ iterativ an. Der Skalierungsfaktor $\sqrt{2}$ ist dabei sehr wichtig, da die Funktionswerte sonst bei jedem Iterationsschritt um den Faktor $\frac{1}{\sqrt{2}}$ kleiner werden würden[19].

Das Wavelet ergibt sich durch einen ähnlichen Algorithmus. Hierbei wird jedoch zu Beginn einmal die Matrix $\mathbf{G}$ angewandt und erst anschließend $\mathbf{H}$:

$$\begin{aligned} x_0 &= \begin{pmatrix} \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \\ x_1 &= \sqrt{2}\mathbf{G}^{\mathrm{T}} x_0 \\ x_{i+1} &= \sqrt{2}\mathbf{H}^{\mathrm{T}} x_i \qquad i \geq 1 \end{aligned} \tag{3.34}$$

In Abbildung 3.1 sind als Beispiel die ersten Iterationsschritte für das Daubechies-Wavelet D_4 mit vier Filterkoeffizienten dargestellt.

[19] dies ist eine Folge der Normalisierung auf $\sqrt{2}$

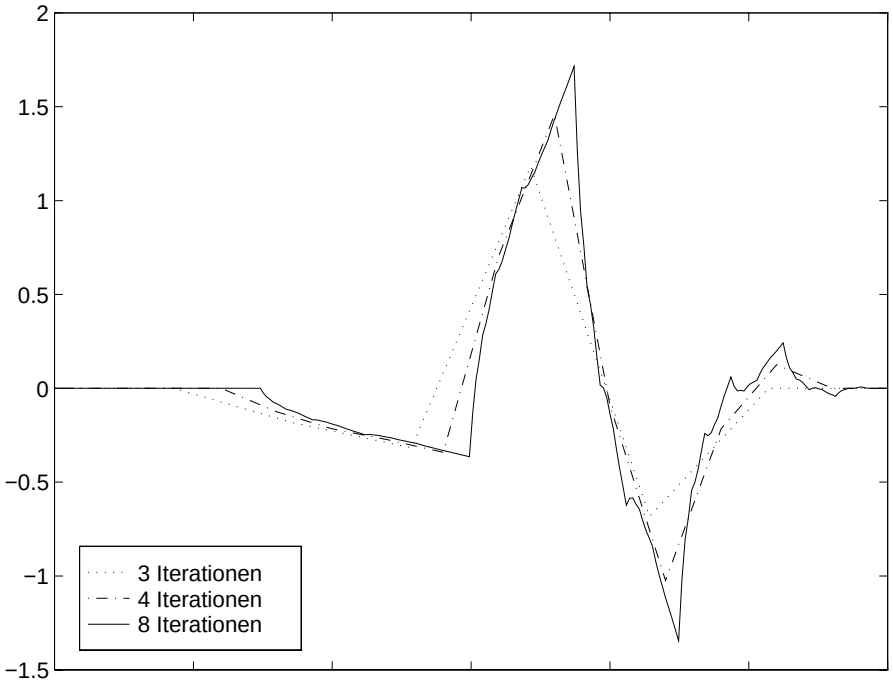


Abbildung 3.1: Verschieden genaue Approximationen des Daubechies-Wavelets D_4

3.5 Die schnelle Wavelet-Transformation

In den folgenden Kapiteln soll nun genauer auf die schnelle diskrete Wavelet-Transformation eingegangen werden, wobei die Grundlagen aus Kapitel 3.3 vorausgesetzt werden. Zunächst wird der Algorithmus für die eindimensionale Transformation vorgestellt, daran anschließend derjenige für die zweidimensionale.

Der Algorithmus wird in der Literatur praktisch immer mehr oder weniger ausführlich beschrieben. Für die vorliegende Abhandlung wurde vor allem [2, 7, 19] verwendet.

3.5.1 Eindimensionale schnelle Wavelet-Transformation

Der zur schnellen diskreten Wavelet-Transformation verwendete Algorithmus[20] basiert auf der iterativen Anwendung von Hoch- und Tiefpaßfiltern. Diese müssen bestimmte Eigenschaften besitzen, welche sich aus der Multiskalen-Analyse wie in 3.3 beschrieben ergeben. Der Algorithmus kann aber auch als Zweikanal-

[20] wegen seiner Struktur auch unter dem Namen *Mallat's herringbone algorithm* (Heringgrätenalgorithmus) bekannt

Teilbandcodierung verstanden werden[21]. Die Vorgehensweise bei der Transformation eines Signals ist in Abbildung 3.2 zu sehen.

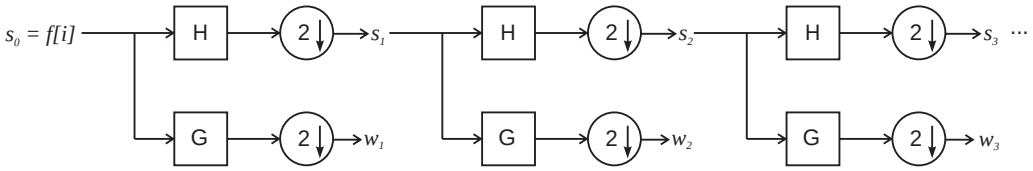


Abbildung 3.2: Der Algorithmus für die schnelle diskrete Wavelet-Transformation

Es wird von nun an der Hochpaßfilter mit den Filterkoeffizienten g_k in Matrixdarstellung mit $\mathbf{G}$ bezeichnet[22], der Tiefpaßfilter mit den Filterkoeffizienten h_k mit $\mathbf{H}$. Die Elemente dieser Matrizen sind $g_{j,k}$ bzw. $h_{j,k}$. Der Tiefpaß wird manchmal auch als *Glättungs-* oder *Skalierungsfilter* bezeichnet, der Hochpaß als *Detail-* oder *Waveletfilter*. Die Bedingungen, die die Filter erfüllen müssen lauten[23]:

- $\sum_k h_k = \sqrt{2}$
- $\sum_k h_k h_{k+2m} = \delta_{0m} \qquad \forall m$
- $g_k = (-1)^k h_{1-k}$
- $\sum_k g_k = 0$

Nach [7, Seite 40] können diese Bedingungen auch in Matrixform dargestellt werden[24]:

- $\mathbf{H}^{\mathrm{T}}\mathbf{H} + \mathbf{G}^{\mathrm{T}}\mathbf{G} = \mathbf{E}$
- $\mathbf{G}\mathbf{H}^{\mathrm{T}} = \mathbf{H}\mathbf{G}^{\mathrm{T}} = \mathbf{0}$
- $\mathbf{G}\mathbf{G}^{\mathrm{T}} = \mathbf{H}\mathbf{H}^{\mathrm{T}} = \mathbf{E}$

Die Matrix $\mathbf{H}$ ist eine Faltungsmatrix, wie sie in Kapitel 3.2 beschrieben wurde. Da im Algorithmus für die Wavelet-Transformation aber nach jedem Filtervorgang ein Downsampling um den Faktor Zwei durchgeführt werden muß,

[21] siehe [2, 4, 15, 19]
[22] entsprechend [7]
[23] siehe Kapitel 3.3
[24] mit $\mathbf{E}$ wird die Einheitsmatrix bezeichnet

fehlt in der Faltungsmatrix jede zweite Zeile. Bei n Filterkoeffizienten h_k ($k = 0, \ldots, n-1$) ergibt sich dann die Matrix:

$$\mathbf{H} = \begin{pmatrix} \ldots & & & & & & & & & & & & \\ \ldots & 0 & h_0 & h_1 & h_2 & \ldots & h_{n-1} & 0 & 0 & 0 & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & 0 & h_0 & h_1 & h_2 & \ldots & h_{n-1} & 0 & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & 0 & h_0 & h_1 & h_2 & \ldots & h_{n-1} & 0 & \ldots \\ \ldots & & & & & & & & & & & & \end{pmatrix} \tag{3.5.1}$$

Die Matrixelemente $\mathrm{H}_{j,k}$ ergeben sich wie beschrieben durch $\mathrm{H}_{j,k} = h_{k-2j}$, wobei die entsprechenden Matrixelemente der Indizes, für die gilt $k - 2j < 0$ oder $k - 2j > n - 1$, null sind.

Weiterhin wird hier davon ausgegangen, daß die zu transformierende diskrete Funktion $f[u]$ aus N Elementen besteht, wobei gilt $N = 2^l$ ($l \in \mathbb{N}$). Es ist zu beachten, daß in der Theorie immer mit unendlich vielen Daten gerechnet wird, was für die Praxis gewisse Auswirkungen hat (siehe weiter unten). Auch die oben beschriebenen Matrizen $\mathbf{H}$ und $\mathbf{G}$ sind prinzipiell unendlich groß, weshalb man zum tatsächlichen Berechnen der Transformation einige Anpassungen (entweder der Matrizen oder Daten) vornehmen muß. Die hier gemachte Beschränkung der Anzahl der Daten auf eine Zweierpotenz kann in der Praxis aufgehoben werden (näheres hierzu folgt ebenfalls später).

Wie man in Abbildung 3.2 sieht, erfolgt die Berechnung der Waveletkoeffizienten $\mathbf{w}_i$ durch Anwendung der Matrix $\mathbf{G}$ auf die Skalierungskoeffizienten $\mathbf{s}_{i-1}$ ($i = 1, \ldots, l$)[25]. Die Skalierungskoeffizienten $\mathbf{s}_i$ enstehen durch Anwendung der Matrix $\mathbf{H}$ auf die Skalierungskoeffizienten $\mathbf{s}_{i-1}$[26]. Diese Vorgehensweise entspricht der Berechnung durch die Formeln 3.31 bzw. 3.28.

Es müssen folglich genau l Schritte durchgeführt werden, bis die Anzahl der Skalierungskoeffizienten durch fortlaufendes Downsampling um den Faktor Zwei bis auf einen Koeffizienten zusammengeschrumpft ist. Es gilt also:

- Die Elemente $\mathbf{s}_{0,k}$ des Eingabevektors $\mathbf{s}_0$ ergeben sich durch $\mathbf{s}_{0,k} = f[k]$, $k = 1, \ldots, 2^l$. Man geht also eigentlich davon aus, daß die diskreten Werte $f[k]$ bereits die Skalierungskoeffizienten einer Funktion im Raum V_0 sind[27].
- Der letzte entstehende Vektor $\mathbf{s}_l$ besteht aus genau einem Skalierungskoeffizienten.
- Die Vektoren $\mathbf{s}_i$ bzw. $\mathbf{w}_i$ haben genau die halbe Anzahl an Elementen wie die Vektoren $\mathbf{s}_{i-1}$ bzw. $\mathbf{w}_{i-1}$ einen Schritt vorher.

[25] dies entspricht einer Hochpaßfilterung
[26] dies entspricht einer Tiefpaßfilterung
[27] siehe auch Kapitel 3.3

Bei dem vorgestellten Verfahren entstehen also insgesamt $N-1$ Waveletkoeffizienten sowie ein Skalierungskoeffizient; dabei sind $\frac{N}{2}$ Koeffizienten aus dem ersten Schritt entstanden, $\frac{N}{4}$ aus dem zweiten, $\frac{N}{8}$ aus dem dritten usw.

Man erhält bei diesem Algorithmus nach jedem Tiefpaßfilter und anschließendem Downsampling um zwei[28] ein Signal $f[u]$, welches durch Anwendung der Skalierungsfunktion ϕ (dem Filter) nun in einer niedrigeren Auflösung vorliegt. Das Signal wird also vom Raum V_0 in einen Raum V_{-i} mit niedrigerer Auflösung abgebildet[29].

Entsprechend der in Kapitel 2.3 vorgestellten dyadischen Wavelets mit kompaktem Träger, welche ja für die schnelle Wavelet-Transformation verwendet werden, benötigt man zur Abdeckung des gesamten Intervalls bei jedem Dilatationsfaktor $-i$ eine bestimmte Anzahl an Wavelets, die dann eine Basis des jeweiligen Waveletraums W_{-i} bilden. Mit zunehmendem i wird dieses Intervall wegen der Reduktion der Auflösung immer geringer, weshalb auch immer weniger Wavelets als Basis benutzt werden und somit auch immer weniger Waveletkoeffizienten $\mathbf{w}_i$ entstehen. Für die Vektoren $\mathbf{w}_i$ bzw. $\mathbf{s}_i$ aus Abbildung 3.2 gilt also:

$$\mathbf{w}_i = \begin{pmatrix} \langle \psi_{-i,1}, f \rangle \\ \langle \psi_{-i,2}, f \rangle \\ \vdots \\ \langle \psi_{-i,\frac{N}{2^i}}, f \rangle \end{pmatrix} \tag{3.36}$$

sowie

$$\mathbf{s}_i = \begin{pmatrix} \langle \phi_{-i,1}, f \rangle \\ \langle \phi_{-i,2}, f \rangle \\ \vdots \\ \langle \phi_{-i,\frac{N}{2^i}}, f \rangle \end{pmatrix} \tag{3.37}$$

Die genauen Formeln für die Berechnung von $\mathbf{w}_i$ bzw. $\mathbf{s}_i$ lauten dann[30]:

$$\begin{aligned} w_{i,j} &= \sum_k g_{k-2j}\, s_{i-1,k} \\ &= (\mathbf{G}\mathbf{s}_{i-1})_j \end{aligned} \tag{3.38}$$

und

$$\begin{aligned} s_{i,j} &= \sum_k h_{k-2j}\, s_{i-1,k} \\ &= (\mathbf{H}\mathbf{s}_{i-1})_j \end{aligned} \tag{3.39}$$

[28] die Reihenfolge - erst Filtern, dann Downsampling oder umgekehrt - ist dabei egal; mit den hier verwendeten Matrizen wird beides in einem Schritt durchgeführt

[29] man darf sich hierbei nicht daran stören, daß der Index des Raums negativ ist. Dieser entsteht nur durch die Wahl der Indizierung bei der Definition der Multiskalen-Analyse nach Mallat (siehe Kapitel 3.3 Definition 4)

[30] entsprechend Kapitel 3.3 Gleichungen 3.28 und 3.31

Die Matrizendarstellung dient dabei nur der einfacheren mathematischen Formulierung; in einer realen Implementierung werden die Summen direkt berechnet, da dies schneller geht.

Nun möchte ich mich noch der inversen Transformation widmen. Will man diese durchführen, so wird im Prinzip einfach der Algorithmus rückwärts berechnet. Das Verfahren ist in Abbildung 3.3 zu sehen.

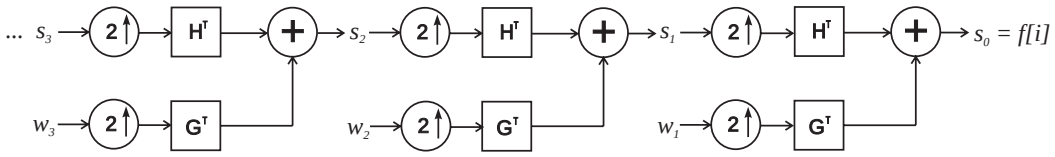


Abbildung 3.3: Der Algorithmus für die inverse schnelle diskrete Wavelet-Transformation

Es werden die gleichen Filterkoeffizienten wie für die Transformation verwendet; für die Darstellung in Matrixform benutzt man statt der Faltungsmatrizen $\mathbf{G}$ und $\mathbf{H}$ deren transponierte $\mathbf{G}^{\mathrm{T}}$ und $\mathbf{H}^{\mathrm{T}}$. Diese besitzen dann immer doppelt soviele Zeilen wie Spalten und bewirken so ein Upsampling um den Faktor Zwei.

Man berechnet also die Skalierungskoeffizienten $\mathbf{s}_i$ aus den Skalierungskoeffizienten $\mathbf{s}_{i+1}$ und den Waveletkoeffizienten $\mathbf{w}_{i+1}$ wie folgt:

$$\mathbf{s}_i = \mathbf{H}^{\mathrm{T}}\,\mathbf{s}_{i+1} + \mathbf{G}^{\mathrm{T}}\,\mathbf{w}_{i+1} \tag{3.40}$$

bzw. als Summe

$$s_{i,j} = \sum_k (h_{j-2k}\, s_{i+1,k} + g_{j-2k}\, w_{i+1,k}) \tag{3.41}$$

Wie bereits erwähnt wurde, wird hier natürlich nicht mit unendlich vielen Daten, sondern mit Daten der Länge N gerechnet. Dadurch ergeben sich einige Konsequenzen:

- Die Größe der Faltungsmatrizen $\mathbf{H}$ und $\mathbf{G}$ nimmt immer mehr ab, je mehr Schritte man durchführt; zu Beginn werden Matrizen der Größe $\frac{N}{2} \times N$ verwendet, wodurch genau $\frac{N}{2}$ Skalierungskoeffizienten und ebenso viele Waveletkoeffizienten entstehen. Für den nächsten Schritt werden die beiden Matrizen so angepaßt, daß sie sowohl in den Zeilen als auch in den Spalten nur noch halb so viele Elemente haben wie vorher — es entstehen also $\frac{N}{4} \times \frac{N}{2}$-Matrizen.

- Es muß eine Anpassung der Daten oder der Matrizen vorgenommen werden, wodurch die endliche Zahl an Daten berücksichtigt wird. Unterläßt

man dies, so kann man durch die inverse Wavelet-Transformation die ursprünglichen Daten nicht mehr wiederherstellen[31].

Will man eine exakte Rekonstruktion der Daten nach einer inversen Transformation erreichen, so hat man folgende Möglichkeiten [7, Seiten 40 und 69]:

- Man benutzt bei der Transformation mehr Daten, als man eigentlich bräuchte, wobei die tatsächlich benötigten Daten von den nicht benötigten eingeschlossen sind. Diese Vorgehensweise ist dann möglich, wenn man eine diskrete Funktion $f[u]$ durch Abtasten eines Signals $f(x)$ gewinnen kann; man digitalisiert dann einfach mehr Daten. Bei der Bildkompression gestaltet sich die Sache schon komplizierter, da man ja im Normalfall wohl das gesamte Bild komprimieren will und nicht etwa nur einen Ausschnitt daraus; hier muß man dann zu einer der anderen Methoden greifen.

- Man nimmt an, daß die Daten periodisch sind. Hierzu müssen nicht zwangsläufig die Daten angepaßt werden, sondern man kann die Matrizen $\mathbf{H}$ und $\mathbf{G}$ entsprechend verändern (siehe unten).

- Man füllt die Daten an den Rändern mit Nullen auf, so daß genügend Werte zur Rekonstruktion zur Verfügung stehen.

- Man gewinnt die Daten an den Rändern durch Reflexion; im Fall eines Bildes bedeutet dies, man setzt an das eigentliche Bild jeweils horizontal bzw. vertikal gespiegelte Versionen des Bildes an, je nachdem an welchem Rand man sich befindet.

- Man verwendet an den Rändern speziell angepaßte Randwavelets. Eine genaue Betrachtung dieser Methode würde hier zu weit führen, ich möchte daher auf die Literatur verweisen [15, Seite 199ff.].

Alle genannten Verfahren haben Vor- und Nachteile: Beim Hinzufügen von mehr Daten wird die Transformation wesentlich rechenaufwendiger, weshalb die Modifikation der Filter im Prinzip zu bevorzugen ist. Außerdem entstehen trotz allem meist mehr oder weniger sichtbare Veränderungen der Daten an den Rändern, wobei diese Randeffekte aber natürlich nicht so stark sind, wie es ohne Anpassung der Fall wäre.

Eine Betrachtung der Randeffekte mit und ohne Anpassung der Daten befindet sich in Kapitel 6.4.

Bei der Annahme, daß man periodische Daten vorliegen hat, müssen die Faltungsmatrizen so verändert werden, wie es hier exemplarisch an einer 4×8-Matrix

[31] exakte Wiederherstellung ist in diesem Fall nur bei Verwendung des Haar-Wavelets möglich; siehe auch [7, Seite 40]

$\mathbf{H_4}$ mit vier Filterkoeffizienten $h_0, \ldots, h_3$ vorgeführt ist[32]:

$$\mathbf{H_8} = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & h_0 & h_1 \end{pmatrix} \tag{3.42}$$

In der praktischen Anwendung des Algorithmus ist man natürlich nicht auf Daten der Länge $N = 2^l$ beschränkt. Hat man als Länge keine Zweierpotenz vorliegen, so kann man die Daten durch Auffüllen mit Nullen evtl. auf die geforderte Größe bringen. Andere Möglichkeiten zur Behandlung dieses Problems werden in [20] geschildert, ich möchte mich hier weiterhin auf Zweierpotenzen beschränken.

Da der Algorithmus bei der Verwendung zur Datenkompression nicht unbedingt bis zum Ende durchgeführt werden muß, sondern nach einer beliebigen Anzahl an Schritten abgebrochen werden kann, ist es in einigen Fällen auch möglich ihn so lange laufen zu lassen, bis die entstehende Sequenz der Skalierungskoeffizienten nicht mehr durch zwei teilbar ist; meist ist das Ergebnis trotzdem recht ansehnlich[33].

3.5.2 Zweidimensionale schnelle Wavelet-Transformation

Für die Anwendung der Wavelet-Transformation in der Bildkompression ist es natürlich nötig, daß die schnelle Wavelet-Transformation nicht nur mit eindimensionalen Signalen, sondern auch mit zweidimensionalen (Bildern) durchgeführt werden kann. Die hierzu nötige Erweiterung des Algorithmus von Mallat ist das Thema dieses Kapitels.

Prinzipiell werden hierfür die bereits in Kapitel 2.2.2 vorgestellten zweidimensionalen Wavelets verwendet. Dabei muß nun unterschieden werden, ob eine separierbare Transformation verwendet wird oder nicht. Da in der Bildkompression zur Zeit praktisch nur mit separierbaren Transformationen gearbeitet wird, sollen hier auch nur diese berücksichtigt werden. Dies Vereinfacht den Algorithmus für die zweidimensionale Transformation auf die Durchführung von hintereinandergeschalteten eindimensionalen Transformationen. Dadurch ist er einfacher zu implementieren und schneller in der Ausführung. Der hier vorgestellte Algorithmus findet sich dann auch in dieser Form in praktisch jeder Literatur zum diesem Thema. Den nicht separierbaren Fall dagegen findet man eher selten; er wird in [15, Seite 123ff.] und [19, Seite 410ff.] kurz besprochen.

Man geht also von einer separierbaren zweidimensionalen Skalierungsfunktion $\phi(x, y)$ aus, deren Dilatationen und Translationen $\{\phi_{j,k_x,k_y}(x, y) = 2^j\phi(2^j x -$

[32] vgl. mit Gleichung 3.5.1

[33] einige Vorschläge zur Behandlung von beliebig langen Daten sind in [20] zu finden

$k_x, 2^j y - k_y)\}$ eine orthonormale Basis eines zweidimensionalen Raumes $V_j \subset L^2(\mathbb{R}^2)$ bilden. Diese zweidimensionale Skalierungsfunktion kann dann als Produkt zweier eindimensionaler Skalierungsfunktionen $\phi(x)$ und $\phi(y)$ dargestellt werden:

$$\phi(x, y) = \phi(x)\phi(y) \tag{3.43}$$

Aufbauend auf die sich dadurch ergebenden zweidimensionalen Basiswavelets[34] $\psi^m(x, y)$ ($m = 1, 2, 3$), ist dann eine orthonormale Basis des $L^2(\mathbb{R}^2)$ definiert durch [15, Seite 208]:

$$\{\psi^m_{j,k_x,k_y}(x, y) \quad | \quad j, k_x, k_y \in \mathbb{Z} \qquad m = 1, 2, 3\} \tag{3.44}$$

mit

$$\psi^m_{j,k_x,k_y}(x, y) = 2^j \psi^m(2^j x - k_x, 2^j y - k_y) \tag{3.45}$$

Die zweidimensionalen Basiswavelets $\psi^m(x, y)$ ergeben sich aus den zu den Skalierungsfunktionen $\phi(x)$ und $\phi(y)$ gehörenden Basiswavelets $\psi(x)$ bzw. $\psi(y)$, welche wie in Kapitel 3.3 beschrieben ermittelt werden:

$$\begin{aligned} \psi^1(x, y) &= \phi(x)\psi(y) \\ \psi^2(x, y) &= \psi(x)\phi(y) \\ \psi^3(x, y) &= \psi(x)\psi(y) \end{aligned} \tag{3.46}$$

Diese Wavelets betonen unterschiedliche Richtungen in der Ebene: ψ^1 ist ein vertikales Wavelet, ψ^2 ein horizontales und ψ^3 ein vertikales. Diese Richtungen sind auch in den während der Transformation entstehenden hochpaßgefilterten Bildern sichtbar, wo Kanten in der jeweiligen Richtung hervorgehoben werden (siehe hierzu auch Abbildung 3.5).

Im Folgenden wird von einer zweidimensionalen Funktion $f(x, y)$ (einem Bild) ausgegangen, wobei die Anzahl der Werte in den Zeilen und Spalten gleich sein soll ($N = 2^l$), das Bild ist also quadratisch; diese Einschränkung kann für eine Implementierung mit einigem Aufwand aufgehoben werden.

Zu berechnen sind dann in jedem Schritt i ($i = 1, \ldots, l$) insgesamt vier Skalarprodukte, wobei aus jedem Skalarprodukt ein um den Faktor Vier kleineres Bild entsteht. Drei der Skalarprodukte ergeben dann bereits Waveletkoeffizienten $w^m_i(k_x, k_y)$ ($m = 1, 2, 3$) und eines das Originalbild in einer um den Faktor Vier niedrigeren Auflösung (die Skalierungskoeffizienten $s_i(k_x, k_y)$):

$$\begin{aligned} s_i(k_x, k_y) &= \langle f, \phi_{-i,k_x,k_y} \rangle \\ w^1_i(k_x, k_y) &= \langle f, \psi^1_{-i,k_x,k_y} \rangle \\ w^2_i(k_x, k_y) &= \langle f, \psi^2_{-i,k_x,k_y} \rangle \\ w^3_i(k_x, k_y) &= \langle f, \psi^3_{-i,k_x,k_y} \rangle \end{aligned} \tag{3.47}$$

Damit kann der Algorithmus nun in jedem Schritt getrennt für die Zeilen und Spalten durchgeführt werden (siehe Abbildung 3.4):

[34] das hochgestellte m wird hier nicht als Exponent verwendet, sondern als Index

- Tiefpaßfilterung von $f(x, y)$ entlang der x-Werte mit anschließendem Downsampling um zwei (Anwendung von $\phi(x)$). Es entsteht ein halb so schmales Bild $f_T(x, y)$.

- Hochpaßfilterung von $f(x, y)$ entlang der x-Werte mit anschließendem Downsampling um zwei (Anwendung von $\psi(x)$). Es entsteht ein halb so schmales Bild $f_H(x, y)$.

- Tiefpaßfilterung von $f_T(x, y)$ entlang der y-Werte mit anschließendem Downsampling um zwei (Anwendung von $\phi(y)$). Es entsteht als Resultat durch Anwendung der Skalierungsfunktion $\phi(x, y)$ auf $f(x, y)$ ein um den Faktor Vier kleineres Bild $f_{TT}(x, y)$: Dieses enthält die Skalierungskoeffizienten s.

- Hochpaßfilterung von $f_T(x, y)$ entlang der y-Werte mit anschließendem Downsampling um zwei (Anwendung von $\psi(y)$). Es entsteht durch Anwendung des Wavelets ψ^1 auf $f(x, y)$ das Bild $f_{TH}(x, y)$: Dies sind die Waveletkoeffizienten w^1.

- Tiefpaßfilterung von $f_H(x, y)$ entlang der y-Werte mit anschließendem Downsampling um zwei (Anwendung von $\phi(y)$). Es entsteht durch Anwendung des Wavelets ψ^2 auf $f(x, y)$ das Bild $f_{HT}(x, y)$: Dies sind die Waveletkoeffizienten w^2.

- Hochpaßfilterung von $f_H(x, y)$ entlang der y-Werte mit anschließendem Downsampling um zwei (Anwendung von $\psi(y)$). Es entsteht durch Anwendung des Wavelets ψ^3 auf $f(x, y)$ das Bild $f_{HH}(x, y)$: Dies sind die Waveletkoeffizienten w^3.

Das Downsampling um zwei erfolgt einfach durch Weglassen jeder zweiten Zeile bzw. Spalte.

Es ist bei diesem Verfahren egal, ob erst ein Downsampling durchgeführt und dann gefiltert wird oder umgekehrt; ebenso kann man die Reihenfolge der Berechnung ändern: statt zuerst die x-Werte zu behandeln, kann man auch erst in y-Richtung downsamplen und filtern und anschließend in x-Richtung.

In Abbildung 3.5 sind für einige Schritte die Zwischenergebnisse dargestellt, die bei der Durchführung des Algorithmus entstehen. Dabei befindet sich im linken oberen Eck jeweils das Originalbild in einer im Vergleich zum vorhergehenden Schritt um den Faktor Vier geringeren Auflösung. Der Rest sind die Waveletkoeffizienten, wobei das Teilbild rechts oben f_{TH} entspricht, das Teilbild links unten f_{HT} und das Teilbild rechts unten f_{HH}.

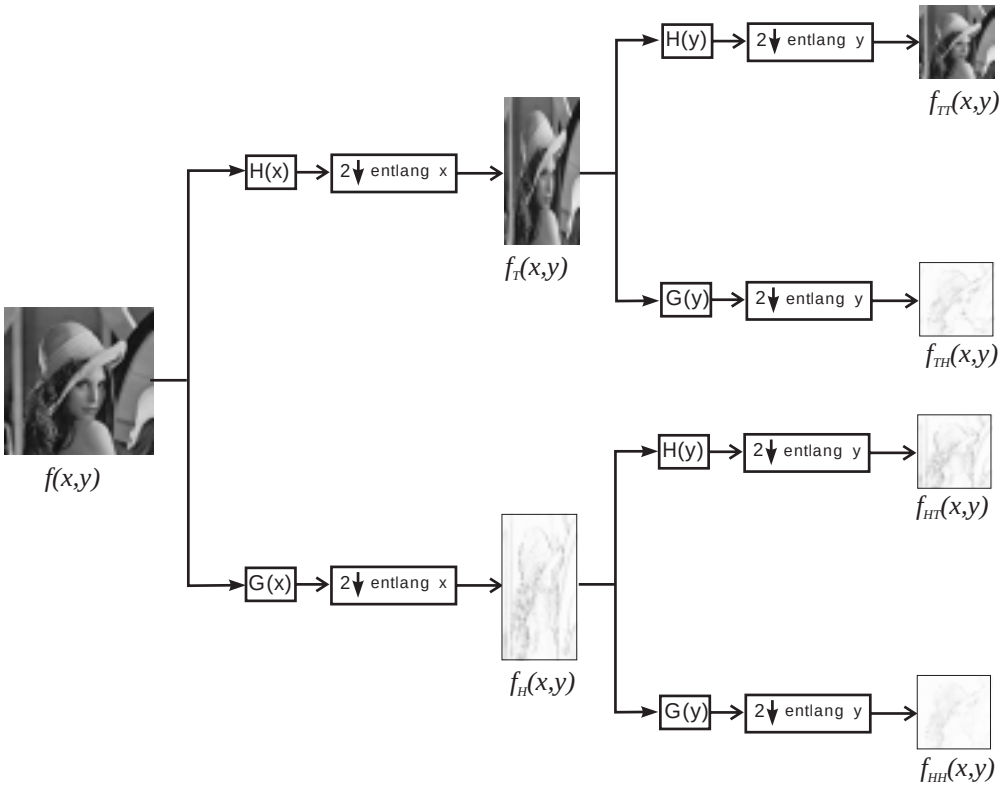


Abbildung 3.4: Der Algorithmus für die zweidimensionale schnelle Wavelet-Transformation (ein Schritt)

(a) Originalbild (b) 1. Schritt

(c) 2. Schritt (d) 3. Schritt

Abbildung 3.5: Mehrere Schritte bei der Transformation eines Bildes

Die inverse Transformation wird nun genau wie im eindimensionalen Fall einfach umgekehrt durchgeführt (siehe Abbildung 3.6):

- Man beginnt mit dem Upsampling der vier Teile f_{TT}, f_{TH}, f_{HT} und f_{HH} entlang der y-Werte um den Faktor Zwei.
- Anschließend werden diese entlang der y-Werte gefiltert, wie es im vorherigen Kapitel bei der eindimensionalen Transformation beschrieben wurde.
- Es ergibt sich nun f_T durch Addition der gefilterten Teile f_{TT} und f_{TH}; f_H durch Addition der gefilterten Teile f_{HT} und f_{HH}.
- Nun erfolgt ein Upsampling von f_T sowie f_H um zwei entlang der x-Werte.
- Durch Filterung und anschließender Addition der beiden Teile erhält man so wieder das Originalbild (bzw. den nächsthöheren Schritt).

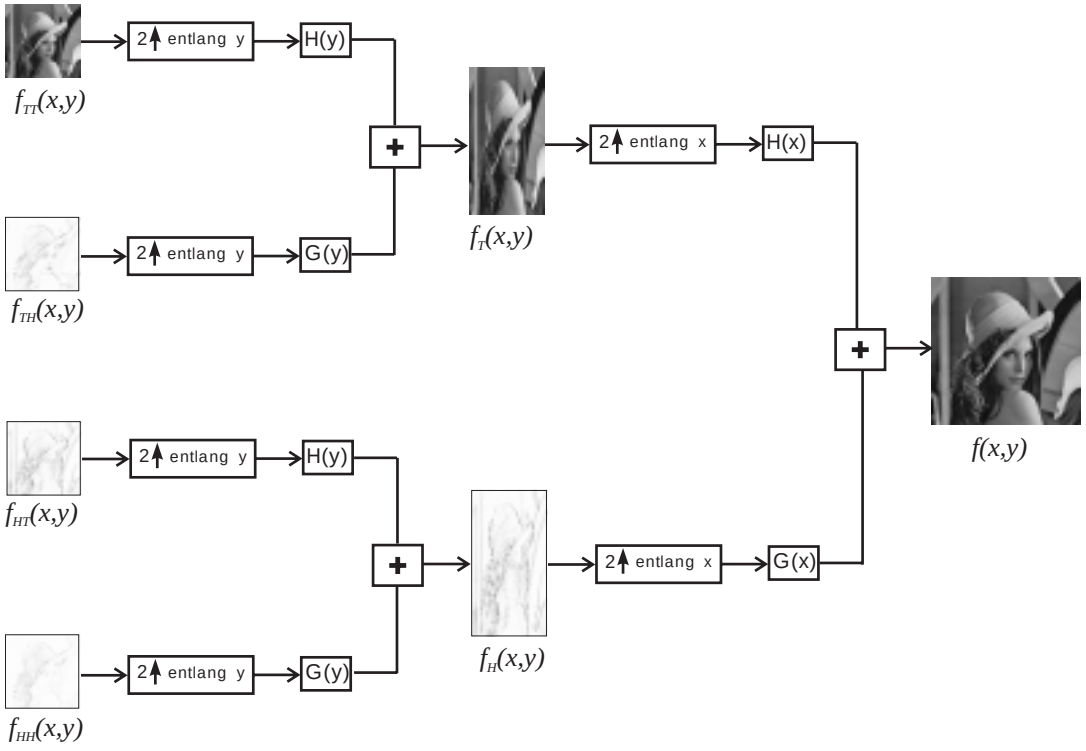


Abbildung 3.6: Der Algorithmus für die inverse zweidimensionale schnelle Wavelet-Transformation (ein Schritt)

3.5.3 Komplexität des Algorithmus

Ich möchte nun kurz auf den Rechenaufwand für die Durchführung der vorgestellten eindimensionalen schnellen Wavelet-Transformation eingehen. Das Ziel dieses Kapitels ist es, auf einfache Weise plausibel zu machen, warum der Algorithmus von Mallat eine Komplexität von $O(N)$ hat, ohne ihn bis in jede Einzelheit zu analysieren. Für weiterführende Angaben, die sich allerdings nicht speziell auf diesen Algorithmus beziehen, wird auf die Literatur [1], [15, Seite 130] verwiesen.

Es ist sehr bemerkenswert, daß die Berechnung der Waveletkoeffizienten für eine diskrete Funktion $f[i]$ ($i = 1, \ldots, N$) nur *linearen* Aufwand erfordert. Bemerkenswert besonders deshalb, da die schnelle Wavelet-Transformation damit besser abschneidet als die schnelle Fourier-Transformation, die ja bekanntermaßen eine Komplexität von $O(N \log N)$ hat.

Bei den folgenden Berechnungen wird davon ausgegangen, daß man eine diskrete Funktion $f[i]$ bestehend aus N Werten ($i = 1, \ldots, N$) vorliegen hat, für die die Waveletkoeffizienten ermittelt werden sollen. Hierbei soll N eine Zweierpotenz sein ($N = 2^n$). Die Anzahl der in den Matrizen $\mathbf{H}$ und $\mathbf{G}$ verwendeten Filterkoeffizienten h_k bzw. g_k wird mit j bezeichnet (also $k = 1, \ldots, j$).

Entsprechend Abbildung 3.2 geschieht die Berechnung der Waveletkoeffizienten $\mathbf{w}_l$ bzw. der Skalierungskoeffizienten $\mathbf{s}_l$ durch iterative Anwendung der Faltungsmatrizen $\mathbf{H}$ und $\mathbf{G}$ auf die Skalierungskoeffizienten. Dazu sind bis zur Reduktion des Vektors $\mathbf{s}_l$ auf ein einziges Element genau n Schritte nötig. Die Größe der Matrizen nimmt dabei immer weiter ab, wie es bereits in 3.5.1 beschrieben wurde.

Wichtig ist nun, daß die meisten Elemente der Matrizen null sind und daher keine vollständigen Matrizenmultiplikationen durchgeführt werden müssen; die Matrizen dienen nur der einfacheren mathematischen Beschreibung des Vorgangs. Dies ist auch gut so, da pro Multiplikation einer $N \times N$-Matrix mit einem N-elementigen Vektor prinzipiell ein Aufwand von $O(N^2)$ nötig ist. In jeder Zeile von $\mathbf{H}$ bzw. $\mathbf{G}$ gibt es j Elemente, die von null verschieden sind (eben die Filterkoeffizienten).

Berücksichtigt man bei der Berechnung des Produkts aus Matrix und Vektor nur diese von null verschiedenen Elemente, so ergibt sich für die Multiplikation einer $\frac{N}{2} \times N$-Matrix mit einem N-elementigem Vektor folgender Aufwand:

- j Multiplikationen pro Zeile der Matrix $\Rightarrow$
- bei $\frac{N}{2}$ Zeilen $j\frac{N}{2}$ Multiplikationen

Der Aufwand für die Additionen soll hier nicht weiter betrachtet werden, man erkennt aber leicht, daß für die obige Matrizenmultiplikation $(j-1)\frac{N}{2}$ Additionen nötig sind.

Der gerade errechnete Wert für die Anzahl der Multiplikationen ergibt sich für den ersten Schritt des Algorithmus. Führt man ihn bis zum Ende durch, so ergibt sich insgesamt für den Aufwand der Berechnung der Skalierungskoeffizienten[35]:

$$\begin{aligned} &j\frac{N}{2} + j\frac{N}{4} + j\frac{N}{8} + \cdots + j\frac{N}{2^n} = \\ &jN\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n}\right) \end{aligned} \tag{3.48}$$

Für $n \to \infty$ ergibt sich für den Ausdruck in Klammern eine unendliche Reihe, welche gegen eins konvergiert:

$$\sum_{v=1}^{\infty} \frac{1}{2^v} = 1 \tag{3.49}$$

Wie man sieht, ergibt sich also ein Aufwand von $O(jN)$.

Berücksichtigt man nun noch, daß pro Schritt nicht nur eine, sondern zwei Matrizenmultiplikationen durchzuführen sind, so muß der eben berechnete Aufwand noch mit dem Faktor Zwei multipliziert werden und man erhält $O(2jN)$.

Da $2j$ ein konstanter Faktor ist, gilt $O(2jN) = O(N)$. An diesem Ergebnis ändert auch eine Einbeziehung der benötigten Additionen in die Berechnung nichts, da ja nur lineare Summanden auftreten.

3.6 Biorthogonale Wavelets

Bisher wurde ausschließlich mit orthogonalen Wavelets und Skalierungsfunktionen gearbeitet; nun soll kurz auf die Verwendung von *biorthogonalen* Wavelets eingegangen werden, da diese in der Bildkompression des öfteren Verwendung finden[36].

Es soll hier nur die Anwendung solcher Wavelets erläutert werden; genaueres findet sich in der Literatur[37].

Der prinzipielle Unterschied zwischen orthogonalen und biorthogonalen Wavelets bei der Durchführung der diskreten Wavelet-Transformation ist schnell erklärt: Im orthogonalen Fall verwendet man für die Transformation die gleichen Hoch- und Tiefpaßfilterkoeffizienten wie für die Rekonstruktion, also eine orthogonale Basis bestehend aus einem Basiswavelet. Im biorthogonalen Fall dagegen wird für die Transformation ein anderes Basiswavelet verwendet als für die Rekonstruktion, man benötigt also insgesamt zwei Hoch- und zwei Tiefpaßfilter. Dieser Zusammenhang ist auch in Abbildung 3.7 illustriert.

[35] d.h. es werden zunächst nur die Anwendungen der Matrix **H** berücksichtigt

[36] z. B. im FBI Standard zur Wavelet-Bildkompression; siehe hierzu Kapitel 4.3

[37] siehe [4, Seite 259ff.], [7, Seite 66ff.], [15, Seite 178ff.] sowie [19, Seite 271ff.]

Biorthogonal bedeutet hierbei: das eine Wavelet $\psi(x)$ ist dual zum anderen Wavelet $\tilde{\psi}(x)$ und ihre Dilatationen und Translationen bilden eine duale Basis [4], es gilt also:

$$\langle \psi_{j,k}, \tilde{\psi}_{l,m} \rangle = \delta_{j,l}\delta_{k,m} \tag{3.50}$$

und

$$f = \sum_{j,k} \langle f, \psi_{j,k} \rangle \tilde{\psi}_{j,k} = \sum_{j,k} \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k} \tag{3.51}$$

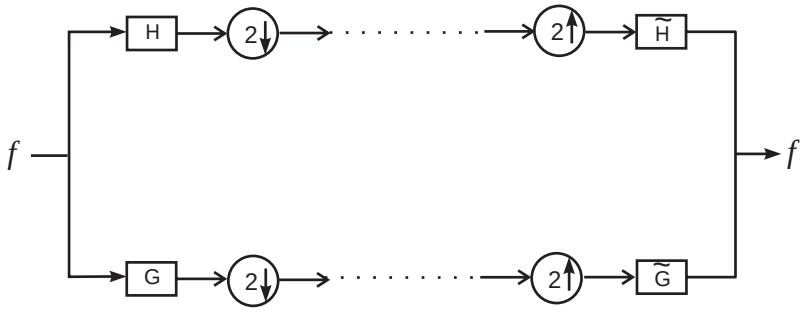


Abbildung 3.7: Transformation und Rekonstruktion unter Verwendung biorthogonaler Wavelets (ein Transformationsschritt)

Man erhält also bezüglich der Multiskalen-Analyse nicht nur einen Raum V_i, sondern zwei Räume V_i und $\tilde{V}_i$ mit den zugehörigen Waveleträumen W_i sowie $\tilde{W}_i$, wobei jetzt aber nicht mehr automatisch $W_i \perp V_i$ gilt, sondern aufgrund der Biorthogonalität $W_i \perp \tilde{V}_i$ und $\tilde{W}_i \perp V_i$ [4, 7].

Zu beachten ist weiterhin, daß sich wegen der Biorthogonalität die Beziehung zwischen den Hoch- und Tiefpaßfiltern ändert[38]:

$$\begin{aligned} g_k &= (-1)^k \tilde{h}_{1-k} \\ \tilde{g}_k &= (-1)^k h_{1-k} \end{aligned} \tag{3.52}$$

Der Vorteil biorthogonaler Wavelets gegenüber orthogonalen liegt vor allem darin, daß gewisse Restriktionen, die einem durch Wahl einer orthogonalen Waveletbasis auferlegt werden, aufgehoben werden, insbesondere:

- die Transformation mit symmetrischen Wavelets mit kompaktem Träger wird möglich: bis auf das Haar-Wavelet ist kein Wavelet mit kompaktem Träger, welches eine orthogonale Basis bildet, symmetrisch. Bei Verwendung von symmetrischen Wavelets, sowohl zur Transformation als auch für die inverse Transformation, wäre keine exakte Rekonstruktion möglich. Symmetrie ist aber gerade in der Bildkompression wünschenswert. Quantisierungsfehler im Bereich von Kanten in Bildern stören den optischen Eindruck eines Menschen nämlich mehr, wenn diese Fehler asymmetrisch um

[38] Diese Gleichung ist eine Verallgemeinerung von 3.25

die Kante verteilt sind als wenn sie symmetrisch sind. Das bedeutet, durch Verwendung von symmetrischen Wavelets kann man eine grobere Quantisierung durchführen und so die Kompressionsrate erhöhen ohne daß die Bildqualität darunter leiden würde [4].

- Wavelets, die eine orthogonale Basis bilden sind wesentlich schwieriger zu konstruieren als biorthogonale [7].

Trotz der genannten Vorteile ist es nicht einfach möglich, in einer Implementierung die zur Transformation und inversen Transformation verwendeten Filter in der beschriebenen Weise auszutauschen, damit alles funktioniert. Dies ist schon wegen der meist unterschiedlichen Anzahl der Filterkoeffizienten nicht möglich. Da mir auch keine genaueren Angaben vorlagen, wie eine Implementierung aussehen muß, können in dem in Kapitel 5 vorgestellten Programm leider keine biorthogonalen Wavelets verwendet werden.

Kapitel 4

Wavelets in der Bildkompression

In den vorangegangenen Kapiteln wurde die Grundlage zur Kompression von Bildern mittels der Wavelet-Transformation gelegt. Nun soll erläutert werden, wie die genaue Vorgehensweise aussieht und welche Eigenschaften ein Basiswavelet haben sollte, damit es für die geforderte Anwendung gut geeignet ist.

4.1 Vorgehensweise bei der Transformationscodierung von Bildern

Bei der verlustbehafteten Kompression von Bildern will man hohe Kompressionsraten durch geschicktes weglassen von Information erreichen, wobei im komprimierten Bild so wenig Veränderung wie möglich sichtbar sein sollen, am besten natürlich gar keine. Die verlustbehaftete Bildkompression wird im allgemeinen durch Transformation eines Bildes in einen Frequenzraum ermöglicht; bestes Beispiel hierfür ist das wohl allseits bekannte JPEG-Format. Bei diesem wird eine diskrete Cosinustransformation[1] (DCT) durchgeführt an die sich noch weitere Verarbeitungsschritte anschließen, die prinzipiell die gleichen sind wie bei der hier verwendeten diskreten Wavelet-Transformation.

Folgende Schritte sind bei der Transformationscodierung nötig:

- Transformation des Originalbildes
- Quantisierung der Transformationskoeffizienten
- Entropiecodierung

Bei der Dekomprimierung eines auf diese Weise gespeicherten Bildes werden die genannten Schritte in umgekehrter Reihenfolge durchlaufen. Dabei entstehen rein

[1] es ist mit JPEG auch eine verlustfreie Kompression möglich; dann wird allerdings **keine** DCT verwendet

theoretisch nur bei der Quantisierung nicht mehr rückgängig zu machende Verluste, das heißt wenn keine Quantisierung vorgenommen wird, kann man, abgesehen von den durch endliche Zahlendarstellung und endliche Daten entstehenden Fehlern, das Bild exakt rekonstruieren. Bei der Dekomprimierung können die Quantisierungsverluste natürlich nicht wiederhergestellt werden, weshalb dieser Schritt bei der Dekomprimierung auch entfällt. Es existieren aber kompliziertere Verfahren, wie z. B. die Vektorquantisierung [7, 19], wo evtl. noch ein Verarbeitungsschritt nötig ist. Das beschriebene Verfahren ist in Abbildung 4.1 zu sehen. Genaueres folgt in den nächsten Kapiteln.

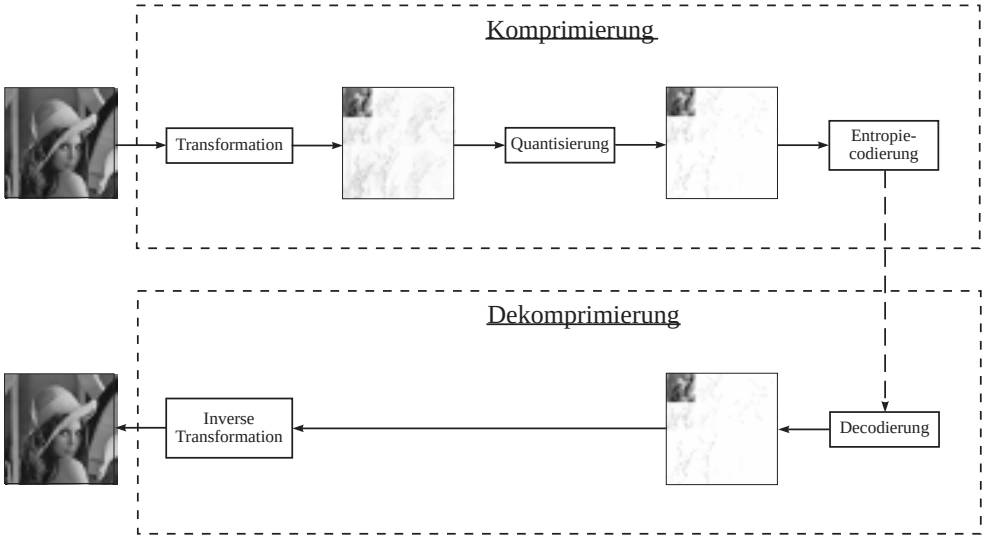


Abbildung 4.1: Bildkompression mittels Transformationscodierung

4.1.1 Transformation

In diesem Schritt wird das Bild mit Hilfe einer geeigneten Transformation in eine andere Darstellung umgewandelt, wobei bei JPEG eine DCT verwendet wird, während wir uns hier mit der Wavelet-Transformation befassen. Es existieren natürlich auch noch diverse andere Transformationsarten, wie z. B. die Walsh-Transformation oder die Karhunen-Loéve-Transformation; auf diese soll hier aber nicht näher eingegangen werden.

Das Ziel einer Transformation ist es, die Bildpunkte zu dekorrelieren; dadurch entstehen dann im Transformationsraum viele Werte, die ungefähr null sind, womit eine kompakte Speicherung möglich wird. Eine für die Bildkompression gut geeignete Transformation sollte folgende Eigenschaften haben[2]:

1. Unabhängigkeit von den zu transformierenden Daten.

[2] siehe auch [7, Seite 109]

2. es gibt einen schnellen Algorithmus um die Transformationskoeffizienten zu berechnen, also einen mit Aufwand $O(n)$ oder $O(n \log n)$.

3. eine Dekorrelation der Daten soll auch mit großen Datenmengen noch möglich sein.

Die erste Eigenschaft wird z. B. von der Karhunen-Loéve-Transformation nicht erfüllt; die Vorgehensweise bei der Transformation ist abhängig von den verwendeten Daten. Die schnelle Fourier-Transformation bzw. die bei JPEG verwendete DCT erfüllen die ersten beiden Eigenschaften, aber nicht immer die letzte, weshalb JPEG auch eine auf 8×8-Matrizen basierende DCT verwendet. Der tiefere Grund hierfür liegt in den Basisfunktionen, die zwar im Frequenzraum lokal sind, im Ortsbereich hingegen überhaupt nicht; durch die genannte Beschränkung auf einen 8×8-Punkte großen Bereich wird diese Lokalität künstlich erzeugt, wodurch aber auch die für JPEG typischen Blockeffekte bei hohen Kompressionsraten entstehen.

Die hier verwendete Wavelet-Transformation erfüllt dagegen alle drei Eigenschaften, ohne daß eine Anpassung wie bei der Fourier-Transformation vorgenommen werden muß. Im Transformationsschritt wird also einfach eine schnelle Wavelet-Transformation mit dem gewählten Basiswavelet durchgeführt.

4.1.2 Quantisierung

Auf die Transformation eines Bildes folgt die Quantisierung der entstandenen Transformationskoeffizienten. Dieser Schritt ist prinzipiell unabhängig von der Art der verwendeten Transformation, obwohl bei fortgeschrittenen Quantisierungsverfahren durchaus auf die Zusammenhänge der Koeffizienten untereinander geachtet wird.

Eine Quantisierung ist schon allein deshalb nötig, da bei der Transformation nahezu alle denkbaren Werte in Form von Fließkommazahlen entstehen können. Das Mindeste was man also tun sollte ist das Runden dieser reellen Zahlen auf ganze Zahlen, damit man beim Speichern nicht zu viel Platz verschwendet. Wie man bereits bei dieser sehr einfachen Art der Quantisierung sehen kann, ist dieser Schritt irreversibel. Alle Verluste, die bei einem Verfahren zur Bildkompression mittels Transformationen entstehen, kommen prinzipiell nur bei der Quantisierung zustande[3]. Aus diesem Grund hat die Quantisierung auch großen Einfluß auf die Bildqualität: Je grober man quantisiert, desto höher wird die erreichbare Kompressionsrate und desto schlechter wird die Bildqualität.

Es muß hierbei zwischen der Art der Quantisierung (linear, nichtlinear) sowie der Art der Daten, mit denen der Quantisierer operiert (Skalare oder Vektoren)[4] unterschieden werden.

[3] wenn man von Rundungsfehlern durch begrenzte Stellenzahl sowie Verlusten durch eine begrenzte Anzahl an Daten absieht

[4] siehe auch [19, Seite 376ff. und 415ff.]

Nun soll zuerst auf die skalare Quantisierung eingegangen werden:

lineare Quantisierung Die skalare lineare Quantisierung[5] ist die einfachste und auch am häufigsten verwendete Art der Quantisierung. Bei anschließender Entropiecodierung, wie sie ja meist durchgeführt wird, ist sie auch nahezu optimal [19]. Man unterteilt hierbei den durch die Koeffizienten vorgegebenen Wertebereich in 2^n gleich große Intervalle, die sogenannten *Bins*. Alle in einem Intervall liegenden Werte werden dann auf die Mitte des Intervalls abgebildet, wodurch man nach der Quantisierung nur noch 2^n verschiedene Werte hat. Zur Speicherung sind dann n Bit nötig.

Dasjenige Intervall, in dem der Wert Null liegt, spielt dabei allerdings eine besondere Rolle: erstens wird es um die Null herum zentriert und alle anderen Intervalle schließen sich dann links und rechts davon an; zweitens kann das Nullintervall als einziges größer sein als die anderen (i.a. doppelt so groß). Diese Sonderrolle liegt bereits in der Transformation begründet: Es sollen ja bereits hier sehr viele Werte entstehen, die gleich null sind oder doch nahe bei null liegen. Alle entstandenen Nullen werden ja nach der Quantisierung einer Lauflängencodierung unterzogen, weshalb sich viele Nullkoeffizienten günstig auf die Kompressionsrate auswirken ohne jedoch die Bildqualität zu sehr zu beeinträchtigen. Die Vorgehensweise bei der linearen Quantisierung ist in Abbildung 4.2 nochmals dargestellt.

nichtlineare Quantisierung Bei der nichtlinearen Quantisierung ist die Vorgehensweise prinzipiell die gleiche wie bei der linearen, wobei sich nun aber die Breite der Bins ändert und nicht gleich bleibt. Diese Änderung der Intervallgröße hat den Vorteil, daß man in Bereichen, in denen das menschliche Auge empfindlicher ist, feiner quantisieren kann als in weniger empfindlichen Bereichen. Auch eine Anpassung an die Verteilung der Koeffizienten ist möglich: liegen diese nahe zusammen, so wird das Intervall kleiner; liegen die Werte weit auseinander, so kann das Intervall größer werden.

Ein optimaler Quantisierer ist der nichtlineare Lloyd-Max-Quantisierer, der an die Verteilung der Koeffizienten angepaßt wird. Er minimiert die Varianz des Quantisierungsfehlers. Zur exakten Quantisierung muß dann allerdings auch die Wahrscheinlichkeitsverteilung der Koeffizienten bekannt sein. Eine Implementierung ist daher auch um einiges aufwendiger als bei einem linearen Quantisierer.

Der Unterschied zwischen skalaren und Vektorquantisierern liegt in der Art, wie die Daten behandelt werden: ein skalarer Quantisierer verarbeitet einzelne Werte, die im Normalfall auf die Mitte eines Quantisierungsintervalls abgebildet werden. Ein Vektorquantisierer hingegen faßt mehrere Werte zu einem Vektor zusammen und bildet diese auf einen neuen Wert ab. Hierzu müssen dann natürlich

[5] im englischen auch als *uniform quantizers* bezeichnet

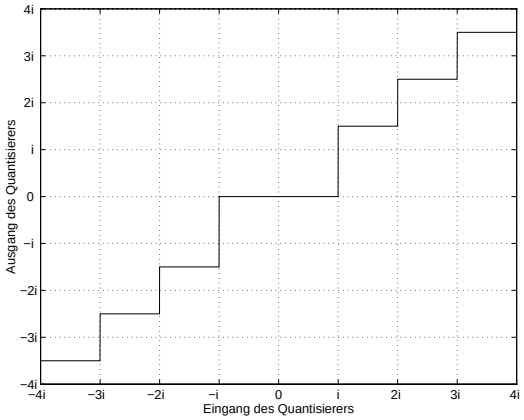


Abbildung 4.2: Lineare Quantisierung

Koeffizienten verwendet werden, die miteinander in Zusammenhang stehen. Dies ist z. B. der Fall, wenn man die bei der schnellen Wavelet-Transformation entstehenden Frequenzbänder betrachtet. Diese sind untereinander natürlich korreliert: eine horizontale Kante in einem Band tritt beispielsweise immer im tiefpaßgefilterten Bild auf und außerdem in jedem Band, welches in vertikaler Richtung hochpaßgefiltert wurde. Vektorquantisierer sind nicht ganz einfach zu handhaben, und man sollte sich daher überlegen, ob das Ergebnis den Implementierungsaufwand rechtfertigt.

Da in dem in dieser Arbeit vorgestellten Programm nur skalare lineare Quantisierung verwendet wird, möchte ich nun auch nicht weiter auf die Vektorquantisierung eingehen und den Leser statt dessen auf die Literatur verweisen [19].

Es folgen noch einige Anmerkungen, die sich speziell auf die Quantisierung der bei der schnellen Wavelet-Transformation entstehenden Koeffizienten beziehen:

- die Bänder mit den höchsten Frequenzen können meist ohne große Verluste auf null gesetzt werden; dies trifft speziell auf das im ersten Schritt sowohl in vertikaler als auch in horizontaler Richtung hochpaßgefilterte Bild zu.
- Oft ist es sinnvoll, die in verschiedenen Transformationsschritten entstandenen Koeffizienten getrennt zu quantisieren, da die Werte hier in etwa im gleichen Größenbereich liegen. Die Quantisierung wird dann letztendlich feiner.

4.1.3 Entropiecodierung

Die Entropiecodierung ist der letzte Schritt bei der Transformationscodierung von Bildern. Die quantisierten Transformationskoeffizienten sollen ja mit möglichst wenig Platzbedarf gespeichert werden. In der Bildkompression werden hierzu oft

zwei Standardverfahren verwendet, die hintereinandergeschaltet werden. Zuerst werden die Koeffizienten *lauflängencodiert*; die so entstandenen neuen Symbole unterzieht man dann noch einer *Huffman-Codierung*. Selbstverständlich existieren noch andere Verfahren, die meist wesentlich komplizierter sind als die hier vorgestellten. Dazu möchte ich auf ein Verfahren hinweisen, welches speziell auf die Codierung der bei der schnellen Wavelet-Transformation entstehenden Teilbänder zugeschnitten ist: der *EZW-Algorithmus*[6]. Eine genaue Beschreibung dieses Algorithmus würde an dieser Stelle allerdings zu weit führen; er ist in *Wavelets and Subband Coding* [19] genauer erläutert.

Nun zu den beiden Techniken, die besonders verbreitet sind:

Lauflängencodierung Die Lauflängencodierung[7] ist in vielen Büchern zur digitalen Bildverarbeitung beschrieben [2, 8], da sie ein einfaches und doch wirkungsvolles Verfahren zur Verringerung der Redundanz ist. Hierzu werden k aufeinanderfolgende Koeffizienten gleichen Wertes (W) zu einem Tupel (W, k) zusammengefaßt. Man bezeichnet k dann als *Lauflänge*. Wie man sieht, ist dieses Verfahren besonders effizient, wenn man sehr große Bereiche hat, die den gleichen Wert haben. Dies wurde ja auch bei den vorangegangenen Schritten Transformation und Quantisierung berücksichtigt: es sollten dabei unter anderem möglichst viele Nullkoeffizienten entstehen.

Das Verfahren ist natürlich nicht mehr effizient, wenn jeder Koeffizient einen anderen Wert hat. Dies ist bei den Koeffizienten der Wavelet-Transformation dann der Fall, wenn diese nicht null sind; solche treten meist nur einzeln auf. Daher werden auch oftmals nur diejenigen Koeffizienten lauflängencodiert, die den Wert Null haben, während die anderen auf normale Weise gespeichert werden.

Huffman-Codierung Die Huffman-Codierung wird oft anschließend an eine Lauflängencodierung durchgeführt. Man erreicht dadurch minimale Redundanz mit Codewörtern variabler Länge. Auf dieses Verfahren soll hier nicht weiter eingegangen werden, da es nicht auf Anwendungen in der Bildverarbeitung beschränkt ist und in vielen Büchern bereits ausführlich beschrieben wurde [9, 19].

4.2 Eigenschaften von Wavelets für die Bildkompression

Bei der Wavelet-Transformation hat man im Gegensatz zur Fourier-Transformation nicht nur eine Basisfunktion, sondern es existieren vielmehr sehr viele ver-

[6] **E**mbedded **Z**ero-tree **W**avelet
[7] engl.: **R**un **L**ength **E**ncoding (RLE)

schiedene Basiswavelets, welche auch sehr unterschiedliche Eigenschaften besitzen. An dieser Stelle soll nun kurz erläutert werden, welche dieser Eigenschaften[8] ein Wavelet besitzen sollte, damit man bei der Kompression gute Ergebnisse erzielt[9].

Kompakter Träger Diese Eigenschaft ist für die Berechnung der diskreten Wavelet-Transformation sehr wichtig, da nur bei Wavelets mit kompaktem Träger die Filter eine endliche Anzahl an Koeffizienten haben. Man kann zwar auch andere Wavelets verwenden, wobei diese dann aber durch eine endliche Zahl von Filterkoeffizienten angenähert werden müssen. Damit ist dann natürlich keine exakte Rekonstruktion mehr möglich.

Glattheit Je glatter ein Wavelet ist, desto besser ist die Lokalität der Filter im Frequenzbereich. Außerdem sind durch Quantisierung der Transformationskoeffizienten entstehende Fehler umso besser erkennbar, je weniger glatt das verwendete Wavelet ist; dies gilt insbesondere für das Nullsetzen von Koeffizienten.

Anzahl der verschwindenden Momente Je mehr verschwindende Momente ein Wavelet hat, desto kleiner werden die Koeffizienten in Bereichen eines Bildes, wo dieses glatt ist; in nicht glatten Bildbereichen hingegen werden die Koeffizienten größer. Für die Verwendung von biorthogonalen Wavelets gilt: Es besteht eine Verbindung zwischen der Glattheit des Wavelets und der Anzahl der verschwindenden Momente des zugehörigen dualen Wavelets.

Filterlänge Prinzipiell sind Filter mit wenigen Koeffizienten schon aus Gründen der Rechenzeit zu bevorzugen. Man muß allerdings einen Kompromiß finden zwischen Filterlänge und Anzahl der verschwindenden Momente sowie Glattheit des Wavelets, da diese mit zunehmender Filterlänge ebenfalls zunehmen.

4.3 Anwendungen der Wavelet-Bildkompression in der Praxis

Obwohl für die Kompression von Bildern mit Wavelets noch kein standardisiertes Dateiformat existiert, wie es mit JPEG für die DCT der Fall ist, wird es bereits in vielen Fällen verwendet, da man gegenüber JPEG einige Vorteile hat. Diese liegen vor allem in der Möglichkeit auch künstliche Bilder wie technische Zeichnungen

[8] siehe auch [7, Seite 112f.] sowie [19, Seite 265ff., Seite 412ff.]

[9] damit also möglichst viele Koeffizienten entstehen, die den Wert Null haben

oder, wie es beim FBI[10] der Fall ist, Fingerabdrücke mit hohen Kompressionsraten in guter Bildqualität zu speichern. JPEG ist hierzu praktisch nicht geeignet.

Wavelet-Bildkompression wird bereits in folgenden Bereichen eingesetzt:

- beim FBI zur effizienten Speicherung von Fingerabdrücken,
- bei der Übertragung von Bildern im Internet mit Hilfe eines Netscape-Plug-Ins und
- als Ersatz für M-JPEG[11] bzw. MPEG[12] zur Kompression von Filmen bei Videoschnittsystemen

Es gibt auch bereits einige Firmen, welche Software zur Bildkompression mit Wavelets anbieten.

Ich möchte nun kurz auf den sogenannten FBI WSQ[13] Standard zur Kompression von Fingerabdrücken eingehen [11]. Das FBI begann mit dem Sammeln von Fingerabdrücken im Jahr 1924. Alle Abdrücke werden seit dieser Zeit mit Tinte auf Papierkarten gemacht, wobei jede dieser Karten nicht nur einen Abdruck enthält, sondern insgesamt 14: von jedem Finger einen der durch Abrollen der Finger entsteht; zusätzlich von jedem Daumen sowie von der kompletten linken und rechten Hand jeweils einen flachen Abdruck. Seitdem haben sich über 200 Millionen dieser Karten angesammelt, womit das Auffinden eines bestimmten Abdrucks nicht leicht war. Auch die schnelle Übertragung von einem Ort zum anderen war unbefriedigend gelöst, da sie mit Hilfe von Faxgeräten erfolgte, wobei natürlich die Qualität stark litt. Man mußte sich also eine Möglichkeit überlegen, die Unmengen an Daten digital zu speichern. Die hierfür erforderliche Speicherkapazität darf dabei nicht vernachlässigt werden: Da ein einzelner Abdruck mit einer Auflösung von 500 dpi mit 256 Graustufen digitalisiert wird, entstehen durch eine einzige Karte ungefähr 10 MB Daten, womit man insgesamt auf eine Menge von 2000 Terabyte kommt.

Man wollte bei den Bildern verständlicherweise möglichst hohe Kompressionsraten erreichen; mit verlustfreien Verfahren käme man dabei auf Kompressionsraten von ca. 2:1, weshalb auch verlustbehaftete Kompressionsverfahren in Betracht gezogen wurden, sofern dadurch keine sichtbaren Qualitätsverluste entstehen. Dadurch kam JPEG nicht mehr in Frage, da es für Bilder mit den Charakteristiken von Fingerabdrücken nicht gerade gut geeignet ist. Nach einigen Testreihen kam man zu dem Schluß, daß sich mit einem auf Wavelets basierenden Verfahren Kompressionsraten von 15:1 ohne große Verluste erreichen lassen.

[10] Federal Bureau of Investigation
[11] **M**otion-**JPEG**
[12] **M**otion **P**icture **E**xperts **G**roup
[13] FBI Wavelet/Scalar Quantization

Nun kurz zur Durchführung der Codierung eines Bildes. Der FBI-Standard erlaubt die Verwendung von biorthogonalen Wavelets mit Filterlängen bis zur Größe 32. Tatsächlich angewandt werden zur Zeit symmetrische biorthogonale Wavelets mit Filtergrößen neun (Tiefpaß) und sieben (Hochpaß)[14]. Bei der Transformation wird allerdings nicht die hier vorgestellte Zerlegung eines Bildes in vier Frequenzbänder verwendet (bei jedem Schritt); vielmehr werden diese noch weiter aufgeteilt, um eine genauere Quantisierung zu ermöglichen: bei hohen Frequenzen ist die Unterteilung grober als bei tiefen Frequenzen. Zur Reduktion der Daten wird dann angenommen, daß die Bänder mit den höchsten Frequenzen[15] alle null sind; diese werden weder berechnet noch übertragen.

Das Problem der Anpassung der Daten wegen der sonst entstehenden Artefakte an den Rändern wurde hier mittels eines Verfahren gelöst, welches unter dem Namen *SET*[16] bekannt ist. Dabei werden die Daten an den Rändern auf eine besondere Weise symmetrisch erweitert, die es erlaubt, eine verlustfreie Transformation durchzuführen und trotzdem nur soviele Koeffizienten übertragen zu müssen, wie das Bild Pixel besitzt. Zusätzlich wird auch die Beschränkung auf Bilder der Größe 2^n aufgehoben; es können ohne weitere Probleme Bilder jeglicher Größe verwendet werden.

Eine genauere Beschreibung (oder auch Implementierung) dieser Art der Datenanpassung ist leider aufgrund fehlender Literatur nicht möglich.

Die Quantisierung erfolgt für jedes Frequenzband getrennt, wobei eine einfache lineare Quantisierung durchgeführt wird. Die Breite des Intervalls um null kann dabei von der Breite der restlichen Intervalle verschieden sein. Beide Intervallgrößen müssen für jedes der Bänder übertragen werden, da sie für jedes Bild neu bestimmt werden.

An die Quantisierung schließt sich eine Huffman-Codierung an, wobei die Nullen vorher noch lauflängencodiert werden. Die Huffman-Codierung wurde stark an diejenige angelehnt, wie sie auch bei JPEG Verwendung findet. Die Tabelle der Huffman-Symbole findet sich in [11].

Es existieren außer der Bildkompression auch noch viele andere Anwendungen, bei denen Wavelets bereits eingesetzt werden; ich möchte hier nur eine kleine Auswahl aufzählen:

- Kantendetektion in Bildern bei der Mustererkennung.
- Optische Verbesserung von Bildern durch Entfernen von Rauschen; der Einsatzbereich geht hierbei bis hin zur Nachbearbeitung von Satellitenbildern und astronomischen Aufnahmen.

[14] die Filterkoeffizienten sind in [11] zu finden

[15] bei dem in Kapitel 3.5.2 vorgestellten Verfahren entspricht das dem Teil der Koeffizienten, der durch zweimalige Hochpaßfilterung entstanden ist

[16] **S**ymmetric **E**xtension **T**ransform

- Lösen von partiellen Differentialgleichungen.
- Analyse von Fraktalen.
- Analyse von Signalen.

4.4 Erweiterung auf Kompression von Farbbildern

In dieser Diplomarbeit wurde sowohl bei der Beschreibung der Algorithmen zur Wavelet-Transformation als auch bei der Implementierung (siehe Kapitel 5) von Graustufenbildern ausgegangen; dies ist zur Verdeutlichung der prinzipiellen Funktionsweise der Bildkompression mit Wavelets auch völlig ausreichend. Da aber in der praktischen Anwendung häufig Echtfarbbilder[17] zu verarbeiten sind, möchte ich kurz erläutern, wie diese behandelt werden müssen.

Obwohl es durchaus möglich wäre im RGB-Format vorliegende Bitmaps so zu behandeln, als wären es drei Graustufenbilder, ist dies nicht gerade die günstigste Vorgehensweise. Der Grund hierfür ist, daß man bei Umrechnung des RGB-Wertes eines Pixels in ein anderes, für die Bildkompression besser geeignetes, Farbsystem bereits vor der eigentlichen Transformation eine Datenreduktion vornehmen kann, wodurch schon an dieser Stelle ein Kompressionseffekt entsteht. Hierfür ist es notwendig ein Farbsystem zu verwenden, welches eine Farbe nicht als RGB-Wert (drei Farbinformationen) darstellt, sondern getrennt als Helligkeits- und Farbsignale (Luminanz und Chrominanz).

Diese Trennung von Helligkeitsinformation und Farbe eines Bildpunktes hat folgenden Vorteil: Da das menschliche Auge Helligkeitsunterschiede zwischen zwei Bildpunkten wesentlich besser wahrnimmt als Farbunterschiede, kann man hier durch weglassen von Farbinformation eine (natürlich verlustbehaftete) Kompression erreichen.

Zunächst aber einige Worte zum verwendeten Farbsystem: YUV. In diesem System wird ein Farbwert, genau wie bei RGB, mit drei Werten angegeben; einem Helligkeitswert (Y) und zwei Farbwerten (U und V). Diese lassen sich aus den RGB-Werten wie folgt berechnen[18]:

$$\begin{pmatrix} \mathrm{Y} \\ \mathrm{U} \\ \mathrm{V} \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,147 & -0,289 & 0,436 \\ 0,615 & -0,515 & -0,100 \end{pmatrix} \cdot \begin{pmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{pmatrix} \tag{4.1}$$

Die Farbwerte U bzw. V sind also (gewichtete) Differenzsignale aus dem Blau-

[17] also Bilder mit einer Farbtiefe von 24 Bit: je ein Byte für den Rot-, Grün- und Blauwert eines Pixels (RGB-Format)

[18] vgl. [17]

bzw. Rotwert und dem Helligkeitssignal Y:

$$\begin{aligned} \mathrm{U} &= 0,493 \cdot (\mathrm{B} - \mathrm{Y}) \\ \mathrm{V} &= 0,877 \cdot (\mathrm{R} - \mathrm{Y}) \end{aligned} \tag{4.2}$$

Die Umrechnung von YUV nach RGB ist ebenso möglich:

$$\begin{pmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{pmatrix} = \begin{pmatrix} 1,000 & 0,000 & 1,140 \\ 1,000 & -0,396 & -0,581 \\ 1,000 & 2,029 & 0,000 \end{pmatrix} \cdot \begin{pmatrix} \mathrm{Y} \\ \mathrm{U} \\ \mathrm{V} \end{pmatrix} \tag{4.3}$$

Nach der Umrechnung der RGB-Werte aller Bildpunkte erhält man die YUV-Darstellung, wobei bis auf Rundungsfehler (man will ja für jeden der drei Werte nur ein Byte benutzen) keine Verluste auftreten. Jetzt wird eine Reduktion der Daten vorgenommen. Man behält für jeden Pixel das Helligkeitssignal Y bei, die Farbinformationen U und V dagegen werden nur für jeden vierten Bildpunkt gespeichert (bzw. der Mittelwert aus vier Punkten). Dies wird auch als 4:1:1-Subsampling bezeichnet. Durch diese Vorgehensweise muß man an Stelle von 12 Byte für vier Pixel nur noch sechs Byte speichern, das heißt, man erhält allein dadurch einen Kompressionsfaktor von 2:1 ohne entscheidenden Qualitätsverlust. Dieses Verfahren ist natürlich nicht auf waveletkomprimierte Bilder beschränkt; es kann bei jedem verlustbehafteten Kompressionsverfahren verwendet werden und wird unter anderem auch bei JPEG eingesetzt.

Kapitel 5

Ein Programm zur Wavelet-Bildkompression

Um die in den vorangegangenen Kapiteln beschriebenen Verfahren auch einmal in der Praxis auszuprobieren und einige Auswertungen bezüglich der Leistungsfähigkeit der Bildkompression mit Wavelets zu machen, wurde ein Programm entwickelt, welches Graustufenbilder komprimieren kann. Die Entwicklung erfolgte unter Windows NT 4.0 mit Microsoft Visual C++ 4. Bevor ich aber auf Details der Implementierung eingehe, möchte ich zuerst den Leistungsumfang des Programms erläutern und anschließend kurz auf dessen Bedienung eingehen.

5.1 Funktionsumfang des Programms

Das Programm hat folgende Merkmale:

- Lauffähig unter Windows NT 4.0 und Windows 95
- Kompression von Bildern im Bitmap-Format mit 256 Graustufen
- Anzeige der Transformationskoeffizienten
- Auswahl des verwendeten Wavelets aus einer umfangreichen Liste
- Vorgabe der Iterationstiefe für die schnelle Wavelet-Transformation
- Wahl der Art der Datenanpassung
- Verschiedene Arten der Quantisierung
- Anzeige von Daten über die Qualität der durchgeführten Kompression sowie die dafür benötigte Rechenzeit
- Speicherung des komprimierten Bildes in einem eigenen Dateiformat (WLT-Format)

- Zeichnen des verwendeten Wavelets und der zugehörigen Skalierungsfunktion
- Speichern der berechneten Funktionswerte in einem von MATLAB lesbaren Dateiformat
- Speichern aller am Bildschirm angezeigten Bilder im Bitmap-Format
- Durchführung einer einfachen eindimensionalen Wavelet-Transformation mit Anzeige der Transformationskoeffizienten

Ich möchte außerdem darauf hinweisen, daß keine Implementierung biorthogonaler Wavelets erfolgte, da ein einfacher Austausch der zur Transformation und inversen Transformation verwendeten Filter nicht so einfach möglich ist, wie es in der Theorie aussieht.

An einigen Stellen in der Beschreibung ist dennoch von biorthogonalen Wavelets die Rede, da das Programm auf eine Erweiterung in dieser Hinsicht bereits vorbereitet wurde.

5.2 Bedienung

In diesem Kapitel soll die Bedienung des Programms beschrieben werden. Dazu werden im ersten Teil des Kapitels zunächst alle verfügbaren Menüpunkte erläutert. Im daran anschließenden Teil werde ich darauf eingehen, wie man genau vorgehen muß, wenn man ein Bild komprimieren möchte.
Es gelten folgende Konventionen:

- Der Name eines Menüpunktes wird in **Fettdruck** dargestellt
- Verschiedene Ebenen eines Menüs werden durch Schrägstrich (/) getrennt, also z. B. **Datei/Öffnen**

5.2.1 Menüpunkte

Abbildung 5.1 zeigt, wie sich das Programm nach dem Start und anschließendem Laden eines Bildes[1] präsentiert.
In der Menüleiste sind folgende Punkte verfügbar:

Datei Hier befinden sich alle Menüpunkte, die zum Laden und Speichern eines Bildes nötig sind.

Bearbeiten Dieses Menü dient dem Einfügen von Bildern aus und in die Windows-Zwischenablage.

[1] mit dem Menüpunkt **Datei/Öffnen**

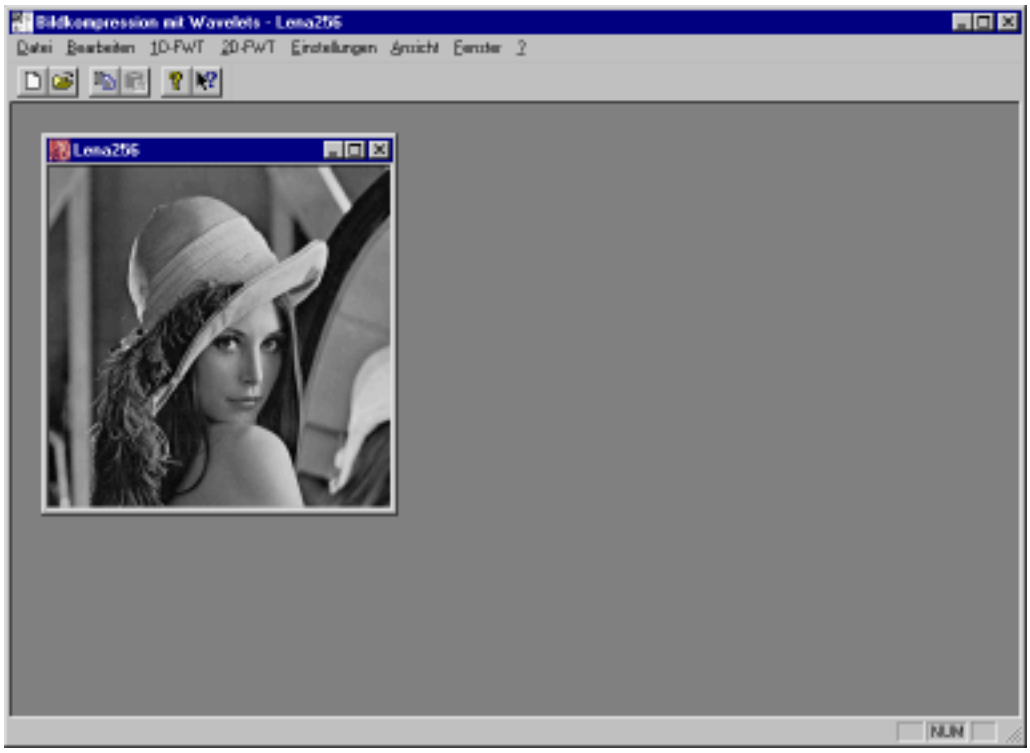


Abbildung 5.1: Das Programm nach dem Laden eines Bildes

1D-FWT Mit diesen Menüpunkten kann man einerseits das aktuelle Wavelet und die dazugehörende Skalierungsfunktion zeichnen lassen und andererseits eine einfache eindimensionale Wavelet-Transformation durchführen.

2D-FWT Alle Operationen zum Komprimieren eines Bildes oder zum Anzeigen der dabei entstehenden Transformationskoeffizienten befinden sich hier.

Einstellungen Zum Einstellen der Parameter für die Kompression benutzt man dieses Menü.

Ansicht enthält Menüpunkte zum Ein- und Ausschalten von Symbol- und Statusleiste.

Fenster Dies ist ein Standardmenü von Windows; es dient dem Anordnen von Fenstern auf der Arbeitsoberfläche.

? Dieser Menüpunkt enthält den Aufruf für die Online-Hilfe

Die in diesen Menüs enthaltenen Untermenüs sollen nun genauer betrachtet werden.

5.2.1.1 Datei

Neu erstellt ein leeres Dokument. Dieser Menüpunkt wird nur dann benötigt, wenn man entweder eine eindimensionale Wavelet-Transformation durchführen oder den Graphen eines Wavelets zeichnen lassen will (siehe Menü **1D-FWT**).

Öffnen dient zum Laden eines Bildes im Bitmap-Format oder im programmeigenen WLT-Format[2].

Schließen schließt das Fenster eines Dokuments.

Speichern unter Dieser Menüpunkt wird zum Speichern eines Fensterinhalts als Bitmap oder eines komprimierten Bildes im WLT-Format verwendet. Alle angezeigten Grafiken können auch als Bitmap gespeichert werden. Für den Ausdruck eines Wavelet- oder Skalierungsfunktionsgraphen in hoher Qualität sollte allerdings statt dessen der Punkt **Datei/MATLAB Export** verwendet werden, sofern MATLAB zur Verfügung steht.

Bei der Speicherung als komprimiertes Bild (*.WLT) wird nach der Speicherung ein Informationsfenster geöffnet, welches den erreichten Kompressionsgrad anzeigt.

MATLAB Export Mit Hilfe dieses Punktes kann ein Funktionsgraph in einem von MATLAB lesbaren Format gespeichert werden. Dieser Punkt wurde implementiert, da es mit MATLAB möglich ist einen gezeichneten Graphen im Encapsulatet-Postscript-Format (EPS) zu speichern, wodurch ein Ausdruck in sehr hoher Qualität erfolgen kann. Die hierfür nötigen MATLAB-Befehle sind im Anhang D abgedruckt.

Exportiert werden können folgende Arten von Graphen:

- Wavelets (Menü **1D-FWT/Zeichne Wavelet**),
- Skalierungsfunktionen (Menü **1D-FWT/Zeichne Skalierungsfunktion**),
- eindimensionale Funktionen (Menü **1D-FWT/Zeichne Funktion**),
- Transformationskoeffizienten eindimensionaler Funktionen (Menü **1D-FWT/Zeichne Waveletkoeffizienten**) und
- rücktransformierte eindimensionale Funktionen (Menü **1D-FWT/ Zeichne rücktransformierte Funktion**)

Beenden beendet das Programm.

[2] in diesem Format werden waveletkomprimierte Bilder gespeichert

5.2.1.2 Bearbeiten

Kopieren Kopiert den Inhalt des aktuellen Fensters als Bitmap in die Windows-Zwischenablage; von dort aus kann das Bild mit anderen Programmen weiterverarbeitet werden.

Einfügen Fügt eine in der Zwischenablage enthaltene Bitmap in das aktuelle Fenster ein. Dieser Befehl steht nur für zwei Arten von Fenstern zur Verfügung: dasjenige, welches ein gerade geladenes Bild enthält und das Fenster, in dem das Ergebnis einer Kompression angezeigt wird. Er ist nicht für Fenster verfügbar, mit denen eine eindimensionale Transformation durchgeführt wurde und für das Fenster, welches die Transformationskoeffizienten eines Bildes anzeigt (Menüpunkt **2D-FWT/Anzeige der Koeffizienten**).

5.2.1.3 1D-FWT

Die in diesem Menü enthaltenen Punkte dienen einerseits dem Zeichnen von Graphen des eingestellten Wavelets und der Skalierungsfunktion und andererseits zur Durchführung einer eindimensionalen Wavelet-Transformation anhand von einigen Testfunktionen.

Alle angezeigten Bilder können entweder als Bitmap abgespeichert werden (Menüpunkt **Datei/Speichern unter**) oder in einem von MATLAB lesbaren Format exportiert werden (Menüpunkt **Datei/MATLAB Export**).

Zeichne Wavelet Nach dem Anwählen dieses Punktes wird das aktuell eingestellte Wavelet (siehe Menü **Einstellungen/Wavelet**) gezeichnet. Die Optionen zum Zeichnen können in dem Dialog geändert werden, welcher nach Anklicken des Menüs **Einstellungen/1D-Transformation/Zeichenoptionen** erscheint.

Zeichne Skalierungsfunktion Die zum Wavelet gehörende Skalierungsfunktion wird entsprechend den Einstellungen im Dialog des Menüpunktes **Einstellungen/1D-Transformation/Zeichenoptionen** gezeichnet.

orthogonales Wavelet zeichnen Dieser Menüpunkt kann ein- und ausgeschaltet werden; die Einstellung hat nur Auswirkungen, wenn im Menü **Einstellungen/Wavelet** biorthogonale Wavelets ausgewählt wurden. Ist der Menüpunkt eingeschaltet, so wird das zur Zerlegung verwendete Wavelet gezeichnet, sonst das zur Synthese verwendete duale.

Zeichne Funktion Dieser Punkt ist nicht anwählbar, wenn das aktuelle Dokument ein Bild enthält. Er dient zum Testen einer eindimensionalen Transformation. Will man diesen Punkt verwenden, so ist zunächst ein neues Dokument mit dem Menüpunkt **Datei/Neu** zu erstellen. Anschließend kann

mit Hilfe von **Zeichne Funktion** die im Menü **Einstellungen/1D-Transformation/Funktion** eingestellte Funktion gezeichnet werden. Mit den beiden folgenden Menüpunkten läßt sich diese dann transformieren.

Zeichne Waveletkoeffizienten Dieser Punkt dient der Transformation einer eindimensionalen Funktion und kann folglich nur dann ausgewählt werden, wenn vorher eine solche mit Hilfe des Menüpunktes **1D-FWT/Zeichne Funktion** gezeichnet wurde. Es erscheint dann ein neues Fenster, welches die Transformationskoeffizienten der Funktion enthält.

Zeichne zurücktransformierte Funktion Mit diesem Menüpunkt kann eine eindimensionale inverse Wavelet-Transformation durchgeführt werden. Deshalb kann er auch nur dann angewählt werden, wenn vorher mit dem vorangegangenen Punkt eine Funktion transformiert wurde. Es erscheint ein neues Fenster, welches das Ergebnis der Rücktransformation anzeigt.

Anmerkung: Da für die eindimensionale Transformation aus Zeitgründen keine Datenanpassung implementiert wurde, kann eine transformierte Funktion nur bei Verwendung des Haar-Wavelets exakt rekonstruiert werden. Bei anderen Basiswavelets entstehen teilweise so große Verluste, daß die ursprüngliche Funktion nicht mehr wiederzuerkennen ist. Etwas Abhilfe kann allerdings geschaffen werden, wenn man beim Zeichnen der Funktion sehr viele Daten verwendet (eine hohe Auflösung) und bei der inversen Transformation nur die am linken Rand entstehenden Werte beachtet. Dies ist möglich, da die genannten Randeffekte eben nur am rechten Rand auftreten.

5.2.1.4 2D-FWT

Die Unterpunkte dieses Menüs stellen Funktionen zur Transformation und Quantisierung von Bildern zur Verfügung. Die Parameter der jeweiligen Menüpunkte können in **Einstellungen** verändert werden.

Transformation durchführen Mit Hilfe dieses Punktes wird das aktuelle Bild transformiert, quantisiert und zurücktransformiert. Anschließend wird das Ergebnis in einem neuen Fenster angezeigt. Um den erreichten Kompressionsgrad zu ermitteln, muß das Bild mit **Datei/Speichern unter** abgespeichert werden; nach dem Speichern wird der erreichte Kompressionsgrad angezeigt.

Anzeige der Koeffizienten transformiert das aktuelle Graustufenbild unter Verwendung des eingestellten Wavelets und zeigt anschließend in einem neuen Fenster die Transformationskoeffizienten an.

Quantisierung Dieser Punkt ist nur verfügbar, wenn ein Fenster aktiv ist, welches die Waveletkoeffizienten eines Bildes anzeigt. Diese werden dann quantisiert und im gleichen Fenster wieder angezeigt.

Inverse Transformation Auch dieser Punkt ist nur bei aktivem Fenster mit Transformationskoeffizienten anwählbar. Er zeigt das sich aus diesen durch inverse Transformation ergebende Bild in einem neuen Fenster an.

Skalennormalisierung für Anzeige Dieser Menüpunkt bezieht sich auf ein Fenster, welches Waveletkoeffizienten enthält. Aufgrund der beschränkten Farbanzahl von 256 Graustufen ist es bei mehreren Iterationen nämlich nicht mehr möglich die Transformationskoeffizienten so anzuzeigen, daß man auch in den Bändern mit sehr hohen Frequenzen noch Unterschiede erkennen kann, da die Koeffizienten betragsmäßig von Band zu Band immer größer werden. Daher erscheint es ohne Anpassung der Farben so, als ob alle Koeffizienten dieser Bänder null wären.

Mit Hilfe dieses Menüpunktes ist es nun möglich, für jeden Iterationsschritt eine getrennte Anpassung der Farben vorzunehmen, womit man dann auch wieder verschieden große Koeffizienten in den zuerst entstandenen Bändern erkennen kann. Das Einschalten dieses Punktes wirkt sich nur auf die Darstellung am Bildschirm aus, nicht jedoch auf die Koeffizienten selbst. Er ist standardmäßig aktiviert.

Bei Datenanpassung nur Bild anzeigen Wurde eine Art der Datenanpassung eingestellt, bei der mehr Daten entstehen, so wird das zu transformierende Bild viermal größer (siehe hierzu **Einstellungen/2D-Transformation**). Dieser Menüpunkt legt nun fest, ob im zurücktransformierten Bild nur der Datenausschnitt gezeigt werden soll, der dem ursprünglichen Bild entspricht, oder das gesamte vergrößerte Bild. Standardmäßig ist dieser Punkt aktiviert, womit nur der wichtige Bildausschnitt zu sehen ist.

negative Darstellung Dieser Punkt bezieht sich auf ein Fenster, das Transformationskoeffizienten enthält, weshalb er auch nur dann anwählbar ist, wenn ein solches aktiv ist. Aufgrund der Farbbelegung eines Graustufenbildes werden nämlich Nullkoeffizienten schwarz dargestellt und Koeffizienten mit großem Betrag sind weiß. Bei dieser Art der Anzeige ist es aber teilweise schwierig, diejenigen Koeffizienten zu erkennen, welche nicht null sind. Daher kann man mit diesem Menüpunkt die Farben invertieren, das heißt Nullkoeffizienten werden nun weiß dargestellt und alle anderen unter Ausnutzung der verfügbaren Graustufen mehr oder weniger schwarz. Diese Art der Anzeige ist besonders dann von Vorteil, wenn die Bitmaps ausgedruckt werden sollen. Sie wirkt sich ebenfalls nur auf die Bildschirmdarstellung aus, nicht auf die Koeffizienten selbst.

Ein sich aus dieser Farbdarstellung ergebende Nachteil soll aber nicht verschwiegen werden: Da nur 256 verschiedene Graustufen verfügbar sind, wird natürlich auch das mehrmals tiefpaßgefilterte Bild invertiert, wodurch es nicht mehr natürlich erscheint. Dies ist auch der Grund, weshalb dieser Punkt standardmäßig inaktiv ist.

Info Dieser Menüpunkt zeigt ein Dialogfenster an, in dem Informationen zum transformierten Bild zu sehen sind (siehe Abbildung 5.2). Der Punkt ist nur verfügbar, wenn ein Fenster aktiv ist, das ein zurücktransformiertes Bild enthält.

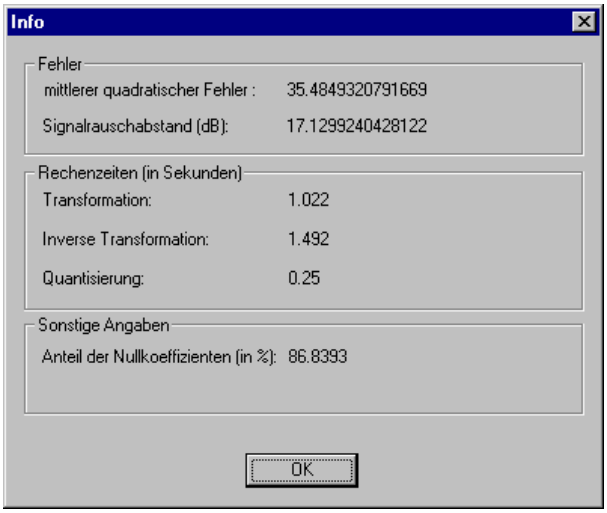


Abbildung 5.2: Der Dialog *Info*

5.2.1.5 Einstellungen

Das Menü **Einstellungen** enthält folgende Untermenüs:

Wavelet Hier wird das zur Transformation zu verwendende Wavelet eingestellt. Es erscheint ein Dialogfenster (siehe Abbildung 5.3), in welchem man auswählen kann, ob man eine orthogonale oder eine biorthogonale Transformation durchführen will. Je nachdem, welche Art gewählt wurde, kann man ein Wavelet zur Transformation aus der oberen Liste aussuchen, welches im orthogonalen Fall sowohl für die Transformation als auch für die Rücktransformation verwendet wird. Im biorthogonalen Fall wird auch die untere Auswahlliste aktiv; hier erscheint eine Auswahl von Dualen zum Wavelet in der oberen Liste. Die Einstellungen der unteren Liste werden für die inverse Transformation verwendet. Es ist zu beachten, daß sich der Inhalt der unteren Liste in Abhängigkeit von der oberen ändert.

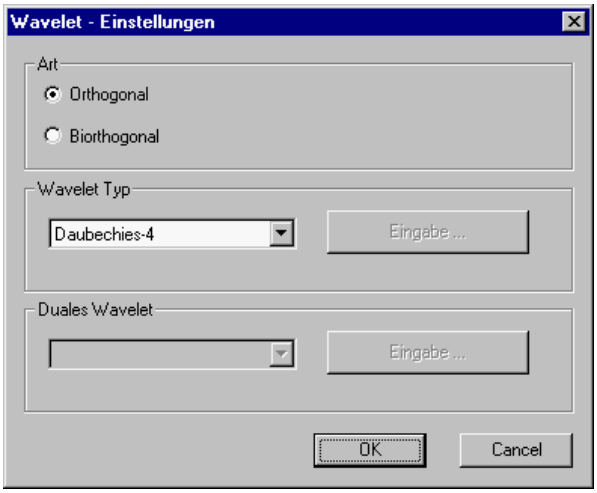


Abbildung 5.3: Der Dialog zur Auswahl eines Wavelets

1D-Transformation Dieser Punkt dient zum Einstellen der im Menü **1D-FWT** zusammengefaßten Operationen. Er enthält wiederum zwei Unterpunkte:

Zeichenoptionen Wird der Punkt **Zeichenoptionen** angewählt, so erscheint das in Abbildung 5.4 gezeigte Dialogfenster. Die hier mögli-

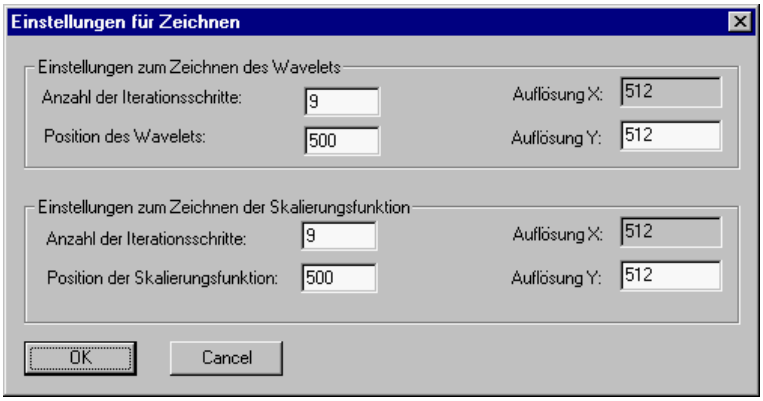


Abbildung 5.4: Der Dialog *Zeichenoptionen*

chen Einstellungen betreffen das Zeichnen des Wavelet- und Skalierungsfunktionsgraphen, wie bereits aus der Zweiteilung des Dialogs erkennbar ist. Die Parameter, die man ändern kann sind jeweils:

- Anzahl der Iterationen: Hier wird die Anzahl der zur Berechnung der Funktionswerte zu verwendenden Iterationsschritte eingegeben[3]. Es ist zu beachten, daß sich mit jedem Schritt mehr die Auflösung der entstehenden Grafik in X-Richtung verdoppelt,

[3]das Verfahren wurde in Kapitel 3.4 beschrieben

weshalb diese auch nicht direkt eingegeben werden kann, sondern aus der Zahl der Iterationsschritte berechnet wird.
Der Defaultwert dieses Feldes ist neun, was einer horizontalen Auflösung von 512 Punkten entspricht.

- Position des Wavelets bzw. der Skalierungsfunktion: Wie in Kapitel 3.4 erläutert wurde, wird zur Berechnung der Funktionswerte ein Startvektor verwendet, dessen Elemente mit Ausnahme eines einzigen alle null sind. Dasjenige, das den Wert Eins hat, gibt die Stelle an, in deren Umgebung der Graph entsteht. Eben diese Position wird hier angegeben. Dabei ist zu beachten, daß der eingegebene Wert zwischen eins und der im Feld *Auflösung-X* angezeigten Zahl liegen muß. Bei Fehleingabe wird eine entsprechende Meldung angezeigt.
  Der Defaultwert ist 500.
- Auflösung Y: Hier wird die vertikale Auflösung der entstehenden Bitmap angegeben.
  Der Defaultwert ist 512.

Funktion In dem hier erscheinenden Dialog (Abbildung 5.5) kann man eine Funktion auswählen, welche bei der eindimensionalen Transformation verwendet werden soll. Weiterhin kann die horizontale und

Abbildung 5.5: Der Dialog *Funktion*

vertikale Auflösung der Bitmap angegeben werden, in der der Funktionsgraph gezeichnet wird. Die Wert für die horizontale Auflösung entspricht dabei der Anzahl der für eine eindimensionale Transformation zur Verfügung stehenden Funktionswerte. Man kann den Funktionsgraphen mit Hilfe des Menüpunktes **1D-FWT/Zeichne Funktion** darstellen lassen. Anschließend ist eine Transformation der Funktion möglich (siehe Beschreibung des Menüs **1D-FWT**).

2D-Transformation Bei Anwahl dieses Menüpunktes erscheint der in Abbildung

5.6 gezeigte Dialog. Die hier gemachten Einstellungen beziehen sich nur

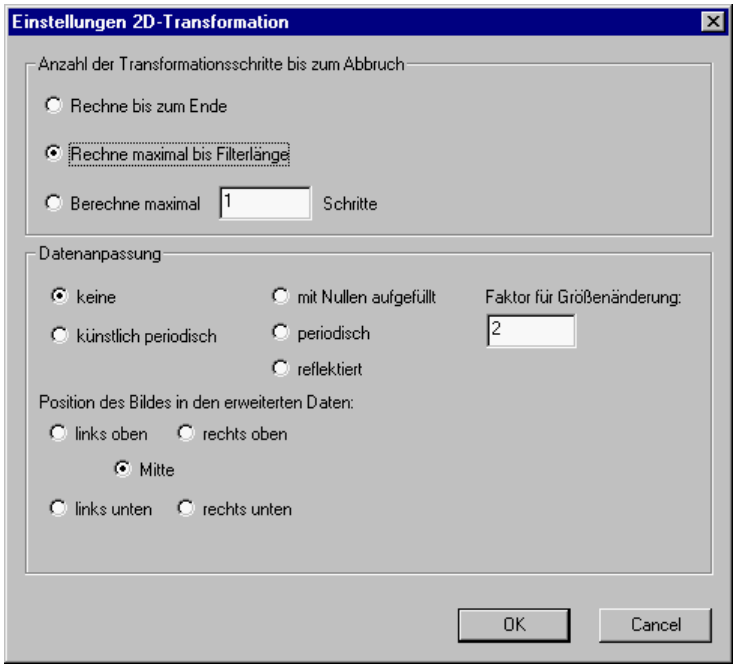


Abbildung 5.6: Der Dialog *2D-Transformation*

auf die im Menü **2D-FWT** stehenden Operationen. Beeinflussen lassen sich bei der Transformation eines Bildes die Tiefe der Berechnung (*Anzahl der Transformationsschritte*) sowie die Art der Datenanpassung (*Datenanpassung*):

- Anzahl der Transformationsschritte: Prinzipiell wird die schnelle Wavelet-Transformation so lange durchgeführt, bis das zu transformierende Bild nur noch einen Pixel groß ist. Da dies aber nicht immer erwünscht ist und ohne Anpassung der Filterkoeffizienten oftmals auch Probleme macht, kann hier angegeben werden, wie viele Schritte bei der Transformation berechnet werden. Die Auswahlmöglichkeiten sind: *Rechne bis zum Ende*, *Rechne maximal bis Filterlänge* und *Berechne maximal x Schritte*.

  Ist der erste Punkt angewählt, so wird die Transformation durchgeführt, bis nur noch ein Pixel übrig ist, sofern dies möglich ist (siehe dritter Punkt). Beim zweiten Punkt wird die Transformation beendet, wenn vom Bild nur noch so viele Pixel in X- oder Y-Richtung vorhanden sind wie das ausgewählte Wavelet Filterkoeffizienten hat. Im dritten Fall kann man die Anzahl der Schritte vorgeben; ist die Zahl hier größer als die maximale, so ist die Wirkung die gleiche wie beim ersten Punkt. Die maximale Anzahl ergibt sich, wenn man abzählt, wie oft die vertikale und horizontale Auflösung des Bildes

durch zwei teilbar ist; der kleinere der beiden Werte wird für die Zahl der Transformationsschritte hergenommen. Ist das Bild quadratisch und die Auflösung eine Zweierpotenz, so kann also bis zur Größe von einem Punkt gerechnet werden.

- Datenanpassung: Wie bereits erwähnt wurde, ist bei allen Wavelets, außer dem Haar-Wavelet, eine Anpassung der Daten nötig. Eine exakte Rekonstruktion des Bildes aus den Transformationskoeffizienten ist sonst unmöglich, da zu viele Koeffizienten an den Rändern verloren gehen würden. Für die Anpassung der Daten hat man folgende Alternativen zur Auswahl:
  - **keine** es wird keine Datenanpassung vorgenommen.
  - **künstlich periodisch** Die Daten werden durch Änderung der Faltungsmatrizen wie periodische Daten behandelt (siehe Kapitel 3.5.1); die Menge der zu transformierenden Bildpunkte bleibt dabei gleich.

  Bei den nächsten drei Methoden wird die Menge der zu transformierenden Bildpunkte durch die Datenanpassung vervierfacht[4], wodurch die Ergebnisse bezüglich der Rekonstruktion wesentlich besser werden. Dafür steigt die Rechenzeit bei der Transformation, und die Kompressionsrate verschlechtert sich zum Teil (siehe auch Kapitel 6.4). Bei allen drei Verfahren hat man die Möglichkeit, das Originalbild an fünf verschiedenen Stellen im neuen Datenbereich zu positionieren, wodurch man ermitteln kann, welchen Einfluß der Ort, an dem man auffüllt, auf die Transformation hat.
  - **mit Nullen aufgefüllt** Um das Originalbild herum werden alle Pixel auf null gesetzt.
  - **periodisch** Das Bild wird in alle Richtung periodisch fortgesetzt.
  - **reflektiert** Das Bild wird in jeder Richtung entsprechend gespiegelt fortgesetzt.

  Das Ergebnis der Datenanpassung mit den eben genannten drei Methoden ist in Abbildung 5.7 dargestellt.

Quantisierung Hier werden die Einstellungen für die Quantisierung der Transformationskoeffizienten vorgenommen. Das zugehörige Dialogfenster ist in Abbildung 5.8 dargestellt. Man hat folgende Möglichkeiten, die Quantisierung zu beeinflussen:

- **keine** Es wird keine Quantisierung durchgeführt, sondern mit allen verfügbaren Nachkommastellen gerechnet. Diese Einstellung dient dem Testen der Transformation, da nur ohne Quantisierung eine exakte

[4] es erfolgt eine Verdopplung der Auflösung in X- und in Y-Richtung

(a) Original (b) mit Nullen aufgefüllt

(c) periodisch (d) reflektiert

Abbildung 5.7: Datenanpassung durch Erweiterung des Bildes. Das Originalbild befindet sich hier immer in der Mitte des erweiterten Bildes

Rekonstruktion wenigstens prinzipiell möglich ist (bis auf Rundungsfehler und die Art der Datenanpassung). Mit dieser Quantisierungseinstellung können Bilder nicht gespeichert werden, weshalb beim Speichern die Koeffizienten auch automatisch auf ganze Zahlen gerundet werden.

- **nur auf ganze Zahlen runden** Mit dieser Einstellung wird als einziges eine Rundung der (reellen) Koeffizienten auf ganze Zahlen durchgeführt, womit diese als Integer gespeichert werden können.
- **ab folgendem Schwellwert Koeffizient auf null setzen** Hier kann ein Schwellwert angegeben werden, der positiv sein muß. Alle Koeffizienten, die betragsmäßig kleiner als dieser Schwellwert sind, werden auf null gesetzt. Alle anderen Koeffizienten werden auf ganze Zahlen gerundet.

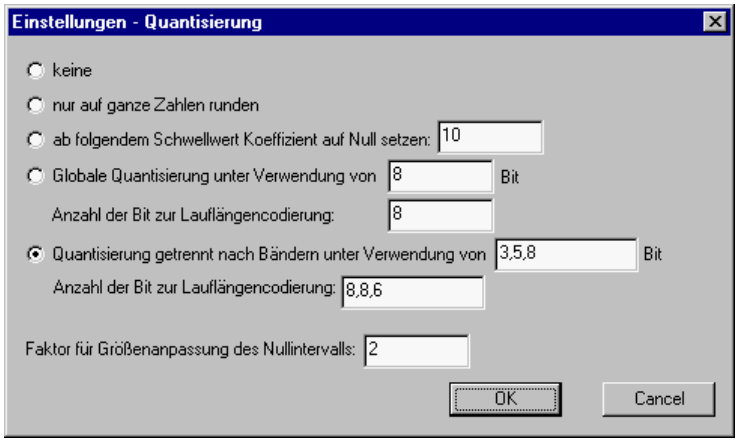


Abbildung 5.8: Der Dialog *Quantisierung*

- **Globale Quantisierung unter Verwendung von** k **Bit** Wählt man diese Einstellung, so kann man angeben, wie viele Bit man für das Speichern eines einzelnen Koeffizienten verwenden möchte, wobei die angegebene Anzahl für alle Koeffizienten wirksam ist. Aus dieser Angabe ergibt sich die Anzahl der entstehenden Intervalle zu 2^k. Dabei sind bis auf eines alle Intervalle gleich groß. Eine Ausnahme ist das um null herum zentrierte Intervall, an welches sich alle anderen links und rechts davon anschließen. Für die Größe des Nullintervalls kann in dem Eingabefeld **Faktor für Größenanpassung des Nullintervalls** ein Faktor angegeben werden, um den das Intervall größer (oder, wenn gewünscht auch kleiner) als die anderen sein soll (siehe letzter Punkt). Der kleinste hier angebbare Wert ist zwei, der größtmögliche ist 16 Bit.

  Weiterhin kann man im unteren der beiden Eingabefelder angeben, mit wievielen Bit die Lauflänge der Nullkoeffizienten codiert werden soll. Es ist nämlich meist nicht sinnvoll die gleiche Anzahl wie bei der Quantisierung zu verwenden, da bei wenigen Bit bei der Quantisierung mehr Nullkoeffizienten entstehen als bei einer großen Anzahl (dies ist natürlich auch noch von der Größe des Nullintervalls abhängig).

- **Quantisierung getrennt nach Bändern unter Verwendung von** k **Bit** Aufgrund der von Frequenzband zu Frequenzband größer werdenden Koeffizienten ist es meist vorteilhaft, jedes Band getrennt zu quantisieren. Dies ist mit dieser Einstellung möglich. Im zugehörigen Eingabefeld können für jedes Band die zu verwendenden Bit eingegeben werden. Die Zahlen müssen zwischen 2 und 16 liegen und durch Komma getrennt sein. Die Eingabe erfolgt von außen nach innen, das heißt die erste Zahl gibt die Bitanzahl für das durch den ersten Iterationsschritt entstandene Band an, die nächste für den zweiten usw.

Es muß übrigens nicht für jedes Band ein Wert vorhanden sein; sind weniger Zahlen enthalten als Iterationsschritte durchgeführt wurden, so wird die letzte für alle weiteren verwendet. Genau wie beim vorherigen Punkt sind alle Intervalle um das Nullintervall zentriert, wobei auch hier ein Faktor für die Größenänderung angegeben werden kann. Außerdem kann wiederum die Anzahl der Bit für die Codierung der Lauflänge für jeden Iterationsschritt getrennt eingegeben werden, wobei die gleichen Bedingungen gelten wie bei der Eingabe der Bit zur Quantisierung.

Beispiel: Eingegeben wurde: 2,3,8,14
Es wird also das erste Frequenzband mit zwei Bit, das zweite mit drei, das dritte mit acht und alle weiteren mit 14 Bit quantisiert.

Achtung: Bei der Eingabe ist es wichtig, daß sich keine Leerzeichen zwischen den Zahlen und den Kommata befinden. Leerzeichen am Anfang oder Ende der Eingabe haben dagegen keine negativen Auswirkungen.

- **Faktor für Größenanpassung des Nullintervalls** Der hier eingegebene Wert hat nur Auswirkungen, wenn entweder der Punkt **Globale Quantisierung unter Verwendung von** k **Bit** oder **Quantisierung getrennt nach Bändern unter Verwendung von** k **Bit** ausgewählt wurde. Dieser Wert gibt einen Faktor an, um den das um null zentrierte Intervall größer ist als alle anderen. Hier kann eine Fließkommazahl eingegeben werden, wodurch z. B. auch 1.3 mal größere Nullintervalle möglich sind. Der Standardwert ist Zwei.

5.2.2 Komprimierung eines Bildes

Nachdem nun im vorangegangenen Kapitel alle verfügbaren Menüpunkte beschrieben wurden, möchte ich jetzt noch kurz eine stichpunktartige Anleitung für die bei der Kompression eines Bildes nacheinander durchzuführenden Schritte geben:

- Laden eines Bildes mit **Datei/Öffnen** oder Einfügen eines Bildes aus der Zwischenablage mit **Bearbeiten/Einfügen**.
- Festlegen des zu verwendenden Wavelets mit **Einstellungen/Wavelet**.
- Einstellen der Zahl der Iterationsschritte und der Art der Datenanpassung durch Anwahl des Menüpunktes **Einstellungen/2D-FWT**.
- Gewünschte Art der Quantisierung im Menü **Einstellungen/Quantisierung** auswählen.

- Durchführen der Transformation, Quantisierung und inversen Transformation entweder in einem Schritt (Menüpunkt **2D-FWT/Transformation durchführen**) oder in Einzelschritten (Menüpunkte **2D-FWT/Transformationskoeffizienten anzeigen**, **2D-FWT/Quantisierung** und **2D-FWT/Inverse Transformation** in dieser Reihenfolge).

- Nun können durch Anwählen des Menüpunktes **2D-FWT/Info** Informationen zur benötigten Rechenzeit sowie dem Signalrauschabstand des komprimierten Bildes vom Original angezeigt werden. Damit dieser Menüpunkt angeklickt werden kann, muß das Fenster aktiv sein, welches das komprimierte Bild anzeigt.

- Nach dem Speichern des Bildes im WLT-Format (**Datei/Speichern unter**) wird der erreichte Kompressionsgrad ermittelt und angezeigt.

5.3 Aufbau und Funktionsweise des Programms

Bevor ich auf die Implementierung des Programms zu sprechen komme, möchte ich noch ein paar Vorbemerkungen machen:

1. Es soll hier nicht eine genaue Beschreibung des Programmcodes erfolgen; vielmehr möchte ich hier eine Übersicht über die grobe Struktur des Programms geben und auf die wichtigsten Teile kurz eingehen.

2. Vom Leser werden Kenntnisse in der objektorientierten Programmierung mit C++ sowie den Microsoft Foundation Classes (MFC) vorausgesetzt. Der grundlegende Aufbau eines MFC-Programms wird hier nicht weiter erläutert; näheres dazu findet sich in der Literatur [14].

Bevor ich auf den Aufbau des Programms eingehe, erfolgt die Beschreibung einiger Basisklassen, die im Programm immer wieder verwendet werden. Zunächst eine Übersicht:

Feld Diese Klasse wurde als Template realisiert, damit sie möglichst flexibel einsetzbar ist. Sie dient der einfacheren Handhabung zweidimensionaler Arrays.

Matrix Die Klasse *Matrix* ist von der Klasse *Feld* abgeleitet und erweitert diese um Operatoren, mit denen ein zweidimensionales Array als Matrix betrachtet werden kann (Addition, Subtraktion, Multiplikation, etc.). Sie ist ebenfalls ein Template.

Wavelet Hier werden die Operationen zum Durchführen einer ein- und zweidimensionalen Wavelet-Transformation bereitgestellt.

CWLT_Daten Diese Klasse dient zum Laden und Speichern eines transformierten Bildes.

CDib Die Klasse *CDib* wurde aus dem Buch *Inside Visual C++* [14] entnommen. Sie bietet grundlegende Funktionen, die zum Laden, Speichern und Anzeigen von Bitmaps erforderlich sind. Sie wird im Programm jedoch nicht direkt verwendet, statt dessen dient sie als Basisklasse für *CPicture*.

CPicture wird verwendet, um auf einfache Weise mit Bitmaps umgehen zu können. Sie ist abgeleitet von den Klassen *CDib* und *Matrix*. Der Vorteil dieser Kombination wird in der genauen Beschreibung von *CPicture* erläutert.

Bis auf *CDib* wird nun die Funktionalität jeder dieser Klassen genauer erläutert. Eine kurze Beschreibung von *CDib* ist im Abschnitt der Klasse *CPicture* enthalten.

5.3.1 Feld

Die Klasse *Feld* wurde entwickelt, um eine einfache Handhabung von zweidimensionalen Arrays zu ermöglichen, welche im Programm in Form von Bildern oder zum Speichern von Transformationskoeffizienten sehr oft auftauchen. Die Verwendung der von der Programmiersprache bereitgestellten Arrays wird vermieden, da diese zu unflexibel sind: will man die Größe des Arrays erst zur Laufzeit festlegen oder während des Programmablaufs dynamisch ändern, so muß der benötigte Speicherplatz jedesmal im Programm angefordert und nach Benutzung wieder freigegeben werden. Ein Zugriff auf die Elemente dieses Speicherbereichs ist außerdem nur über Zeiger möglich, wobei jedesmal diverse Berechnungen anfallen. Da das Hantieren mit Zeigern und Speicherbereichen der Übersichtlichkeit des Programms nicht gerade dienlich ist, wurde die Klasse *Feld* entwickelt. Diese übernimmt die Verwaltung des Speichers und regelt den Zugriff darauf.

Beim Anlegen eines Objektes vom Typ *Feld* kann man die Anzahl der Zeilen und Spalten übergeben, die das Feld haben soll. Das Objekt alloziert den erforderlichen Speicherbereich bei der Initialisierung und gibt ihn automatisch wieder frei, sobald das Objekt nicht mehr benötigt wird (z. B. beim Verlassen eines Gültigkeitsbereichs). Weiterhin kann die Größe des Feldes zur Laufzeit verändert werden, wobei allerdings der Inhalt verloren geht. Der Zugriff auf ein Element des Feldes geschieht durch Anwendung des Klammeroperators (), bei dem die gewünschte Zeile und Spalte angegeben wird; die Zählung beginnt hierbei bei eins, nicht bei null! Eine weitere wichtige Funktion ist durch den =-Operator realisiert. Hiermit läßt sich ein Objekt vom Typ *Feld* einem anderen zuweisen.

Bevor ich zu den Beispielen komme, bleibt noch anzumerken, daß die Klasse *Feld* selbst im Programm nicht direkt verwendet wird, sondern immer die hiervon abgeleitete Klasse *Matrix*, da diese noch einige zusätzliche Operationen erlaubt.

Beispiele:

Anlegen eines Feldes vom Typ „double“ mit zwei Zeilen und fünf Spalten:

```
Feld<double> Test(2, 5);
```

Zugriff auf ein Feldelement:

```
Test(2, 1) = 12.4;
double x = Feld(1, 1);
```

Zuweisen eines kompletten Feldes:

```
Testfeld2 = Test;
```

Größe eines Feldes ändern:

```
Test.Resize(10, 10);
```

Größenänderung bei gleichzeitiger Initialisierung des Feldes mit einem Wert (hier: 0.0):

```
Test.Resize(10, 10, 0.0);
```

5.3.2 Matrix

Die Klasse *Matrix* wurde von *Feld* abgeleitet und bietet daher die gleiche Funktionalität wie ihre Basisklasse; sie wurde ebenfalls als Template implementiert. Zusätzlich zu den von *Feld* vererbten Methoden sind Operatoren vorhanden, mit deren Hilfe Matrixoperationen auf die Feldelemente angewandt werden können. Dies sind im einzelnen:

- **Rechnenoperatoren:** Addition, Subtraktion und Multiplikation zweier Matrizen (Operatoren +, -, *) sowie Multiplikation einer Matrix mit einem Skalar (Operator %); transponieren einer Matrix (Operator !).

  Hierbei ist darauf zu achten, daß bei Addition und Subtraktion die Anzahl der Zeilen und Spalten beider Matrizen übereinstimmen müssen; bei der Multiplikation zweier Matrizen gelten die üblichen Regeln. Für die Multiplikation einer Matrix mit einem Skalar ist zu beachten, daß der Skalar nur von rechts multipliziert werden kann (siehe auch Beispiel). Der Grund hierfür liegt in der Behandlung des Operators durch die Programmiersprache.

- **Vergleichsoperatoren:** Es ist ein Vergleich auf Gleichheit (Operator ==) und Ungleichheit möglich (Operator !=).

- **Sonstige Methoden:** Zusätzlich zu den Operatoren stehen Methoden zum Finden des kleinsten und größten Wertes einer Matrix zur Verfügung (Min() bzw. Max()) sowie zum Speichern der Matrix in einem von MATLAB lesbaren Dateiformat (Write_Mat(ofstream &os)).

Beispiele:

Anlegen einer Matrix vom Typ „double“:

Matrix<double> M1(100, 200);

Addition von zwei Matrizen:

M3 = M2 + M1;

Die Matrizen M1 und M2 müssen hierbei gleich groß sein; die Matrix M3 kann eine beliebige Größe haben, da sie bei der Zuweisung sowieso neu angelegt wird.

Multiplikation einer Matrix mit einem Skalar:

M2 = M1 % 3.6;

nicht möglich ist dagegen:

M2 = 3.6 % M1;

Die Klasse *Matrix* wird im Programm insbesondere zur Speicherung von Transformationskoeffizienten verwendet.

5.3.3 Wavelet

Die Klasse *Wavelet* enthält alle Daten und Methoden, die zur Durchführung einer Wavelet-Transformation benötigt werden. Außerdem sind Funktionen zum Berechnen der diskreten Werte des Wavelets und der Skalierungsfunktion aus den Filterkoeffizienten vorhanden. Ein Objekt dieser Klasse steht jedem Dokument des Programms genau einmal zur Verfügung.

Es folgt eine Beschreibung derjenigen Methoden, die von außerhalb der Klasse aufgerufen werden können:

Initialisierung einer Instanz dieser Klasse:
Zur Initialisierung einer Instanz der Klasse *Wavelet* existieren zwei Methoden; die erste wird verwendet, wenn es sich bei den ausgewählten Basisfunktionen um eine orthogonale Basis handelt, die zweite wenn es sich um eine biorthogonale Basis handelt. Die Instanz wird jedesmal neu initialisiert, wenn der Anwender im Dialog **Einstellungen/Wavelet** eine Änderung vorgenommen hat. Die beiden Methoden haben den gleichen Namen **InitWavelet**; sie unterscheiden sich nur in der Zahl der übergebenen Parameter.

- Der Methode **InitWavelet(Matrix<double> H)** wird ein Zeilenvektor übergeben, der die Tiefpaßfilterkoeffizienten eines Basiswavelets enthält, dessen Dilatationen und Translationen eine orthogonale Basis bilden. Aus diesen werden die zugehörigen Hochpaßfilterkoeffizienten wie in der Arbeit beschrieben berechnet. Die Koeffizienten werden in den internen Variablen *H* und *G* gespeichert.

- Der Methode **InitWavelet(Matrix<double> H, Matrix<double> H_Biorth)** werden zwei Matrizen übergeben: die Matrix *H* enthält die zur

Analyse zu verwendenden Tiefpaßfilterkoeffizienten, die Matrix *H_Biorth* die zur Synthese zu verwendenden. Die Speicherung der jeweils hierzu gehörenden Hochpaßfilterkoeffizienten werden in den internen Variablen *G* und *G_Biorth* gespeichert.

Methoden zur eindimensionalen Wavelet-Transformation:
Für die eindimensionale Wavelet-Transformation stehen nur zwei Methoden zur Verfügung: eine zur Transformation und eine zur inversen Transformation. Transformiert wird unter Verwendung der mit **InitWavelet()** festgelegten Basis.

- Zur Transformation wird die Methode **FWT_1D(Matrix<double> Werte)** verwendet. Diese führt eine Wavelet-Transformation durch, wobei soviele Iterationsschritte durchgeführt werden, bis die zu transformierende Funktion nur noch aus einem Punkt besteht; daher muß die Anzahl der Spalten von *Werte* auch eine Zweierpotenz sein. Es erfolgt keine Anpassung der Daten. Der Parameter *Werte* ist ein Zeilenvektor, der die diskreten Funktionswerte enthält. Das Ergebnis der Transformation wird in einer Matrix vom Typ *double* zurückgeliefert, die ebenso groß ist wie *Werte.*

- Für die inverse Transformation steht die Methode **IFWT_1D(Matrix<double> Werte, int Anzahl_Schritte)** zur Verfügung. *Werte* ist ein Zeilenvektor, der die mit **FWT_1D()** ermittelten Transformationskoeffizienten enthält; *Anzahl_Schritte* gibt die Zahl der bei der inversen Transformation durchzuführenden Iterationsschritte an.

Methoden zur zweidimensionalen Wavelet-Transformation:
Auch für die zweidimensionale Transformation stehen nur zwei Methoden zur Verfügung, die von Außen zugänglich sind. Diese rufen wiederum interne Funktionen auf.

- **FWT_2D(Matrix<double> Werte, int Anzahl_Schritte, int Datenanpassung, int dlgende)**: Diese Methode führt die zweidimensionale Wavelet-Transformation mit den in *Werte* enthaltenen (nach *double* konvertierten) Bildpunkten durch. Das Ergebnis der Transformation wird in einer Matrix gleicher Größe vom Typ *double* zurückgeliefert. Als Parameter wird in *Anzahl_Schritte* die Zahl der durchzuführenden Iterationsschritte übergeben. *Datenanpassung* enthält einen Wert, der aus dem Dialog **Einstellungen/2D-FWT** übernommen wird und angibt, welche Art der Datenanpassung gewählt wurde; der Inhalt dieser Variablen ist nur dann von Bedeutung, wenn die Einstellung *künstlich periodisch* ausgesucht wurde, da nur dann während der Transformation anders gerechnet werden muß. Der Parameter *dlgende* dient nur zur Anzeige des Fortschrittsbalkens während der Berechnung. Er gibt an, welcher Wert in der Fortschrittsanzeige als letzter angezeigt werden soll. Der Parameter ist optional; wird er nicht übergeben, so wird der Standardwert 100 (Prozent) verwendet.

- **IFWT_2D(Matrix<double> Werte, int Anzahl_Schritte, int Datenanpassung, int dlgende)**: Die Parameter dieser Methode entsprechen denen von **FWT_2D()**. Zurückgeliefert wird eine Matrix vom Typ *double*, welche das Ergebnis der inversen Transformation enthält.

Berechnung von Graphen des Wavelets und der Skalierungsfunktion:
Mit den beiden folgenden Methoden können aus den Filterkoeffizienten die Graphen des zugehörigen Wavelets bzw. der Skalierungsfunktion wie in Kapitel 3.4 beschrieben berechnet werden.

- **Berechne_Wavelet_Graph(Matrix<double> Werte, int Anzahl_Iterationen, BOOLEAN Orth)**: Diese Methode bekommt im Parameter *Werte* einen Zeilenvektor übergeben, der nur einen einzigen Wert ungleich null enthalten darf, wie in Kapitel 3.4 beschrieben. Zurückgeliefert wird ein gleich großer Vektor, der die diskreten Werte des Wavelets enthält. Zu beachten ist, daß die Anzahl der Elemente dieses Vektors eine Zweierpotenz sein muß. *Anzahl_Iterationen* gibt die Anzahl der durchzuführenden Iterationen an; *Orth* zeigt an, ob bei einer biorthogonalen Basis das Analyse-Wavelet (*Orth* = FALSE) oder das Synthese-Wavelet (*Orth* = TRUE) berechnet werden soll.

- **Berechne_Skalierung_Graph(Matrix<double> Werte, int Anzahl_Iterationen, BOOLEAN Orth)**: Die Parameter dieser Funktion entsprechen denen von **Berechne_Wavelet_Graph()**. Es wird die Skalierungsfunktion berechnet.

Sonstiges:
Weiterhin stehen dem Benutzer der Klasse noch folgende Methoden zur Verfügung:

- **Get_H(), Get_G():** Diese beiden Funktionen liefern jeweils einen Zeilenvektor (Matrix<double>) zurück, der die Filterkoeffizienten des Tief- bzw. Hochpaßfilters enthält.

- **Get_H_Biorth(), Get_G_Biorth():** Wie oben, nur für das Synthese-Wavelet bei Verwendung einer biorthogonalen Basis.

- **Skalennormierung(Matrix<double> Werte, int Anzahl_Schritte):** Diese Funktion führt eine Anpassung der Transformationskoeffizienten für die Bildschirmdarstellung durch: die Werte der Koeffizienten werden für jeden Iterationsschritt getrennt auf den Wertebereich 0-255 abgebildet. Diese Methode wird nach dem Anklicken des Menüpunktes **2D-FWT/Skalennormalisierung für Anzeige** aufgerufen. Das Ergebnis wird in einer Matrix vom Typ *double* zurückgeliefert.

Weiterhin existieren noch Methoden, die nur klassenintern aufgerufen werden:

- **Transform_H(Matrix<double> Werte, int Datenanpassung):** Diese Methode wird von **FWT_2D()** aufgerufen. Sie führt eine Filterung des Inhalts von *Werte* in X-Richtung mit dem Filter *H* durch. Das Ergebnis wird als Funktionswert zurückgegeben; es ist eine Matrix, die ebenso viele Zeilen hat wie *Werte* aber nur halb soviele Spalten, da ein Downsampling um zwei durchgeführt wurde.

- **Transform_G(Matrix<double> Werte, int Datenanpassung):** Wie **Transform_H()**, nur wird an Stelle von *H* der Filter *G* verwendet.

- **Transform_H_Y(Matrix<double> Werte, int Datenanpassung):** Diese Methode wird von **FWT_2D()** aufgerufen. Sie führt eine Filterung des Inhalts von *Werte* in Y-Richtung mit dem Filter *H* durch. Das Ergebnis wird als Funktionswert zurückgegeben; es ist eine Matrix, die ebenso viele Spalten hat wie *Werte* aber nur halb soviele Zeilen, da ein Downsampling um zwei durchgeführt wurde.

- **Transform_G_Y(Matrix<double> Werte, int Datenanpassung):** Wie **Transform_H_Y()**, nur wird an Stelle von *H* der Filter *G* verwendet.

- **I_Transform_H_X(Matrix<double> Werte, int Datenanpassung):** Diese Methode wird von **IFWT_2D()** aufgerufen. Sie führt eine inverse Filterung des Inhalts von *Werte* in X-Richtung mit dem Filter *H* durch. Das Ergebnis wird als Funktionswert zurückgegeben; es ist eine Matrix, die ebenso viele Zeilen hat wie *Werte* aber doppelt soviele Spalten, da ein Upsampling um zwei durchgeführt wurde.

- **I_Transform_G_X(Matrix<double> Werte, int Datenanpassung):** Wie **I_Transform_H_X()**, nur wird an Stelle von *H* der Filter *G* verwendet.

- **I_Transform_H_Y(Matrix<double> Werte, int Datenanpassung):** Diese Methode wird aufgerufen von **IFWT_2D()**. Sie führt eine inverse Filterung des Inhalts von *Werte* in Y-Richtung mit dem Filter *H* durch. Das Ergebnis wird als Rückgabewert zurückgegeben; es ist eine Matrix, die ebenso viele Spalten hat wie *Werte* aber doppelt soviele Zeilen, da ein Upsampling um zwei durchgeführt wurde.

- **I_Transform_G_Y(Matrix<double> Werte, int Datenanpassung):** Wie **I_Transform_H_Y()**, nur wird an Stelle von *H* der Filter *G* verwendet.

5.3.4 CWLT_Daten

Diese Klasse wird zur Speicherung eines Bildes nach einer Wavelet-Transformation verwendet. Hier sollen nur die Methoden der Klasse beschrieben werden. Die

dort benutzten Datenstrukturen entsprechen nämlich dem Dateiformat, welches im Anhang B erläutert wird.

Die Methoden zum Laden bzw. Speichern eines Bildes sind:

- **WLT_Load(CFile *pFile):** Lädt die in der Datei *pFile* enthaltenen Daten und legt sie in den Datenstrukturen des Objekts ab. Nach dem Laden muß die Methode **Entpacken()** (s.u.) aufgerufen werden.

- **WLT_Save(CFile *pFile):** Speichert den Inhalt der Datenstrukturen des Objekts in der Datei *pFile.* Vor dem Speichern muß die Methode **Packen()** (s.u.) aufgerufen werden.

Da die Transformationskoeffizienten in einer Matrix vom Typ *double* vorliegen, werden von jedem Koeffizienten acht Byte Speicherplatz belegt. In der Datei soll aber so wenig Platz verwendet werden wie möglich. Daher müssen die Koeffizienten vor dem Speichern in eine platzsparende Form umgewandelt werden, die durch die Einstellungen im Dialog **Einstellungen/Quantisierung** beeinflußt wird. Hierbei wird folgendermaßen vorgegangen:

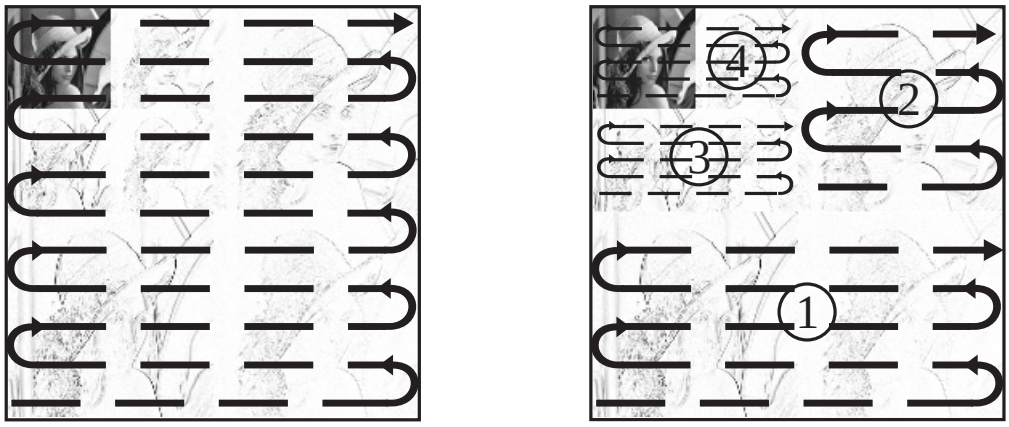
1. **Einstellung „keine Quantisierung“, „nur auf ganze Zahlen runden“ oder „Schwellwert“:** Hier wird jeder Koeffizient in eine ganze Zahl umgewandelt und auch als solche gespeichert, das heißt es werden pro Koeffizient vier Byte verwendet. Für diejenigen Koeffizienten, die null sind, wird eine Lauflängencodierung durchgeführt; für die Lauflänge werden ebenfalls vier Byte benutzt; sie folgt immer direkt nach dem Nullkoeffizienten. Die Koeffizienten werden von unten nach oben zeilenweise abgelegt, wie sie auch auf dem Bildschirm erscheinen.

2. **Einstellung „globale Quantisierung“:** Bei dieser Einstellung werden für jeden Koeffizienten die im Dialog angegebene Anzahl an Bit zur Speicherung verwendet. Gespeichert wird dann nicht der Wert des Koeffizienten, sondern das Quantisierungsintervall, in welchem er enthalten ist. Da die Bitanzahl zwischen zwei und 16 frei wählbar ist, müssen die Koeffizienten „zusammengerückt“ werden, damit keine ungenutzten Lücken entstehen. Für die Speicherung der Lauflänge nach einem Nullkoeffizienten[5] wird ebenfalls der im Dialog angegebene Wert verwendet, der von dem zur Speicherung der Koeffizienten verschieden sein kann. Auch bei dieser Art der Speicherung werden die Koeffizienten von unten nach oben zeilenweise abgelegt.

3. **Einstellung „getrennte Quantisierung für jedes Band“:** Die Vorgehensweise entspricht prinzipiell der bei „globale Quantisierung“, wobei jetzt die

[5] eigentlich dem Quantisierungsintervall, das der Null entspricht

Anzahl der zu verwendenden Bit sich von Iterationsschritt zu Iterationsschritt ändert. Weiterhin ist es im Gegensatz zu den oben genannten Arten der Speicherung aufgrund der verschiedenen Bitanzahlen nötig, die zu einem Iterationsschritt gehörenden Koeffizienten auch in der Datei hintereinander abzulegen; hierbei werden zunächst die im Bild unten links und unten rechts erscheinenden Teile von unten nach oben zeilenweise gespeichert. Daran anschließend wird der Teil rechts oben gespeichert. Ist dies erledigt, wird zum nächsten Band übergegangen. Die unterschiedliche Art der Speicherung ist in Abbildung 5.9 nochmals verdeutlicht.



(a) Fall 1. und 2.

(b) Fall 3.

Abbildung 5.9: Speicherung der Transformationskoeffizienten

Die eben beschriebene Umwandlung der Koeffizienten in eine platzsparende Form wird von der Methode **Packen(Matrix<double> Werte)** erledigt. Diese muß vor dem Speichern aufgerufen werden. Sie bekommt als Parameter die Transformationskoeffizienten übergeben; nach dem Packen der Daten sind diese in einem neuen Speicherbereich enthalten, auf den der Zeiger *pDaten* des Objekts zeigt. Die Länge dieses Speicherbereichs in Byte ist in *Anzahl_komp_Daten* enthalten.

Nach dem Laden eines komprimierten Bildes muß diese kompakte Form natürlich wieder rückgängig gemacht werden, so daß jeder Koeffizient einer *double*-Zahl entspricht. Dies wird von der Methode **Entpacken()** erledigt, die nach dem Laden eines Bildes ausgeführt werden muß. Sie liefert eine Matrix vom Typ *double* zurück, die die Koeffizienten enthält.

5.3.5 CDib

Die Klasse *CDib* wurde dem Buch „Inside Visual C++“ [14] entnommen, wo auch eine Beschreibung der Funktionalität enthalten ist[6]. Sie stellt alle Operationen bereit, die zum Laden, Speichern und Anzeigen von Bitmaps in einem Programm benötigt werden. Die Klasse konnte praktisch unverändert übernommen werden; nach der Beseitigung eines Fehlers in der Prozedur zum Speichern einer Bitmap (**Write()**) funktionierte sie dann auch einwandfrei.

Trotzdem dient sie nur als Basisklasse für die Klasse *CPicture*, welche einen einfacheren Zugriff auf einzelne Pixel einer Bitmap ermöglicht.

5.3.6 CPicture

CPicture wurde abgeleitet von den Klassen *CDib* und *Matrix*. Sie vereinigt die Vorteile eines einfachen Zugriffs auf die Pixel der Bitmap durch die von *Matrix* bereitgestellten Operatoren mit den Funktionen zum Hantieren mit Bitmaps unter Windows aus der Klasse *CDib*. Zusätzlich wurden noch folgende Funktionen implementiert:

- **Löschen des Bildes:** Dies erfolgt durch Aufruf der Funktion **Clear()**.

- **Zeichnen einer Linie in die Bitmap:** Die hierfür verwendete Methode trägt den Namen **Line256**. Sie kann benutzt werden, wenn die Bitmap eine Farbtiefe von acht Bit hat (wenn also ein Bildpunkt einem Byte entspricht). Der Aufruf erfolgt durch **Line256(int x1, int y1, int x2, y2, BYTE Farbe)**. Die zu übergebenden Parameter sind der Anfangs- und Endpunkt der Linie ((*x1*, *y1*) bzw. (*x2*, *y2*)) sowie die Farbe, mit der die Linie gezeichnet werden soll. Der Parameter *Farbe* ist optional und darf Werte zwischen null und 255 annehmen; wird *Farbe* nicht mit übergeben, so wird der Standardwert Eins verwendet.

- **Plotten eines Funktionsgraphen:** Die Methode **Plot_Function()** stellt die als Zeilenvektor übergebenen Werte als Funktionsgraphen dar. Die X-Achse befindet sich dabei immer in der Bildmitte; weiterhin werden alle Funktionswerte so skaliert, daß der gesamte verfügbare Bereich in Y-Richtung ausgenutzt wird. Zu beachten ist, daß der Vektor genau soviele Werte enthält wie die Bitmap in X-Richtung. Die genaue Syntax für den Aufruf lautet: **Plot_Function(Matrix<int> Vektor, BYTE Farbe, BOOLEAN Lin_Interpol)**. Der Parameter *Farbe* gibt wiederum die Farbe des Graphen an; ist *Lin_Interpol* FALSE, so werden nur die angegebenen Punkte in der Bitmap gesetzt; ist der Parameter TRUE, so wird zwischen den Funktionswerten linear interpoliert.

[6] Seite 252ff.

- **Plotten der Werte einer Matrix:** Mit der Methode **Plot_2D_Koeff()** kann der Inhalt einer Matrix in einer Bitmap dargestellt werden. Diese Funktion wird zum Anzeigen von Transformationskoeffizienten verwendet, die bei der Transformation eines Bildes entstanden sind. Die übergebene Matrix muß dabei die gleiche Anzahl an Zeilen und Spalten enthalten wie die Bitmap; der Aufruf lautet: **Plot_2D_Koeff(Matrix<BYTE> Werte)**.

Der Zugriff auf einen Bildpunkt erfolgt durch den Klammeroperator, genau wie bei einer Matrix. Hierbei ist zu beachten:

- Die Zählung der Koordinaten beginnt nicht bei null sondern bei eins.
- Der Ursprung des Bildes liegt in der linken unteren Fensterecke.
- Aufgrund der Behandlung des Bildes als Matrix müssen die Koordinaten eines Bildpunktes in der Form (y, x) angegeben werden und nicht wie normalerweise üblich mit (x, y).

Eine neue Bitmap kann auf zweierlei Weise angelegt werden: durch Laden eines Bildes aus einer Datei mit der Methode **Read()** oder durch den Konstruktor **CPicture(CSize Größe, int Tiefe, BYTE Palette)**. Hierbei werden in der Struktur *CSize* die Größe der neuen Bitmap in X- und Y-Richtung angegeben; *Tiefe* gibt die Farbtiefe in Bit an (bei 256 Graustufen ist der Wert für diesen Parameter immer acht); der Parameter *Palette* ist optional: wird er nicht übergeben, so wird der Standardwert Null verwendet. Dieser gibt an, daß die verwendete Palette der eines Graustufenbildes entspricht. Außer null kann hierfür nur noch der Wert Eins übergeben werden; dieser entspricht einer negativen Graustufenpalette.

5.3.7 Aufbau des Programms

Nun zum prinzipiellen Aufbau des Programms:

Es kommt, entsprechend dem Document/View-Konzept der MFC, eine Art von Document zur Anwendung dessen Daten in mehreren verschiedenen Views angezeigt werden können. Das Document selbst enthält alle Datenstrukturen, die zur Wavelet-Transformation benötigt werden. Dies sind unter anderem:

- Ein Objekt vom Typ *Wavelet*, welches die Transformationen durchführt (Variable *m_Wavelet*),
- ein Objekt vom Typ *CPicture**, in welchem das zu transformierende Bild abgelegt wird (Variable *m_Bitmap*),
- vier weitere Objekte vom Typ *CPicture**; alle Bitmaps werden erst angelegt, wenn sie verwendet werden sollen, um nicht unnötig Speicherplatz zu verschwenden. Je eine Bitmap wird verwendet, um den Graphen eines Wavelets bzw. einer Skalierungsfunktion anzuzeigen (*m_Wavelet_Pic*

bzw. *m_Skalierung_Pic*); eine Bitmap dient der Darstellung der Transformationskoeffizienten (*m_Koeffizienten_Pic*) und eine weitere zur Anzeige des Ergebnisses der inversen Transformation (*m_Ruecktrans_Pic*).

- zwei Objekte vom Typ *Matrix<double>*, die zum Speichern. der Transformationskoeffizienten (*m_WLT_Koeff_2D*) bzw. der rücktransformierten Daten (*m_Rueck*) benutzt werden.

- des weiteren enthält das Document noch diverse Methoden zum Umgang mit der Erweiterung von Daten.

- es sind außerdem noch einige Variablen vorhanden, die zum Speichern verschiedenster Daten benötigt werden (PSNR, RMSE, Datenanpassung, Quantisierung, etc.). Andere wiederum dienen zum Aktivieren und Deaktivieren von Menüpunkten bzw. allgemein der Verwaltung der Benutzeroberfläche. Darunter fallen insbesondere einige Dialoge, die mit dem Menüpunkt **Einstellungen** aufgerufen werden.

Um die prinzipielle Funktionsweise des Programms zu verstehen, möchte ich zudem kurz das Zusammenspiel von Document und Views erläutern, wie sie bei der vorliegenden Implementierung verwendet werden.

Hierzu ist zunächst zu sagen, daß nur ein Typ von Document verwendet wird, für das allerdings, je nach Inhalt, verschiedene Views zur Anzeige der darin enthaltenen Daten benutzt werden. Dies sind im einzelnen:

COriginalView In diesem View wird die Bitmap angezeigt, die in der Variablen *m_Bitmap* enthalten ist. Dies ist entweder ein von der Festplatte geladenes Bild im BMP- oder WLT-Format oder der Graph einer mit **1D-FWT/Zeichen Funktion** gezeichneten Funktion zur eindimensionalen Transformation.

CKoeffizientenView Dieser View wird zur Darstellung der Transformationskoeffizienten nach einer ein- oder zweidimensionalen Wavelet-Transformation verwendet, das heißt er zeigt den Inhalt der Bitmap *m_Koeffizienten_Pic* an. Ein Fenster mit diesem View öffnet sich nach der Anwahl des Menüpunktes **2D-FWT/Anzeige der Koeffizienten** oder **1D-FWT/Zeichne Waveletkoeffizienten**.

CRuecktransView Hier wird die Bitmap *m_Rueck* angezeigt, also das Ergebnis einer inversen Transformation. Dieser View wird nach dem Anklicken der Menüpunkte **2D-FWT/Transformation durchführen**, **2D-FWT/Inverse Transformation** oder **1D-FWT/Zeichne zurücktransformierte Funktion** geöffnet.

CWaveletView dient zur Anzeige des Graphen eines Wavelets (Bitmap *m_Wavelet_Pic*). Er wird durch den Menüpunkt **1D-FWT/Zeichne Wavelet** geöffnet.

CSkalierungView dient zur Anzeige des Graphen einer Skalierungsfunktion (Bitmap *m_Skalierung_Pic*). Er wird durch den Menüpunkt **1D-FWT/Zeichne Skalierungsfunktion** geöffnet.

Die Beschreibungen in diesem Kapitel sollten nur einen groben Überblick über das Programm verschaffen. Auf die genaue Implementierung der Wavelet-Transformation gehe ich hier nicht näher ein, da diese eine exakte Umsetzung des in Kapitel 3.5 beschriebenen Algorithmus ist. Statt dessen folgen nun einige Auswertungen und Vergleiche, die die Stärken und Schwächen der Wavelet-Bildkompression aufzeigen.

Kapitel 6

Vergleich von Wavelet-Kompressionsverfahren

Nachdem nun die theoretischen Grundlagen der Wavelet-Transformation und der damit möglichen Bildkompression erläutert wurden, soll in diesem Kapitel der Einfluß der Parameterwahl auf die bei einem bestimmten Kompressionsfaktor erreichbare Bildqualität untersucht werden.
Betrachtet werden sollen insbesondere:

- Auswirkung der Wahl des Basiswavelets,
- Einfluß des Bildmaterials auf den Kompressionsgrad,
- verschiedene Möglichkeiten der Datenanpassung,
- Unterschiede bei der Wahl der Quantisierung,
- Auswirkung der Anzahl der durchgeführten Iterationsschritte und
- Qualität der Kompression im Vergleich mit JPEG.

6.1 Maßzahlen für die Bildqualität

Um einen Vergleich der Bildqualität der komprimierten Bilder anstellen zu können, benötigt man eine Metrik, die den Unterschied zwischen zwei Bildern angibt, hier speziell zwischen dem Original und dem komprimierten Bild. Dies ist, wie sich herausstellte, kein leichtes Unterfangen, da man zwar den Abstand zweier Bilder mit den verschiedensten Verfahren berechnen kann, die aber alle meist eher schlecht den tatsächlichen optischen Eindruck wiedergeben. Das heißt, ein Bild kann nach den hier verwendeten Maßzahlen eine schlechtere Bildqualität haben als ein anderes, obwohl ein menschlicher Betrachter dennoch das vermeintlich schlechtere als näher am Original liegend bezeichnen würde. Gerade

die beim JPEG-Verfahren bei hohen Kompressionsraten auftretenden Blockeffekte sind ein gutes Beispiel hierfür, da solche regelmäßigen Strukturen vom Auge als störend empfunden werden; von den Maßzahlen hingegen werden sie überhaupt nicht berücksichtigt.

Diese Überlegungen sollte man also bei der Betrachtung der folgenden Auswertungen immer im Hinterkopf behalten.

Nun zu den hier verwendeten Maßen: dem mittleren quadratischen Fehler[1] und dem sich hieraus ergebenden Signalrauschabstand[2]: Für die folgenden Gleichungen wird ein Originalbild $f[i,j]$ bestehend aus Z Zeilen und S Spalten betrachtet sowie ein ebenso großes Bild $\tilde{f}[i,j]$, welches aus $f[i,j]$ durch Kompression gewonnenen wurde und nun wieder als Bitmap vorliegt.

DEFINITION 6 (MITTLERER QUADRATISCHER FEHLER)
Der mittlere quadratische Fehler RMSE ist definiert durch

$$\textit{RMSE} = \sqrt{\frac{1}{ZS}\sum_{i=1}^{Z}\sum_{j=1}^{S}\left(f[i,j]-\tilde{f}[i,j]\right)^2} \tag{6.1}$$

Aus dem RMSE kann man dann den Signalrauschabstand wie folgt berechnen:

DEFINITION 7 (SIGNALRAUSCHABSTAND)
Der Signalrauschabstand PSNR errechnet sich durch

$$\textit{PSNR} = 20\log_{10}\frac{max|f[i,j]|}{RMSE} \tag{6.2}$$

wobei $max|f[i,j]|$ der größte Wert ist, der im Bild auftreten kann.

Da hier nur Bilder mit 256 Graustufen betrachtet werden sollen, gilt also $max|f[i,j]| = 255$.

6.2 Vorbemerkungen

Als Grundlage für die Vergleiche wurde das Bild „Lena“ in einer Auflösung von 512×512 Punkten verwendet. Teilweise wurden aber auch andere Bilder mit einbezogen, die den Typ eines nicht-natürlichen Bildes repräsentieren („Floor“ und „Door“). Zu den Bildern „Floor“ und „Door“ ist noch zu sagen, daß diese nur einen Ausschnitt aus den Originalbildern darstellen, da für die Auswertungen mit

[1] oft auch als RMSE bezeichnet: **R***oot* **M***ean-***S***quared* **E***rror*

[2] hier wird der sog. PSNR (**P***eak* **S***ignal-to-***N***oise* **R***atio*) verwendet. Bei Vergleichen mit in der Literatur angegebenen Werten muß berücksichtigt werden, daß auch andere Arten des Signalrauschabstands gebräuchlich sind. Siehe auch [7, Seite 110] und [10, Seite 2].

dem bereits vorgestellten Programm zur Wavelet-Bildkompression Bilder mit einer horizontalen und vertikalen Auflösung benötigt werden, die eine Zweierpotenz sind.

Alle verwendeten Bilder sind im Anhang A abgebildet.

In diesem Kapitel wird des öfteren von „natürlichen“ und „künstlichen“ Bildern gesprochen. Unter „natürlich“ sollen hier Bilder verstanden werden, die ähnlich dem Bild „Lena“ sind, das heißt Bilder mit relativ weichen Farbübergängen, die meist durch Digitalisierung einer Photographie entstanden sind (oder auch durch Raytracing). Ein „künstliches“ Bild ist ein Bild, welches einer technischen Zeichnung ähnlich ist, es besitzt also sehr viele, mehr oder wenige dicke, schwarze Linien oder Bögen auf weißem Grund (oder natürlich auch umgekehrt).

Zu den in den folgenden Kapiteln abgedruckten Ergebnissen der Auswertungen muß außerdem angemerkt werden, daß der bei einem bestimmten Kompressionsgrad erreichte Signalrauschabstand nicht direkt mit den Ergebnissen anderer Kompressionsverfahren (JPEG, fraktale Bildkompression) verglichen werden kann, da nach der Lauflängencodierung keine Huffman-Codierung mehr durchgeführt wurde, weshalb die Werte teilweise schlechter erscheinen als sie eigentlich sind. Ein kurzer Vergleich mit JPEG wurde dennoch vorgenommen, mehr dazu folgt in Kapitel 6.7.

Bei fast allen Diagrammen wurde der Signalrauschabstand (PSNR) über dem Kompressionsgrad angetragen. Der Kompressionsgrad wurde vom Programm als Quotient aus dem Speicherplatzbedarf des Bildes als Bitmap und dem Speicherplatzbedarf des Bildes nach Wavelet-Transformation, Quantisierung und Lauflängencodierung errechnet. Hierbei ist zu beachten, daß nur der Speicherplatz des eigentlichen Bildes (das heißt ohne Header etc.) im Hauptspeicher berücksichtigt wurde, und nicht etwa die Dateigröße auf der Festplatte.

Um den Kompressionsgrad in großen Schritten zu beeinflussen ist es nötig die Einstellungen zur Quantisierung (also die Anzahl der verwendeten Bit) zu verändern. Um dennoch zwischen den einzelnen Messungen Vergleiche anstellen zu können, wurden die Einstellungen für die Quantisierung immer gleich gelassen, auch wenn dadurch die erreichbare Bildqualität bei einem bestimmten Kompressiongrad etwas schlechter ausfällt als es mit anderen Einstellungen möglich gewesen wäre (betrifft speziell die Anzahl der Bit zur Lauflängencodierung).

Um festzustellen, wie stark sich die Kompressionsrate durch die Quantisierung beeinflussen läßt, wurden zunächst 60 Bildkompressionen mit dem Bild „Lena“ mit verschiedenen Einstellungen für die Quantisierung (getrennt nach Frequenzbändern) und der Anzahl der Bit zur Lauflängencodierung durchgeführt. Die Ergebnisse dieses Tests sollen hier allerdings nicht alle angegeben werden, da sie nur dazu dienen sollten, die in den folgenden Kapiteln beschriebenen Messungen vorzubereiten, das heißt es sollten Meßpunkte ermittelt werden, mit denen der Kompressionsgrad geändert werden kann. Diese Werte für die Quantisierung wurden bei allen Auswertungen dann immer wieder verwendet.

In den Tabellen wurde die zur Kompression benötigte Rechenzeit nicht mit aufgenommen. Dies liegt zum einen daran, daß diese bei allen Einstellungen in etwa gleich ist und zum größten Teil von der Bildgröße und der Art der Datenanpassung abhängt. Für ein Bild der Größe 512×512 Punkte mit Datenanpassung durch Auffüllen mit Nullen werden insgesamt (also für Transformation, Quantisierung und inverse Transformation) ungefähr eineinhalb Minuten benötigt. Würde man kleinere Bilder verwenden (256×256 Punkte), so sind die Rechenzeiten bei sonst gleichen Einstellungen wesentlich geringer (ca. 15 Sekunden). Der große Unterschied liegt vor allem darin begründet, da das Programm zur Speicherung der Koeffizienten Variablen vom Typ *double* verwendet, die einen Speicherplatz von je acht Byte benötigen. Es wird also sehr viel Speicher zur Transformation benötigt, wobei bei kleinen Bildern der physikalische Speicher noch ausreicht, bei größeren hingegen muß das Betriebssystem Speicherbereiche auf die Festplatte auslagern, wodurch natürlich erhebliche Geschwindigkeitseinbußen entstehen. Die Werte für die verschieden großen Bilder sind also nicht direkt miteinander vergleichbar. Die Rechenzeiten sind auch von daher nicht unbedingt ein Maßstab für Vergleiche mit anderen Kompressionsverfahren, da das Programm aus Gründen der Übersichtlichkeit nicht alle möglichen Optimierungen ausnutzt.

Die angegeben Zeiten beziehen sich auf einen Rechner mit 40 MB Hauptspeicher und Pentium-100 CPU unter Windows NT 4.0.

Alle hier enthaltenen Diagramme wurden mit Microsoft Excel 7.0 erstellt.

6.3 Auswirkungen der Wahl des Basiswavelets

Zunächst wurde untersucht, wie die Wahl eines Wavelets die Bildqualität bei verschiedenen Kompressionsfaktoren beeinflußt. Der Kompressionsgrad wurde, wie auch bei allen anderen Auswertungen über die Quantisierung geändert. Um vergleichbare Werte zu erhalten, wurden immer die gleichen Einstellungen für die Quantisierung gewählt. Aus diesem Grund entstehen Werte für Kompressionsgrad und PSNR, die in der Tabelle nicht direkt miteinander vergleichbar sind, da sich allein durch Änderung des Wavelets beide ändern; daher ist der graphische Vergleich anhand des Diagramms besser geeignet.

Die konstanten Parameter bei der ersten Auswertung sind:

- das Bild: Lena, 512×512
- die Anzahl der Iterationsschritte: 5
- die Art der Datenanpassung: mit Nullen aufgefüllt
- die Position des Bildes in den erweiterten Daten: rechts oben
- die Art der Quantisierung: getrennt nach Frequenzbändern mit einem Faktor von 2 für das Nullintervall

In Tabelle 6.1 sind alle ermittelten Werte abgedruckt, wobei die ungefähre Größe des Kompressionsgrades durch die Quantisierungseinstellungen beeinflußt wurde, die jeweils in der ersten Zeile vor den Daten für die einzelnen Wavelets stehen. Die Wavelets, mit denen Kompressionen durchgeführt wurden, sind[3]:

- Haar-Wavelet,
- Daubechies-Wavelets mit 4, 6, 8 und 20 Filterkoeffizienten (D-4, D-6, D-8, D-20),
- Coifman-Wavelets mit 6 und 12 Filterkoeffizienten (C-6, C-12) und
- Beylkin-Wavelet mit 18 Filterkoeffizienten (B-18).

Die in der Tabelle enthaltenen Daten sind in Abbildung 6.1 graphisch dargestellt. Da die Kurven allerdings meist sehr eng zusammenliegen, wurden Vergrößerungen des Diagramms bei niedrigen und mittleren Kompressionsgraden gemacht, die in den Abbildungen 6.2 bzw. 6.3 zu sehen sind. Als Ergebnis kann für das Bild „Lena“ festgehalten werden:

Bei niedrigen Kompressionsraten (bis ca. 10:1) hat die Wahl des Basiswavelets relativ wenig Auswirkungen auf die Bildqualität, sofern nicht gerade das

[3] die zugehörigen Filterkoeffizienten sowie die Graphen der Wavelets und der Skalierungsfunktionen sind im Anhang C abgedruckt

Haar-Wavelet benutzt wurde. Dies ändert sich mit zunehmender Kompressionsrate immer mehr, wobei die Unterschiede allerdings auch dann nur ungefähr 1dB betragen, was natürlich nicht unbedingt viel über die tatsächliche Bildqualität aussagen muß. Bei Kompressionraten ab 50:1 werden die Unterschiede noch größer; da die Bildqualität dann aber bereits sehr schlecht ist, sind diese Werte eher theoretischer Natur.

Zu beobachten ist beim Haar-Wavelet weiterhin ein „Blockeffekt“ bei hohen Kompressionsraten, wie er in ähnlicher Form auch bei JPEG auftritt; die Ursache liegt im Aussehen des Haar-Wavelets, welches sehr stark von null ansteigt und auch wieder steil abfällt. Die Blöcke beim Haar-Wavelet sind allerdings unterschiedlich groß, weshalb sie vielleicht nicht ganz so störend empfunden werden wie bei JPEG[4].

Bei Verwendung anderer Wavelets treten zwar keine Blockstrukturen auf, aber die durch die Quantisierung entstandenen Artefakte äußern sich ebenfalls in Form eines Effekts, der in der englischsprachigen Literatur auch als „ringing around edges“ bezeichnet wird, was den Sachverhalt ganz gut beschreibt[5]. Dieser Effekt wird mit zunehmender Anzahl der Filterkoeffizienten eines Wavelets immer stärker, weshalb solche Wavelets auch eher ungeeignet für die Bildkompression sind. Der PSNR eines komprimierten Bildes wird hierdurch natürlich teilweise stärker beeinflußt, als es vom optischen Eindruck her erscheint. Dies ist besonders bei Vergleichen mit JPEG zu beachten.

Zu Abbildung 6.6 muß noch angemerkt werden, daß die Kompressionsraten zwischen den mit dem Haar-Wavelet komprimierten Bildern nicht ganz vergleichbar sind mit denen, die mit dem D-6-Wavelet komprimiert wurden; dies liegt einfach daran, daß die Messungen immer mit den gleichen Einstellungen für die Quantisierung durchgeführt wurden. Vergleichbar sind dagegen die ermittelten Werte für den Signalrauschabstand, weshalb man gut die Unterschiede in der Bildqualität bei Verwendung verschiedener Wavelets bei gleichem PSNR sehen kann.

[4] vgl. hierzu auch Abbildung 6.6 mit Abbildung 6.17, wobei allerdings die Kompressionsraten nicht vergleichbar sind (siehe Kapitel 6.7).

[5] siehe hierzu auch Abbildung 6.6

Wavelet	Kompressionsgrad	RMSE	PSNR
Bit pro Band: 6,7,8; Lauflänge: 8,7,6,5			
Haar	1,677	1,811	42,974
D-4	1,641	1,698	43,532
D-6	2,103	2,072	41,801
D-8	2,126	2,091	41,722
D-20	1,572	1,686	43,590
C-6	2,206	2,124	41,589
C-12	1,874	1,919	42,470
B-18	1,570	1,682	43,612
Bit pro Band: 4,5,6,7,8; Lauflänge: 10,8,7,7,6			
Haar	6,204	4,535	35,000
D-4	6,829	3,946	36,208
D-6	2,103	2,072	41,801
D-8	2,126	2,091	41,722
D-20	1,572	1,686	43,590
C-6	2,206	2,124	41,589
C-12	1,874	1,919	42,470
B-18	1,570	1,682	43,612
Bit pro Band: 3,3,4,6,7; Lauflänge: 11,9,8,7			
Haar	17,338	8,795	29,246
D-4	18,313	7,422	30,720
D-6	22,616	8,116	29,944
D-8	22,308	7,899	30,180
D-20	18,804	7,661	30,445
C-6	23,205	8,380	29,666
C-12	23,092	8,046	30,019
B-18	19,071	7,706	30,394
Bit pro Band: 2,2,4,7,8; Lauflänge: 13,11,8,7			
Haar	29,772	10,744	27,508
D-4	26,900	9,097	28,952
D-6	28,389	9,593	28,492
D-8	28,426	9,481	28,594
D-20	25,695	9,587	28,497
C-6	29,806	9,787	28,318
C-12	29,925	9,228	28,829
B-18	25,062	9,356	28,709
Bit pro Band: 2,2,4,5,7; Lauflänge: 13,11,8,7			
Haar	35,358	11,012	27,293

Wavelet	Kompressionsgrad	RMSE	PSNR
D-4	32,165	9,443	28,628
D-6	34,575	9,917	28,203
D-8	35,244	9,846	28,265
D-20	31,675	9,928	28,194
C-6	36,238	10,060	28,079
C-12	37,443	9,600	28,486
B-18	30,968	9,686	28,408
Bit pro Band: 2,2,3,4,7; Lauflänge: 13,11,8,7			
Haar	47,741	12,660	26,082
D-4	42,800	11,205	27,143
D-6	47,968	11,595	26,846
D-8	48,277	11,467	26,942
D-20	45,143	11,597	26,844
C-6	50,325	11,787	26,703
C-12	51,552	11,426	26,973
B-18	43,778	11,554	26,876
Bit pro Band: 2,2,2,3,4; Lauflänge: 13,11,9,7			
Haar	74,749	18,571	22,754
D-4	63,504	18,345	22,860
D-6	78,486	18,787	22,654
D-8	84,836	18,955	22,576
D-20	80,610	19,361	22,392
C-6	86,090	18,279	22,892
C-12	87,850	18,656	22,714
B-18	81,361	20,054	22,087

Tabelle 6.1: Abhängigkeit der Bildqualität vom Basiswavelet („Lena“)

Da sich bei einer späteren Messung herausstellte, daß sich Bilder mit schwarzen Linien, hier speziell die Bilder „Floor“ und „Door“, wesentlich schlechter komprimieren lassen als das Bild „Lena“, wurde auch der Einfluß des zur Transformation verwendeten Wavelets auf diesen Bildtyp untersucht. Die Parameter waren die gleichen wie bei der ersten Auswertung mit „Lena“; es wurden lediglich weniger Messungen durchgeführt. Die Daten für „Floor“ sind in Tabelle 6.2 enthalten, diejenigen des Bildes „Door“ in Tabelle 6.3. Eine graphische Darstellungen der Daten zeigen die Abbildungen 6.4 bzw. 6.5.

Das Verhalten der beiden Bilder ist ähnlich, wenn man auch sagen muß, daß sich das Bild „Door“ noch schlechter komprimieren läßt als „Floor“. Zunächst fällt auf, daß sich beide Bilder wesentlich schlechter komprimieren lassen als zum Beispiel „Lena“, was doch etwas überrascht, da gerade die Eignung der Wavelet-Bildkompression für jeden Bildtyp in der Literatur immer wieder herausgestellt wird. Für eine nähere Betrachtung der Abhängigkeit der erreichbaren Kompres-

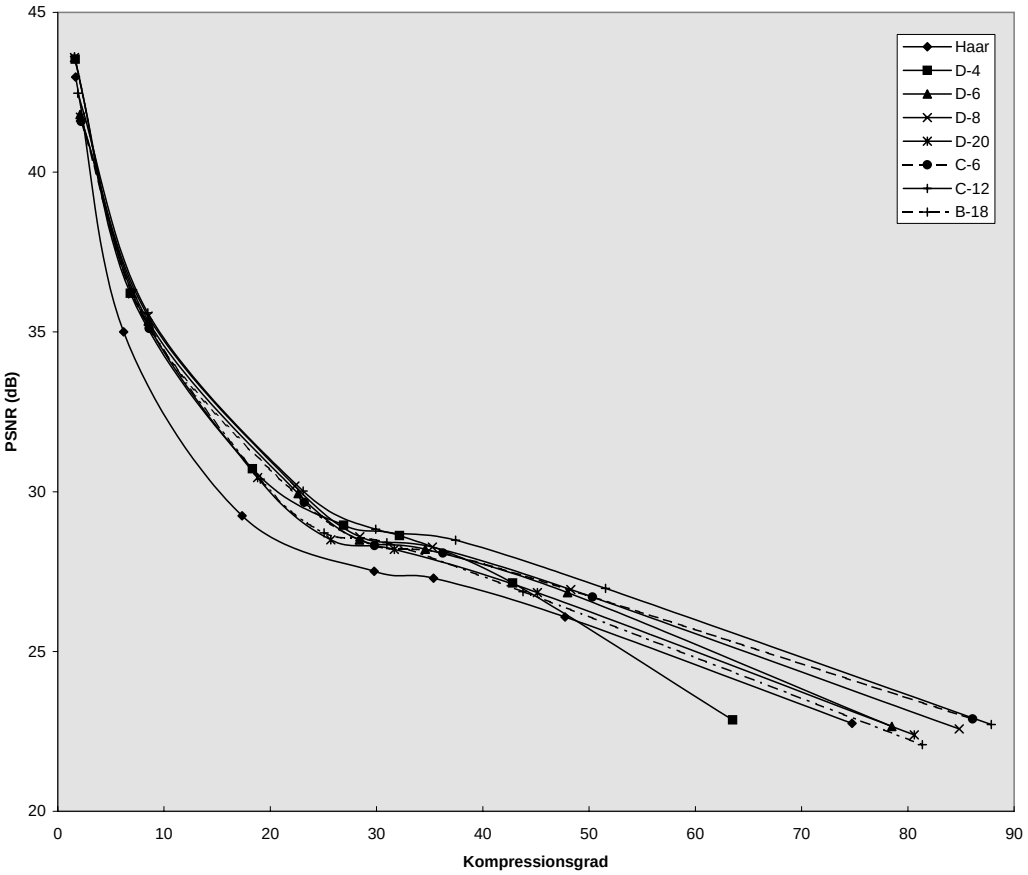


Abbildung 6.1: Diagramm: Abhängigkeit der Bildqualität vom Basiswavelet („Lena“)

sionsraten vom Typ des Bildes möchte ich den Leser jedoch auf Kapitel 6.6 verweisen. Daß Wavelets zur Kompression solcher Bilder immer noch besser geeignet sind als JPEG, zeigt sich in Kapitel 6.7.

Auffallend bei den beiden Bildern „Floor“ und „Door“ ist, daß sie sich genau anders verhalten als das Bild „Lena“: Das Haar-Wavelet ist sehr gut zur Bildkompression geeignet, während die Bildqualität bei gleichem Kompressionsgrad bei der Verwendung von anderen Wavelets (hier D-4 und D-6) doch ein ganzes Stück schlechter ist. Als Beispiel hierfür sind in Abbildung 6.7 komprimierte Versionen von „Floor“ zu sehen, die auf sechs verschiedene Arten entstanden sind. Die Bilder auf der linken Seite wurden mit dem Haar-Wavelet komprimiert, die auf der rechten Seite mit dem D-6-Wavelet. Bei den Bildern dieser Abbildung sind die Kompressionsraten vergleichbar, wodurch die Unterschiede in der Bildqualität bei den beiden Wavelets deutlich sichtbar werden. Besonders auffällig ist bei höheren Kompressionsraten, daß ehemals weiße Flächen im Bild nun einen Grauschleier haben.

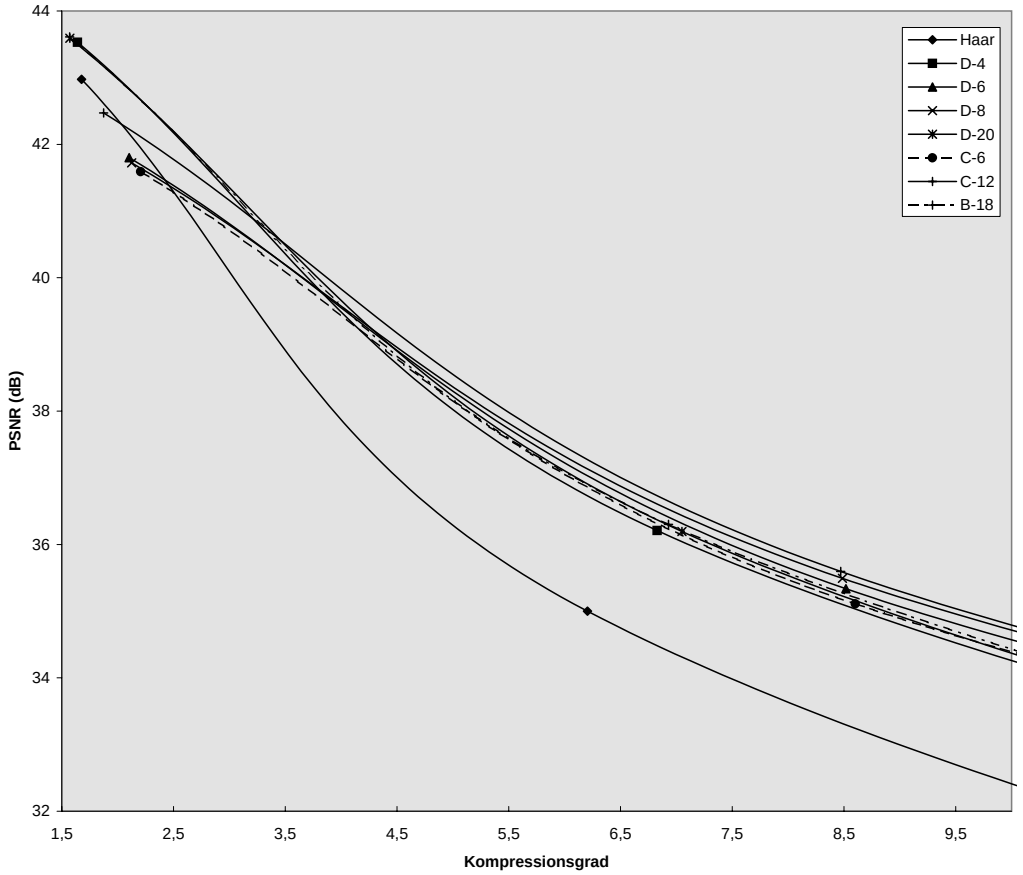


Abbildung 6.2: Diagramm: Abhängigkeit der Bildqualität vom Basiswavelet (Ausschnitt 1)

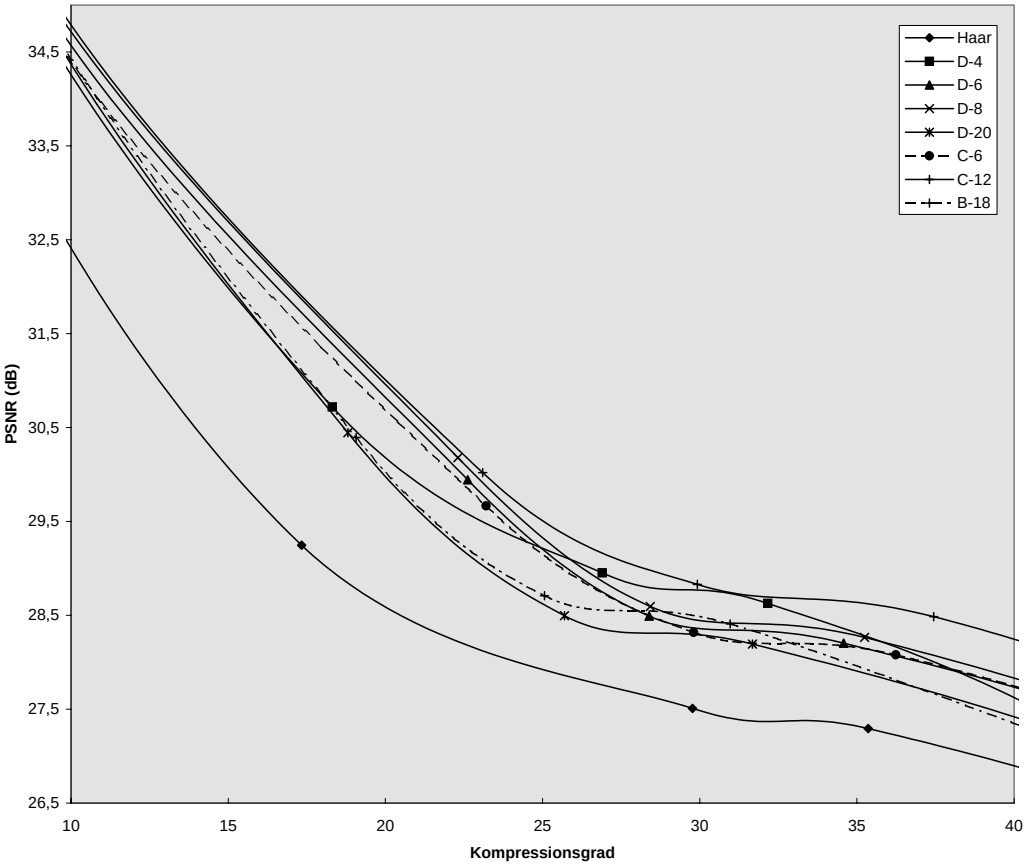


Abbildung 6.3: Diagramm: Abhängigkeit der Bildqualität vom Basiswavelet (Ausschnitt 2)

Wavelet	Kompressionsgrad	RMSE	PSNR
Bit pro Band: 6,7,8; Lauflänge: 8,7,6,5			
Haar	4,778	1,076	47,493
D-4	3,356	1,651	43,775
D-6	3,052	2,171	41,399
Bit pro Band: 4,5,6,7,8; Lauflänge: 10,8,7,7,6			
Haar	6,611	2,966	38,688
D-4	5,277	6,024	32,533
D-6	5,122	7,113	31,089
Bit pro Band: 3,3,4,6,7; Lauflänge: 11,9,8,7			
Haar	8,971	9,788	28,317
D-4	8,687	14,125	25,131
D-6	8,608	16,515	23,773
Bit pro Band: 2,2,4,7,8; Lauflänge: 13,11,8,7			
Haar	13,165	21,496	21,483
D-4	13,636	25,925	19,857
D-6	13,617	28,930	18,904

Tabelle 6.2: Abhängigkeit der Bildqualität vom Basiswavelet („Floor“)

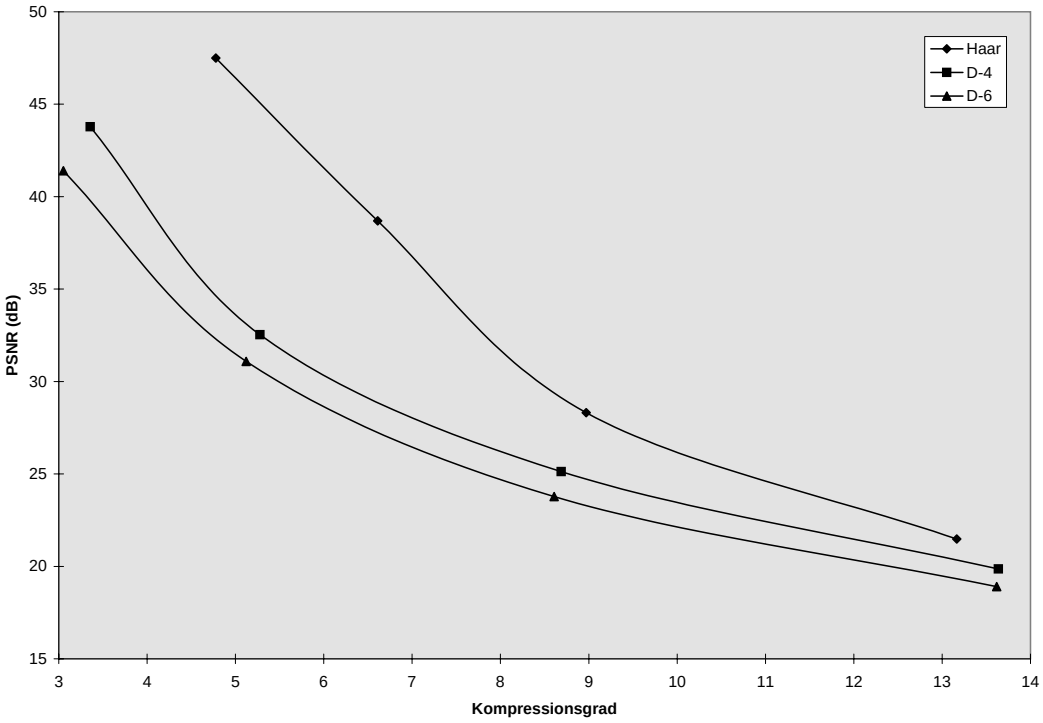


Abbildung 6.4: Diagramm: Abhängigkeit der Bildqualität vom Basiswavelet („Floor“)

Wavelet	Kompressionsgrad	RMSE	PSNR
Bit pro Band: 6,7,8; Lauflänge: 8,7,6,5			
Haar	3,585	1,327	45,676
D-6	2,387	1,864	42,720
Bit pro Band: 4,5,6,7,8; Lauflänge: 10,8,7,7,6			
Haar	4,690	4,631	34,817
D-6	3,864	8,198	29,857
Bit pro Band: 3,3,4,6,7; Lauflänge: 11,9,8,7			
Haar	5,905	11,255	27,104
D-6	5,932	20,331	21,968
Bit pro Band: 2,2,4,7,8; Lauflänge: 13,11,8,7			
Haar	6,738	22,453	21,105
D-6	8,508	41,435	15,783

Tabelle 6.3: Abhängigkeit der Bildqualität vom Basiswavelet („Door“)

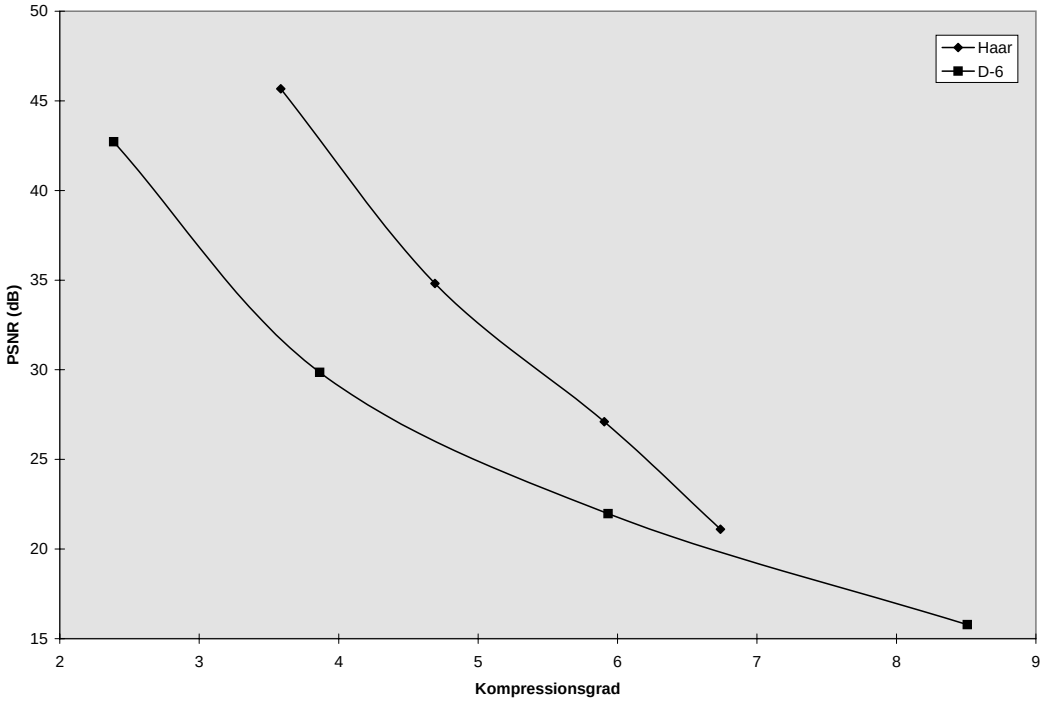


Abbildung 6.5: Diagramm: Abhängigkeit der Bildqualität vom Basiswavelet („Door“)

(a) Haar, 6,2:1, 35dB

(b) D-6, 8,5:1, 35,3dB

(c) Haar, 17,3:1, 29,2dB

(d) D-6, 22,6:1, 29,9dB

(e) Haar, 47,1:1, 26,1dB

(f) D-6, 48:1, 26,8dB

Abbildung 6.6: „Lena“ komprimiert mit Haar-Wavelet und D-6-Wavelet

(a) Haar, 4,8:1, 47,5dB

(b) D-6, 5,1:1, 31,1dB

(c) Haar, 9:1, 28,3dB

(d) D-6, 8,6:1, 23,8dB

(e) Haar, 13,1:1, 21,5dB

(f) D-6, 13,6:1, 18,9dB

Abbildung 6.7: „Floor“ komprimiert mit Haar-Wavelet und D-6-Wavelet

6.4 Auswirkungen der Datenanpassung

In diesem Kapitel möchte ich anhand einiger Auswertungen kurz erläutern, warum bei allen Messungen als Art der Datenanpassung „mit Nullen aufgefüllt“ und als Position des Originalbildes in den erweiterten Daten „rechts oben“ gewählt wurde. Der Grund ist natürlich, daß sich mit diesen Einstellungen die beste Bildqualität bei gleicher Kompressionsrate erreichen läßt.

Folgende Parameter wurden für die folgenden Messungen konstant gelassen:

- das Bild: Lena, 512×512
- das Wavelet: D-6
- die Anzahl der Iterationsschritte: 5
- die Art der Quantisierung: getrennt nach Frequenzbändern mit einem Faktor von 2 für das Nullintervall, (4,5,6,7,8) Bit pro Band und (10,8,7,7,6) Bit für die Lauflänge

Geändert wurde einmal die Position des Bildes innerhalb der erweiterten Daten und einmal die Art der Datenanpassung.

Zunächst zur Bildposition. Für die Ermittlung der in Tabelle 6.4 enthaltenen Werte wurde als Datenanpassung „mit Nullen aufgefüllt“ gewählt, die Bildposition wurde jeweils geändert. Der Inhalt der Tabelle ist in Form eines Balkendiagramms in Abbildung 6.8 auch graphisch dargestellt, wobei sich die in der Tabelle in Spalte „Position“ in Klammern angegebene Nummer auf die Beschriftung der horizontalen Achse der Abbildung bezieht.

Es zeigt sich, daß die beste Position des Bildes im rechten oberen Eck ist, da die Koeffizienten, die verloren gehen, sich am linken und am unteren Rand befinden. Unterschiede zwischen den Positionen „Mitte“ und „rechts oben“ werden bei dieser Messung nicht sichtbar, da sich die verlorengegangenen Koeffizienten nicht bis zur Mitte des Bildes auswirken. Dies ist erst bei Wavelets mit mehr Filterkoeffizienten der Fall, da bei einer größeren Zahl an Filterkoeffizienten auch mehr Daten an den Rändern verloren gehen.

Abbildung 6.10 zeigt das Bild „Lena“ mit verschiedenen Werten für die Bildposition komprimiert.

Nun soll die Bildposition gleich bleiben, sofern sie überhaupt benötigt wird (rechts oben) und die Art der Datenanpassung wird geändert. Die Ergebnisse befinden sich in Tabelle 6.5, das zugehörige Balkendiagramm zeigt Abbildung 6.9.

Wie man sieht, ist eine Erweiterung der Daten dringend nötig, da die Ergebnisse bei „keine Datenanpassung“ und „künstlich periodisch“ doch sehr unbefriedigend ausfallen. Der Kompressionsgrad ist beim Auffüllen der Daten mit Nullen und sonst gleichgebliebenen Parametern der gleiche, es wird jedoch eine wesentlich bessere Bildqualität erreicht. Dies mag im ersten Moment verwunderlich

Position	Kompressionsgrad	RMSE	PSNR
rechts oben (1)	8,518	4,362	35,337
links oben (2)	8,970	15,960	24,070
rechts unten (3)	8,472	17,908	23,070
links unten (4)	8,569	23,531	20,698
Mitte (5)	8,518	4,362	35,337

Tabelle 6.4: Einfluß der Position des Bildes in den erweiterten Daten

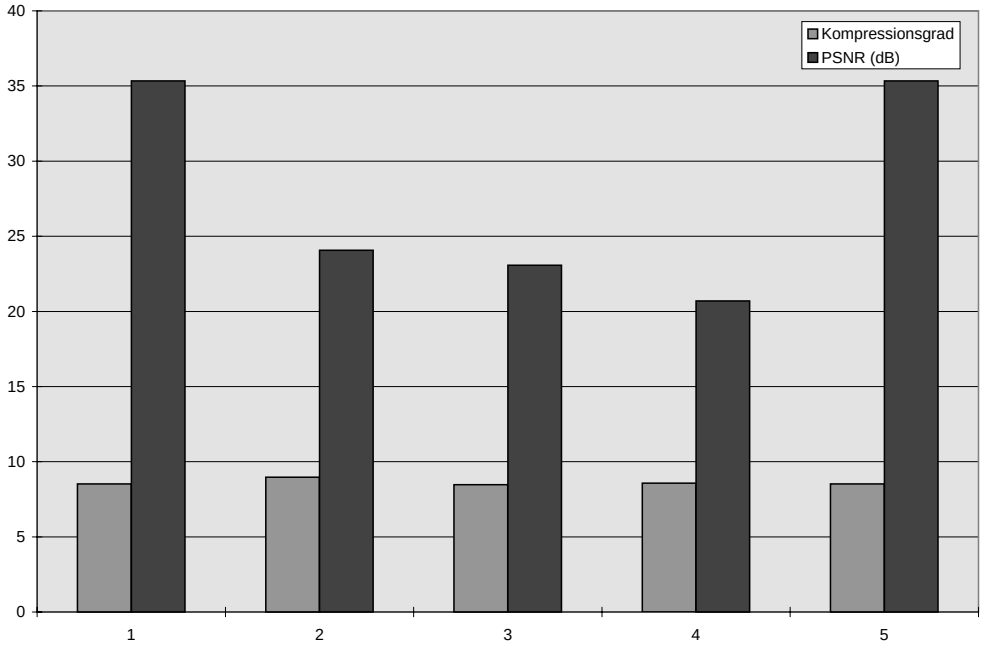


Abbildung 6.8: Diagramm: Einfluß der Position des Bildes in den erweiterten Daten: (1) rechts oben, (2) links oben, (3) rechts unten, (4) links unten, (5) Mitte

erscheinen, da wegen der Datenanpassung die Menge der zu komprimierenden Bildpunkte um den Faktor Vier zugenommen hat. Dies wird jedoch durch die Lauflängencodierung wieder wettgemacht.

Eben diese Lauflängencodierung ist bei den Einstellungen „reflektiert“ und „periodisch“ nicht möglich, weshalb die Kompressionsrate auch wesentlich kleiner ist als vorher, wogegen die Bildqualität etwas besser ist. Auch hier wurden wieder alle anderen Parameter beibehalten.

Die hier ermittelten Daten lassen allerdings noch nicht den Schluß zu, daß für eine gute Bildqualität auch eine Erweiterung der Daten zwingend erforderlich ist, obwohl dies bei den hier vorgestellten Verfahren tatsächlich der Fall ist. Es besteht immer noch die Möglichkeit eine Datenanpassung ohne Erweiterung vorzunehmen, zum Beispiel mit Hilfe der SET-Methode[6], über die jedoch leider

[6]siehe auch Kapitel 4.3

keine näheren Informationen zu bekommen waren.

Auch für diese Auswertung ist in Abbildung 6.11 das Bild „Lena“ gezeigt, komprimiert mit verschiedenen Werten für die Bildposition.

Datenanpassung	Kompressionsgrad	RMSE	PSNR
keine (1)	9,160	23,531	20,698
künstlich periodisch (2)	8,139	23,009	20,893
mit Nullen aufgefüllt (3)	8,518	4,362	35,337
reflektiert (4)	2,410	4,217	35,631
periodisch (5)	2,300	4,159	35,752

Tabelle 6.5: Auswirkungen der Art der Datenanpassung

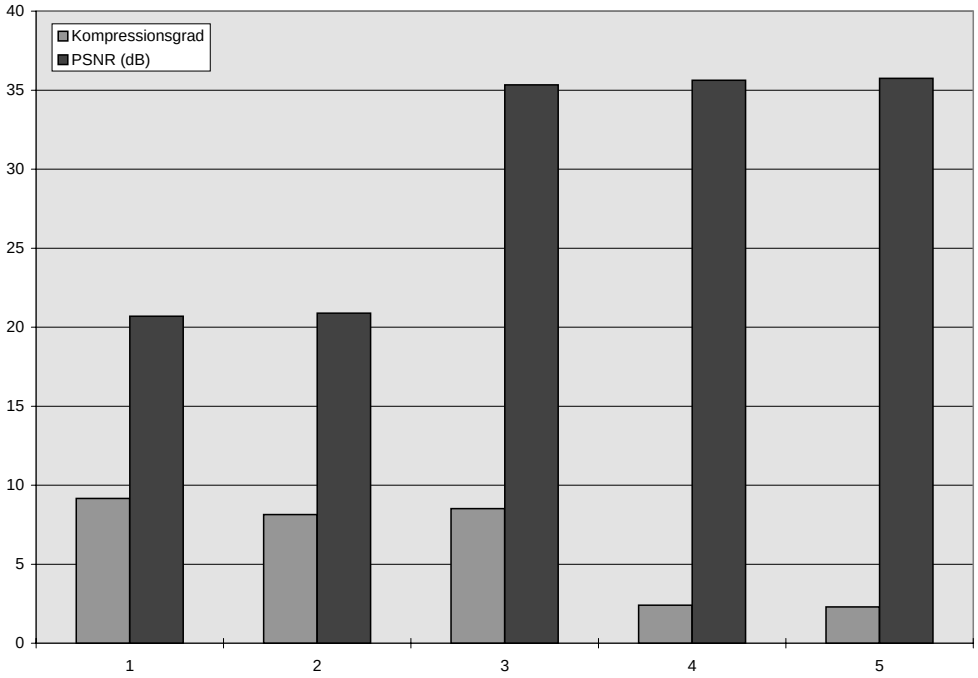


Abbildung 6.9: Diagramm: Auswirkungen der Art der Datenanpassung: (1) keine, (2) künstlich periodisch, (3) mit Nullen aufgefüllt, (4) reflektiert, (5) periodisch

(a) links oben, 9:1, 24,1dB

(b) rechts oben, 8,5:1, 35,3dB

(c) links unten, 8,6:1, 20,7dB

(d) rechts unten, 8,5:1, 23,1dB

Abbildung 6.10: „Lena“ komprimiert mit verschiedenen Positionen des Bildes in den erweiterten Daten; die Position „Mitte“ fehlt hier, da sie mit „rechts oben“ identisch ist.

(a) keine, 9,2:1, 20,7dB

(b) künstl. periodisch, 8,1:1, 23dB

(c) Nullen, 8,5:1, 35,3dB

(d) reflektiert, 2,4:1, 35,6dB

(e) periodisch, 2,3:1, 35,8dB

Abbildung 6.11: „Lena“ komprimiert mit verschiedenen Arten der Datenanpassung

6.5 Auswirkungen der Quantisierung

Nun wird untersucht, wie sich die verschiedenen Einstellungen für die Quantisierung auf die erreichbare Kompressionsrate und Bildqualität auswirken. Folgende Parameter blieben hierbei immer konstant:

- das Bild: Lena, 512×512
- das Wavelet: D-6
- die Art der Datenanpassung: mit Nullen aufgefüllt
- die Position des Bildes in den erweiterten Daten: rechts oben

Verändert wurden zunächst die Einstellungen für die Quantisierung, wobei der Faktor für die Größenänderung des Nullintervalls gleich blieb (2). Es wurden Daten ermittelt für verschiedene Schwellwerte, globale und nach Frequenzbändern getrennte Quantisierung. Um Unterschiede in der Wirkung der globalen und getrennten Quantisierung zeigen zu können, wurden hierfür ebenfalls verschiedene Meßreihen mit verschiedenen Iterationstiefen ermittelt. Die Ergebnisse sind in Tabelle 6.6 enthalten. Zu den bei „Schwelle“ für die Anzahl der Bit zur Quantisierung bzw. Lauflängencodierung in Klammern angegebenen Werten (32) ist zu sagen, daß diese nicht direkt eingestellt werden können, da bei Quantisierung mit Schwellwert alle Koeffizienten und Lauflängen in der Größe einer Integer-Zahl (32 Bit) gespeichert werden. Eine graphische Darstellung der Ergebnisse befindet sich in Abbildung 6.12.

Beim Betrachten der Kurven fällt zunächst auf, daß die Quantisierung mit Schwellwert die schlechteste Möglichkeit ist. Bei der globalen Quantisierung sieht man, daß die bei einer bestimmten Kompressionsrate erreichbare Bildqualität stark von der Anzahl der durchgeführten Iterationsschritte abhängt. Je weniger Schritte verwendet wurden, desto höher der Signalrauschabstand. Dies liegt daran, daß bei jedem Schritt die Koeffizienten betragsmäßig immer größer werden, weshalb bei gleichbleibender globaler Quantisierung die Intervalle immer größer werden, wodurch die Quantisierungsfehler ebenfalls größer werden.

Dieser Effekt tritt bei Verwendung der getrennten Quantisierung nicht auf, weshalb die Meßkurven hier auch bei veränderter Schrittzahl deutlich sichtbar enger zusammenliegen.

Das Bild „Lena“, komprimiert mit verschiedenen Arten der Quantisierung wird in Abbildung 6.14 gezeigt.

Jetzt wurde die Größe des Quantisierungsintervalls verändert, in dem die Null liegt; hierzu wurden verschiedene Werte für den „Nullfaktor“ verwendet, wobei eine Quantisierung getrennt nach Bändern mit den gleichen Werten wie schon früher durchgeführt wurde. Tabelle 6.7 enthält die ermittelten Werte, die graphische Darstellung befindet sich in Abbildung 6.13.

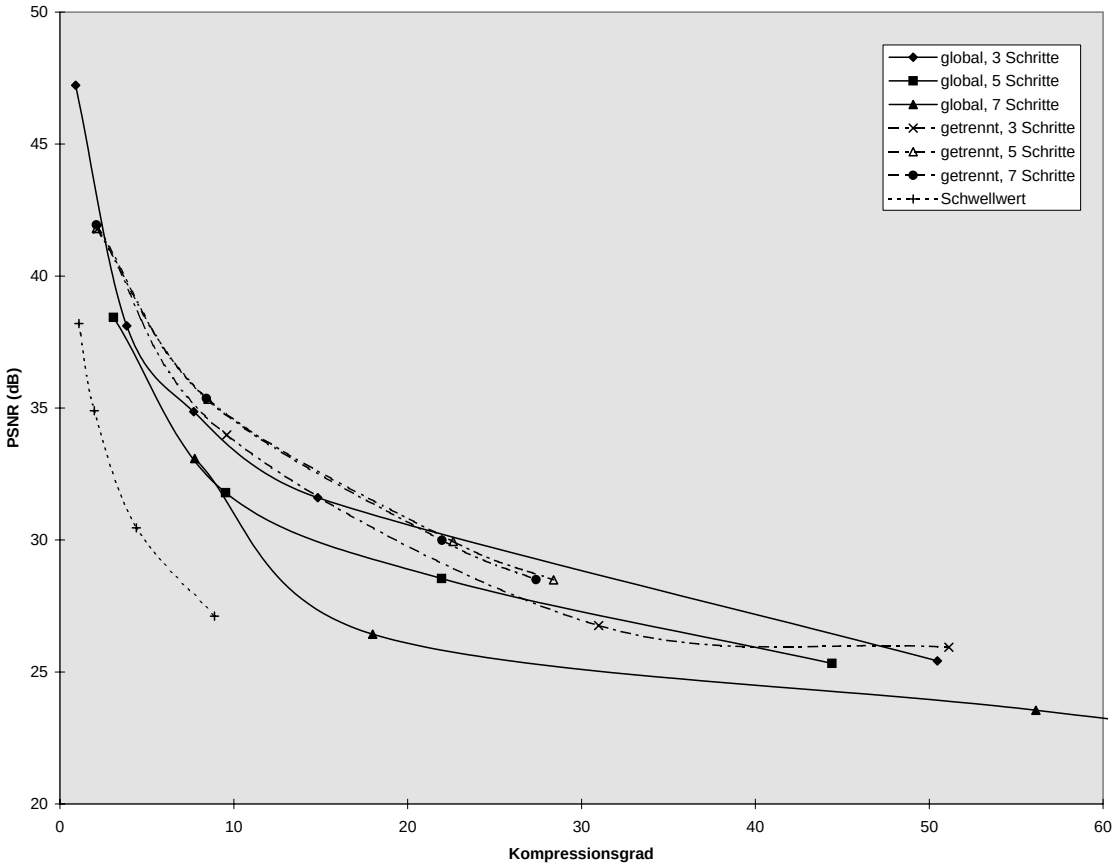


Abbildung 6.12: Diagramm: Einfluß der Art der Quantisierung

Es zeigt sich, daß ein Faktor größer als Zwei bei Kompressionsraten bis ungefähr 20:1 das Ergebnis negativ beeinflußt. Werden höhere Kompressionsraten gewünscht, so empfiehlt sich evtl. ein Wechsel vom Faktor Zwei zum Faktor Vier oder Sechs, wobei dies aber keine gravierenden Unterschiede machen dürfte, da die Kurven doch recht eng beieinander liegen.

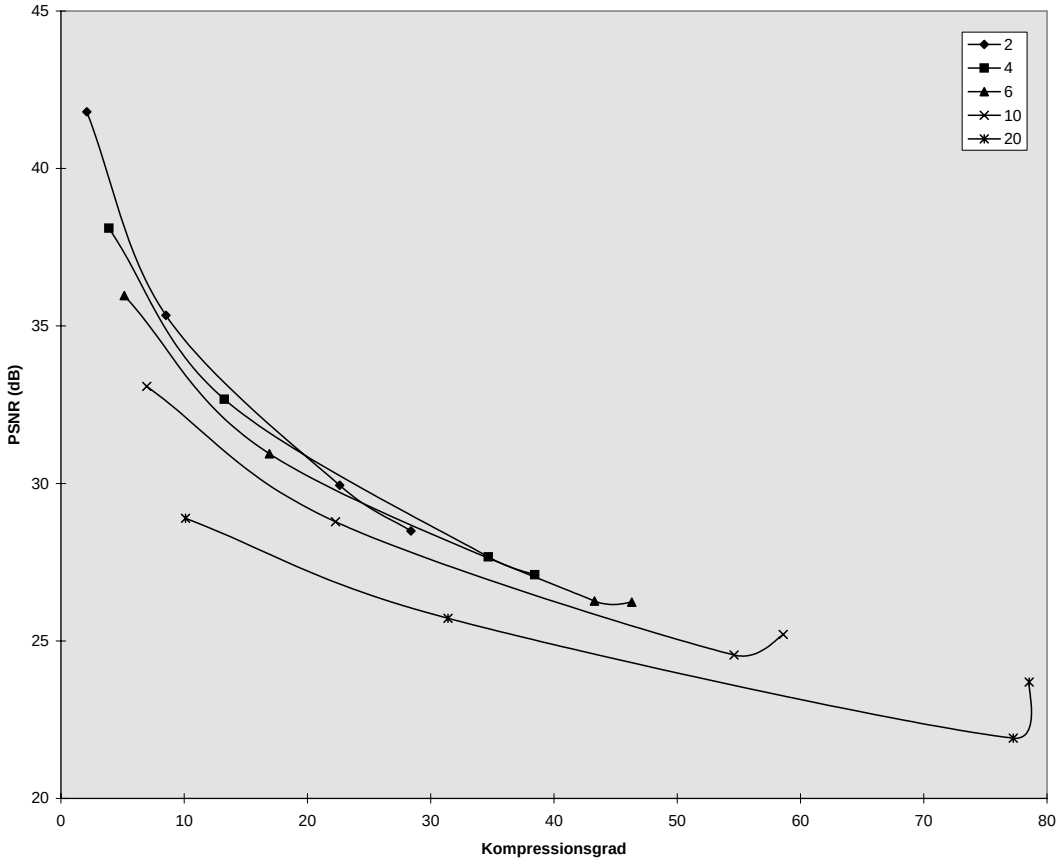


Abbildung 6.13: Diagramm: Einfluß der Größe des Nullintervalls

Quantisierung	Bit Quant.	Bit Lauflänge	Anz. Iterat.	Komp.-grad	RMSE	PSNR
ganze Zahlen	(32)	(32)	5	0,224	0,286	59,015
Schwelle: 10	(32)	(32)	5	1,098	3,136	38,202
Schwelle: 20	(32)	(32)	5	1,972	4,587	34,900
Schwelle: 50	(32)	(32)	5	4,397	7,650	30,458
Schwelle: 100	(32)	(32)	5	8,893	11,246	27,111
global	10	10	3	0,906	1,110	47,226
global	8	8	3	3,842	3,167	38,117
global	7	10	3	7,697	4,609	34,859
global	6	10	3	14,831	6,705	31,603
global	4	10	3	50,471	13,667	25,418
global	10	10	5	3,082	3,054	38,432
global	8	8	5	9,527	6,560	31,792
global	7	10	5	21,950	9,541	28,539
global	6	10	5	44,401	13,812	25,326
global	10	10	7	7,753	5,651	33,089
global	7	10	7	17,995	12,160	26,432
global	6	10	7	56,134	16,939	23,553
global	4	10	7	89,347	22,965	20,910
getrennt	6,7,8	8,7,6,5	3	2,171	2,076	41,785
getrennt	4,5,6,7,8	10,8,7,7,6	3	9,594	5,097	33,984
getrennt	3,3,4,6,7	11,9,8,7	3	30,990	11,714	26,756
getrennt	2,2,4,7,8	13,11,8,7	3	51,130	12,876	25,935
getrennt	6,7,8	8,7,6,5	5	2,103	2,072	41,801
getrennt	4,5,6,7,8	10,8,7,7,6	5	8,518	4,362	35,337
getrennt	3,3,4,6,7	11,9,8,7	5	22,616	8,116	29,944
getrennt	2,2,4,7,8	13,11,8,7	5	28,389	9,593	28,492
getrennt	6,7,8	8,7,6,5	7	2,098	2,040	41,940
getrennt	4,5,6,7,8	10,8,7,7,6	7	8,427	4,347	35,366
getrennt	3,3,4,6,7	11,9,8,7	7	21,975	8,073	29,990
getrennt	2,2,4,7,8	13,11,8,7	7	27,387	9,585	28,499

Tabelle 6.6: Einfluß der Art der Quantisierung

Nullfaktor	Kompressionsgrad	RMSE	PSNR
Bit pro Band: 6,7,8; Lauflänge: 8,7,6,5			
2	2,103	2,072	41,801
4	3,893	3,172	38,105
6	5,144	4,057	35,968
10	6,962	5,657	33,080
20	10,109	9,519	28,894
Bit pro Band: 4,5,6,7,8; Lauflänge: 10,8,7,7,6			
2	8,518	4,362	35,337
4	13,248	5,927	32,673
6	16,918	7,234	30,945
10	22,280	9,273	28,786
20	31,398	13,205	25,716
Bit pro Band: 3,3,4,6,7; Lauflänge: 11,9,8,7			
2	22,616	8,116	29,944
4	34,680	10,548	27,667
6	43,294	12,396	26,265
10	54,613	15,093	24,556
20	77,260	20,460	21,913
Bit pro Band: 2,2,4,7,8; Lauflänge: 13,11,8,7			
2	28,389	9,593	28,492
4	38,449	11,259	27,101
6	46,315	12,446	26,230
10	58,593	14,004	25,206
20	78,557	16,667	23,694

Tabelle 6.7: Einfluß der Größe des Nullintervalls

(a) global, 7,8:1, 33,1dB

(b) getrennt, 8,4:1, 35,3dB

(c) global, 22:1, 28,5dB

(d) getrennt, 22,6:1, 29,9dB

Abbildung 6.14: „Lena“ komprimiert mit globaler und nach Iterationsschritten getrennter Quantisierung

6.6 Auswirkungen des Bildmaterials

Es stellt sich die Frage, ob die Kompression von Bildern mit Wavelets für alle Arten von Bildern gleichermaßen gut geeignet ist, oder ob es, abhängig vom Typ des Bildes, Unterschiede im erreichbaren Kompressionsgrad bzw. der Bildqualität gibt. Auf diesen Punkt wurde bereits in Kapitel 6.3 kurz eingegangen, wo der Einfluß des Basiswavelets auf die Kompressionsrate untersucht wurde. Es stellte sich heraus, das nicht ein einziges Wavelet für jedes Bild verwendet werden sollte, sondern daß sich die Auswahl nach dem zu komprimierenden Bild richten muß. Der Auslöser für die dort durchgeführten Messungen waren dabei die nun vorgestellten Ergebnisse. Es sollten, bei sonst gleicher Parameterwahl, verschiedene Bilder komprimiert werden. Erfaßt wurden dann die bei einer bestimmten Einstellung erreichte Kompressionsrate und Bildqualität.
Verwendet wurden die folgenden Parameter:

- Anzahl der Iterationsschritte: 5
- Art der Datenanpassung: mit Nullen aufgefüllt
- Position des Bildes in den erweiterten Daten: rechts oben
- Art der Quantisierung: getrennt nach Frequenzbändern mit einem Faktor von Zwei für das Nullintervall und unterschiedlich starken Quantisierungen zur Änderung des Kompressionsgrades

Mit diesen Einstellungen sollten sechs verschiedene Bilder mit unterschiedlichen Charakteristiken komprimiert werden. Darunter befinden sich drei „natürliche“ Bilder, also Bilder wie sie durch eine Photographie entstehen (Lena, Boat, Peppers) und zwei Bilder, die in die Richtung von technischen Zeichnungen gehen, die also sehr viele schwarze Linien in unterschiedlichen Stärken auf weißem Grund enthalten (Floor, Door). Das letzte Bild (Mandrill) spielt eine Sonderrolle, da es zwar auch ein „natürliches“ Bild ist, es aber sehr viele, teilweise starke Änderungen der Pixelwerte im Fell um das Gesicht herum enthält (siehe auch Abbildung im Anhang A).

Tabelle 6.8 enthält die Ergebnisse der Auswertung, die dazugehörenden Kurven befinden sich in Abbildung 6.15. Das Ergebnis sieht wie folgt aus:

Am besten komprimieren ließen sich die Bilder „Lena“ und „Peppers“, das Bild „Boat“ erreichte bei gleichen Kompressionsraten einen etwas schlechteren PSNR, es bewegt sich aber dennoch etwa im selben Rahmen wie die beiden erstgenannten.

Nicht unbedingt überraschend ist das Ergebnis für „Mandrill“, da nach der Transformation dieses Bildes bedingt durch die starken Änderungen der Grauwerte wesentlich weniger Nullkoeffizienten bei hohen Frequenzen auftreten als bei den ersten drei Bildern. Daher war es zu erwarten, daß die Meßkurve unterhalb derjenigen der obigen Bilder liegt. Dennoch fallen die Werte ein ganzes

Stück schlechter aus als zunächst angenommen. Hierzu ist allerdings anzumerken, daß gerade bei diesem Bild der Signalrauschabstand die tatsächlichen Verhältnisse schlecht wiedergibt, da Änderungen der Grauwerte im Fell des „Mandrill“ vom Auge auch bei relativ großen Abweichungen fast nicht wahrgenommen werden.

Die Bilder „Floor“ sowie „Door“ gehen tendenziell in die gleiche Richtung wie die von „Mandrill“, bis zu Kompressionsraten von ungefähr 4:1 bzw. 8:1 liefern sie sogar bessere Ergebnisse. Im Gegensatz zu „Mandrill“, wo die Kurve mit zunehmendem Kompressionsgrad flacher wird, nimmt die Steigung bei diesen beiden Bildern nicht so stark ab. Bei (für diese Bilder) hohen Kompressionsraten werden die Fehler im Bild durch Grauschlieren auf dem weißen Hintergrund sichtbar, die schwarzen Linien verschmieren sozusagen.

Da das Ergebnis an dieser Stelle so schlecht ausfiel, wurden anschließend die bereits vorgestellten Versuche mit dem Haar-Wavelet durchgeführt, wo die Ergebnisse deutlich besser sind als hier. Es sind jedoch auch damit keine so hohen Kompressionsgrade erreichbar wie mit „Lena“.

Es bleibt also festzuhalten, daß sich „natürliche“ Bilder mit Wavelet-Bildkompressionsverfahren wesentlich besser komprimieren lassen als künstlich erzeugte, wie die hier vorliegenden „Floor“ und „Door“.

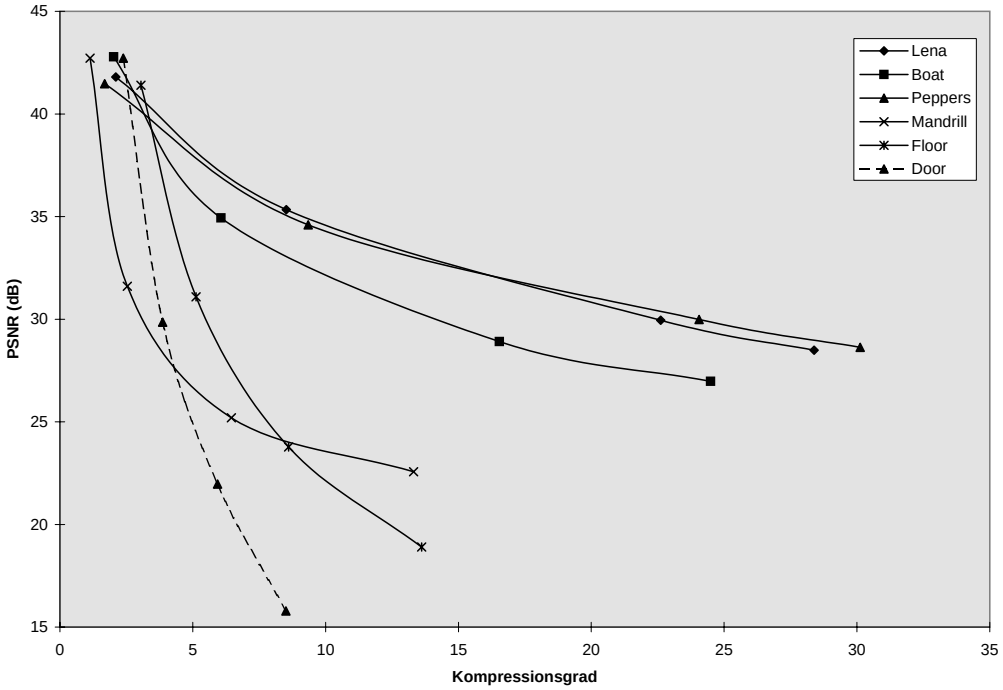


Abbildung 6.15: Diagramm: Einfluß des Bildmaterials auf die Kompressionsrate

Bild	Kompressionsgrad	RMSE	PSNR
Bit pro Band: 6,7,8; Lauflänge: 8,7,6,5			
Lena	2,103	2,072	41,801
Boat	2,020	1,851	42,781
Peppers	1,683	2,153	41,470
Mandrill	1,137	1,865	42,717
Floor	3,052	2,171	41,399
Door	2,387	1,864	42,720
Bit pro Band: 4,5,6,7,8; Lauflänge: 10,8,7,7,6			
Lena	8,518	4,362	35,337
Boat	6,061	4,572	34,930
Peppers	9,344	4,752	34,594
Mandrill	2,538	6,704	31,604
Floor	5,122	7,113	31,089
Door	3,864	8,198	29,857
Bit pro Band: 3,3,4,6,7; Lauflänge: 11,9,8,7			
Lena	22,616	8,116	29,944
Boat	16,543	9,145	28,908
Peppers	24,061	8,075	29,988
Mandrill	6,453	14,026	25,192
Floor	8,608	16,515	23,773
Door	5,932	20,331	21,968
Bit pro Band: 2,2,4,7,8; Lauflänge: 13,11,8,7			
Lena	28,389	9,593	28,492
Boat	24,495	11,419	26,978
Peppers	30,121	9,445	28,626
Mandrill	13,309	18,974	22,568
Floor	13,617	28,930	18,904
Door	8,508	41,435	15,783

Tabelle 6.8: Einfluß des Bildmaterials auf die Kompressionsrate

6.7 Vergleich mit JPEG

Zum Vergleich mit JPEG wurden die jeweiligen Bilder mit einem Grafikprogramm[7] komprimiert. Das entstandene Bild wurde anschließend über die Zwischenablage als Bitmap an Stelle eines komprimierten Bildes ins Programm zur Wavelet-Bildkompression eingefügt, um den mittleren quadratischen Fehler sowie den Signalrauschabstand zu berechnen.

Da bei JPEG nach Transformation und Lauflängencodierung noch eine Huffman-Codierung erfolgt, sind die entstehenden Dateien nicht direkt mit denen des Waveletkompressionsprogramms vergleichbar. Um dennoch einen ungefähren Vergleichswert zu bekommen, wurden die vom Programm gespeicherten WLT-Dateien mit Hilfe des Kompressionsprogramms ARJ huffmancodiert. Der Kompressionsgrad wurde anschließend aus den Größen der Dateien auf der Festplatte errechnet und nicht wie vorher aus dem im Hauptspeicher belegten Platz (was ja nun nicht mehr möglich war).

Es soll an dieser Stelle kein aufwendiger Vergleich der beiden Verfahren erfolgen, es soll vielmehr nur gezeigt werden, was sich mit den in dieser Diplomarbeit implementierten Methoden gegenüber JPEG bereits erreichen läßt. Dabei möchte ich auch darauf hinweisen, daß die Möglichkeiten, die sich mit der Wavelet-Bildkompression ergeben mit den in dieser Auswertung verwendeten Parametern noch lange nicht erschöpft ist. Doch dazu mehr in Kapitel 7.

Zum Vergleich wurden die Bilder „Lena“ und „Floor“ herangezogen, wobei das erste Bild mit dem D-6-Wavelet komprimiert wurde und das zweite Bild mit dem Haar-Wavelet. Für beide Bilder wurden JPEG-Kompressionen mit Kompressionsraten von ca. 2:1 bis 50:1 durchgeführt. Die Meßwerte sind in den Tabellen 6.9 und 6.10 für das Bild „Lena“ bzw. 6.11 und 6.12 für das Bild „Floor“ angegeben. Die Größe der der Originaldateien im Bitmap-Format auf die sich die Werte beziehen betrug 263224 Byte.

Das Ergebnis des Vergleichs kann als sehr positiv bezeichnet werden: Betrachtet man die Kurven für das Bild „Lena“, so stellt man fest, daß die Werte für JPEG und Wavelet-Kompression bis zu Kompressionsraten von ungefähr 35:1 eng beieinander liegen; ab diesem Punkt wird die Bildqualität bei JPEG sehr schnell sehr schlecht, während sie bei der Wavelet-Kompression bis 100:1 (das war der letzte Meßwert) nur langsam abnimmt[8].

Für das Bild „Floor“ sieht die Sache im Vergleich zu JPEG sogar noch besser aus: Obwohl bei diesem Bild bei weitem keine so hohen Kompressionsraten erreicht wurden wie mit „Lena“, sind die Werte durchgehend besser als bei JPEG. Zum Vergleich sind auch für dieses Bild in Abbildung 6.18 die Resultate der Kompression abgebildet.

[7] PixFolio, Shareware

[8] siehe auch Abbildung 6.17

Dateigröße komprimiert	Kompressionsgrad	RMSE	PSNR
162400	1,621	0,296	58,706
61858	4,255	2,336	40,760
40227	6,543	3,029	38,503
31210	8,434	3,476	37,310
25683	10,249	3,838	36,450
22618	11,638	4,136	35,799
19570	13,450	4,471	35,122
16593	15,864	4,927	34,280
13355	19,710	5,736	32,959
9521	27,647	7,692	30,410
7155	36,789	10,967	27,329
5110	51,512	32,260	17,957

Tabelle 6.9: „Lena“ komprimiert mit JPEG

Dateigröße komprimiert	Kompressions-grad	RMSE	PSNR	Bit Quant.	Bit Lauflänge
83616	3,148	2,072	41,801	6,7,8	8,7,6,5
25293	10,407	4,362	35,337	4,5,6,7,8	10,8,7,7,6
9872	26,664	8,116	29,944	3,3,4,6,7	11,9,8,7
7874	33,430	9,593	28,492	2,2,4,7,8	13,11,8,7
6236	42,210	9,917	28,203	2,2,4,5,7	13,11,8,7
4609	57,111	11,595	26,846	2,2,3,4,7	13,11,8,7
2625	100,276	18,787	22,654	2,2,2,3,4	13,11,9,7

Tabelle 6.10: „Lena“ komprimiert mit Wavelet-Bildkompression

Dateigröße komprimiert	Kompressionsgrad	RMSE	PSNR
113630	2,317	0,136	65,473
64474	4,083	1,842	42,827
52386	5,025	3,333	37,675
45858	5,740	4,664	34,756
41568	6,332	5,948	32,643
38429	6,850	7,195	30,990
35149	7,489	8,704	29,337
31465	8,366	10,670	27,568
26539	9,918	14,045	25,180
19236	13,684	19,927	22,142
13167	19,991	28,328	19,087
5322	49,460	51,553	13,886

Tabelle 6.11: „Floor“ komprimiert mit JPEG

Dateigröße komprimiert	Kompressions-grad	RMSE	PSNR	Bit Quant.	Bit Lauflänge
35591	7,396	1,076	47,493	6,7,8	8,7,6,5
28897	9,109	2,966	38,688	4,5,6,7,8	10,8,7,7,6
21107	12,471	9,788	28,317	3,3,4,6,7	11,9,8,7
13642	19,295	21,496	21,483	2,2,4,7,8	13,11,8,7

Tabelle 6.12: „Floor“ komprimiert mit Wavelet-Bildkompression

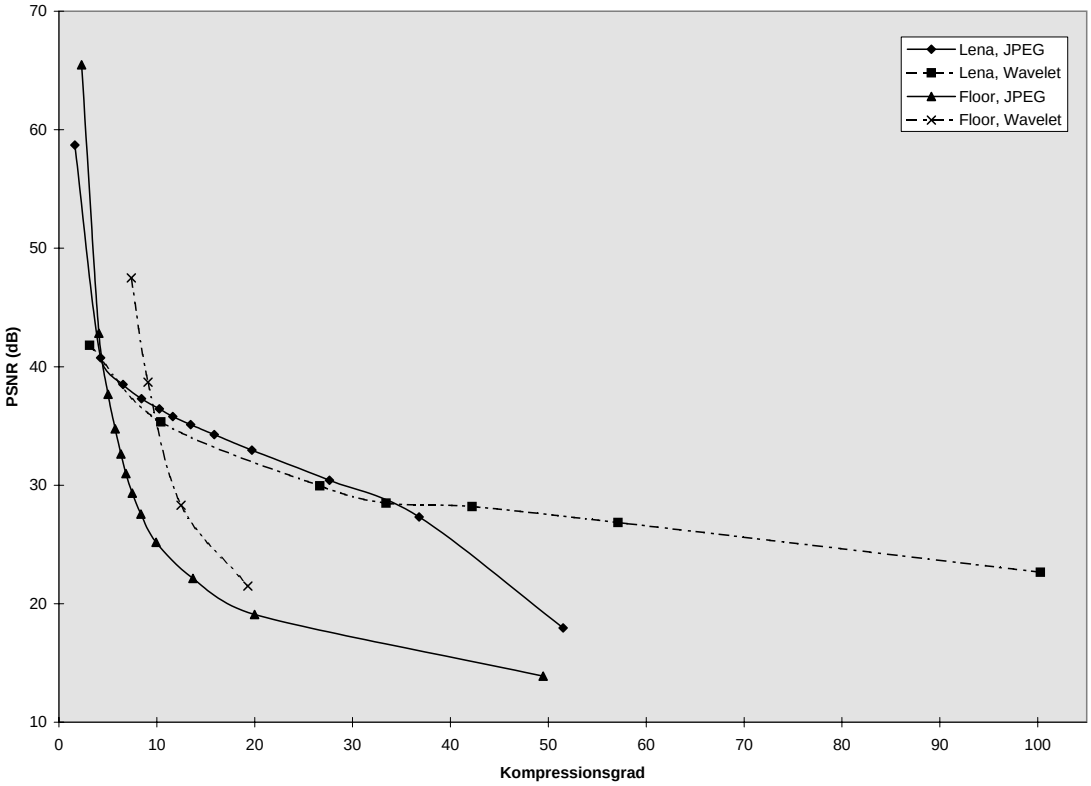


Abbildung 6.16: Diagramm: Vergleich mit JPEG

(a) JPEG, 10,2:1, 36,5dB

(b) D-6, 10,4:1, 35,3dB

(c) JPEG, 27,6:1, 30,4dB

(d) D-6, 26,7:1, 29,9dB

(e) JPEG, 51,5:1, 18dB

(f) D-6, 57,1:1, 26,8dB

Abbildung 6.17: „Lena“ komprimiert mit JPEG und D-6-Wavelet

(a) JPEG, 7,5:1, 29,3dB

(b) Haar, 7,4:1, 47,5dB

(c) JPEG, 13,7:1, 22,1dB

(d) Haar, 12,5:1, 28,3dB

(e) JPEG, 20:1, 19,1dB

(f) Haar, 19,3:1, 21,5dB

Abbildung 6.18: „Floor“ komprimiert mit JPEG und Haar-Wavelet

6.8 Zusammenfassung

Die Ergebnisse der Auswertungen lassen sich wie folgt zusammenfassen:

- Die erreichbare Bildqualität hängt vom verwendeten Basiswavelet ab, wobei man allerdings nicht ein einziges Wavelet für jeden Bildtyp empfehlen kann.
- Auch die Kompressionsraten sind stark vom Typ des Bildes abhängig: Natürliche Bilder lassen sich besser komprimieren als künstlich erzeugte.
- Eine Erweiterung der Daten ist für eine gute Bildqualität dringend erforderlich, zumindest solange man keine aufwendigeren Verfahren einsetzt, die ohne diese auskommen. Auf die Kompressionsrate hat dies, wenn man die Erweiterung in Form eines Auffüllens mit Nullen vornimmt, praktisch keinen Einfluß. Die Rechenzeit und der Speicherbedarf des Bildes während der Transformation steigt natürlich an, was für manche Anwendungen vielleicht ein Problem sein könnte.
- Zur Quantisierung ist zu sagen, daß man in den allermeisten Fällen die beste Wahl mit einer nach Frequenzbändern getrennten Quantisierung trifft. Damit ist es möglich, hohe Frequenzen stärker zu quantisieren als niedrige, was sich auf die Bildqualität positiv auswirkt.
- Die Bildkompression mit Wavelets schneidet im Vergleich mit JPEG, zumindest was die hier untersuchten Bilder betrifft, recht gut ab: Natürliche Bilder lassen sich ungefähr genauso gut, teilweise sogar besser komprimieren als mit JPEG. Künstliche Bilder liefern durchgehend bessere Ergebnisse, wenn diese auch nicht so gut sind wie bei natürlichen. Dies ist besonders deshalb von Bedeutung, da die Wavelet-Bildkompression mit den hier vorgestellten Methoden noch lange nicht am Ende ist.

Kapitel 7

Ausblick

Zum Schluß möchte ich noch einige Vorschläge unterbreiten, welche Möglichkeiten man zur Erweiterung der hier behandelten Verfahren zur Wavelet-Bildkompression hat, womit sich die Bildqualität bei feststehender Kompressionsrate sicher noch verbessern ließe.

Zunächst einmal ist zu den hier verwendeten Wavelets zu sagen, daß diese mehr oder weniger willkürlich ausgesucht wurden, da mir leider über die Eignung orthogonaler Wavelets zur Bildkompression keinerlei Versuchsergebnisse vorlagen, das heißt, es kann keine verläßliche Aussage darüber gemacht werden, ob diese nun besonders gut oder aber vielleicht sogar besonders schlecht zur Bildkompression geeignet sind.

Damit komme ich zu den biorthogonalen Wavelets, die in dieser Arbeit zwar vorgestellt wurden, aber nicht implementiert werden konnten, da keine genauen Angaben zur Implementierung verfügbar waren. Jedenfalls sollte sich mit ihrer Hilfe die Bildqualität nochmals steigern lassen.

Weiterhin ist es für eine tatsächliche Anwendung natürlich erforderlich, daß anschließend an die Lauflängencodierung eine Huffman-Codierung erfolgt, welche hier zum Vergleich mit JPEG mittels des Komprimierungsprogramms ARJ simuliert wurde. Hier gibt es auch Ansatzpunkte für gänzlich andere Verfahren zur Entropiecodierung, wie zum Beispiel den in Kapitel 4.1.3 kurz erwähnten EZW-Algorithmus.

Auch bezüglich der Behandlung der Daten an den Rändern gibt es noch andere Möglichkeiten als die Vergrößerung der Datenmenge. Hier ist zum einen die Verwendung von speziell angepaßten Wavelets an den Rändern zu nennen [15] und zum anderen die vom FBI verwendete SET-Methode (siehe Kapitel 4.3), über die jedoch keine näheren Informationen verfügbar waren.

Zudem existieren noch Erweiterungen der Wavelet-Bildkompression, die bisher unerwähnt blieben. Hier sind auf jeden Fall die sogenannten Wavelet-Pakete[1]

[1] engl.: *wavelet packets*

zu nennen, welche in [19] und [20] beschrieben sind. Dies sind Gebilde, die den Algorithmus zur schnellen diskreten Wavelet-Transformation dahingehend erweitern, daß nicht nur die nach der Tiefpaßfilterung entstandenen kleineren Bilder weiterverarbeitet werden, sondern auch die in jedem Schritt nach der Hochpaßfilterung entstandenen Waveletkoeffizienten. Es entsteht dann statt des sich beim Algorithmus von Mallat ergebenden Bildes (Heringgräte) ein Baum, der auf beiden Seiten gleich weit in die Tiefe geht. Eine genaue Abhandlung hätte im Rahmen dieser Arbeit zu weit geführt, ist aber in der oben genannten Literatur zu finden.

Anhang A

Bilder

In diesem Anhang sind alle für Auswertungen verwendete Bilder im Original enthalten. Alle haben eine Auflösung von 512x512 Punkten bei 256 Graustufen. Zu den Bildern „Floor“ und „Door“ ist zu sagen, daß diese Ausschnitte aus größeren Bildern darstellen, die aufgrund der Auflösung nicht im Original verwendet werden konnten (keine Zweierpotenzen).

A.1 Lena

Abbildung A.1: Lena

A.2 Boat

Abbildung A.2: Boat

A.3 Peppers

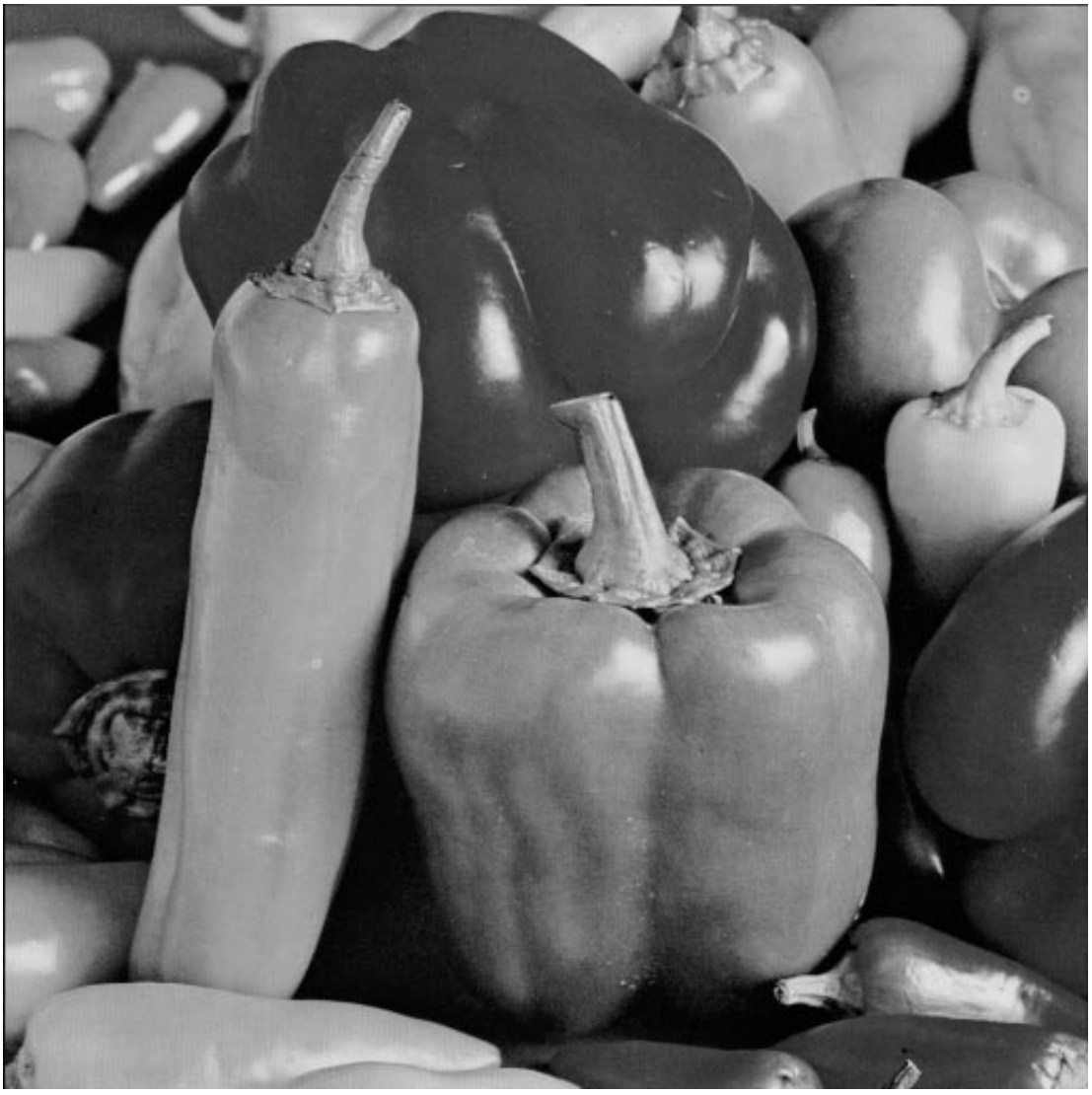


Abbildung A.3: Peppers

A.4 Mandrill

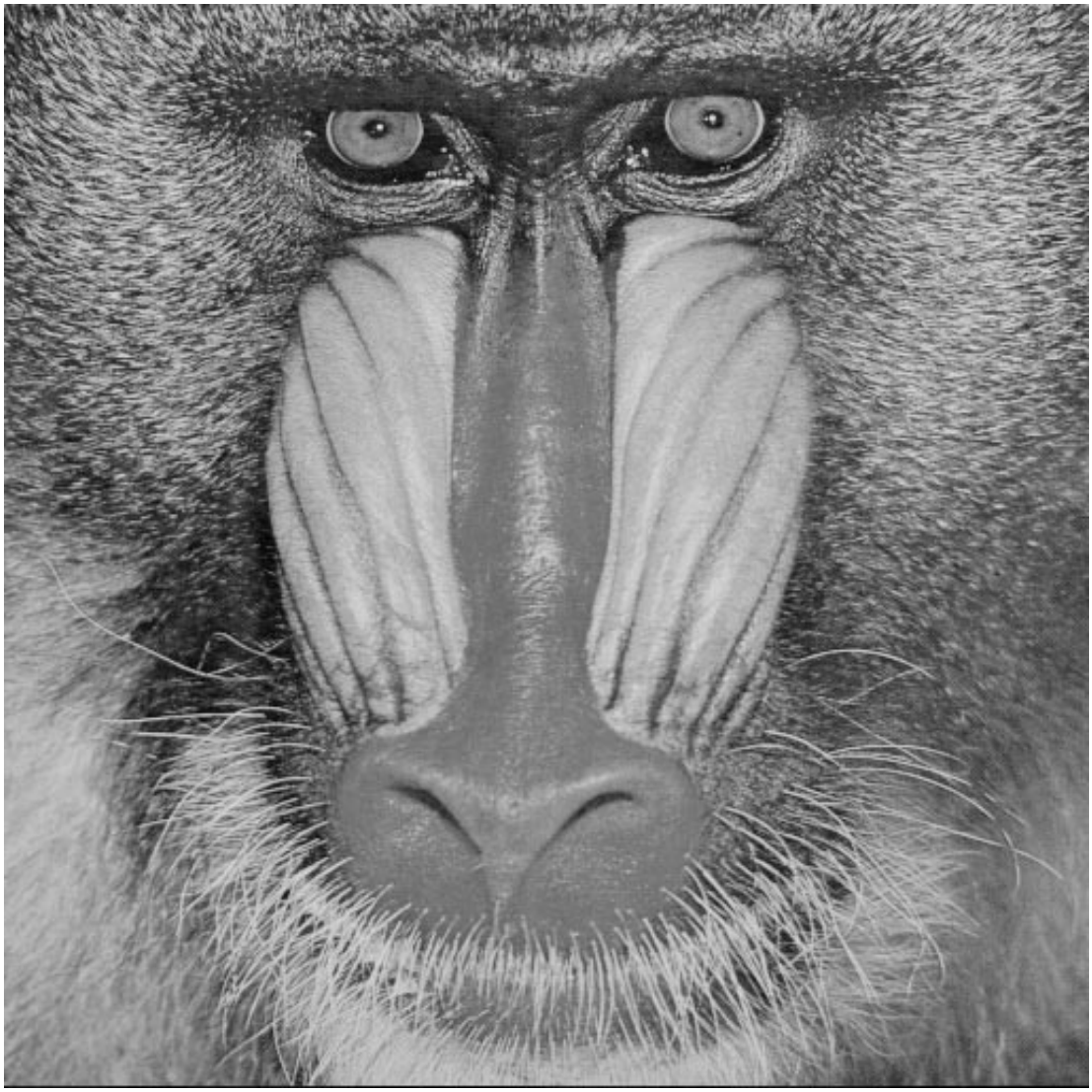


Abbildung A.4: Mandrill

A.5 Floor

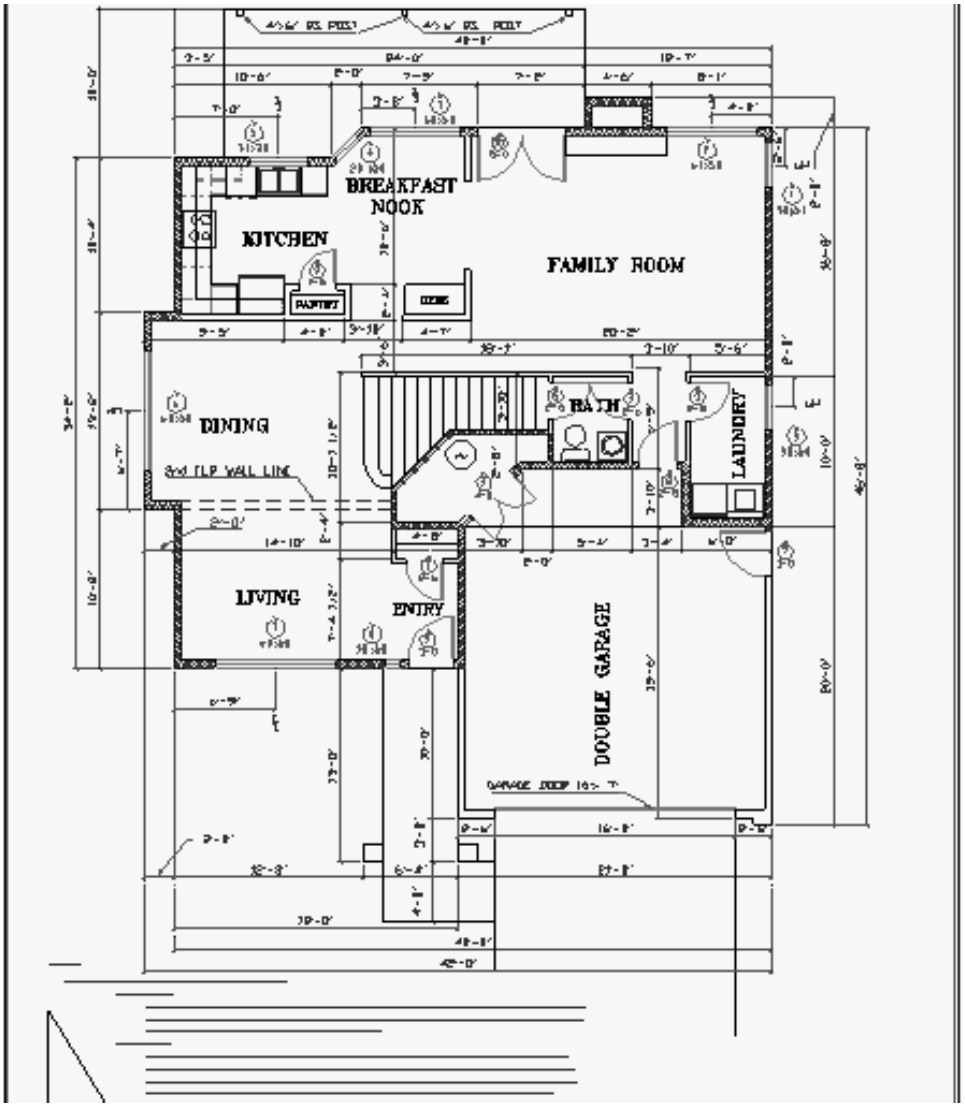


Abbildung A.5: Floor

A.6 Door

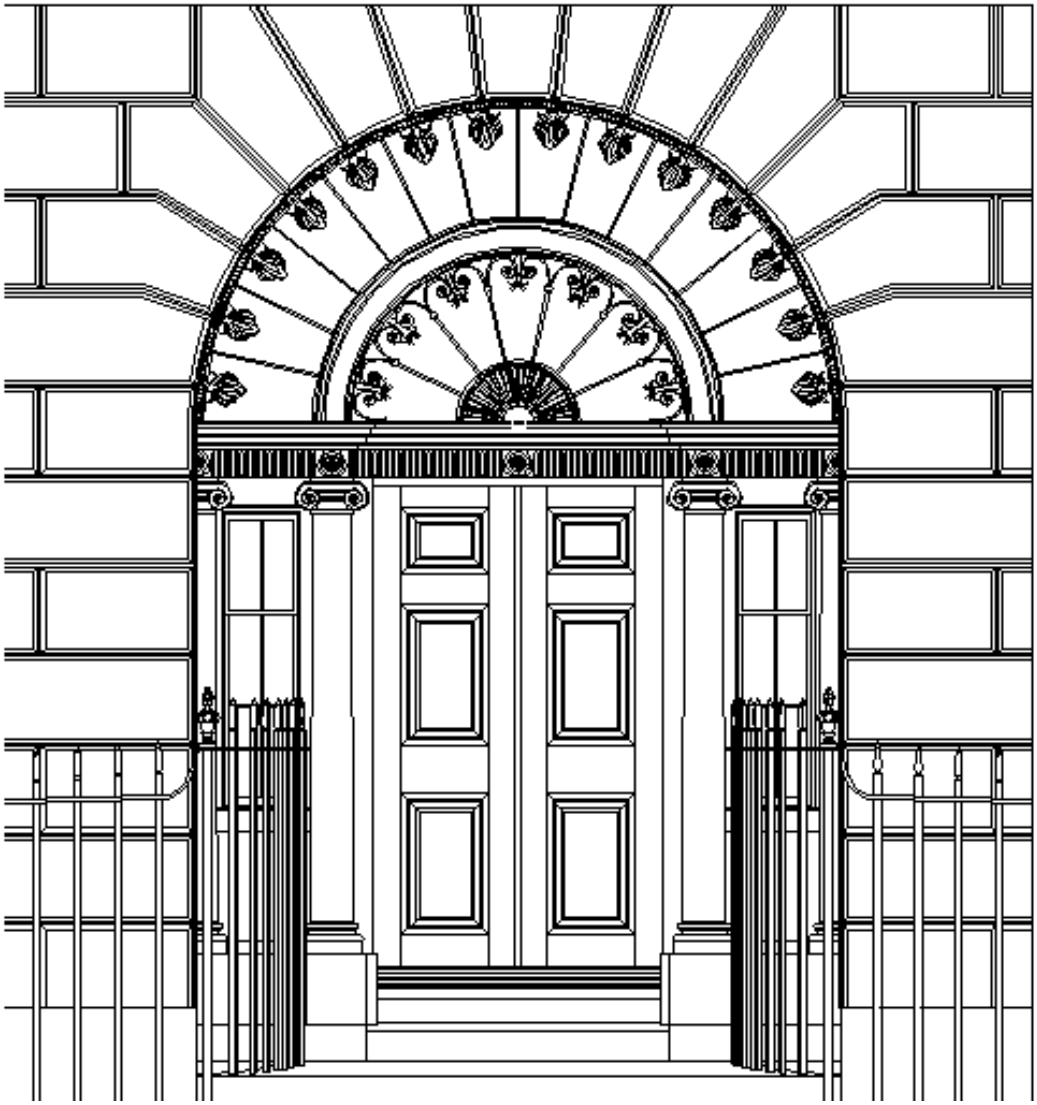


Abbildung A.6: Door

Anhang B

Dateiformat für komprimierte Bilder

Beschrieben ist hier das zum Speichern von komprimierten Bildern verwendete Dateiformat.

Feld Nr.	Typ	Anzahl Byte	Beschreibung
1	BYTE	1	Bildkennung: 0 bei Graustufenbild, 255 für Echtfarbenbild
2	BYTE	1	Typ des verwendeten Basiswavelets: 0 orthogonale Basis, 1 biorthogonale Basis
3	BYTE	1	hier wird das zur Analyse (und bei einer orthogonalen Basis auch zur Synthese) verwendete Wavelet codiert. Die Werte, die dieses Feld annehmen kann, sind in der Tabelle weiter unten beschrieben.
4	BYTE	1	dieses Feld existiert nur, wenn in Feld Nr. 2 als Typ „1“ (biorthogonal) eingetragen ist; es enthält das zur Synthese verwendete Wavelet, wobei die gleiche Codierung benutzt wird wie in Feld Nr. 3
5	BYTE	1	dieses Feld ist nur dann vorhanden, wenn als Analyse-Wavelet in Feld Nr. 3 der Typ „benutzerdefiniert“ enthalten ist. Dann wird hier die Anzahl der Tiefpaßfilterkoeffizienten gespeichert.

Feld Nr.	Typ	Anzahl Byte	Beschreibung
6	double	8*Inhalt von Feld 5	hier werden die Koeffizienten des Tiefpaß-filters für das Analyse-Wavelet abgelegt; nur bei Typ „benutzerdefiniert“
7	BYTE	1	dieses Feld ist nur dann vorhanden, wenn als Synthese-Wavelet in Feld Nr. 4 der Typ „benutzerdefiniert“ enthalten ist. Dann wird hier die Anzahl der Tiefpaßfilterkoeffizienten gespeichert.
8	double	8*Inhalt von Feld 7	hier werden die Koeffizienten des Tiefpaßfilters für das Synthese-Wavelet abgelegt; nur bei Typ „benutzerdefiniert“
9	BYTE	1	Art der Datenanpassung: 0: keine, 1: künstlich periodisch, 2: mit Nullen aufgefüllt, 3: periodisch, 4: reflektiert
10	BYTE	1	Vergrößerungsfaktor für das Bild; wird nur benutzt, wenn in Feld Nr. 9 der Wert „2“, „3“ oder „4“ steht
11	BYTE	1	Anzahl der durchgeführten Iterationsschritte
12	unsigned short	2	Gesamtgröße des Bildes in X-Richtung (in Pixel) nach der Datenanpassung
13	unsigned short	2	Gesamtgröße des Bildes in Y-Richtung (in Pixel) nach der Datenanpassung
14	unsigned short	2	Größe des Originalbildes in X-Richtung (in Pixel)
15	unsigned short	2	Größe des Originalbildes in Y-Richtung (in Pixel)
16	BYTE	1	Position des Bildes bei Datenanpassung: 0: links oben, 1: rechts oben, 2: links unten, 3: rechts unten, 4: Mitte
17	BYTE	1	Art der verwendeten Quantisierung: 1: nur auf ganze Zahlen gerundet, 2: Schwellwert, 3: globale Quantisierung, 4: Quantisierung getrennt nach Bändern
18	BYTE	1	dieses Feld existiert nur, wenn in Feld Nr. 17 der Wert „4“ eingetragen ist. Dann steht hier die Anzahl der Werte, die bei der Quantisierung unter „Anzahl der zur Quantisierung verwendeten Bit pro Band“ angegeben wurden; dies ist erforderlich, da weniger Werte eingegeben werden dürfen als Iterationsschritte durchgeführt werden.

Feld Nr.	Typ	Anzahl Byte	Beschreibung
19	BYTE	1 oder Inhalt von Feld 18	Anzahl der pro Band zur Quantisierung verwendeten Bit (1 Byte, wenn Feld 17 den Wert „3“ hat; hat Feld 17 den Wert „4“, so ergibt sich die Anzahl der Byte aus dem Inhalt von Feld 18)
20	BYTE	1	dieses Feld existiert nur, wenn in Feld Nr. 17 der Wert „4“ eingetragen ist. Dann steht hier die Anzahl der Werte, die bei der Quantisierung unter „Anzahl der zur Lauflängencodierung verwendeten Bit pro Band“ angegeben wurden; auch hier ist es möglich weniger Werte anzugeben als Iterationsschritte durchgeführt werden
21	BYTE	1 oder Inhalt von Feld 20	Anzahl der zur Lauflängencodierung verwendeten Bit pro Band (1 Byte, wenn Feld 17 den Wert „3“ hat; hat Feld 17 den Wert „4“, so ergibt sich die Anzahl der Byte aus dem Inhalt von Feld 20)
22	int	4*Inhalt von Feld 11	enthält für jeden durchgeführten Iterationsschritt den Wert des größten Koeffizienten des jeweiligen Frequenzbandes
23	int	4*Inhalt von Feld 11	enthält für jeden durchgeführten Iterationsschritt den Wert des kleinsten Koeffizienten des jeweiligen Frequenzbandes
24	float	4*Inhalt von Feld 11	enthält für jeden durchgeführten Iterationsschritt die Größe eines Quantisierungsintervalls
25	unsigned long	4*Inhalt von Feld 11	enthält für jeden durchgeführten Iterationsschritt den Wert, mit dem das Nullintervall codiert wurde
26	float	4	enthält den Faktor, um den das um null zentrierte Quantisierungsintervall größer ist als die anderen Intervalle
27	unsigned long	4	Anzahl der ab dem nächsten Byte gespeicherten komprimierten Daten in Byte
28	BYTE	Inhalt von Feld 27	Lauflängencodierte Daten

Feld Nr.	Typ	Anzahl Byte	Beschreibung

Tabelle B.1: Dateiformat

Die folgende Tabelle enthält die Codierung des verwendeten Basiswavelets für die Felder drei und vier der obigen Beschreibung.

Wert	Wavelet
0	Haar
1	Daubechies-4
2	Daubechies-6
3	Daubechies-8
4	Daubechies-10
5	Daubechies-12
6	Daubechies-14
7	Daubechies-16
8	Daubechies-18
9	Daubechies-20
10	Beylkin-18
11	Coifman-6
12	Coifman-12
13	Coifman-18
14	Coifman-24
15	Coifman-30
16	Vaidyanathan-24
255	benutzerdefiniert

Tabelle B.2: Codierung des Basiswavelets

Anhang C

Filterkoeffizienten

In den folgenden Kapiteln dieses Anhangs werden die zur Berechnung der schnellen Wavelet-Transformation nötigen Filterkoeffizienten angegeben. Bis auf das Haar-Wavelet und das Daubechies-Wavelet mit vier Koeffizienten sind hier nur die Tiefpaßfilter angegeben; die Koeffizienten des Hochpaßfilters lassen sich aus denen des Tiefpasses wie in der Arbeit beschrieben berechnen. Bei den meisten Filtern sind keine exakten Werte angegeben, sondern nur Näherungen (wenn auch recht genaue), da die zu berechnenden Ausdrücke sehr komplex werden. Die Komplexität steigt mit zunehmender Koeffizientenzahl der Filter.

Weiterhin ist zu jedem Filter ein Graph des zugehörigen Wavelets und der Skalierungsfunktion abgebildet; diese wurden mit dem Programm zur Wavelet-Bildkompression berechnet, nach MATLAB exportiert und anschließend damit gezeichnet.

Die Werte wurden aus [20] entnommen.

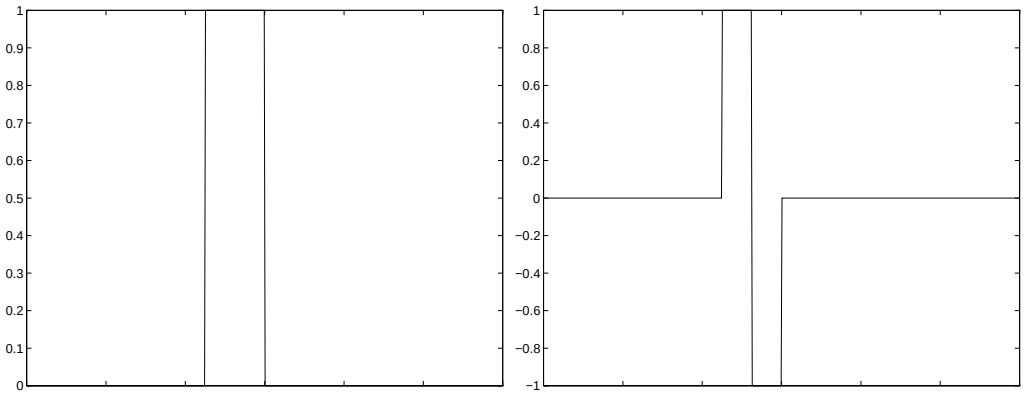
C.1 Orthogonale Wavelets

C.1.1 Haar-Filter

Der Haar-Filter (oder auch Haar-Walsh-Filter) entspricht dem Daubechies-2-Filter ${}_1\psi$. Er hat zwei Koeffizienten, die ungleich null sind, was einem verschwindenden Moment entspricht. Weiterhin ist sehr bemerkenswert, daß das hieraus entstehende Haar-Wavelet als einziges orthogonales Wavelet Symmetrieeigenschaften besitzt. Eine genauere Beschreibung dieses Wavelets befindet sich in Kapitel 2.4.1.

k	Tiefpaß h_k	Hochpaß g_k
0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
1	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$

Tabelle C.1: Haar-Filterkoeffizienten



(a) Skalierungsfunktion (b) Wavelet

Abbildung C.1: Haar-Wavelet und Skalierungsfunktion

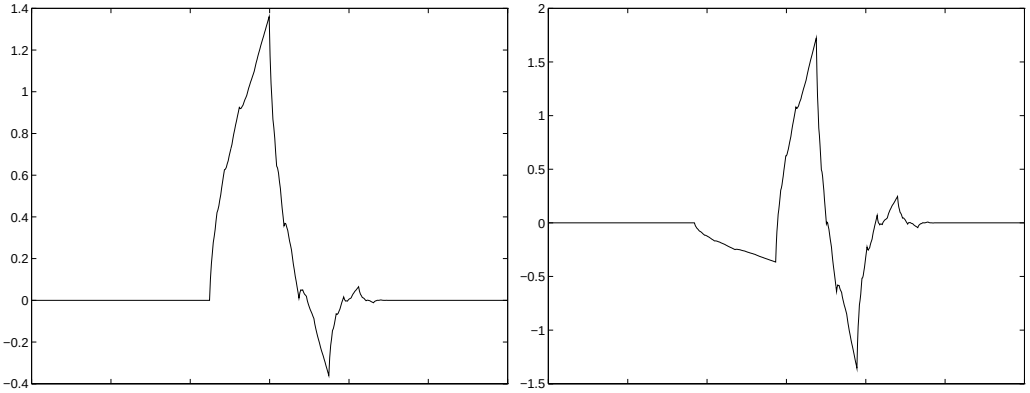
C.1.2 Standard-Daubechies-Filter

Die Daubechies-Filter wurden so entworfen, daß die Skalierungsfunktion für die jeweilige Filtergröße die maximale Glattheit sowie die maximal mögliche Anzahl verschwindender Momente besitzt (diese entspricht der halben Filterlänge). An dieser Stelle fehlen die Daten für den Daubechies-2-Filter, da dieser dem Haar-Filter entspricht (siehe Anhang C.1.1). Weiteres zu den Daubechies-Wavelets findet sich in Kapitel 2.4.3. Für die Bezeichnung der im Folgenden beschriebenen Wavelets gilt, daß hinter dem Namen die Filterlänge angegeben ist. Ein „Daubechies-$2N$-Wavelet“ würde also mit ${}_N\psi$ bezeichnet, da es N verschwindende Momente hat.

C.1.2.1 Daubechies-4

k	Tiefpaß h_k	Hochpaß g_k
0	$\frac{1+\sqrt{3}}{4\sqrt{2}}$	$\frac{1-\sqrt{3}}{4\sqrt{2}}$
1	$\frac{3+\sqrt{3}}{4\sqrt{2}}$	$-\frac{3-\sqrt{3}}{4\sqrt{2}}$
2	$\frac{3-\sqrt{3}}{4\sqrt{2}}$	$\frac{3+\sqrt{3}}{4\sqrt{2}}$
3	$\frac{1-\sqrt{3}}{4\sqrt{2}}$	$-\frac{1+\sqrt{3}}{4\sqrt{2}}$

Tabelle C.2: Daubechies-4-Filterkoeffizienten



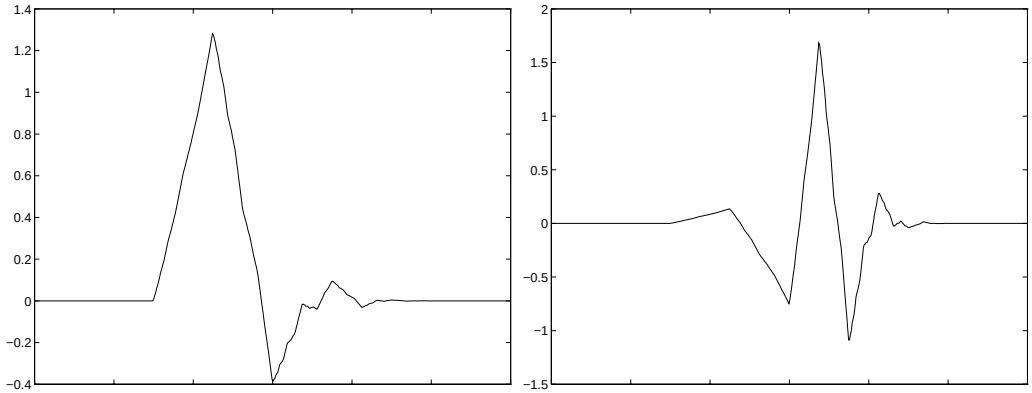
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.2: Daubechies-4-Wavelet und Skalierungsfunktion

C.1.2.2 Daubechies-6

k	Tiefpaß h_k
0	$3.32670552950082630 \cdot 10^{-1}$
1	$8.06891509311092550 \cdot 10^{-1}$
2	$4.59877502118491540 \cdot 10^{-1}$
3	$-1.35011020010254580 \cdot 10^{-1}$
4	$-8.54412738820266580 \cdot 10^{-2}$
5	$3.52262918857095330 \cdot 10^{-2}$

Tabelle C.3: Daubechies-6-Filterkoeffizienten



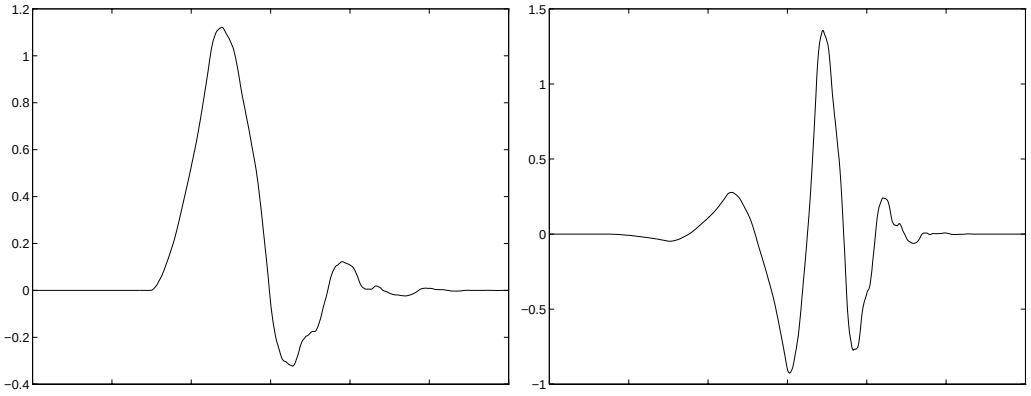
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.3: Daubechies-6-Wavelet und Skalierungsfunktion

C.1.2.3 Daubechies-8

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$2.303778133090 \cdot 10^{-1}$	4	$-1.870348117190 \cdot 10^{-1}$
1	$7.148465705530 \cdot 10^{-1}$	5	$3.084138183600 \cdot 10^{-2}$
2	$6.308807679300 \cdot 10^{-1}$	6	$3.288301166700 \cdot 10^{-2}$
3	$-2.798376941700 \cdot 10^{-2}$	7	$-1.059740178500 \cdot 10^{-2}$

Tabelle C.4: Daubechies-8-Filterkoeffizienten



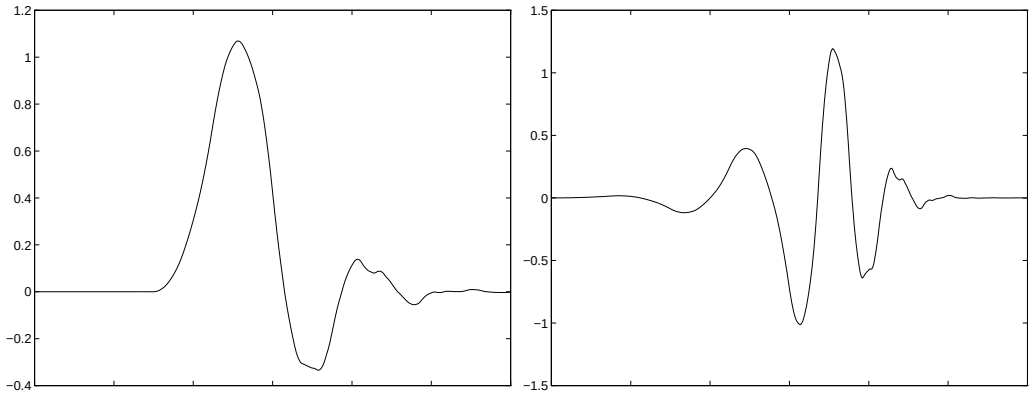
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.4: Daubechies-8-Wavelet und Skalierungsfunktion

C.1.2.4 Daubechies-10

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$1.601023979740 \cdot 10^{-1}$	5	$-3.224486958500 \cdot 10^{-2}$
1	$6.038292697970 \cdot 10^{-1}$	6	$7.757149384000 \cdot 10^{-2}$
2	$7.243085284380 \cdot 10^{-1}$	7	$-6.241490213000 \cdot 10^{-3}$
3	$1.384281459010 \cdot 10^{-1}$	8	$-1.258075199900 \cdot 10^{-2}$
4	$-2.422948870660 \cdot 10^{-1}$	9	$3.335725285000 \cdot 10^{-3}$

Tabelle C.5: Daubechies-10-Filterkoeffizienten



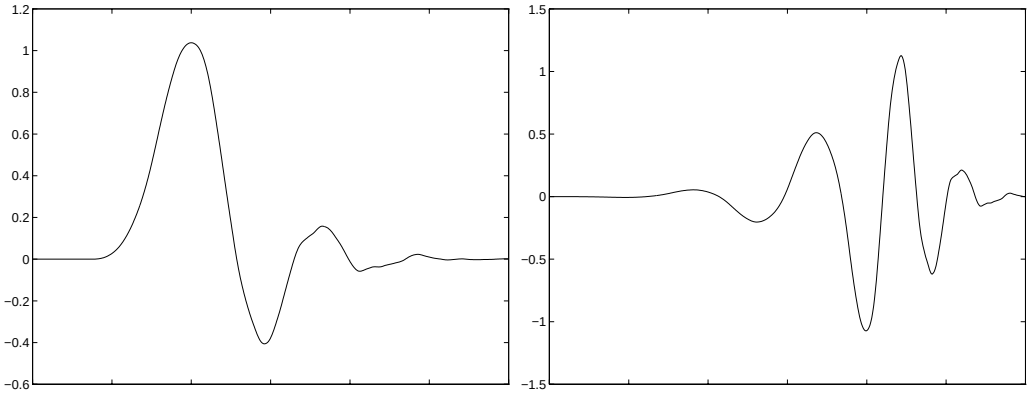
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.5: Daubechies-10-Wavelet und Skalierungsfunktion

C.1.2.5 Daubechies-12

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$1.115407433500 \cdot 10^{-1}$	6	$9.750160558700 \cdot 10^{-2}$
1	$4.946238903980 \cdot 10^{-1}$	7	$2.752286553000 \cdot 10^{-2}$
2	$7.511339080210 \cdot 10^{-1}$	8	$-3.158203931800 \cdot 10^{-2}$
3	$3.152503517090 \cdot 10^{-1}$	9	$5.538422010000 \cdot 10^{-4}$
4	$-2.262646939650 \cdot 10^{-1}$	10	$4.777257511000 \cdot 10^{-3}$
5	$-1.297668675670 \cdot 10^{-1}$	11	$-1.077301085000 \cdot 10^{-3}$

Tabelle C.6: Daubechies-12-Filterkoeffizienten



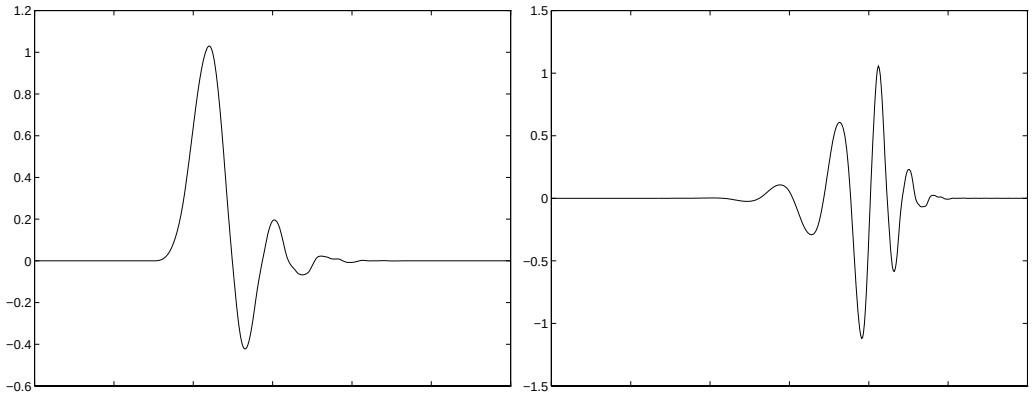
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.6: Daubechies-12-Wavelet und Skalierungsfunktion

C.1.2.6 Daubechies-14

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$7.785205408500 \cdot 10^{-2}$	7	$8.061260915100 \cdot 10^{-2}$
1	$3.965393194820 \cdot 10^{-1}$	8	$-3.802993693500 \cdot 10^{-2}$
2	$7.291320908460 \cdot 10^{-1}$	9	$-1.657454163100 \cdot 10^{-2}$
3	$4.697822874050 \cdot 10^{-1}$	10	$1.255099855600 \cdot 10^{-2}$
4	$-1.439060039290 \cdot 10^{-1}$	11	$4.295779730000 \cdot 10^{-4}$
5	$-2.240361849940 \cdot 10^{-1}$	12	$-1.801640704000 \cdot 10^{-3}$
6	$7.130921926700 \cdot 10^{-2}$	13	$3.537138000000 \cdot 10^{-4}$

Tabelle C.7: Daubechies-14-Filterkoeffizienten



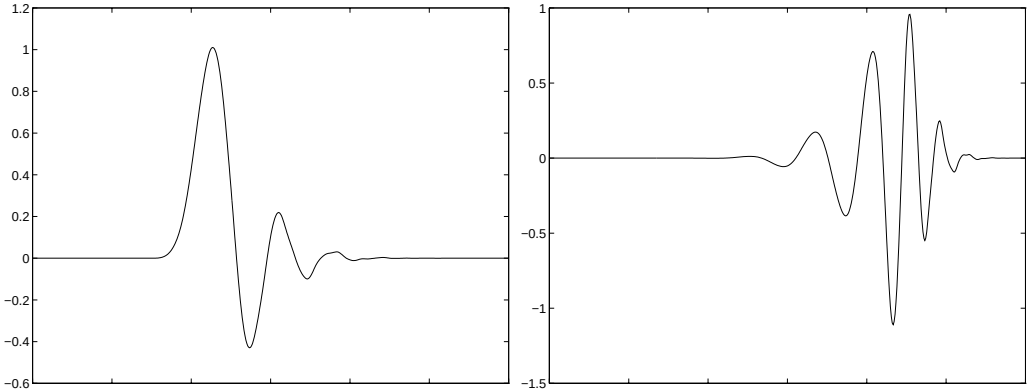
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.7: Daubechies-14-Wavelet und Skalierungsfunktion

C.1.2.7 Daubechies-16

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$5.441584224300 \cdot 10^{-2}$	8	$-1.736930100200 \cdot 10^{-2}$
1	$3.128715909140 \cdot 10^{-1}$	9	$-4.408825393100 \cdot 10^{-2}$
2	$6.756307362970 \cdot 10^{-1}$	10	$1.398102791700 \cdot 10^{-2}$
3	$5.853546836540 \cdot 10^{-1}$	11	$8.746094047000 \cdot 10^{-3}$
4	$-1.582910525600 \cdot 10^{-2}$	12	$-4.870352993000 \cdot 10^{-3}$
5	$-2.840155429620 \cdot 10^{-1}$	13	$-3.917403730000 \cdot 10^{-4}$
6	$4.724845740000 \cdot 10^{-4}$	14	$6.754494060000 \cdot 10^{-4}$
7	$1.287474266200 \cdot 10^{-1}$	15	$-1.174767840000 \cdot 10^{-4}$

Tabelle C.8: Daubechies-16-Filterkoeffizienten



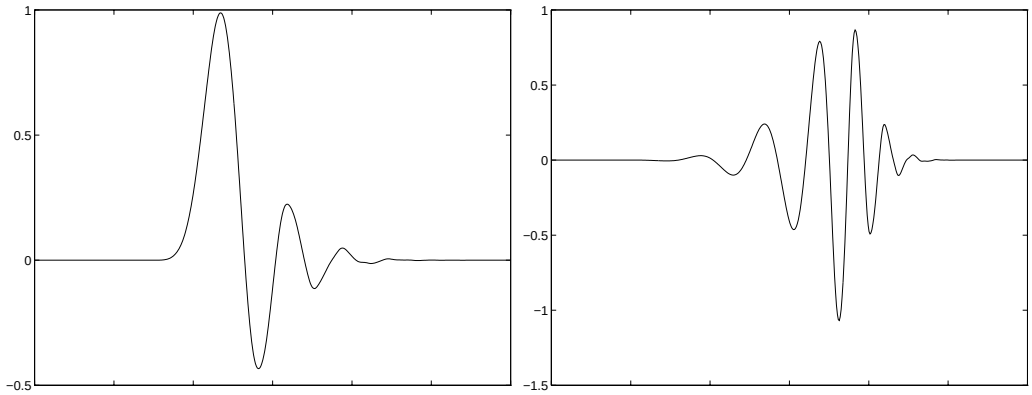
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.8: Daubechies-16-Wavelet und Skalierungsfunktion

C.1.2.8 Daubechies-18

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$3.807794736400 \cdot 10^{-2}$	9	$-6.763282906100 \cdot 10^{-2}$
1	$2.438346746130 \cdot 10^{-1}$	10	$2.509471150000 \cdot 10^{-4}$
2	$6.048231236900 \cdot 10^{-1}$	11	$2.236166212400 \cdot 10^{-2}$
3	$6.572880780510 \cdot 10^{-1}$	12	$-4.723204758000 \cdot 10^{-3}$
4	$1.331973858250 \cdot 10^{-1}$	13	$-4.281503682000 \cdot 10^{-3}$
5	$-2.932737832790 \cdot 10^{-1}$	14	$1.847646883000 \cdot 10^{-3}$
6	$-9.684078322300 \cdot 10^{-2}$	15	$2.303857640000 \cdot 10^{-4}$
7	$1.485407493380 \cdot 10^{-1}$	16	$-2.519631890000 \cdot 10^{-4}$
8	$3.072568147900 \cdot 10^{-2}$	17	$3.934732000000 \cdot 10^{-5}$

Tabelle C.9: Daubechies-18-Filterkoeffizienten



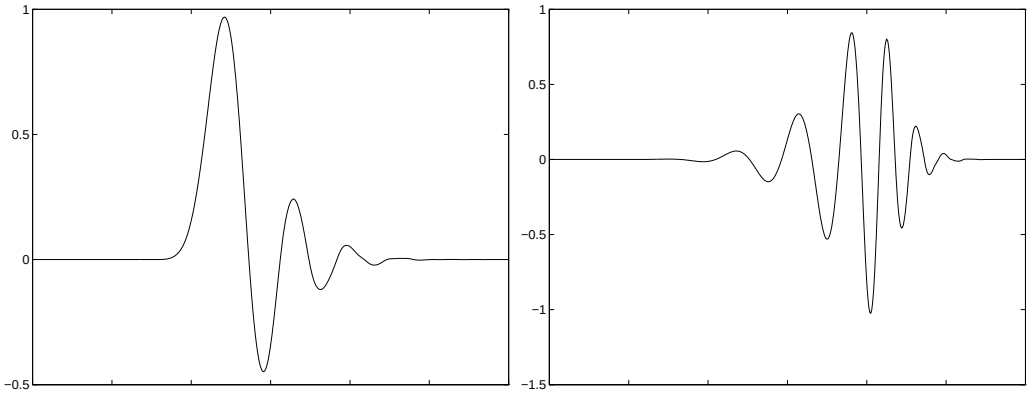
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.9: Daubechies-18-Wavelet und Skalierungsfunktion

C.1.2.9 Daubechies-20

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$2.667005790100 \cdot 10^{-2}$	10	$-2.945753682200 \cdot 10^{-2}$
1	$1.881768000780 \cdot 10^{-1}$	11	$3.321267405900 \cdot 10^{-2}$
2	$5.272011889320 \cdot 10^{-1}$	12	$3.606553567000 \cdot 10^{-3}$
3	$6.884590394540 \cdot 10^{-1}$	13	$-1.073317548300 \cdot 10^{-2}$
4	$2.811723436610 \cdot 10^{-1}$	14	$1.395351747000 \cdot 10^{-3}$
5	$-2.498464243270 \cdot 10^{-1}$	15	$1.992405295000 \cdot 10^{-3}$
6	$-1.959462743770 \cdot 10^{-1}$	16	$-6.858566950000 \cdot 10^{-4}$
7	$1.273693403360 \cdot 10^{-1}$	17	$-1.164668550000 \cdot 10^{-4}$
8	$9.305736460400 \cdot 10^{-2}$	18	$9.358867000000 \cdot 10^{-5}$
9	$-7.139414716600 \cdot 10^{-2}$	19	$-1.326420300000 \cdot 10^{-5}$

Tabelle C.10: Daubechies-20-Filterkoeffizienten



(a) Skalierungsfunktion (b) Wavelet

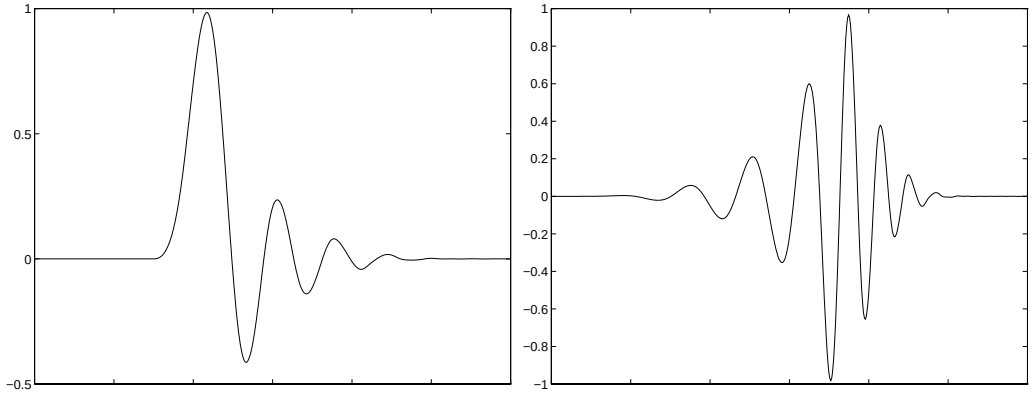
Abbildung C.10: Daubechies-20-Wavelet und Skalierungsfunktion

C.1.3 Beylkin-Filter

Der Beylkin-18-Filter wurde so erstellt, daß die Energie des Leistungsspektrums in einem bestimmten Band konzentriert ist [20].

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$9.93057653743539270 \cdot 10^{-2}$	9	$-8.85436306229248350 \cdot 10^{-2}$
1	$4.24215360812961410 \cdot 10^{-1}$	10	$1.96798660443221200 \cdot 10^{-2}$
2	$6.99825214056600590 \cdot 10^{-1}$	11	$4.29163872741922730 \cdot 10^{-2}$
3	$4.49718251149468670 \cdot 10^{-1}$	12	$-1.74604086960288290 \cdot 10^{-2}$
4	$-1.10927598348234300 \cdot 10^{-1}$	13	$-1.43658079688526110 \cdot 10^{-2}$
5	$-2.64497231446384820 \cdot 10^{-1}$	14	$1.00404118446319900 \cdot 10^{-2}$
6	$2.69003088036903200 \cdot 10^{-2}$	15	$1.48423478247234610 \cdot 10^{-3}$
7	$1.55538731877093800 \cdot 10^{-1}$	16	$-2.73603162625860610 \cdot 10^{-3}$
8	$-1.75207462665296490 \cdot 10^{-2}$	17	$6.40485328521245350 \cdot 10^{-4}$

Tabelle C.11: Beylkin-18-Filterkoeffizienten



(a) Skalierungsfunktion (b) Wavelet

Abbildung C.11: Beylkin-18-Wavelet und Skalierungsfunktion

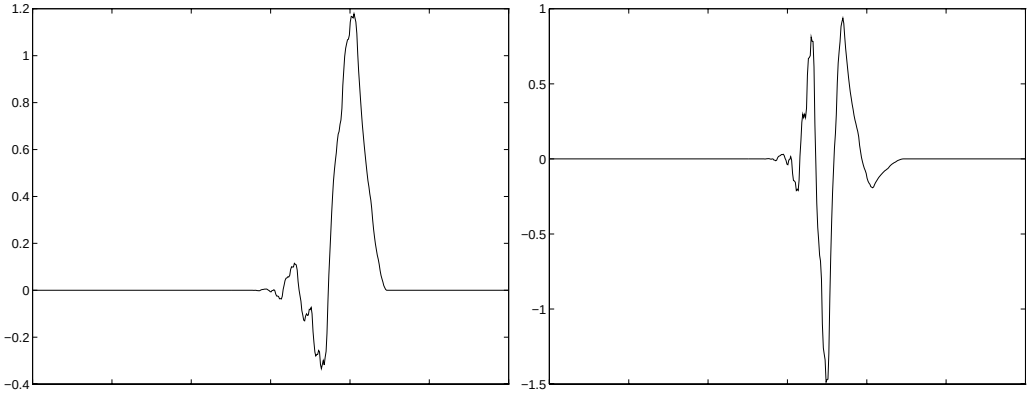
C.1.4 Coifman- oder „Coiflet“-Filter

Diese Wavelets sind in [4] genauer beschrieben. Sie wurden von Daubechies „Coiflets“ getauft, nach ihrem Entdecker Coifman. Dieser forderte nicht nur für das Wavelet verschwindende Momente, sondern gleichzeitig auch für die Skalierungsfunktion. Coiflets sind wesentlich symmetrischer als Daubechies-Wavelets. Aufgrund der verschwindenden Momente der Skalierunsfunktion besitzen sie aber bei gleicher Anzahl verschwindender Momente des Wavelets einen größeren Träger: für Daubechies-Wavelets mit $2N$ verschwindenen Momenten beträgt die Intervallgröße $4N-1$, für ein vergleichbares Coiflet hingegen $6N-1$.

C.1.4.1 Coifman-6

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$\frac{\sqrt{15}-3}{32}\sqrt{2}$	3	$\frac{\sqrt{15}+3}{16}\sqrt{2}$
1	$\frac{1-\sqrt{15}}{32}\sqrt{2}$	4	$\frac{\sqrt{15}+13}{32}\sqrt{2}$
2	$\frac{3-\sqrt{15}}{16}\sqrt{2}$	5	$\frac{9-\sqrt{15}}{32}\sqrt{2}$

Tabelle C.12: Coifman-6-Filterkoeffizienten



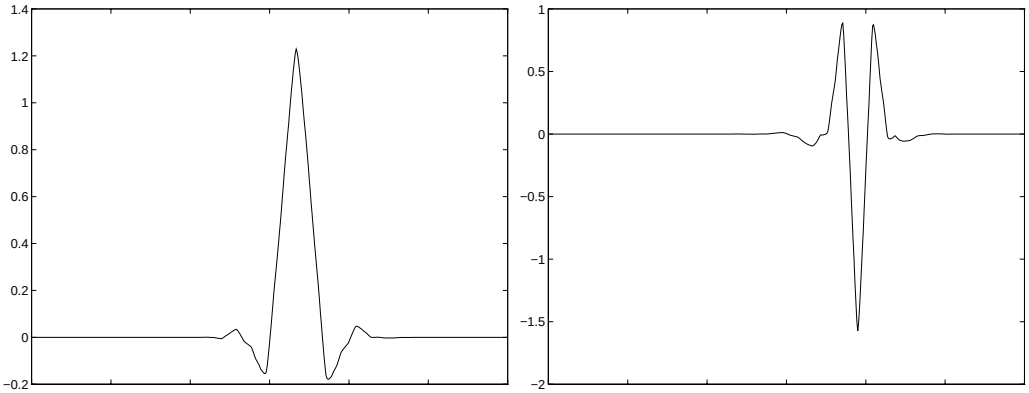
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.12: Coifman-6-Wavelet und Skalierungsfunktion

C.1.4.2 Coifman-12

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$1.63873364631797850 \cdot 10^{-2}$	6	$-7.64885990782645940 \cdot 10^{-2}$
1	$-4.14649367819664850 \cdot 10^{-2}$	7	$-5.94344186464712400 \cdot 10^{-2}$
2	$-6.73725547222998740 \cdot 10^{-2}$	8	$2.36801719468767500 \cdot 10^{-2}$
3	$3.86110066823092900 \cdot 10^{-1}$	9	$5.61143481936598850 \cdot 10^{-3}$
4	$8.12723635449606130 \cdot 10^{-1}$	10	$-1.82320887091009920 \cdot 10^{-3}$
5	$4.17005184423777600 \cdot 10^{-1}$	11	$-7.20549445368115120 \cdot 10^{-4}$

Tabelle C.13: Coifman-12-Filterkoeffizienten



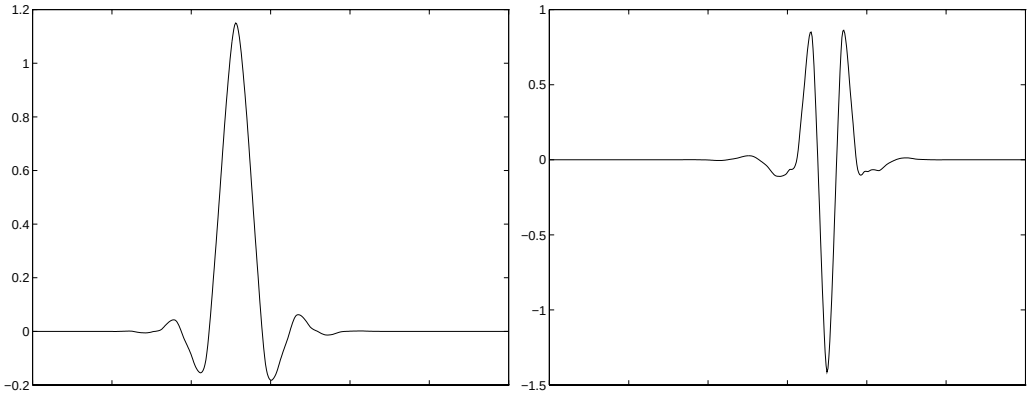
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.13: Coifman-12-Wavelet und Skalierungsfunktion

C.1.4.3 Coifman-18

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$-3.79351286437787590 \cdot 10^{-3}$	9	$-8.23019271063202830 \cdot 10^{-2}$
1	$7.78259642567078690 \cdot 10^{-3}$	10	$3.45550275733444640 \cdot 10^{-2}$
2	$2.34526961421191030 \cdot 10^{-2}$	11	$1.58805448636159010 \cdot 10^{-2}$
3	$-6.57719112814312280 \cdot 10^{-2}$	12	$-9.00797613673228960 \cdot 10^{-3}$
4	$-6.11233900029556980 \cdot 10^{-2}$	13	$-2.57451768812796920 \cdot 10^{-3}$
5	$4.05176902409616790 \cdot 10^{-1}$	14	$1.11751877082696180 \cdot 10^{-3}$
6	$7.93777222625620340 \cdot 10^{-1}$	15	$4.66216959820144030 \cdot 10^{-4}$
7	$4.28483476377618690 \cdot 10^{-1}$	16	$-7.09833025057049280 \cdot 10^{-5}$
8	$-7.17998216191705900 \cdot 10^{-2}$	17	$-3.45997731974026950 \cdot 10^{-4}$

Tabelle C.14: Coifman-18-Filterkoeffizienten



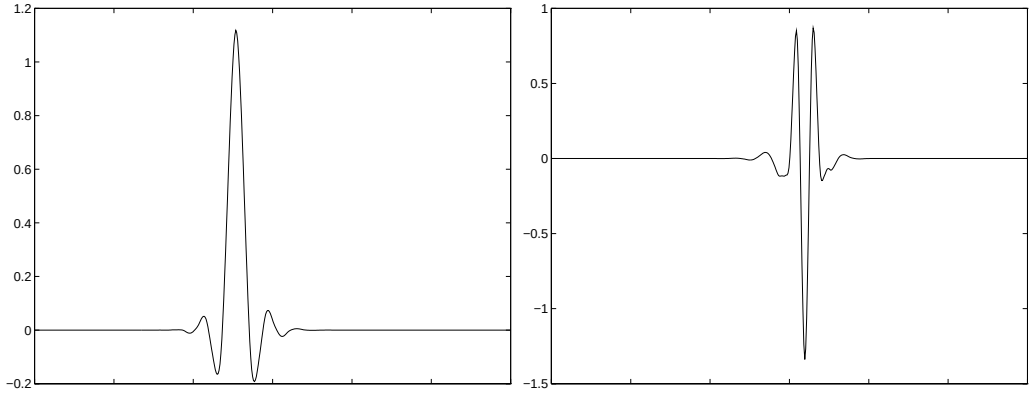
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.14: Coifman-18-Wavelet und Skalierungsfunktion

C.1.4.4 Coifman-24

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$8.92313668220275710 \cdot 10^{-4}$	12	$3.93344271229132190 \cdot 10^{-2}$
1	$-1.62949201311084900 \cdot 10^{-3}$	13	$2.50822618451469330 \cdot 10^{-2}$
2	$-7.34616632765623490 \cdot 10^{-3}$	14	$-1.52117315272391490 \cdot 10^{-2}$
3	$1.60689439640692360 \cdot 10^{-2}$	15	$-5.65828668594603800 \cdot 10^{-3}$
4	$2.66823001556288040 \cdot 10^{-2}$	16	$3.75143615692490270 \cdot 10^{-3}$
5	$-8.12666996803130540 \cdot 10^{-2}$	17	$1.26656192867951870 \cdot 10^{-3}$
6	$-5.60773133164719500 \cdot 10^{-2}$	18	$-5.89020756811437840 \cdot 10^{-4}$
7	$4.15308407030430150 \cdot 10^{-1}$	19	$-2.59974552319421750 \cdot 10^{-4}$
8	$7.82238930920498790 \cdot 10^{-1}$	20	$6.23390338657646180 \cdot 10^{-5}$
9	$4.34386056491468390 \cdot 10^{-1}$	21	$3.12298760780433580 \cdot 10^{-5}$
10	$-6.66274742630007520 \cdot 10^{-2}$	22	$-3.25968044485761290 \cdot 10^{-6}$
11	$-9.62204420335636970 \cdot 10^{-2}$	23	$-1.78498455869993380 \cdot 10^{-6}$

Tabelle C.15: Coifman-24-Filterkoeffizienten



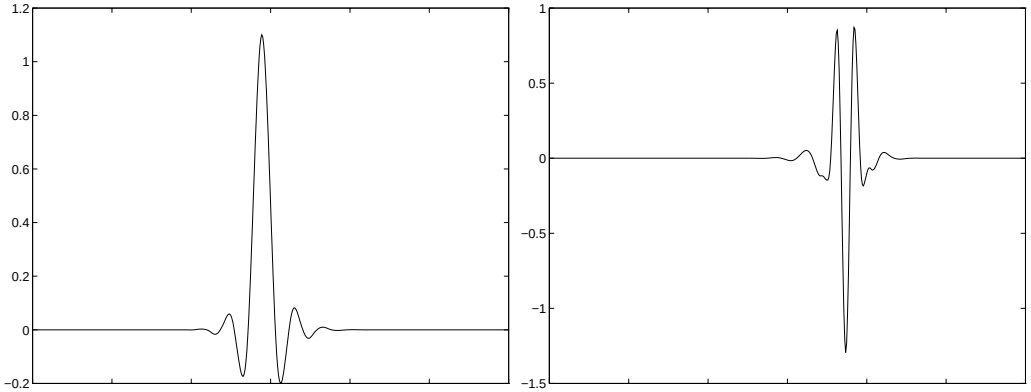
(a) Skalierungsfunktion (b) Wavelet

Abbildung C.15: Coifman-24-Wavelet und Skalierungsfunktion

C.1.4.5 Coifman-30

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$-2.12080863336306810 \cdot 10^{-4}$	15	$3.26835742832495350 \cdot 10^{-2}$
1	$3.58589677255698600 \cdot 10^{-4}$	16	$-1.97617790117239590 \cdot 10^{-2}$
2	$2.17823630484128470 \cdot 10^{-3}$	17	$-9.16423115304622680 \cdot 10^{-3}$
3	$-4.15935878160399350 \cdot 10^{-3}$	18	$6.76418541866332000 \cdot 10^{-3}$
4	$-1.01311175380455940 \cdot 10^{-2}$	19	$2.43337320922405380 \cdot 10^{-3}$
5	$2.34081567615927950 \cdot 10^{-2}$	20	$-1.66286376908581340 \cdot 10^{-3}$
6	$2.81680290621414970 \cdot 10^{-2}$	21	$-6.38131296151377520 \cdot 10^{-4}$
7	$-9.19200105488064130 \cdot 10^{-2}$	22	$3.02259519791840680 \cdot 10^{-4}$
8	$-5.20431632162377390 \cdot 10^{-2}$	23	$1.40541148901077230 \cdot 10^{-4}$
9	$4.21566206728765440 \cdot 10^{-1}$	24	$-4.13404844919568560 \cdot 10^{-5}$
10	$7.74289603740284550 \cdot 10^{-1}$	25	$-2.13150140622449170 \cdot 10^{-5}$
11	$4.37991626228364130 \cdot 10^{-1}$	26	$3.73459674967156050 \cdot 10^{-6}$
12	$-6.20359639056089690 \cdot 10^{-2}$	27	$2.06380639023316330 \cdot 10^{-6}$
13	$-1.05574208705835340 \cdot 10^{-1}$	28	$-1.67408293749300630 \cdot 10^{-7}$
14	$4.12892087407341690 \cdot 10^{-2}$	29	$-9.51579170468293560 \cdot 10^{-8}$

Tabelle C.16: Coifman-30-Filterkoeffizienten



(a) Skalierungsfunktion (b) Wavelet

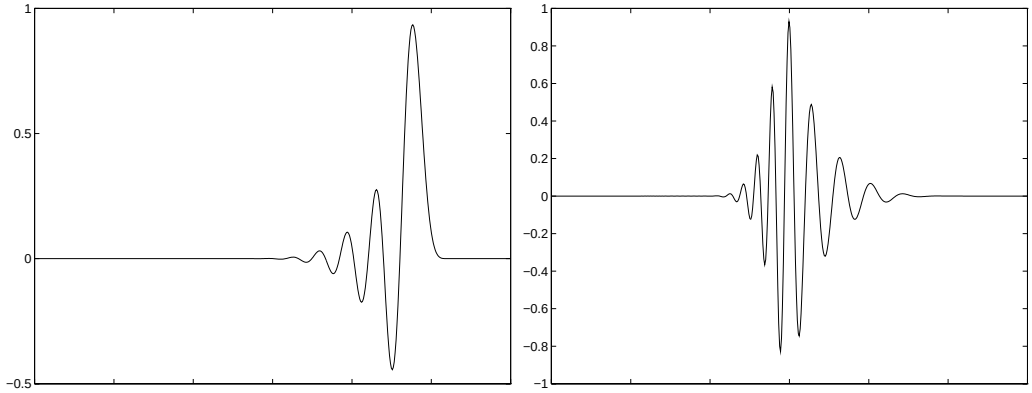
Abbildung C.16: Coifman-30-Wavelet und Skalierungsfunktion

C.1.5 Vaidyanathan-Filter

Beschrieben ist hier der Vaidyanathan-24-Filter, mit dem eine exakte Rekonstruktion möglich ist, obwohl der Filter nicht exakt normiert ist (die Summe der Filterkoeffizienten ist nicht genau $\sqrt{2}$). Dieser Filter wird vor allem für die Codierung von Sprache eingesetzt [20].

k	Tiefpaß h_k	k	Tiefpaß h_k
0	$-6.29061181907475230 \cdot 10^{-5}$	12	$5.58925236913735480 \cdot 10^{-2}$
1	$3.43631904821029190 \cdot 10^{-4}$	13	$-7.77097509019694100 \cdot 10^{-2}$
2	$-4.53956619637219290 \cdot 10^{-4}$	14	$-8.39288843661128300 \cdot 10^{-2}$
3	$-9.44897136321949270 \cdot 10^{-4}$	15	$1.31971661416977720 \cdot 10^{-1}$
4	$2.84383454683556460 \cdot 10^{-3}$	16	$1.35084227129481260 \cdot 10^{-1}$
5	$7.08137504052444710 \cdot 10^{-4}$	17	$-1.94450471766478170 \cdot 10^{-1}$
6	$-8.83910340861387800 \cdot 10^{-3}$	18	$-2.63494802488459910 \cdot 10^{-1}$
7	$3.15384705589700400 \cdot 10^{-3}$	19	$2.01612161775308660 \cdot 10^{-1}$
8	$1.96872150100727140 \cdot 10^{-2}$	20	$6.35601059872214940 \cdot 10^{-1}$
9	$-1.48534480052300990 \cdot 10^{-2}$	21	$5.72797793210734320 \cdot 10^{-1}$
10	$-3.54703986072834530 \cdot 10^{-2}$	22	$2.50184129504662180 \cdot 10^{-1}$
11	$3.87426192934114400 \cdot 10^{-2}$	23	$4.57993341109767180 \cdot 10^{-2}$

Tabelle C.17: Vaidyanathan-24-Filterkoeffizienten



(a) Skalierungsfunktion (b) Wavelet

Abbildung C.17: Vaidyanathan-24-Wavelet und Skalierungsfunktion

Anhang D

Zeichnen einer Funktion mit MATLAB

Um die mit dem Menüpunkt **Datei/MATLAB Export** gespeicherten Daten der Graphen eines Wavelets oder einer Skalierungsfunkltion mit MATALB zeichnen zu können, ist folgender MATLAB-Programmcode nötig:

```
t = linspace(0, 10, <AUFLÖSUNG-X>);
load <NAME>.dat;
plot(t, <NAME>);
set(gca, 'XTickLabels', []);
```

Hierbei muß der Platzhalter <AUFLÖSUNG-X> durch die beim Zeichnen der jeweiligen Funktion verwendete horizontale Auflösung ersetzt werden (Menü **Einstellungen/1D-FWT/Zeichenoptionen**); diese entspricht der Anzahl der gespeicherten Werte.

Für <NAME> wird der Dateiname eingesetzt, unter dem die Werte abgespeichert wurden, jedoch ohne die Dateierweiterung „.dat“.

Für den Ausdruck oder die Weiterverarbeitung der geplotteten Graphen in Textverarbeitungen oder LaTeX empfiehlt sich das Speichern der Bilder im EPS-Format (encapsulated postscript). Diese geschieht mit folgendem Kommando in der MATLAB-Eingabeaufforderung:

```
print -deps <Dateiname>
```

Durch dieses Vorgehen kann eine bestmögliche Qualität erreicht werden. Auch sämtliche in dieser Arbeit gezeigten Graphen von Wavelets und Skalierungsfunktionen wurden auf diese Weise erstellt.

Literaturverzeichnis

[1] G. Beylkin, R. Coifman, V. Rokhlin. *Fast Wavelet Transforms And Numerical Algorithms I.*. Paper 06520. Yale University, New Haven, Connecticut.

[2] Kenneth R. Castleman. *Digital Image Processing*. Prentice-Hall International, Englewood Cliffs, New Jersey, 1996^2

[3] Charles K. Chui. *An Introduction to WAVELETS*, Wavelet Analysis and its Applications, Volume 1. Academic Press, 1995^5

[4] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1995^4

[5] Ronald A. DeVore, Bradley J. Lucier. *Wavelets*. To appear in *Acta Numerica*. Department of Mathematics, University of South Carolina, Columbia, SC 29208. Department of Mathematics, Purdue University, West Lafayette, IN 47907

[6] Gordon Erlebacher, M. Yousuff Hussaini, Leland M. Jameson (Editors). *Wavelets - Theory and Applications*, ICASE/LaRC Series in Computational Science and Engineering. Oxford University Press, New York, 1996

[7] Alain Fournier (Organizer) u.a. *Wavelets and their Applications in Computer Graphics*. SIGGRAPH'95 Conference Course Notes: #26 Wavelets. Department of Computer Science at the University of British Columbia, 1995

[8] Rafael C. Gonzalez, Paul Wintz. *Digital Image Processing*. Addison-Wesley, 1987^2

[9] Richard W. Hamming. *Information und Codierung*. VCH, Weinheim, 1987^2

[10] Michael L. Hilton, Björn D. Jawerth, Ayan Sengupta. *Compressing Still and Moving Images with Wavelets*. To appear in *Multimedia Systems*, Vol. 2, No. 3, April 1994

[11] Tom Hopper, Jonathan N. Bradley, Christopher M. Brislawn, Remigius J. Onyshczak. *The FBI compression standard for digitized fingerprint images*. to appear in *Proc. SPIE*, Vol. 2847, Denver, 1996

[12] Björn Jawerth, Wim Sweldens. *AN OVERVIEW OF WAVELET BASED MULTIRESOLUTION ANALYSES*. Department of Mathematics, University of South Carolina, Columbia SC 29208

[13] Gerald Kaiser. *A Friendly Guide to Wavelets.* Birkhäuser, Boston, 1995^2

[14] David J. Kruglinski. *Inside Visual C++ - Version 4.* Microsoft Press Deutschland, Unterschleißheim, 1996

[15] Alfred K. Louis, Peter Maaß, Andreas Rieder. *Wavelets.* Teubner, Stuttgart, 1994

[16] Jeffrey Richter. *Windows Programmierung für Experten. Der Entwicklerleitfaden zur Win32-API für Windows NT 3.5 und Windows 95.* Microsoft Press Deutschland, Unterschleißheim, 1996

[17] Alan Roberts, Adrian Ford. *Colour Space Conversions FAQ*. Aus: ftp.westminster.ac.uk, 1995

[18] Robert Sedgewick. *Algorithmen in C++*. Addison-Wesley, Bonn, München, Paris, 1992

[19] Martin Vetterli, Jelena Kovačević. *Wavelets and Subband Coding.* Prentice-Hall, Englewood Cliffs, New Jersey, 1995^3

[20] Mladen Victor Wickerhauser. *Adaptive Wavelet-Analysis.* Vieweg, Braunschweig/Wiesbaden, 1996

[21] Mladen Victor Wickerhauser. *High-Resolution Still Picture Compression.* Department of Mathematics, Washington University, St. Louis, Missouri 63130, April 1992

[22] Meinrad Zeller. *Flinkes Wellenspiel, Signalverarbeitung mit Wavelets*, c't - magazin für computertechnik, Heft 11/1994, Seiten 258-264. Verlag Heinz Heise, 1994

Hiermit erkläre ich, daß ich diese Arbeit selbständig verfaßt, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Nürnberg, den 4. April 1997