Wavelets und Filterbänke

Alexander Stoffel
Institut für Nachrichtentechnik
Fakultät für Informations-, Medien- und Elektrotechnik
Fachhochschule Köln

19. Dezember 2010

Inhaltsverzeichnis

Vorbemerkung

Dieses Skript ist als Ergänzung zu meinen Lehrveranstaltungen und zur Unterstützung der Einarbeitung für Diplomanden gedacht. Es bedarf an vielen Stellen der zusätzlichen Erklärung und es enthält vorläufiges Material, das noch aufzuarbeiten ist. Für Kritik und Verbesserungsvorschläge bin ich sehr dankbar. Ein herzliches Dankeschön an alle, die durch aufmerksames und kritisches Lesen schon sehr zur Verbesserung dieses Skriptes beigetragen haben.

Kapitel 1

Einführung

1.1 Wozu sind Wavelets nützlich?

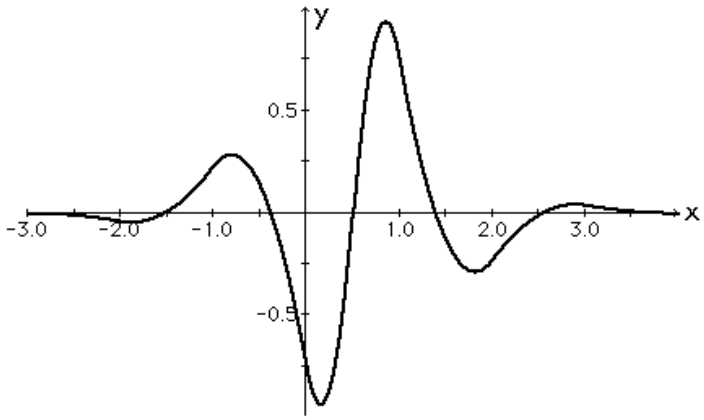


Abbildung 1.1: Beispiel für ein „Mutter“-Wavelet

Wer sich mit Wavelets beschäftigt, wird häufig nach einer möglichst knappen Antwort auf die Frage „Was sind Wavelets?“ gefragt. Eine sehr prägnante Antwort gibt Wim Sweldens in [32]:

„Wavelets sind Bausteine, die schnell Daten dekorrelieren können.“

Auch die Fourier-Transformation dient der Dekorrelation von Daten. Dabei werden als Bausteine der Sinus und Cosinus mit verschiedenen Frequenzen oder die komplexe Exponentialfunktion $f(t) = e^{j\omega t}$ mit beliebigen Werten $\omega \in \mathbb{R}$ benutzt. Diese Funktionen haben den Nachteil, dass sie für große Argumente t nicht „abklingen”, es gilt stets $|e^{j\omega t}| = 1$. Diese Schwierigkeit vermeidet man mit Wavelets.

Man geht von einem festen „Mutter“-Wavelet aus (beispielsweise von der in Abb.1.1 abgebildeten Funktion). Als Mindestvoraussetzung an ein Mutterwavelet verlangt man, daß

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0 \tag{1.1}$$

Man benutzt dann diese Funktion sowie skalierte, d.h. gestauchte oder gestreckte und verschobene Versionen (siehe Abb.1.2), um ein Signal als Summe derartiger Funktionen

darzustellen (so wie man in einer Fourier-Reihe ein periodisches Signal als Summe von Sinus- und Cosinusfunktion darstellt).

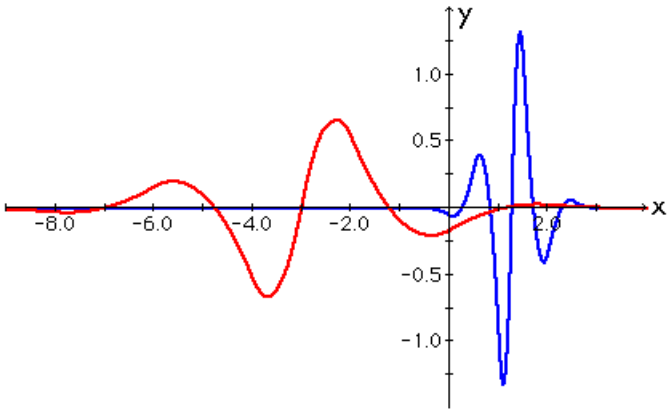


Abbildung 1.2: gestreckte (links) und gestauchte (rechts) sowie verschobene Versionen des „Mutter“-Wavelets aus Abb. 1.1

Dabei kommt es auf die hierzu notwendigen Koeffizienten an. In ihnen steckt oft mehr Information als im ursprünglichen Signal (so wie in der Frequenz eines Tones mehr Information steckt). Sie können also helfen, charakteristische Eigenschaften des Signals zu erkennen. Viele Koeffizienten sind so klein, dass man sie weglassen kann, wenn man kleine Änderungen des ursprünglichen Signals akzeptiert.

Betrachten wir zum Vergleich zunächst die Entwicklung einer periodischen Funktion $f(t)$ mit der Periode T und zugehöriger Kreisfrequenz $\omega = \frac{2\pi}{T}$ in eine Fourier-Reihe

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{jk\omega t} \qquad \text{mit} \qquad c_k = \frac{1}{T} \int_0^T f(t) e^{-jk\omega t} dt$$

Statt der Funktionen $g_k(t) = e^{jk\omega t}$ werden wir, ausgehend von einem geeigneten Mutter-Wavelet $\psi(t)$, hier Funktionen der Form

$$\psi_{m,n}(t) := 2^{\frac{m}{2}} \psi(2^m t - n), \qquad m, n \in \mathbb{Z} \tag{1.2}$$

benutzen. Es hat sich nämlich als sinnvoll herausgestellt, sich auf ganzzahlige Verschiebungen sowie Streckungen oder Stauchungen mit Zweierpotenzen 2^m mit ganzzahligem m zu beschränken. Der Vorfaktor $2^{\frac{m}{2}}$ dient der Normierung. Bei geeignet gewähltem $\psi(t)$ existiert ein „Gegenstück“ dazu, das *duale* Wavelet $\tilde{\psi}(t)$, und es läßt sich jede Funktion $f(t)$, die $\int_{-\infty}^{+\infty} |f(t)|^2 dt < \infty$ erfüllt, als Reihe in der Form

$$f(t) = \sum_{m,n=-\infty}^{+\infty} d_m(n) \psi_{m,n}(t) \tag{1.3}$$

schreiben. Die Koeffizienten $d_m(n)$ ergeben sich aus

$$d_m(n) = \int_{-\infty}^{+\infty} f(t) \tilde{\psi}_{m,n}(t) dt \tag{1.4}$$

wobei $\tilde{\psi}_{m,n}(t)$ in Analogie zu (1.2) verschobene und gestreckte oder gestauchte Versionen des dualen Wavelets $\tilde{\psi}(t)$ sind. Es bleibt anzumerken, dass — unter weitaus schwächeren Voraussetzungen an $\psi(t)$ — eine zu (1.3) analoge Darstellung durch ein Integral existiert. Die Behandlung einer derartigen *kontinuierlichen* Darstellung und der damit verbundenen kontinuierlichen Wavelet-Transformation würde den hier vorliegenden Rahmen bei weitem sprengen. Wir beschränken uns also hier auf die durch (1.3) und (1.4) gegebene *diskrete* Wavelet-Transformation mit den Koeffizienten $d_m(n)$.

Will man die unendliche Reihe (1.3) durch eine Summe mit endlich vielen Termen approximieren, so kommt man mit den Wavelets allein nicht aus. Integriert man eine endliche Summe der Form (1.3), dann ist das Ergebnis stets Null, da Summation und Integration vertauscht werden können und (1.1) als Mindestvoraussetzung an das Wavelet vereinbart wird. Beim Arbeiten mit endlichen Summen (was in der Praxis die Regel ist), ist die Benutzung einer weiteren Funktion, der *Skalierungsfunktion* $\varphi(x)$ unabdingbar, die dann $\int_{-\infty}^{+\infty} \varphi(x)dx = 1$ erfüllt.

Müsste man die Koeffizienten $d_m(n)$ in der Praxis tatsächlich durch Ausrechnen eines Integrals der Form (1.4) ermitteln, so würde dies eine erhebliche Hürde für praktische Anwendungen darstellen, und dann wären die Wavelets kaum über einen kleinen Kreis von Spezialisten bekannt geworden. Bei geschickter Wahl des Wavelets können diese Koeffizienten durch einen *schnellen* Algorithmus mit Hilfe von *Filterbänken* berechnet werden. Die Zahl der Rechenoperationen wächst dabei nur proportional zur Zahl N der Daten. Das ist sogar noch günstiger als bei der Schnellen Fourier-Transformation, bei der die Zahl der Rechenoperationen bei 2^m Daten proportional zu $m \cdot 2^m$ anwächst.

Die Berechnung der Wavelet-Koeffizienten mit Filterbänken macht wesentlich von einer *Näherung* an die Funktion $f(t)$ Gebrauch. Von dieser hat man in der heutigen digitalen Praxis ohnehin nur die Abtastwerte $f(k)$ mit ganzzahligem k zur Verfügung. Bei dieser Schreibweise wurde die Abtastfrequenz durch eine entsprechende Skalierung zu 1 normiert. Die Filterbänke waren im übrigen historisch vor den Wavelets bekannt.

Wavelets als Funktionen, wie sie beispielsweise in Abb. 1.2 gezeigt sind, treten also in der Praxis gar nicht unmittelbar auf. Der Praktiker kümmert sich nur um die direkte Berechnung der Koeffizienten mit Hilfe von Filterbänken — oder durch eine hierzu äquivalente Methode, mit Lifting-Schritten. Daher stehen auch in diesem einleitenden Kapitel die Filterbänke im Vordergrund.

Hier schließlich eine kleine Übersicht über einige Anwendungsbereiche der Wavelet-Transformation:

- Datenkompression (z.B. von Bilddaten)
- Analyse von Daten (Erkennen charakteristischer Eigenschaften, z.B. Hinweise auf Krankheiten in einem EKG-Signal)
- Beseitigung von Rauschen (siehe das Beispiel in Abb. 1.3)

Wesentliche Vorteile gegenüber der Fourier-Analysis:

- gute Lokalisierung der Wavelets im Zeit- und Frequenzbereich (im Gegensatz zum Sinus!)
- große Flexibilität (es gibt eine Vielzahl von Funktionen, die als Wavelets infrage kommen)

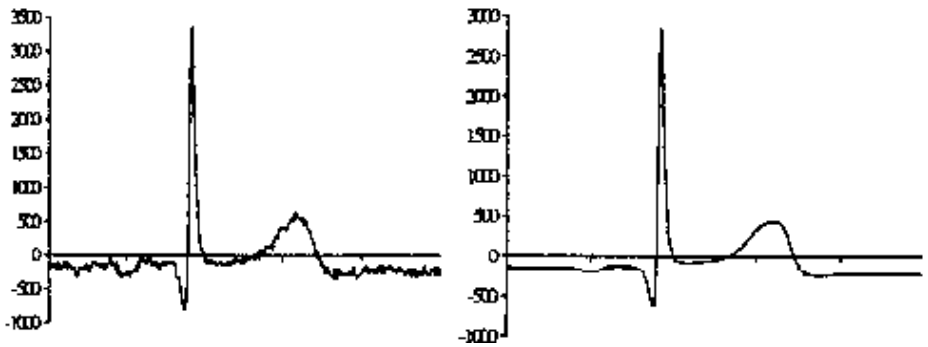


Abbildung 1.3: links verrauschtes EKG-Signal, rechts nach der Bearbeitung (aus der Diplomarbeit von Martin Bandelow)

Literatur: Hervorragend geeignet als allgemeinverständliche Einführung (die erste Hälfte des Buches enthält keine Formeln!) ist [18]. Dieses Buch geht auch ausführlich auf die aufschlußreiche Geschichte der Wavelets ein.

Unter den zahlreichen Lehrbüchern seien hier zum Einstieg und im Hinblick auf praktische Anwendungen vor allem [1, 5, 19, 31] empfohlen. Für eine mathematische Grundlegung wird auf das Standardwerk [10] verwiesen.

Zur einführenden Literatur siehe auch die Literaturhinweise mit den entsprechenden Links in:

`http://alex.nt.fh-koeln.de/wavelet.html`

1.2 Filterbank zum Haar-Wavelet

Wir betrachten hier ein Beispiel, das von Haar zum Beginn des 20. Jahrhunderts untersucht wurde, ohne den Namen „Wavelet“ zu benutzen (der erst Mitte der Achtzigerjahre „erfunden“ wurde). Dieses Beispiel hat Nachteile für praktische Anwendungen, aber entscheidende pädagogische Vorteile, da es ohne komplizierte Rechnungen einen Einblick in die Vorteile der Wavelet-Transformation erlaubt.

Wir gehen hier so vor, wie es in der Praxis üblich ist und betrachten den Algorithmus zur Berechnung der Koeffizienten mit Hilfe einer Filterbank, ohne uns um die zugehörigen Funktionen, also die „Bausteine“ zu kümmern. Diese werden dann in Abschnitt 2.2 vorgestellt. Erst in Abschnitt 2.3 wird aufgezeigt, daß der hier behandelte Algorithmus tatsächlich die Koeffizienten einer Reihenentwicklung berechnet, bei der die Haar-Wavelets als Bausteine auftreten.

Wir betrachten die in Abb. 1.4 gezeigte Filterbank. Was dabei mit dem Eingangssignal $x(k)$ geschieht, wird in Tabelle 1.1 verdeutlicht. Insgesamt wird das Eingangssignal $x(k)$ aufgeteilt in die Mittelwerte

$$s(k) = \frac{1}{2}\Big(x(2k) + x(2k+1)\Big) \tag{1.5}$$

und die Differenzen

$$d(k) = x(2k) - x(2k+1) \tag{1.6}$$

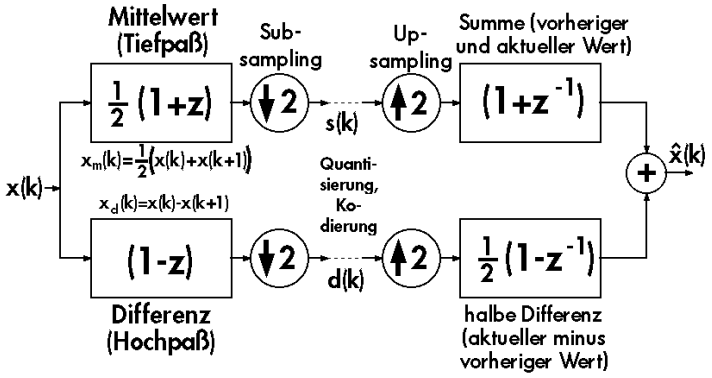


Abbildung 1.4: einfaches Beispiel einer Filterbank

Eingang	Ausgang Hoch/Tiefpass	zu quant.	Eing. Rek.-F.	Ausgang Rekonstr.-Filter
$x(-2)$	$\frac{1}{2}x(-2)+\frac{1}{2}x(-1)$	$s(-1)$	$s(-1)$	$s(-1)=\frac{1}{2}x(-2)+\frac{1}{2}x(-1)$
$x(-1)$	$\frac{1}{2}x(-1)+\frac{1}{2}x(0)$	weg	0	$s(-1)=\frac{1}{2}x(-2)+\frac{1}{2}x(-1)$
$x(0)$	$\frac{1}{2}x(0)+\frac{1}{2}x(1)$	$s(0)$	$s(0)$	$s(0)=\frac{1}{2}x(0)+\frac{1}{2}x(1)$
$x(1)$	$\frac{1}{2}x(1)+\frac{1}{2}x(2)$	weg	0	$s(0)=\frac{1}{2}x(0)+\frac{1}{2}x(1)$
$x(2)$	$\frac{1}{2}x(2)+\frac{1}{2}x(3)$	$s(1)$	$s(1)$	$s(1)=\frac{1}{2}x(2)+\frac{1}{2}x(3)$
$x(3)$	$\frac{1}{2}x(3)+\frac{1}{2}x(4)$	weg	0	$s(1)=\frac{1}{2}x(2)+\frac{1}{2}x(3)$
$x(-2)$	$x(-2)-x(-1)$	$d(-1)$	$d(-1)$	$\frac{1}{2}d(-1)=\frac{1}{2}x(-2)-\frac{1}{2}x(-1)$
$x(-1)$	$x(-1)-x(0)$	weg	0	$-\frac{1}{2}d(-1)=-\frac{1}{2}x(-2)+\frac{1}{2}x(-1)$
$x(0)$	$x(0)-x(1)$	$d(0)$	$d(0)$	$\frac{1}{2}d(0)=\frac{1}{2}x(0)-\frac{1}{2}x(1)$
$x(1)$	$x(1)-x(2)$	weg	0	$-\frac{1}{2}d(0)=-\frac{1}{2}x(0)+\frac{1}{2}x(1)$
$x(2)$	$x(2)-x(3)$	$d(1)$	$d(1)$	$\frac{1}{2}d(1)=\frac{1}{2}x(2)-\frac{1}{2}x(3)$
$x(3)$	$x(3)-x(4)$	weg	0	$-\frac{1}{2}d(1)=-\frac{1}{2}x(2)+\frac{1}{2}x(3)$

Tabelle 1.1: zur Erläuterung der Filterbank von Abb. 1.4

In diesen Formeln ist durch den Faktor 2 bereits berücksichtigt, dass durch das „Subsampling“ (Unterabtastung) um den Faktor 2 nur jeder 2. Mittelwert und jede 2. Differenz beibehalten wird (die weggelassenen Werte sind in der Tabelle in der Spalte „zu quantisieren“ durch „weg“ gekennzeichnet). Das „Upsampling“ fügt stattdessen wieder Nullen zwischen die Werte ein, der Ausgang des „Upsampling“ stimmt mit dem Eingang des Rekonstruktionsfilters überein (siehe die entsprechende Spalte in der Tabelle). Für die Rekonstruktion ergibt sich damit insgesamt

$$x(k)=\begin{cases} s(\frac{k}{2})+\frac{1}{2}d(\frac{k}{2}) & \text{falls } k \text{ gerade} \\ \\ s(\frac{k-1}{2})-\frac{1}{2}d(\frac{k-1}{2}) & \text{falls } k \text{ ungerade} \end{cases} \tag{1.7}$$

Diese Transformation ist durch ihre Anwendung auf ein Testsignal in Abb. 1.5 illustriert. Als Testsignal wurde die 500. Zeile des Testbildes „Lena“ gewählt (vollständiges Bild siehe Abb. 1.6). Die Mittelwerte $s(k)$ (links unten) ergeben ein Signal, das sich nur qulitativ nur wenig vom Eingangssignal unterscheidet, allerdings enthalten sie nur halb so viele Abtastwerte. An einigen Stellen (insbesondere um den 150. Signalwert) ist erkennbar,

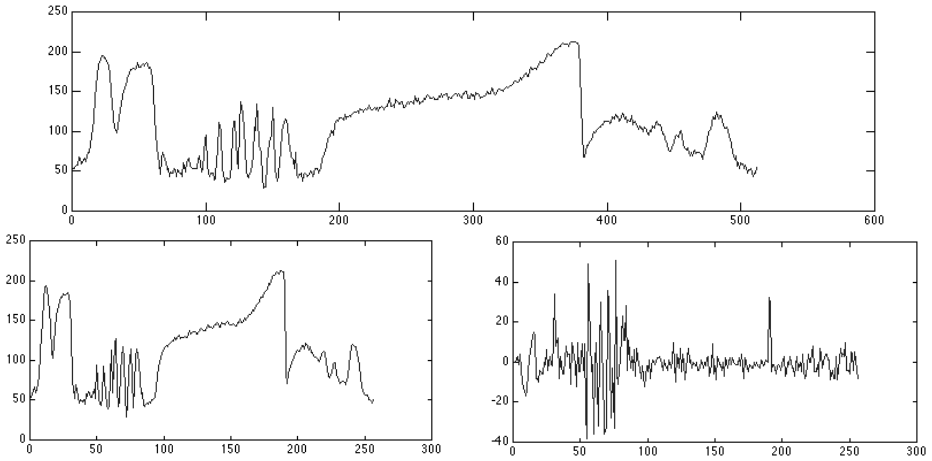


Abbildung 1.5: **Oben:** Ursprüngliches Signal (500. Zeile des Testbilds Lena", 512 Pixel, vollständiges Bild siehe Abb. 1.6). **Links unten:** Mittelwerte $s(k)$ nach Unterabtastung (siehe (1.5)). **Rechts unten:** Differenzen $d(k)$ nach Unterabtastung (siehe (1.6)).

dass das ursprüngliche Signal durch die Mittelwertbildung etwas geglättet wurde. Die Differenzen $d(k)$ (rechts unten) haben dort besonders große Werte, wo sich das ursprünglich Signal stark ändert, insbesondere, wo es Sprünge macht. Dies ist im Bild bei den Kanten der Fall. Ansonsten nehmen Absolutbeträge der Differenzen zahlenmäßig, verglichen mit dem Eingangssignal, nur sehr kleine Werte an.

Die durch Mittelwertbildung und Differenz gegebene Transformation ist zunächst nur für eindimensionale Daten anwendbar. Man kann sie jedoch auf ein digitalisiertes Bild anwendenden, indem man die Bilddaten zunächst zeilenweise transformiert. Dies ist in Abb. 1.6 für ein sehr gebräuchliches Testbild gezeigt. Links ist das ursprüngliche Bild, rechts sind die Bilddaten *zeilenweise* transformiert. Dieses Zwischenergebnis wird anschließend *spaltenweise* derselben Transformation unterzogen. Dies liefert das in Abb. 1.7 gezeigte *transformierte* Bild. Dabei wurden die transformierten Werte wieder so quantisiert, dass Graustufen zwischen 0 und 255 herauskamen. Bei den Differenzen wurde der Kontrast zur besseren Darstellung wieder verstärkt (sonst würde nur ein einheitliches Grau entstehen).

Der praktische Nutzen dieser Transformation für die Datenkompression ist deutlich sichtbar: Die Differenzen enthalten überwiegend sehr kleine Zahlen, die man durch Null ersetzen kann. Lauflängen- und Huffman-Kodierung führen dann zur einer beträchtlichen Reduktion des benötigten Speicherplatzes (siehe Abschnitt 6.4).

Die hier besprochene, so einfache Transformation ist allerdings nur zur Erklärung des Prinzips der Wavelet-Transformation und ihrer Anwendungen hervorragend geeignet. Ihre Nachteile, die einer praktischen Anwendung entgegenstehen, werden in Abschnitt 2.1 deutlich werden.

Häufig wird die hier besprochene Transformation mit anderen Vorfaktoren, d.h. mit einer anderen Normierung angewandt. Durch folgende Abänderung von (1.5) und (1.6)

$$s(k) = \frac{1}{\sqrt{2}}\big(x(2k) + x(2k+1)\big) \tag{1.8}$$

$$d(k) = \frac{1}{\sqrt{2}}\big(x(2k) - x(2k+1)\big) \tag{1.9}$$

Abbildung 1.6: **Links:** gebräuchliches Testbild „Lena“, hier 512×512 Pixel. **Rechts:** zeilenweise transformiertes Testbild

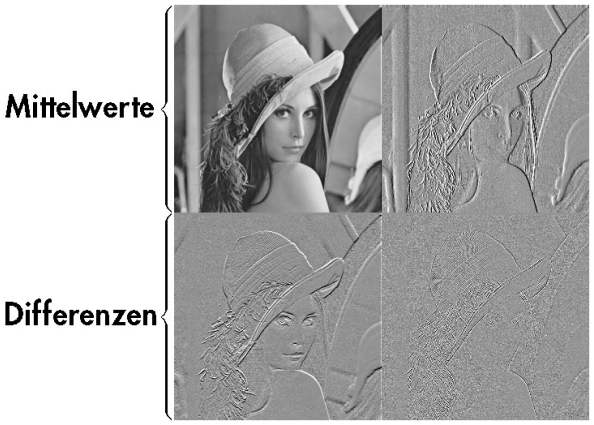


Abbildung 1.7: erst zeilenweise, dann spaltenweise transformiertes Testbild

errecht man eine symmetrischere Verteilung der Faktoren.

1.3 Beschreibung von Filtern

1.3.1 Vorbemerkung

In diesem Abschnitt sollen einige mathematische Techniken besprochen werden, die für eine Verallgemeinerung des im Abschnitt 1.2 besprochenen Beispiels sinnvoll sind. Er dient der Vorbereitung auf die Überlegungen des Abschnitts 1.5.

Als „Filter“ wird eine Vorrichtung oder — mathematisch gesehen — eine Abbildung bezeichnet, die jedem Eingangssignal ein eindeutig bestimmtes Ausgangssignal zuordnet, wenn dabei folgende Eigenschaften erfüllt sind:

- Ein gegenüber dem ursprünglichen Signal verschobenes Eingangssignal soll zu einem entsprechend verschobenen Ausgangssignal führen („Zeitinvarianz“).

- Das Ausgangssignal soll *linear* vom Eingangssignal abhängen, d.h. die Summe zweier Eingangssignale führt zur Summe der entsprechenden Ausgangssignale und das Vielfache eines Eingangssignals zum entsprechenden Vielfachen des Ausgangssignals.

Wir betrachten hier nur diskrete Signale, die in der Praxis meist als Folge $x(k)$ der *Abtastwerte* eines kontinuierlichen Signals $g(t)$ entstehen:

$$x(k) = g(k\Delta t), \quad \Delta t \text{ Abtastzeit, also } \frac{1}{\Delta t} \text{ die Abtastfrequenz, } k \in \mathbb{Z} \tag{1.10}$$

Hier bezeichnet $\mathbb{Z}$ die Menge der ganzen Zahlen. Folge dieser Schreibweise (Δt ist weggelassen bzw. $\Delta t = 1$): Nach dem *Abtasttheorem* ist die höchste Frequenz des Signals $f_{\max} = \frac{1}{2}$ und beim Übergang zur Kreisfrequenz (die in der Mathematik gebräuchliche und meist mit ω bezeichnete Größe)

$$\omega_{\max} = 2\pi f_{\max} = \pi \tag{1.11}$$

1.3.2 Beschreibung durch die Impulsantwort

Hier bleibt man im Zeit- oder Ortsbereich. Das Ausgangssignal erhält man als Faltung mit der Impulsantwort:

$$y(n) = \sum_{k=-\infty}^{+\infty} f(k)x(n-k) = \sum_{k=-\infty}^{+\infty} f(n-k)x(k) \tag{1.12}$$

Dabei ist $x(n)$ das *Eingangssignal* und $y(n)$ das *Ausgangssignal* sowie $f(n)$ die *Impulsantwort*, d.h. der Ausgang des Filters, wenn als Eingangssignal der Einheitsimpuls

$$x(n) = \begin{cases} 1 & \text{falls } n = 0 \\ 0 & \text{falls } n \neq 0 \end{cases} \tag{1.13}$$

vorliegt. Wir werden hier fast ausschließlich FIR-Filter (Finite Impulse Response) betrachten. Bei diesen ist $f(k) \neq 0$ nur für *endlich* viele $k \in \mathbb{Z}$, so dass die Summe (1.12) in Wirklichkeit nur aus *endlich* vielen Summanden besteht.
Beispiel: Impulsantwort für das Tiefpassfilter zum Haar-Wavelet:

$$f(k) = h_0(k) = \begin{cases} \frac{1}{2} & \text{falls } k = 0 \text{ oder } k = -1 \\ 0 & \text{sonst} \end{cases} \tag{1.14}$$

und hierfür erhält man als Ausgang bei einem beliebigen Eingangssignal $x(n)$

$$y(n) = \frac{1}{2}x(n+1) + \frac{1}{2}x(n) \tag{1.15}$$

1.3.3 Beschreibung durch Matrizen

Die durch (1.12) gegebene Zuordnungsvorschrift stellt eine lineare Abbildung dar und kann folglich durch eine Matrix $\mathbf{A}$ mit Matrixelementen a_{ik} beschrieben werden in der Form:

$$\mathbf{y} = \mathbf{A}\mathbf{x} \qquad \text{bzw.} \qquad y_n = \sum_k a_{nk}x_k \tag{1.16}$$

Diese Matrix ist hier allerdings unendlich ausgedehnt. Hierzu hat man sich die Folgen des abgetasteteten Eingangs- und Ausgangssignals als Spaltenvektoren $\mathbf{x}$ und $\mathbf{y}$ mit unendlich vielen Komponenten ($x_n = x(n)$) vorzustellen. Ein Vergleich von (1.12) und (1.16) liefert

$$a_{mn} = f(m-n) \tag{1.17}$$

Die Spalten der Matrix bestehen also aus der Impulsantwort, die nach rechts fortschreitend jeweils um eine Position nach unten verschoben ist:

$$\mathbf{A} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & f(1) & f(0) & f(-1) & f(-2) & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & f(2) & f(1) & f(0) & f(-1) & f(-2) & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & f(2) & f(1) & f(0) & f(-1) & f(-2) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & f(2) & f(1) & f(0) & f(-1) & f(-2) & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & f(2) & f(1) & f(0) & f(-1) & f(-2) & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & f(2) & f(1) & f(0) & f(-1) & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \tag{1.18}$$

In den Zeilen steht ebenfalls die Impulsantwort, allerdings in umgekehrter Reihenfolge, ebenso jeweils um eine Position versetzt. Dadurch sind die Diagonalen dieser Matrix konstant. Im Beispiel des Tiefpassfilters zum Haar-Wavelet erhalten wir

$$\mathbf{A} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots & \cdots \\ \cdots & \cdots & \cdots & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0 & \frac{1}{2} & \frac{1}{2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \tag{1.19}$$

mit einer für FIR-Filter typischen Bandstruktur.

1.3.4 Beschreibung durch die z-Transformierte

Wir beschränken uns hier auf Signale $x(n)$, die nur für endlich viele $n \in \mathbb{Z}$ $x(n) \neq 0$ erfüllen. Die z-Transformierte des Signals ist dann durch

$$X(z) := \sum_{k=-\infty}^{+\infty} x(k) z^{-k} \tag{1.20}$$

definiert. Analog definiert man die z-Transformierte $F(z)$ der Impulsantwort $f(k)$ sowie die z-Transformierte $Y(z)$ des Ausgangssignals. Vereinbarungsgemäß sind von der formal unendlichen Summe nur endlich viele von Null verschieden.

Die z-Transformierte ist hier also stets eine Funktion von z der Form

$$P(z) = \sum_{k=n_1}^{n_2} a_k z^k \qquad \text{mit} \quad n_1, n_2 \in \mathbb{Z} \tag{1.21}$$

die an ein Polynom erinnert. Allerdings können hier auch negative Exponenten auftreten. Solche Funktionen heißen LAURENT-*Polynome*. Durch Ausklammern können sie stets als

Faktor aus einer Potenz z^k mit $k \in \mathbb{Z}$ und einem „normalen“ Polynom dargestellt werden. Der *Grad* eines LAURENT-Polynoms der Form (1.21) ist $n_2 - n_1$ (vorausgesetzt $a_{n_1} \neq 0$ und $a_{n_2} \neq 0$). Man beachte, dass ein Polynom der Form $P(z) = c \cdot z^p$ mit $p \in \mathbb{Z}$ und $c \neq 0$ stets den Grad 0 hat. Alle Polynome vom Grad Null haben diese Gestalt. Ferner ist zu beachten, dass der *Kehrwert* eines LAURENT-Polynoms

$$\frac{1}{P(z)}$$

im allgemeinen *kein* LAURENT-Polynom ist, es sei denn, $P(z)$ hat den Grad Null.

Die z-Transformierte eines FIR-Filters ist also stets ein LAURENT-Polynom. Wenn hier im Zusammenhang mit z-Transformierten von Polynomen die Rede ist, so sind damit stets LAURENT-Polynome gemeint, auf den ausdrücklichen Zusatz „LAURENT“ wird hier also häufig verzichtet.

Einige nützliche Regeln für die z-Transformierte (leicht nachzurechnen):

- $X(z^{-1})$ ist die z-Transformierte des zeitumgekehrten Signals $x(-k)$
- $X(-z)$ ist die z-Transformierte des Signals $(-1)^k x(k)$

Beispiele:

(a) Tiefpassfilter zum Haar-Wavelet

$$F(z) = H_0(z) = h_0(-1)z^1 + h_0(0)z^0 = \frac{1}{2}z + \frac{1}{2} \tag{1.22}$$

(b) zugehöriges Hochpassfilter

$$\begin{aligned} h_1(k) &= \begin{cases} 1 & \text{falls } k = 0 \\ -1 & \text{falls } k = -1 \\ 0 & \text{sonst} \end{cases} \\ F(z) &= H_1(z) = h_1(-1)z^1 + h_1(0)z^0 = 1 - z \end{aligned} \tag{1.23}$$

(c) Verschiebung:

$$Y(z) = z \cdot X(z) \qquad \Longleftrightarrow \qquad y(k) = x(k+1) \tag{1.24}$$

Multiplikation mit z bewirkt also eine Verschiebung der Folge, Multiplikation mit z^{-1} verschiebt in die andere Richtung:

$$Y(z) = z^{-1} \cdot X(z) \qquad \Longleftrightarrow \qquad y(k) = x(k-1) \tag{1.25}$$

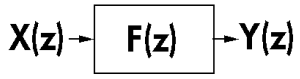
und für beliebiges $m \in \mathbb{Z}$

$$Y(z) = z^m \cdot X(z) \qquad \Longleftrightarrow \qquad y(k) = x(k+m) \tag{1.26}$$

Ein großer Vorteil der z-Transformierten und wohl der Hauptgrund für ihre Benutzung ist die einfache Beschreibung des Zusammenhangs zwischen Eingangssignal $X(z)$ und Ausgangssignal $Y(z)$ bei Filtern: Die z-Transformierte des **Ausgangssignals** erhält man durch

$$Y(z) = F(z) \cdot X(z) \tag{1.27}$$

bildlich dargestellt:



Es genügt also, $F(z)$ anzugeben. Im Beispiel $F(z) = H_0(z)$ gilt

$$Y(z) = \left(\frac{1}{2}z + \frac{1}{2}\right) \cdot X(z) \tag{1.28}$$

Durch ein einfaches Ausmultiplizieren kann man sich überzeugen, dass der einfache Zusammenhang (1.27) tatsächlich mit der kompliziert aussehenden Faltungsvorschrift von (1.12) übereinstimmt. Dabei sind die Summen statt der abstrakten Schreibweise mit $\sum$ durch eine intuitivere mit „Pünktchen“ ersetzt.

$$\begin{aligned} X(z) &= \ldots + x(-2)z^2 + x(-1)z + x(0) + x(1)z^{-1} + x(2)z^{-2} + \ldots \\ F(z) &= \ldots + f(-2)z^2 + f(-1)z + f(0) + f(1)z^{-1} + f(2)z^{-2} + \ldots \end{aligned}$$

Für das Produkt $F(z) \cdot X(z)$ erhält man

$$\begin{aligned} & \Big(\ldots + f(-2)z^2 + f(-1)z + f(0) + f(1)z^{-1} + f(2)z^{-2} + \ldots\Big) \\ & \quad \cdot\Big(\ldots + x(-2)z^2 + x(-1)z + x(0) + x(1)z^{-1} + x(2)z^{-2} + \ldots\Big) \\ = \;& f(0)x(0) + f(-1)x(1) + f(1)x(-1) + f(-2)x(2) + f(2)x(-2) + \ldots \\ & + \Big(f(0)x(1) + f(1)x(0) + f(-1)x(2) + f(2)x(-1) + f(-2)x(3) + \ldots\Big)z^{-1} \\ & + \Big(f(0)x(-1) + f(1)x(-2) + f(-1)x(0) + f(2)x(-3) + f(-2)x(1) + \ldots\Big)z \\ & + \Big(f(0)x(2) + f(1)x(1) + f(-1)x(3) + f(2)x(0) + f(-2)x(4) + \ldots\Big)z^{-2} \\ & + \Big(f(0)x(-2) + f(1)x(-3) + f(-1)x(-1) + f(2)x(-4) + f(-2)x(1) + \ldots\Big)z^2 \\ & + \ldots \\ = \;& Y(z) = \ldots + y(-2)z^2 + y(-1)z + y(0) + y(1)z^{-1} + y(2)z^{-2} + \ldots \end{aligned}$$

Koeffizientenvergleich liefert tatsächlich die Übereinstimmung mit (1.12).

1.3.5 Frequenzverhalten des Filters

Setzt man in der z-Transformierten $z = e^{j\omega}$, so erhält man die FOURIER-Transformierte

$$X(e^{j\omega}) = \sum_{k=-\infty}^{+\infty} x(k)e^{-jk\omega} \tag{1.29}$$

analog für $Y(e^{j\omega})$ und $F(e^{j\omega})$, also ist das Frequenzverhalten des Ausgangs eines Filters durch eine einfache Multiplikation gegeben:

$$Y(e^{j\omega}) = F(e^{j\omega}) \cdot X(e^{j\omega}) \tag{1.30}$$

Beispiel: (Haar-Filterbank)
$F(e^{j\omega}) = H_0(e^{j\omega})$ (Tiefpass) und $F(e^{j\omega}) = H_1(e^{j\omega})$ (Hochpass)

$$\begin{aligned} H_0(e^{j\omega}) &= \frac{1}{2}(1+e^{j\omega}) = \frac{1}{2}e^{j\frac{\omega}{2}}(e^{-j\frac{\omega}{2}} + e^{j\frac{\omega}{2}}) \\ &= e^{j\frac{\omega}{2}}\cos\left(\frac{\omega}{2}\right) \qquad (1.31) \\ H_1(e^{j\omega}) &= 1 - e^{j\omega} = e^{j\frac{\omega}{2}}(e^{-j\frac{\omega}{2}} - e^{j\frac{\omega}{2}}) \\ &= -2je^{j\frac{\omega}{2}}\sin\left(\frac{\omega}{2}\right) \qquad (1.32) \end{aligned}$$

Die Absolutbeträge des jeweiligen Frequenzgangs sind in Abb. 1.8 und Abb. 1.9 gezeigt.

$|H_0(e^{j\omega})|$

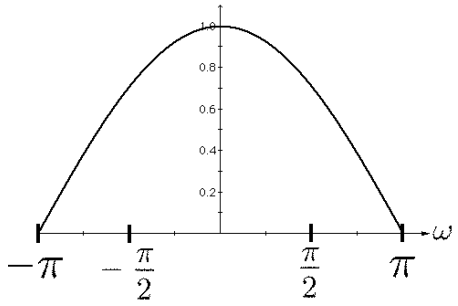


Abbildung 1.8: Absolutbetrag des Frequenzgangs für das Tiefpassfilter zum Haar-Wavelet (siehe (1.31))

$|H_1(e^{j\omega})|$

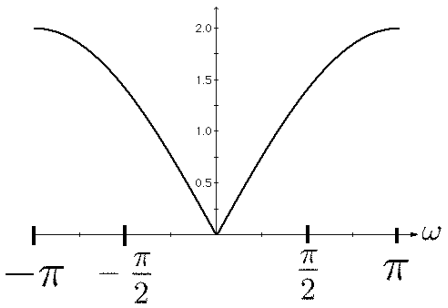


Abbildung 1.9: Absolutbetrag des Frequenzgangs für das Hochpass filter zum Haar-Wavelet (siehe (1.32))

1.4 Beschreibung von Subsampling und Upsampling

Auch dieser Abschnitt dient der Vorbereitung auf die Überlegungen des Abschnitts 1.5.

Die nützliche Eigenschaft von Filtern, dass Verschieben des Eingangssignals zu einem entsprechend verschobenen Ausgangssignal führt, ist am einfachsten an der z-Transformierten sichtbar (siehe hierzu (1.27)). Eine Verschiebung des Eingangssignals in der Zeit

um l Positionen (also um $l \cdot \Delta t$) in die Zukunft entspricht einer Multiplikation mit z^{-l}, und ein solcher Faktor kann mit dem Faktor $F(z)$ vertauscht werden:

$$F(z) \cdot z^{-l} \cdot X(z) = z^{-l} \cdot F(z) \cdot X(z) \tag{1.33}$$

Diese Eigenschaft geht beim Subsampling (Unterabtastung) verloren: Das Eingangssignal

$$x(k) = \begin{cases} 1 & \text{falls } k \text{ gerade} \\ 0 & \text{falls } k \text{ ungerade} \end{cases} \tag{1.34}$$

liefert nach Subsampling die konstante Folge 1, das um 1 verschobene Eingangssignal dagegen die konstante Folge 0. Daher muss die Beschreibung von Sub- und Upsampling im z- und Frequenzbereich etwas komplizierter sein.

Beim *Subsampling* wird nur jeder zweite Abtastwert behalten, d.h. die Abtastwerte mit einer geraden „Nummer“ bleiben übrig. Aus dem Eingangssignal $x(k)$ wird das Ausgangssignal

$$y(k) = (\downarrow 2)x(k) = x(2k) \tag{1.35}$$

Das Eingangssignal mit der z-Transformierten

$$\cdots + x_{-2}z^2 + x_{-1}z + x_0 + x_1 z^{-1} + x_2 z^{-2} + \cdots$$

geht über in das Signal mit der z-Transformierten

$$\cdots + x_{-2}z + x_0 + x_2 z^{-1} + \cdots$$

Bildlich wird Subsampling (Unterabtastung) durch

formelmäßig durch $Y(z) = (\downarrow 2)X(z)$ beschrieben, die z-Transformierte des Ausgangs ist durch

$$Y(z) = (\downarrow 2)X(z) = \frac{1}{2}\big(X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})\big) \tag{1.36}$$

gegeben.

Beispiel: Eingangssignal:

$$x_{-2} = 1; \quad x_{-1} = 4, \quad x_0 = 6, \quad x_1 = 4, \quad x_2 = 1$$

und $x_k = 0$ für $|k| > 2$, also

$$\begin{aligned} X(z) &= z^2 + 4z + 6 + 4z^{-1} + z^{-2} \\ X(z^{\frac{1}{2}}) &= z + 4z^{\frac{1}{2}} + 6 + 4z^{-\frac{1}{2}} + z^{-1} \\ X(-z^{\frac{1}{2}}) &= z - 4z^{\frac{1}{2}} + 6 - 4z^{-\frac{1}{2}} + z^{-1} \\ Y(z) &= z \qquad + 6 \qquad + z^{-1} \end{aligned}$$

Im Frequenzbereich hat man für das Subsampling (Unterabtastung)

$$Y(e^{j\omega}) = \frac{1}{2}\big(X(e^{j\frac{\omega}{2}}) + X(-e^{j\frac{\omega}{2}})\big) = \frac{1}{2}\big(X(e^{j\frac{\omega}{2}}) + X(e^{j(\frac{\omega}{2}+\pi)})\big) \tag{1.37}$$

Es tritt also eine Halbierung der Frequenzen und ein um π verschobener „Aliasing“-Term auf.

Eine Anmerkung ist an dieser Stelle zur Bezeichnungsweise zu machen. Statt des englischen Wortes „Subsampling“ wird hier im weiteren in der Regel das deutsche Wort „Unterabtastung“ verwendet, das den Sachverhalt genauso treffend beschreibt. Außerdem ist hier stets der eigentlich notwendige Hinweis auf den Faktor 2 weggelassen, da hier keine Unterabtastung um einen andern Faktor behandelt wird — außer im Abschnitt 4.2, wo aber stets explizit der dort als Beispiel behandelte Faktor 4 angegeben ist. Im Abschnitt 3.6.1 wird eine andere kompliziertere Unterabtastung behandelt, die mit dem Namen „Schachbrett-Abtastung“ bezeichnet ist, so dass keine Verwechslungen möglich sind.

Beim *Upsampling* wird zwischen jedem zweiten Signalwert eine Null eingefügt. Aus dem Eingangssignal $x(k)$ wird das Ausgangssignal

$$y(k) = (\uparrow 2)x(k) = \begin{cases} x(\frac{k}{2}) & \text{falls } k \text{ gerade} \\ 0 & \text{falls } k \text{ ungerade} \end{cases} \tag{1.38}$$

Aus dem Eingangssignal mit der z-Transformierten

$$\cdots x_{-2}z^2 + x_{-1}z + x_0 + x_1z^{-1} + x_2z^{-2} + \cdots$$

wird also das Ausgangssignal

$$\cdots + x_{-2}z^4 + x_{-1}z^2 + x_0 + x_1z^{-2} + x_2z^{-4} + \cdots$$

Bildlich wird das Upsampling durch

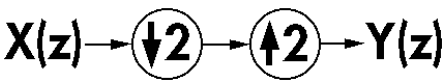
formelmäßig durch $Y(z) = (\uparrow 2)X(z)$ beschrieben, aus dem Eingangssignal $X(z)$ wird das Ausgangssignal

$$Y(z) = (\uparrow 2)X(z) = X(z^2) \tag{1.39}$$

und für den Frequenzbereich wird daraus

$$Y(e^{j\omega}) = X(e^{j\cdot 2\omega}) \tag{1.40}$$

Als kleine Anwendung betrachten wir Subsampling gefolgt von Upsampling



formelmäßig $Y(z) = (\uparrow 2)(\downarrow 2)X(z)$. Dies tritt bei Zwei-Kanal-Filterbänken auf. Zunächst haben wir $U(z) = (\downarrow 2)X(z) = \frac{1}{2}\big(X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})\big)$, danach $Y(z) = (\uparrow 2)U(z) = U(z^2)$. Insgesamt erhalten wir

$$Y(z) = \frac{1}{2}\big(X(z) + X(-z)\big) \tag{1.41}$$

und im Frequenzbereich

$$Y(e^{j\omega}) = \frac{1}{2}\big(X(e^{j\omega}) + X(e^{j(\omega+\pi)})\big) \tag{1.42}$$

1.5 Zwei-Kanal-Filterbänke und perfekte Rekonstruktion

Hier wenden wir die in den vorangegangenen Abschnitten behandelten Techniken an, um das Beispiel der Haar-Filterbank zu verallgemeinern. Wir betrachten also eine Filterbank wie im Beispiel der Abb. 1.4, lassen jedoch allgemeinere Filter zu. Dies ist in Abb. 1.10 dargestellt. Mit den Ergebnissen von Abschnitt 1.3 und 1.4 erhalten wir für die beiden

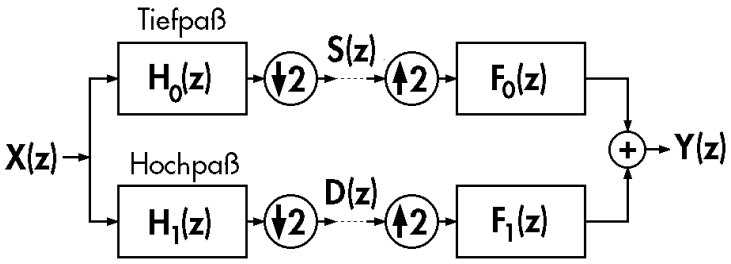


Abbildung 1.10: Zwei-Kanal-Filterbank

Kanäle nach der Unterabtastung

$$S(z) = \frac{1}{2}\Big(H_0(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + H_0(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}})\Big) \tag{1.43}$$

$$D(z) = \frac{1}{2}\Big(H_1(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + H_1(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}})\Big) \tag{1.44}$$

und für den Ausgang nach der Rekonstruktion (als Übungsaufgabe nachrechnen!)

$$\begin{aligned} Y(z) &= \frac{1}{2}\Big(\big(H_0(z)F_0(z) + H_1(z)F_1(z)\big)X(z) \\ &\quad + \frac{1}{2}\Big(H_0(-z)F_0(z) + H_1(-z)F_1(z)\Big)X(-z) \end{aligned} \tag{1.45}$$

Wir wollen erreichen, dass das Ausgangssignal mit dem Eingangssignal übereinstimmt, d.h. $Y(z) = X(z)$ (falls keine Veränderung zur Datenkompression oder zum Beseitigen von Rauschen vorgenommen wurde). Wir könnten auch zulassen, daß $Y(z) = z^{-m}X(z)$, d.h. dass das Ausgangssignal gegenüber dem Eingang verschoben ist, aber eine derartige Verschiebung kann man immer durch einen entsprechenden Faktor in beiden Rekonstruktionsfiltern $F_0(z)$ und $F_1(z)$ kompensieren. Wir erhalten also die folgenden Bedingungen für die *perfekte Rekonstruktion*

$$H_0(-z)F_0(z) + H_1(-z)F_1(z) = 0 \tag{1.46}$$

$$H_0(z)F_0(z) + H_1(z)F_1(z) = 2 \tag{1.47}$$

Gleichung (1.46) wird als „Alias-Auslöschung“ bezeichnet, da die Terme mit $-z$ im Frequenzbereich Termen mit $\omega + \pi$ entsprechen. Gleichung (1.47) drückt aus, dass keine Verzerrung stattfindet.

Bisher haben wir als Lösung dieser beiden Bedingungen lediglich die Filter des Haar-Wavelets kennengelernt, die durch (1.22), (1.23) und

$$F_0(z) = 1 + z^{-1}, \qquad F_1(z) = \frac{1}{2}(1 - z^{-1}) \tag{1.48}$$

k	$4\sqrt{2} \cdot f_0(k)$	$4\sqrt{2} \cdot f_1(k)$	k	$4\sqrt{2} \cdot h_0(k)$	$4\sqrt{2} \cdot h_1(k)$
0	$1+\sqrt{3}$	$1-\sqrt{3}$	0	$1+\sqrt{3}$	$1-\sqrt{3}$
1	$3+\sqrt{3}$	$\sqrt{3}-3$	-1	$3+\sqrt{3}$	$\sqrt{3}-3$
2	$3-\sqrt{3}$	$3+\sqrt{3}$	-2	$3-\sqrt{3}$	$3+\sqrt{3}$
3	$1-\sqrt{3}$	$-1-\sqrt{3}$	-3	$1-\sqrt{3}$	$-1-\sqrt{3}$

Tabelle 1.2: Filterkonstanten „$D4$“ („D“ steht für die Entdeckerin Ingrid Daubechies, „4“ für die Länge)

gegeben sind. Ein etwas komplizierteres Beispiel ist in Tabelle 1.2 gegeben. Durch eine recht mühsame Rechnung (oder mit einem Computer-Algebra-Programm) kann man sich überzeugen, daß diese vier Filter tatsächlich die Bedingungen der perfekten Rekonstruktion erfüllen. Dabei ist zu bemerken, dass bei diesem Beispiel die Filter zur Rekonstruktion aus denen zur Analyse durch Umkehr der Reihenfolge entstehen. Weiteres zu Zwei-Kanal-Filterbänken wird in Abschnitt 2.5 besprochen.

1.6 Transformation von Bilddaten

Im Abschnitt 1.5 wurde die Transformation eindimensionaler Signale mit einer Filterbank besprochen. Für zweidimensionale Signale, in der Praxis meist Bilddaten, kann man genauso vorgehen, wie dies in Abschnitt 1.2 am Beispiel der Filterbank zum Haar-Wavelet gezeigt wurde. Man transformiert das Bild erst zeilenweise und erhält damit zwei Teilbilder, wie dies in Abb. 1.6 rechts sichtbar ist. Durch anschließende spaltenweise Transformation erhält man schließlich vier Subbänder (wie in Abb. 1.7).

Dabei wird üblicherweise das Ergebnis einer Tiefpassfilterung in Zeilen- und Spaltenrichtung mit „HH“ bezeichnet. Es ist eine geglättete und verkleinerte Version des ursprünglichen Bildes. Das Ergebnis einer Hochpassfilterung in Zeilen- und Spaltenrichtung wird mit „GG“ bezeichnet. Es enthält an den Stellen betragsmäßig große Koeffizienten, an denen das ursprüngliche Bild Kanten in Diagonalrichtung hat, die restlichen Koeffizienten sind betragsmäßig klein.

Das Subband „HG“ enststeht durch Hochpassfilterung in Zeilenrichtung und Tiefpassfilterung in Spaltenrichtung. Betragsmäßig große Koeffizienten ergeben sich nur an den Stellen, an denen das Bild *vertikale* Kanten hat. Das Subband „GH“ enststeht durch Tiefpassfilterung in Zeilenrichtung und Hochpassfilterung in Spaltenrichtung und hat entsprechend nur dort betragsmäßig große Koeffizienten, wo das ursprüngliche Bild *horizontale* Kanten hat. Dies ist in Abb. 1.11 verdeutlicht, wobei der Kontrast außer in Subband HH erheblich verstärkt wurde.

Bei der bisher beschriebenen Vorgehensweise wurde eine eindimensionale Transformation einfach zeilen- und spaltenweise auf Bilddaten übertragen. Dies bevorzugt Zeilen- und Spaltenrichtungen, was für einige Anwendungen von Nachteil ist. Man kann auch Transformationen benutzen, die nicht aus einer Hintereinanderausführung zweier eindimensionaler Transformationen bestehen. Dies ist in Abschnitt 3.6.1 beschrieben.

Auch für die Behandlung zweidimensionaler Daten kann die z-Transformation nützlich sein. Wir gehen auch hier davon aus, dass die Daten abgetastet vorliegen, also beispielsweise $x(k_1, k_2)$ die Helligkeit des Bildes in der k_1. Zeile und der k_2. Spalte beschreibt. Zur Vereinfachung denken wir uns hier das Bild dadurch unendlich ausgedehnt, dass wir die Helligkeitswerte $x(k_1, k_2)$ außerhalb des Bildes Null setzen. Die Behandlung der Probleme,

Abbildung 1.11: Beispiel für die Transformation von Bilddaten. Bei den Subbändern HG, GH, GG wurde der Kontrast verschärft. Es wurde das Filter DD(4,2) (siehe Abschnitt 3.3) benutzt.

die daher kommen, dass ein praktisches Bild endlich ausgedehnt ist, wird auf Kapitel 2.9 verschoben. Die zweidimensionale z-Transformierte ist dann definiert durch

$$X(z_1, z_2) = \sum_{k_1,k_2=-\infty}^{+\infty} x(k_1, k_2) z_1^{-k_1} z_2^{-k_2} \tag{1.49}$$

Ein gefiltertes Bild $y(k_1, k_2)$ entsteht im allgemeinsten Fall durch Faltung mit der Impulsantwort

$$y(n_1, n_2) = \sum_{k_1,k_2=-\infty}^{+\infty} f(n_1 - k_1, n_2 - k_2) x(k_1, k_2) \tag{1.50}$$

und für die z-Transformierte hat man

$$Y(z_1, z_2) = F(z_1, z_2) \cdot X(z_1, z_2) \tag{1.51}$$

Mit zwei eindimensionalen Filtern $F_1(z)$ und $F_2(z)$ kann man eine zweidimensionale Filterung durchführen. Man filtert zuerst spaltenweise (nur der 1. Index, der Zeilenindex wird variiert) mit $F_1(z)$ und erhält als Ausgang $F_1(z_1) \cdot X(z_1, z_2)$. Anschließend filtert man zeilenweise mit $F_2(z)$ filtert. Als Ergebnis erhält man insgesamt

$$Y(z_1, z_2) = F_1(z_1) \cdot F_2(z_2) \cdot X(z_1, z_2) \tag{1.52}$$

Man sieht hieran, dass es dabei nicht auf die Reihenfolge der Filterungen ankommt (erst spalten-, dann zeilenweise oder umgekehrt). Allerdings erhält man im allgemeinen ein anderes Ergebnis, wenn man die Rollen der beiden Filter vertauscht, d.h. mit $F_1(z)$ zeilen- und mit $F_2(z)$ spaltenweise filtert, dies würde der Multiplikation mit $F_1(z_2) \cdot F_2(z_1)$ enstprechen. Und nicht jedes Filter $F(z_1, z_2)$ entsteht durch Multiplikation mit zwei eindimensionalen Filtern $F_1(z_1) \cdot F_2(z_2)$.

Um umständliche Formeln der Form (1.49) zu vermeiden, führt man eine vektorielle Schreibweise mit geeigneten Schreibabkürzungen ein. Für Vektoren der Form

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \qquad \text{und} \qquad \mathbf{k} = \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} \tag{1.53}$$

wird eine Potenz durch

$$\mathbf{z}^{\mathbf{k}} := z_1^{k_1} \cdot z_2^{k_2} \tag{1.54}$$

definiert. Ist der Exponent eine Summe, so gilt auch hierfür die Rechenregel

$$\mathbf{z}^{\mathbf{k}+\mathbf{n}} = \mathbf{z}^{\mathbf{k}} \cdot \mathbf{z}^{\mathbf{n}} \tag{1.55}$$

Schreibt man die Variablen ebenfalls vektoriell, so wird (1.49) zu

$$X(\mathbf{z}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} x(\mathbf{k})\mathbf{z}^{-\mathbf{k}} \tag{1.56}$$

Die durch (1.50) beschriebene Faltung wird dann

$$y(\mathbf{n}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} f(\mathbf{n}-\mathbf{k})x(\mathbf{k}) \tag{1.57}$$

Der große Vorteil dieser Schreibweise ist also, dass die Formeln bis auf das Auftreten von Vektoren mit denen des eindimensionalen Falls *übereinstimmen*.

1.7 Mehrstufige Filterbank-Transformation

Das transformierte Testbild in Abb. 1.7 legt nahe, dass man dieselbe Transformation (Mittelwert- und Differenzbildung) erst zeilen- und dann spaltenweise erneut auf das linke obere Teilbild anwendet, und dies weiter wiederholt. In Abb. 1.12 wird gezeigt, was mit eindimensionalen Signalen (z.B. den Bildzeilen) zu tun ist: Die Ausgänge der Tiefpassfilter werden als Eingangssignale an eine weitere Filterbank gegeben. Durch die Unterabtastung enthalten die Ausgänge der Tiefpassfilter immer niederere Frequenzen (um den Faktor 2). Da die Transformation jeder Filterbankstufe einzeln aufgrund der Bedingung der exakten Rekonstruktion (1.46) und (1.47) rückgängig gemacht werden kann, erhält man das ursprüngliche Signal durch eine Anordnung von Filterbänken wie in Abb. 1.13 zurück.

Wenn bei einem eindimensionalen Signal $s_0(k)$ die Gesamtzahl N der Signalwerte gerade ist, dann erhält man für den Ausgang des Tiefpassfilters und des Hochpassfilters jeweils $\frac{N}{2}$ Werte. Man speichert diese häufig in einem Vektor derselben Länge ab, und zwar den Ausgang des Hochpassfilters $d_1(k)$ in der „oberen“ Hälfte, also den Indizes $\frac{N}{2} \cdots N-1$, den des Tiefpassfilters in der „unteren“ Hälfte, also den Indizes $0 \cdots \frac{N}{2}-1$. In der nächsten Stufe gibt man dann nur die „untere“ Hälfte des Vektors als Eingang auf die Filterbank. Man erhält so eine fortgesetzte Halbierung der weiterverarbeiteten Werte. Wenn die ursprüngliche Gesamtzahl N der Werte durch 2^m teilbar ist, dann erhält man nach einer m-stufigen Fiterbanktransformation die Ausgänge in der folgenden Anordnung:

Typ	Speicherplätze		
$d_1(k)$	$\frac{N}{2}$	$\cdots$	$N-1$
$d_2(k)$	$\frac{N}{4}$	$\cdots$	$\frac{N}{2}-1$
$d_3(k)$	$\frac{N}{8}$	$\cdots$	$\frac{N}{4}-1$
$\vdots$		$\vdots$	
$d_m(k)$	$\frac{N}{2^m}$	$\cdots$	$\frac{N}{2^{m-1}}-1$
$s_m(k)$	0	$\cdots$	$\frac{N}{2^m}-1$

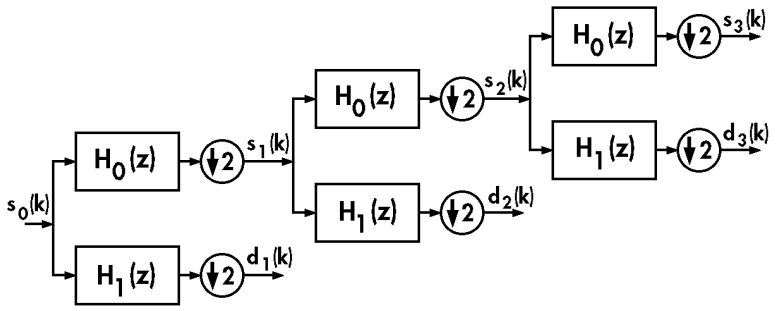


Abbildung 1.12: Dreistufige Filterbank: Analyseteil. $H_0(z)$ ist das Tiefpassfilter.

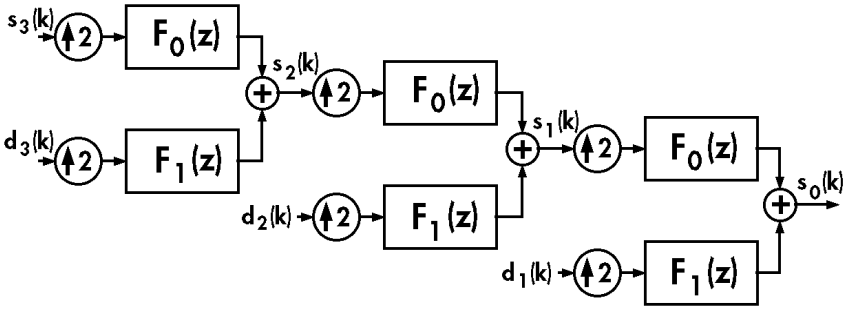


Abbildung 1.13: Dreistufige Filterbank: Syntheseteil. $F_0(z)$ ist das Tiefpassfilter.

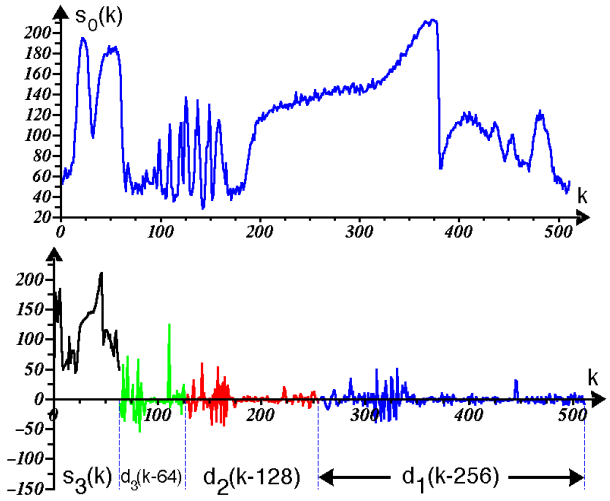


Abbildung 1.14: Dreistufige Filterbanktransformation (Haar-Wavelet) angewandt auf die 500. Zeile des Testbildes „Lena“, der Eingang ist oben, der Ausgang unten in „Mallat“-Anordnung gezeigt.

Diese Anordnung wird „Mallat“-Ordnung genannt. In Abb. 1.14 ist das Ergebnis einer dreistufigen Filterbank-Transformation mit der Haar-Filterbank (Mittelwert und Differenz) in dieser Anordnung gezeigt.

Bei Bildern erhält man die „Mallat“-Ordnung, indem man die nächste Filterbank-Transformation auf das HH-Subband der vorherigen Stufe anwendet. Dieses Subband ist

üblicherweise „links oben“, belegt also in einer $N \times M$-Matrix die Zeilenindizes $0 \cdots \frac{N}{2} - 1$ und die Spaltenindizes $0 \cdots \frac{M}{2} - 1$. Das Ergebnis einer dreistufigen Filterbanktransformation, angewandt auf das Testbild „Lena“ (erst zeilenweise, dann spaltenweise), ist in Abb. 1.15 gezeigt.

Für Abb. 1.15 sind die nach den Gleichungen (1.8) und (1.9) normierten Filter benutzt worden, d.h. für die Analyse

$$H_0(z) = \frac{1}{\sqrt{2}}(1 + z); \qquad H_1(z) = \frac{1}{\sqrt{2}}(1 - z) \tag{1.58}$$

und für die Synthese (Rekonstruktion)

$$F_0(z) = \frac{1}{\sqrt{2}}(1 + z^{-1}); \qquad F_1(z) = \frac{1}{\sqrt{2}}(1 - z^{-1}) \tag{1.59}$$

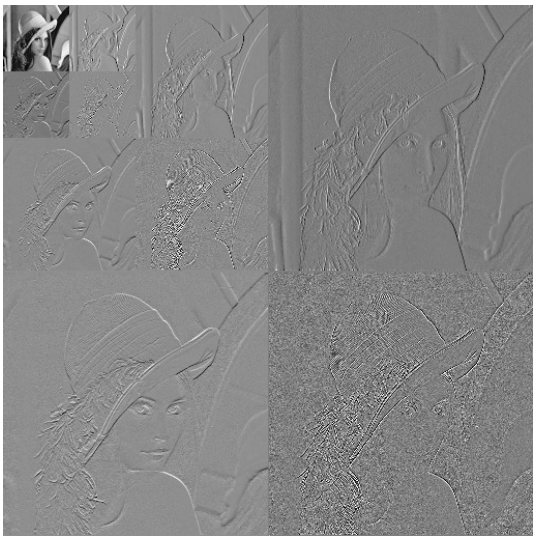


Abbildung 1.15: Dreistufige Filterbanktransformation (Haar-Wavelet) angewandt auf das Testbild „Lena“, die Kontraste der Hochpassanteile enthaltenden Subbänder sind verstärkt.

Kapitel 2

Von den Filterbänken zu den Wavelets

2.1 Einfluß des Quantisierungsfehlers bei der Rekonstruktion

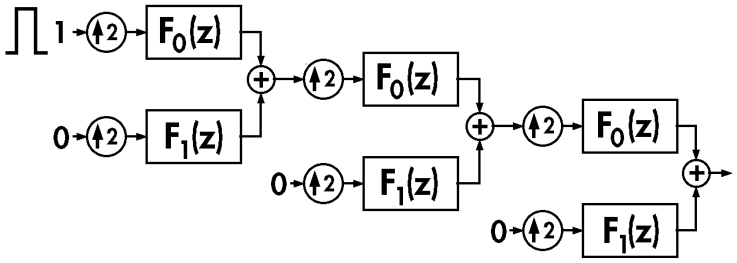


Abbildung 2.1: Einheitsimpuls bei der Rekonstruktion (zur Ermittlung des Einflusses eines Quantisierungsfehlers)

Betrachten wir eine mehrstufige Filterbank zur Analyse (n Stufen), wie sie beispielsweise für $n = 3$ in Abb. 1.12 gezeigt ist. Für die Datenkompression werden die Koeffizienten $d_i(k)$ (wobei $1 \leq i \leq n$) und $s_n(k)$ (von der letzten Stufe) quantisiert, d.h. im Rechner durch die Schrittweite dividiert und durch Runden in eine ganze Zahl umgewandelt. Der Vorgang der Quantisierung wird ausführlich in Abschnitt 6.3 besprochen. Bei der Rekonstruktion rechnet man dann mit einem geänderten Wert (der gerundete wird mit der Schrittweite multipliziert). Da die verwandten Transformationen *linear* sind, genügt es, sich den Einfluß eines Einheitsimpulses bei der Rekonstruktion anzuschauen (ein beliebiger Fehler kann als Linearkombination von Einheitsimpulsen dargestellt werden). Die betrachtete Situation ist in Abb. 2.1 dargestellt. Was aus dem Einheitsimpuls

$$x(k) = \begin{cases} 1 & \text{falls } k = 0 \\ 0 & \text{falls } k \neq 0 \end{cases} \qquad \text{also } X(z) = 1 \tag{2.1}$$

bei der Rekonstruktion entsteht, wenn an allen übrigen Eingängen ein Nullsignal anliegt, ist in Abb. 2.2 gezeigt. Für den Ausgang erhält man (mit den Filtern des Haar-Wavelets, siehe (1.59))

$$Y(z) = F_0(z^4)F_0(z^2)F_0(z) \tag{2.2}$$

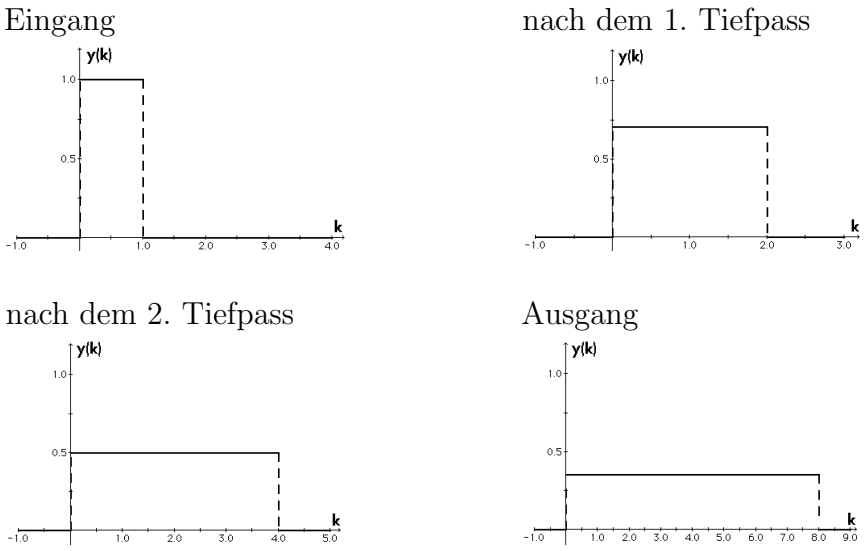


Abbildung 2.2: Ausgang bei der in Abb. 2.1 gezeigten Situation mit den Filterkonstanten für das Haar-Wavelet (zur Ermittlung des Einflusses eines Quantisierungsfehlers)

Das Ergebnis ist für die Praxis niederschmetternd. Der Fehler wird zwar in der Amplitude niedriger, bleibt in der Form aber gleich und breitet sich aus. Der Sprung führt — zu einem zweidimensionalen Bildsignal als Fehler addiert — zu einer sichtbaren *Kante*. Ein erheblich günstigeres Ergebnis erhält man jedoch, wenn man dieselbe Rechnung mit den in Tabelle 1.2 dargestellten Filterkonstanten durchführt. Dies ist in Abb. 2.3 verdeutlicht. Es wurden dabei mehr Stufen als in Abb. 2.1 verwandt. Aus dem Einheitsimpuls wird ein etwas merkwürdig aussehendes Signal, das an seinem „Rand“ *stetig* in 0 übergeht. Dies ist — als Fehler zu Bildsignalen addiert — weniger stark sichtbar.

Analog kann man den Einheitsimpuls als Eingang auf den Hochpass geben und alle übrigen Eingänge auf Null setzen, wie dies in Abb. 2.4 dargestellt ist. Die Ergebnisse sind für die Haar-Filterkonstanten in Abb. 2.5, für die Filterkonstanten „D4“ (siehe Tabelle 1.2) in Abb. 2.6 dargestellt.

Auch hier ist ein entsprechender Einfuß eines Quantisierungsfehlers bei den Haar-Filterkonstanten nicht akzeptabel, bei den Filterkonstanten „D4“ jedoch schon erheblich günstiger. Gibt man den Einheitsimpuls als Eingang auf die andern Hochpassfilter einer mehrstufigen Filterbank, so ändert sich die Situation qualitativ überhaupt nicht. Der Impuls durchläuft dann bei der Rekonstruktion lediglich weniger Tiefpassfilter.

Wir wollen nun die betrachtete Situation unter einem mehr mathemtischen Gesichtspunkt interpretieren. Abb. 2.1 und Abb. 2.4 bedeuten, daß wir alle Koeffizienten bis auf einen Null gesetzt haben. Bei der Fourier-Reihe erhalten wir mit $c_k = 0$ für $k \neq n$ und $c_n = 1$ gerade die Basisfunktion $f(t) = e^{jn\omega t}$. Bei der diskreten Fourier-Transformation würden wir entsprechend abgetastete Werte der e-Funktion erhalten. Der Sachverhalt ist hier ganz ähnlich. Wir werden die bei der Rekonstruktion eines einzelnen Impulses erhaltenen Signale als abgetastete Näherungen unserer Basisfunktionen, nämlich der Skalierungsfunktion (Impuls als Eingang am Tiefpass) und des Wavelets (Impuls als Eingang am Hochpass) ansehen. Dies wird für das Beispiel der Haar-Filterkonstanten in den Abschnitten 2.2 und 2.3 sowie im allgemeinen Fall im Abschnitt 2.4 besprochen.

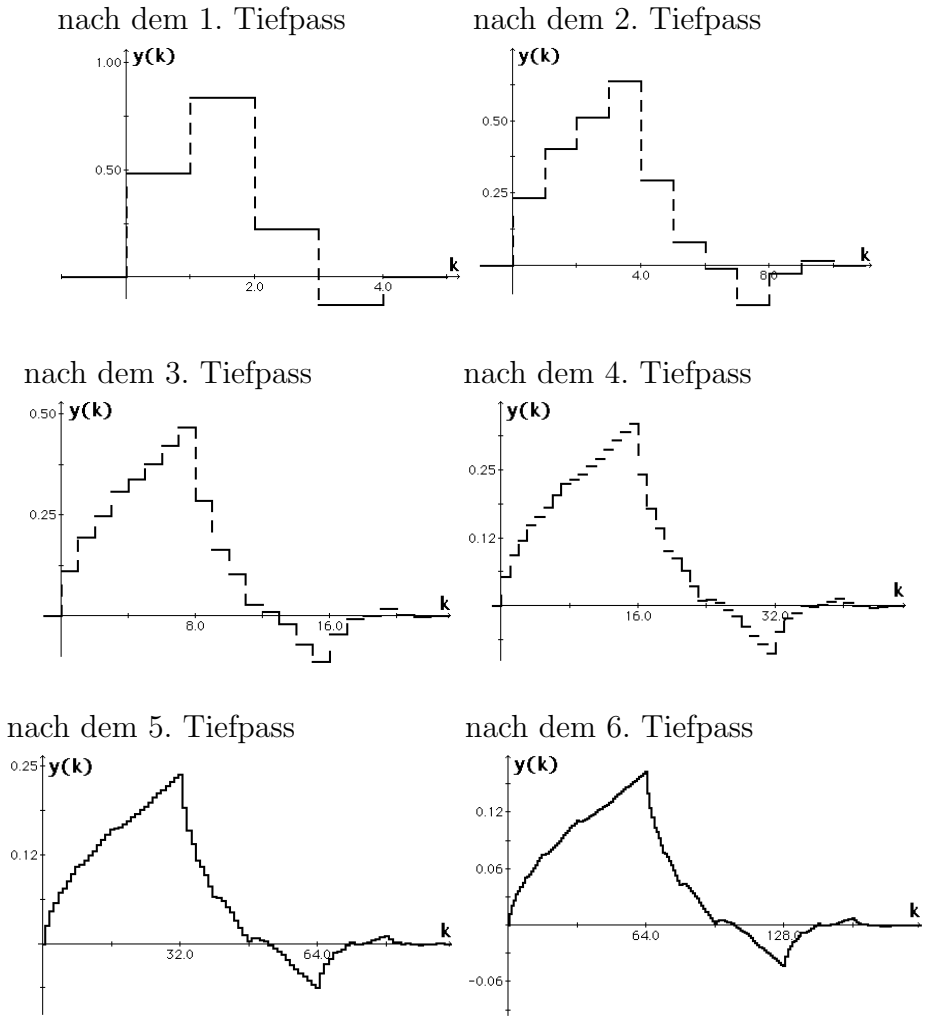


Abbildung 2.3: Ausgang bei der in Abb. 2.1 gezeigten Situation, jedoch mit den Filterkonstanten „$D4$“ (siehe Tabelle 1.2)

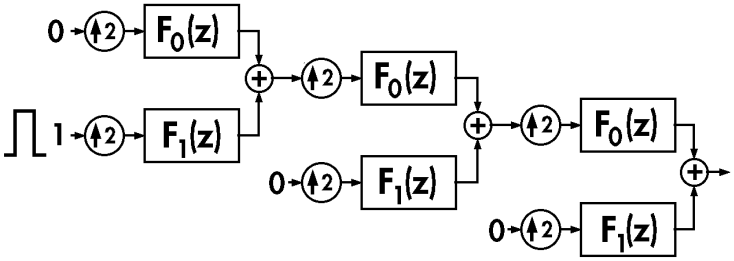


Abbildung 2.4: Einheitsimpuls bei der Rekonstruktion als Eingang beim Hochpass

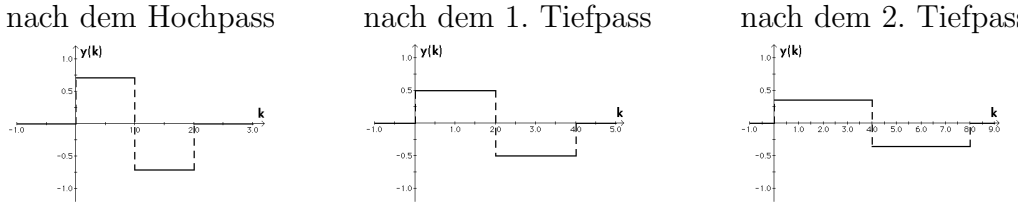


Abbildung 2.5: Ausgang bei der in Abb. 2.4 gezeigten Situation mit den Haar-Filterkonstanten

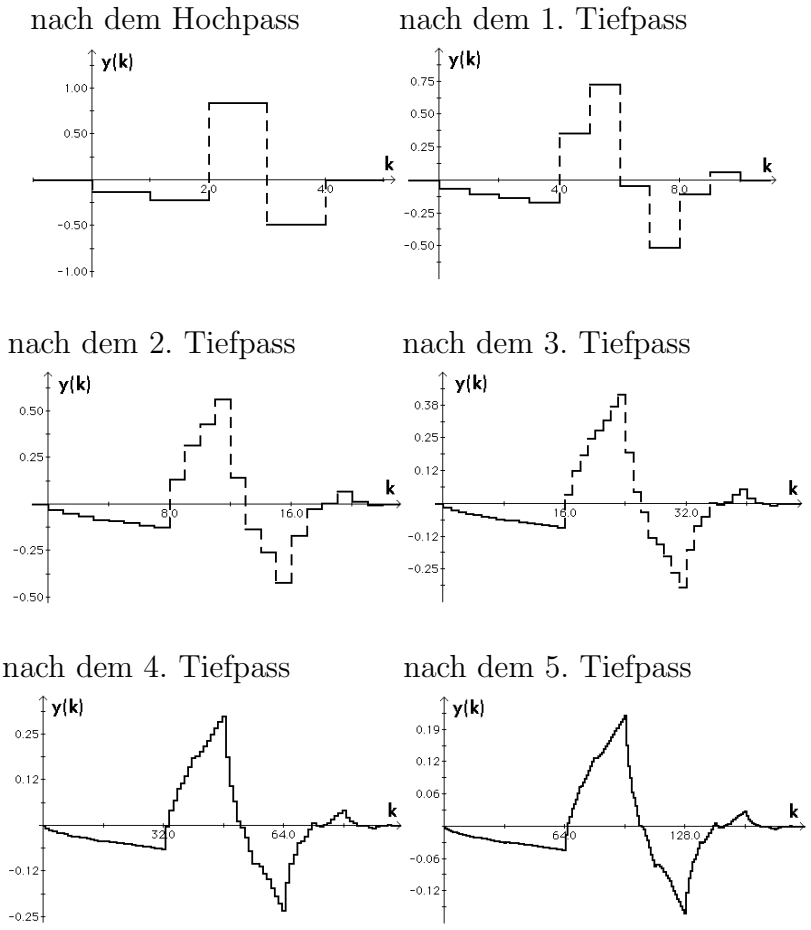


Abbildung 2.6: Ausgang bei der in Abb. 2.4 gezeigten Situation, jedoch mit den Filterkonstanten „$D4$“ (siehe Tabelle 1.2)

2.2 Das Haar-Wavelet und die zugehörige Skalierungsfunktion

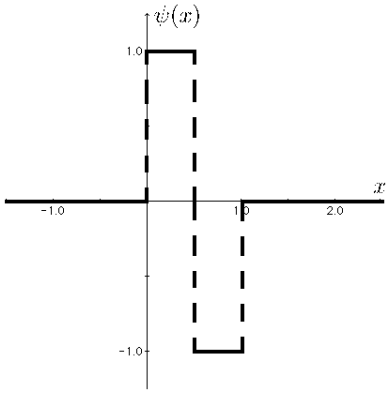


Abbildung 2.7: Das Haar-Wavelet als „Mutter“-Wavelet

Die bisher besprochenen Filterbanktransformationen sind älter als die Untersuchung der zugehörigen Funktionen. Es hat sich erst später herausgestellt, dass man damit die Koeffizienten der zugehörigen Wavelet-Entwicklung ausrechnet. Ist man ausschließlich an der Anwendungspraxis interessiert, so zeigt sich der praktische Nutzen der Betrachtung der zugehörigen Funktion nur bei der Untersuchung des Einflusses der Quantisierungsfehler bei der Rekonstruktion (siehe Abschnitt 2.1).

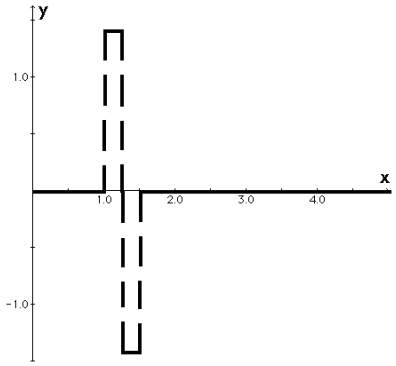


Abbildung 2.8: Gestrecktes ($m = 1$) und verschobenes ($n = 2$) Haar-Wavelet (siehe (2.4))

Die Ergebnisse dieser Untersuchungen (vgl. insbesondere auch Abb. 2.5) legen es nahe, das *Haar-Wavelet* als die folgende, in Abb. 2.7 gezeigte Funktion zu definieren

$$\psi(x) := \begin{cases} +1 & \text{falls } x \in [0, \frac{1}{2}[\\ -1 & \text{falls } x \in [\frac{1}{2}, 1[\\ 0 & \text{sonst} \end{cases} \tag{2.3}$$

Man benutzt skalierte und verschobene Versionen:

$$\psi_{m,n}(x) := 2^{\frac{m}{2}} \psi(2^m x - n), \qquad m, n \in \mathbb{Z} \tag{2.4}$$

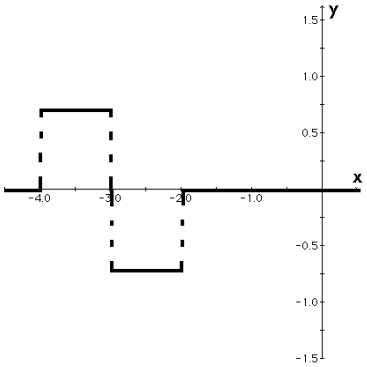


Abbildung 2.9: Gestauchtes ($m = -1$) und verschobenes ($n = -2$) Haar-Wavelet (siehe (2.4))

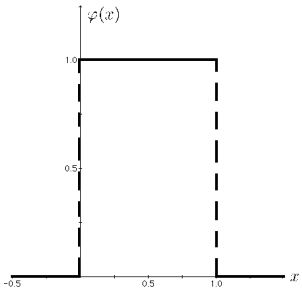


Abbildung 2.10: Skalierungsfunktion zum Haar-Wavelet

In Abb. 2.8 ist $\psi_{1,2}(x)$ und in Abb. 2.9 $\psi_{-1,-2}(x)$ gezeigt.

Die Ergebnisse von Abschnitt 2.1 legen es ebenfalls nahe (siehe insbesondere Abb. 2.2), neben dem Wavelet eine weitere Basisfunktion zu benutzen, die *Skalierungsfunktion* (in der Frühzeit der Wavelets „Vater“ genannt)

$$\varphi(x) := \begin{cases} +1 & \text{falls } x \in [0,1[\\ 0 & \text{sonst} \end{cases} \tag{2.5}$$

Sie ist in Abb. 2.10 gezeigt. Auch hiervon braucht man skalierte und verschobene Versionen:

$$\varphi_{m,n}(x) := 2^{\frac{m}{2}} \varphi(2^m x - n), \qquad m, n \in \mathbb{Z} \tag{2.6}$$

In Abb. 2.11 ist $\varphi_{1,2}(x)$ und in Abb. 2.12 $\varphi_{-1,-2}(x)$ gezeigt. Zu den folgenden wichtigen Gleichungen siehe auch Abb. 2.13.

Die Skalierungsfunktion erfüllt die Zwei-Skalen-Relation:

$$\varphi(x) = \varphi(2x) + \varphi(2x-1) \tag{2.7}$$

und zum Wavelet besteht der Zusammenhang:

$$\psi(x) = \varphi(2x) - \varphi(2x-1) \tag{2.8}$$

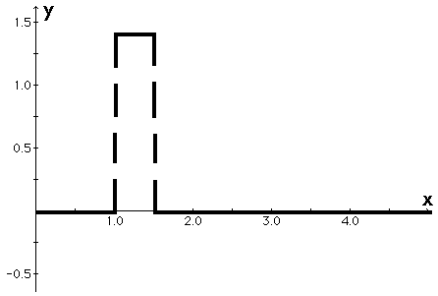


Abbildung 2.11: Gestreckte ($m = 1$) und verschobene ($n = 2$) Skalierungsfunktion $\varphi_{mn}(x)$ (siehe (2.6))

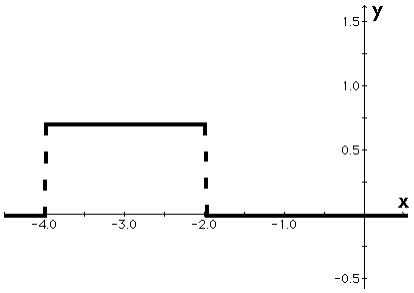


Abbildung 2.12: Gestauchte ($m = -1$) und verschobene ($n = -2$) Skalierungsfunktion $\varphi_{mn}(x)$ (siehe (2.6))

Abbildung 2.13: Zur Zwei-Skalen-Relation (2.7) und zum Zusammenhang Skalierungsfunktion - Wavelet (2.8)

2.3 Die Skalierungsfunktion als Vermittler zwischen diskretem und kontinuierlichem Signal

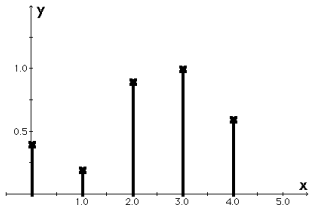


Abbildung 2.14: zeitdiskretes (abgetastetes) Signal $y(k) = s_0(k)$

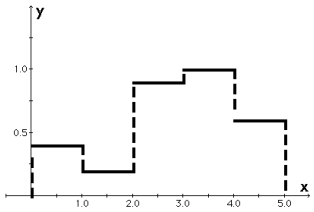


Abbildung 2.15: Aus dem zeitdiskreten Signal von Abb. 2.14 durch (2.9) entstandenes kontinuierliches Signal.

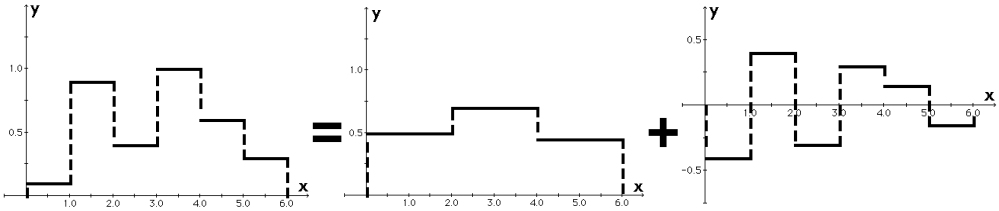


Abbildung 2.16: Zerlegung eines Signals durch Mittelwert- und Differenzbildung, siehe (2.12).

Intuitiv ist es plausibel, dass das Haar-Wavelet etwas mit der Differenz und die Skalierungsfunktion etwas mit dem Mittelwert zu tun hat. Es ist sinnvoll, aus dem in Abb. 2.14 dargestellten abgetasteten Signal $y(k) = s_0(k)$ mit Hilfe der Skalierungsfunktion durch

$$f_0(x) = \sum_{k=-\infty}^{+\infty} s_0(k)\varphi(x-k) \tag{2.9}$$

wieder ein kontinuierliches Signal zu erzeugen. Dieses ist in Abb. 2.15 graphisch dargestellt. Hierdurch hat man jedoch nur eine *Näherung* $f_0(x)$ an das ursprüngliche kontinuierliche Signal $f(x)$ gewonnen, das i.a. nicht konstant zwischen den Abtastzeitpunkten ist.

Eine *gröbere* Näherung an das ursprüngliche Signal erhält man durch

$$f_1(x) = \sum_{k=-\infty}^{+\infty} s_1(k)\varphi_{-1,k}(x) \tag{2.10}$$

Diese Näherung ist auf Intervallen der Länge 2 konstant. Es ist sinnvoll, die zugehörigen Funktionswerte durch Mittelwertbildung aus den ursprünglichen zu berechnen. Für die Koeffizienten haben wir zu berücksichtigen, dass die Funktionswerte $\varphi_{-1,k}(x)$ gegenüber denen von $\varphi(x-k)$ durch die Normierung in der Definition (2.6) um den Faktor $2^{-\frac{1}{2}} = \frac{1}{\sqrt{2}}$ verringert sind. Wir erhalten daher die neuen Koeffizienten mit Hilfe der normierten Filterkonstanten nach (1.8), d.h.

$$s_1(k) = \frac{1}{\sqrt{2}}\big(s_0(2k) + s_0(2k+1)\big)$$

Man kann dieses Ergebnis rechnerisch auch leicht aus der Zwei-Skalen-Relation (2.7) erhalten.

Betrachten wir nun die Differenz zwischen gröberer und Standardnäherung. Sie ist durch verschobene und gedehnte (Skalierungsstufe -1) Versionen des Wavelets darstellbar. Die entsprechenden Funktionswerte kommen durch Differenzbildung zustande. Aufgrund der in (2.4) festgelegten Normierung erhält man die entsprechenden Koeffizienten $d_1(k)$ auch hier nicht aus der Differenz der Koeffizienten, sondern aus der mit $\frac{1}{\sqrt{2}}$ normierten Differenzbildung in (1.9). Mit

$$d_1(k) = \frac{1}{\sqrt{2}}\big(s_0(2k) - s_0(2k+1)\big)$$

haben wir also

$$f_1(x) - f_0(x) = \sum_{k=-\infty}^{+\infty} d_1(k)\psi_{-1,k}(x) \tag{2.11}$$

und damit

$$\sum_{k=-\infty}^{+\infty} s_0(k)\varphi_{0,k}(x) = \sum_{k=-\infty}^{+\infty} s_1(k)\varphi_{-1,k}(x) + \sum_{k=-\infty}^{+\infty} d_1(k)\psi_{-1,k}(x) \tag{2.12}$$

Die oben angegebene Berechnung der Koeffizienten $d_1(k)$ aus $s_0(k)$ ergibt sich auch aus dem wichtigen Zusammenhang zwischen Wavelet und Skalierungsfunktion in (2.8). Die Zerlegung (2.12) ist in Abb. 2.16 symbolisch dargestellt.

Wir können dieses Vorgehen fortsetzen und eine noch gröbere Näherung an das ursprüngliche Signal betrachten durch

$$f_2(x) = \sum_{k=-\infty}^{+\infty} s_2(k)\varphi_{-2,k}(x) \tag{2.13}$$

die auf Intervallen der Länge 4 konstant ist und deren Funktionswerte durch erneute Mittelwertbildung aus der vorigen Näherung entstehen. Die Differenz ist auch hier durch Wavelets darstellbar, diesmal der Skalierungsstufe -2. Die Koeffizienten enstehen erneut aus der Haar-Filterbank mit den normierten Filterkoeffizienten, d.h. mit

$$\begin{aligned} s_2(k) &= \frac{1}{\sqrt{2}}(s_1(2k) + s_1(2k+1) \\ d_2(k) &= \frac{1}{\sqrt{2}}\big(s_1(2k) - s_1(2k+1)\big) \end{aligned}$$

bekommen wir

$$f_2(x) - f_1(x) = \sum_{k=-\infty}^{+\infty} d_2(k)\psi_{-2,k}(x) \tag{2.14}$$

und

$$\begin{aligned}\sum_{k=-\infty}^{+\infty} s_0(k)\varphi_{0,k}(x) &= \sum_{k=-\infty}^{+\infty} d_1(k)\psi_{-1,k}(x) + \sum_{k=-\infty}^{+\infty} d_2(k)\psi_{-2,k}(x) \\ &\quad + \sum_{k=-\infty}^{+\infty} s_2(k)\varphi_{-2,k}(x)\end{aligned} \tag{2.15}$$

Wir können diese Vorgehensweise nochmal wiederholen und erhalten die Zerlegung

$$\begin{aligned}\sum_{k=-\infty}^{+\infty} s_0(k)\varphi_{0,k}(x) &= \sum_{k=-\infty}^{+\infty} d_1(k)\psi_{-1,k}(x) + \sum_{k=-\infty}^{+\infty} d_2(k)\psi_{-2,k}(x) \\ &\quad + \sum_{k=-\infty}^{+\infty} d_3(k)\psi_{-3,k}(x) + \sum_{k=-\infty}^{+\infty} s_3(k)\varphi_{-3,k}(x)\end{aligned} \tag{2.16}$$

Dies entspricht genau der Zerlegung, die durch die Filterbänke in Abb. 1.12 durchgeführt wird mit den durch (1.58) gegebenen Filtern.

Halten wir als Ergebnis fest:

Bildet man aus den Abtastwerten $y(k)$ das kontinuierliche Signal

$$f_0(x) = \sum_{k=-\infty}^{+\infty} s_0(k)\varphi(x-k)$$

so erhält man als Hochpass-Ausgang der Filterbank mit den Filtern

$$H_0(z) = \frac{1}{\sqrt{2}}(1+z); \qquad H_1(z) = \frac{1}{\sqrt{2}}(1-z)$$

die Koeffizienten einer Wavelet-Entwicklung, als Tiefpass-Ausgang die Koeffizienten der Skalierungsfunktion.

Der Sinn dieser Überlegungen war es, eine Einsicht über den Zusammenhang zwischen Filterbank-Operationen und Wavelets zu vermitteln, die im hier betrachteten Beispiel des Haar-Wavelets rechnerisch und intuitiv leicht nachzuvollziehen ist. Die praktische Bedeutung der Eigenschaften von Skalierungsfunktion und Wavelet wird im nächsten Abschnitt sichtbar. Das Vorgehen dieses Abschnitts wird für den allgemeinen Fall biorthogonaler Wavelets in Abschnitt 2.5 wiederholt.

2.4 Skalierungsfunktion und Wavelet als Ergebnis eines Grenzübergangs

Skaliert man das Signal, das in Abb. 2.2 den Einfluß des Quantisierungsfehlers bei der Rekonstruktion beschreibt (siehe Abschnitt 2.1), so entsteht daraus die Skalierungsfunktion zum Haar-Wavelet. Führt man dasselbe mit einem Einheitsimpuls als Eingang beim

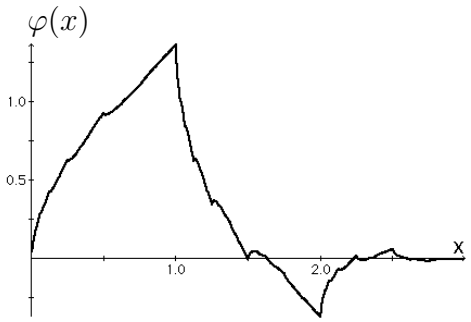


Abbildung 2.17: Skalierungsfunktion zum DAUBECHIES-Wavelet $D4$

Hochpass durch, so entsteht daraus das Haar-Wavelet. Das Filterbeispiel „$D4$“ führt nach jeweiliger Neuskalierung beim Grenzübergang zu unendlich vielen Stufen offensichtlich zu einer stetigen Funktion, die in Abb 2.17 gezeigt ist (vgl. auch Abb. 2.3). Betrachtet man die zugrundegelegten Operationen (Tiefpass filterung, Upsampling, Neuskalierung), so ist dies im Frequenzbereich leicht zu beschreiben.

Signal-Verarbeitung	Wavelet
$H_0(z)$	$\tilde{H}(z^{-1})$
$H_1(z)$	$\tilde{G}(z^{-1})$
$F_0(z)$	$H(z)$
$F_1(z)$	$G(z)$

Tabelle 2.1: Gegenüberstellung der unterschiedlichen Schreibweisen für die Filter bei einer 2-Kanal-Filterbank

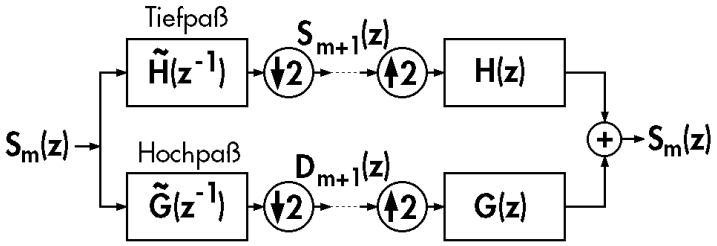


Abbildung 2.18: Zwei-Kanal-Filterbank in „Wavelet-Bezeichnungsweise“ (vgl. Abb. 1.10 und Tabelle 2.1)

Wir gehen dabei über zu der für Wavelets üblichen Schreibweise (siehe Tabelle 2.1), die Filterbank von Abb. 1.10 ist in der neuen Schreibweise in Abb. 2.18 dargestellt. Wir betrachten hier die Rekonstruktion, gehen also von einem Tiefpassfilter $H(z)$ aus, dessen Filterkonstanten im Beispiel $D4$ nun in der neuen Schreibweise lauten

$$\begin{aligned} h(0) &= \frac{1}{4\sqrt{2}}(1+\sqrt{3}), \qquad h(1) = \frac{1}{4\sqrt{2}}(3+\sqrt{3}) \\ h(2) &= \frac{1}{4\sqrt{2}}(3-\sqrt{3}), \qquad h(3) = \frac{1}{4\sqrt{2}}(1-\sqrt{3}) \end{aligned}$$

und $h(k) = 0$ für $k < 0$ und $k > 3$ erfüllen. Der Einheitsimpuls als Eingang (siehe Abb. 2.1) führt auf $X(e^{j\omega}) = 1$. Also haben wir nach der 1. Stufe $H(e^{j\omega})$, nach der 2. Stufe $H(e^{j\omega}) \cdot H(e^{j2\omega})$, nach der 3. Stufe $H(e^{j\omega}) \cdot H(e^{j2\omega}) \cdot H(e^{j4\omega})$. Die Abb. 2.3 legt nahe, dass wir bei jedem Schritt die horizontale Achse um den Faktor 2 skalieren müssen, um einen Grenzwert zu erhalten.

Wir benutzen hier für die Fourier-Transformation

$$F(\omega) := \int_{-\infty}^{+\infty} f(x)e^{-j\omega x}dx; \qquad f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{j\omega x}d\omega$$

die Schreibweise

$$f(x) \;\circ\!\!-\!\!\bullet\; F(\omega)$$

um auszudrücken, dass $F(\omega)$ die Fourier-Transformierte der Funktion $f(x)$ ist. Wir haben nun zu beachten, dass bei der Fourier-Transformation für die Skalierung der x-Achse die folgende Regel gilt:

$$f(ax) \;\circ\!\!-\!\!\bullet\; \frac{1}{a}F\left(\frac{\omega}{a}\right)$$

Eine Sklalierung der x-Achse um den Faktor 2 heißt also, dass wir ω durch $\frac{1}{2}\omega$ zu ersetzen haben. Statt $H(e^{j4\omega})H(e^{j2\omega})H(e^{j\omega})$ müssen wir also $H(e^{j\omega})H(e^{j\frac{1}{2}\omega})H(e^{j\frac{1}{4}\omega})$ nehmen. Und wenn wir Konvergenz wollen, müssen wir auch die Änderungen im Graphen bezüglich der vertikalen Achse im Auge behalten, also die „Amplituden“. Dies ist ganz einfach, denn für $\omega = 0$ erhalten wir (siehe Abschnitt 2.7, insbesondere (2.71)) aufgrund der Normierung $H(1) = \sqrt{2}$, dass wir einen Faktor $\frac{1}{\sqrt{2}}$ bei jedem Faktor anbringen müssen, um Konvergenz zu erhalten.

Wir erwarten also, dass der Grenzwert

$$\Phi(\omega) := \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{2}}) \cdot \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{4}}) \cdot \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{8}}) \cdot \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{16}}) \cdot \cdots \tag{2.17}$$

existiert. Die **Skalierungsfunktion** ist dann durch inverse Fourier-Transformation definiert:

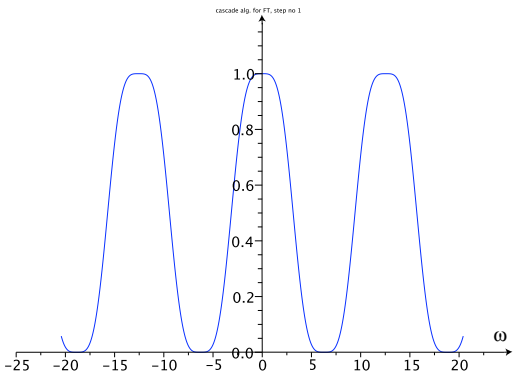
$$\varphi(x) := \frac{1}{2\pi}\int_{-\infty}^{+\infty} \Phi(\omega)e^{j\omega x}d\omega \tag{2.18}$$

D.h. der Grenzübergang (2.17) liefert die Fourier-Transformierte der Skalierungsfunktion $\varphi(x)$.

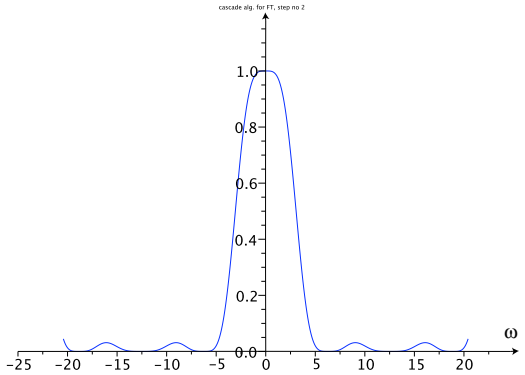
Für das später behandelte Beispiel des „Binlets“ (siehe Abschnitt 3.3, die Filterkonstanten sind in Tabelle 3.2 gezeigt) ist der Grenzübergang (2.17) in Abb. 2.19 veranschaulicht. Beachten Sie, dass $H(e^{j\omega}) = \sum_{k=-\infty}^{+\infty} h(k)e^{-jk\omega}$ Periode 2π, $H(e^{j\frac{\omega}{2}})$ Periode 4π, und $\frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{2}}) \cdot \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{4}})$ Periode 8π hat. Der Grenzwert ist nicht periodisch!

Ersetzt man in (2.17) ω durch $\frac{\omega}{2}$ und vergleicht die entstehende Gleichung mit (2.17), so erhält man die wichtige Beziehung

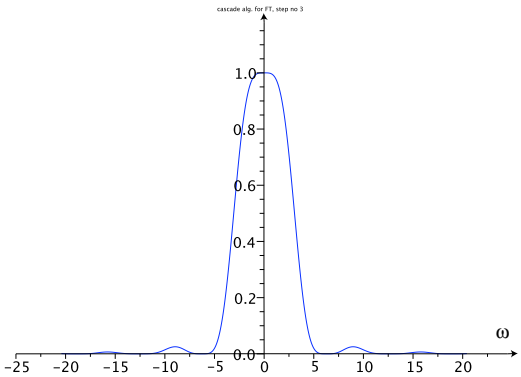
$$\Phi(\omega) = \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{2}})\Phi(\frac{\omega}{2}) \tag{2.19}$$



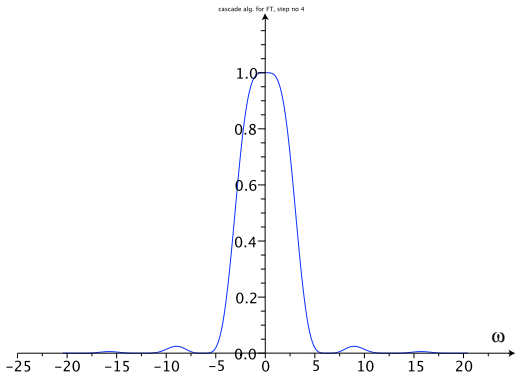
$\frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{2}})$ (Periode 4π, links),



$\frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{2}}) \cdot \frac{1}{\sqrt{2}}H(e^{j\frac{\omega}{4}})$ (rechts)



3 Faktoren (links),



4 Faktoren (rechts)

Abbildung 2.19: Beispiel zur Illustration des Grenzübergangs (2.17), es wurden die Filterkonstanten des „Binlets“ (siehe Abschnitt 3.3) benutzt.

Wir wenden hierauf eine inverse Fourier-Transformation an und benutzen

$$H(e^{j\omega}) = \sum_{k=-\infty}^{+\infty} h(k)e^{-jk\omega}$$

sowie die Substitution $\omega = 2u$

$$\begin{aligned}
\varphi(x) &= \frac{1}{\sqrt{2}}\frac{1}{2\pi}\int_{-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} h(k)e^{-jk\frac{\omega}{2}}\Phi(\frac{\omega}{2})e^{j\omega x}d\omega \\
&= \frac{1}{\sqrt{2}}\sum_{k=-\infty}^{+\infty} h(k)\left(\frac{1}{2\pi}\int_{-\infty}^{+\infty}\Phi(\frac{\omega}{2})e^{j\omega(x-\frac{k}{2})}d\omega\right) \\
&= \sqrt{2}\sum_{k=-\infty}^{+\infty} h(k)\left(\frac{1}{2\pi}\int_{-\infty}^{+\infty}\Phi(u)e^{ju(2x-k)}du\right) \\
&= \sqrt{2}\sum_{k=-\infty}^{+\infty} h(k)\varphi(2x-k)
\end{aligned}$$

Das Ergebnis ist die wichtige

Zwei-Skalen-Relation

$$\varphi(x) = \sqrt{2} \sum_{k=-\infty}^{+\infty} h(k)\varphi(2x-k) \tag{2.20}$$

Für die Skalierungsfunktion zum Haar-Wavelet war uns eine entsprechende Relation bereits in (2.7) begegnet. Ein weniger triviales Beispiel ist durch das Hut-Wavelet CDF(2,2) gegeben (siehe (2.125)). Die Zwei-Skalen-Relation lautet hierfür

$$\varphi(x) = \frac{1}{2}\varphi(2x+1) + \varphi(2x) + \frac{1}{2}\varphi(2x-1) \tag{2.21}$$

und ist in Abb. 2.20 (links) veranschaulicht. Ein weiteres Beispiel ist durch das Wavelet CDF(3,5) gegeben mit der Zwei-Skalen-Relation

$$\varphi(x) = \frac{1}{4}\Big(\varphi(2x+1) + 3\varphi(2x) + 3\varphi(2x-1) + \varphi(2x-2)\Big) \tag{2.22}$$

gegeben; es ist in Abb. 2.20 (rechts) grafisch dargestellt.

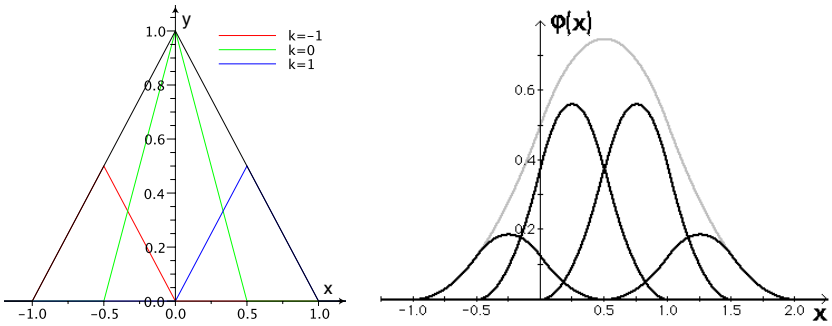


Abbildung 2.20: Beispiele zur Zwei-Skalen-Relation: Die Skalierungsfunktion $\varphi(x)$ (grau) ist darstellbar als Summe „gestauchter“ und verschobener Versionen der Skalierungsfunktion (links für das Hut-Wavelet, rechts für das Wavelet CDF(3,5) siehe auch (2.22))

Die Zwei-Skalen-Relation bleibt erfüllt, wenn wir $\varphi(x)$ durch $c\varphi(x)$ ersetzen (c eine beliebige reelle Konstante), aber die Normierung $H(1) = \sqrt{2}$ liefert für $\omega = 0$ aus (2.17) die Bedingung

$$\Phi(0) = \int_{-\infty}^{+\infty} \varphi(x) e^{-j\cdot 0x} dx = \int_{-\infty}^{+\infty} \varphi(x) dx = 1$$

Die Zwei-Skalen-Relation erlaubt es, die Skalierungsfunktion auch direkt im „Ortsbereich“ als Grenzwert $n \to \infty$ auszurechnen:

$$\varphi_{n+1}(x) = \sqrt{2} \sum_{k=-\infty}^{+\infty} h(k)\varphi_n(2x-k) \tag{2.23}$$

wobei als „Startfunktion“ die Skalierungsfunktion für das Haar-Wavelet

$$\varphi_0(x) = \varphi_{\text{Haar}}(x)$$

entsprechend (2.5) oder die symmetrische Version $\varphi_0(x) = \varphi_{\text{Haar}}(x + \frac{1}{2})$ zu nehmen ist. Dies ist der Kaskade-Algorithmus. Er ist für das später behandelte Beispiel des „Binlets“ (siehe Abschnitt 3.3, Filterkonstanten siehe Tabelle 3.2) in Abb. 2.21 veranschaulicht.

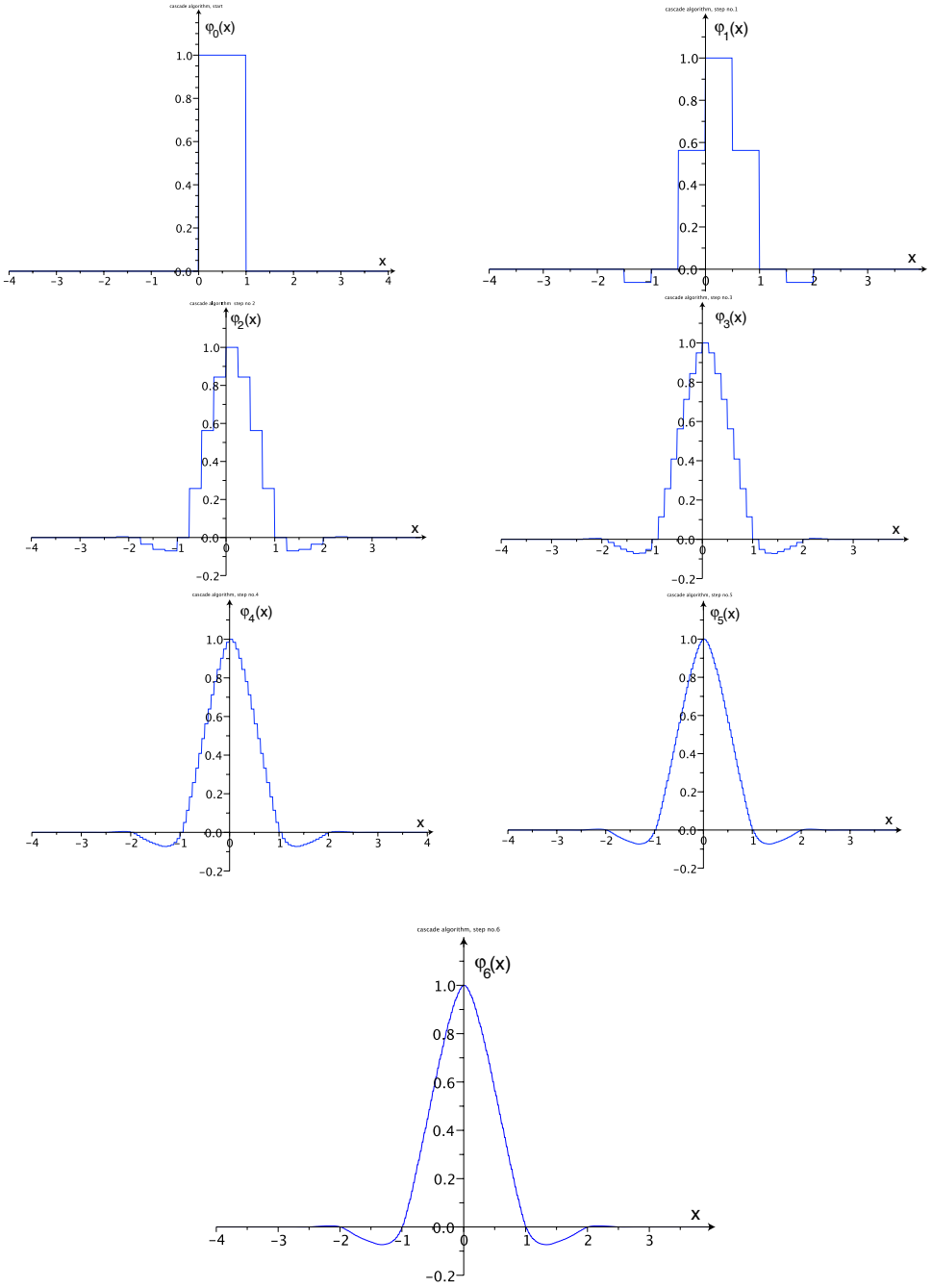


Abbildung 2.21: Beispiel für den Kaskade-Algorithmus (2.23), es wurden die Filterkonstanten des „Binlets“ (siehe Abschnitt 3.3) benutzt.

Dasselbe Vorgehen mit einem Einheitsimpuls beim Eingang des *Hochpassfilters* liefert nach der 1. Stufe $G(e^{j\omega})$, nach der 2. Stufe $G(e^{j\omega}) \cdot H(e^{j2\omega})$, nach der 3. Stufe $G(e^{j\omega}) \cdot H(e^{j2\omega}) \cdot H(e^{j4\omega})$. Wir erwarten also, dass der Grenzwert

$$\Psi(\omega) := \frac{1}{\sqrt{2}} G(e^{j\frac{\omega}{2}}) \cdot \frac{1}{\sqrt{2}} H(e^{j\frac{\omega}{4}}) \cdot \frac{1}{\sqrt{2}} H(e^{j\frac{\omega}{8}}) \cdot \frac{1}{\sqrt{2}} H(e^{j\frac{\omega}{16}}) \cdot \cdots \tag{2.24}$$

und definieren damit das **Wavelet** $\psi(x)$ durch inverse Fourier-Transformation

$$\psi(x) := \frac{1}{2\pi} \int_{-\infty}^{+\infty} \Psi(\omega) e^{j\omega x} d\omega \tag{2.25}$$

Durch Vergleich von (2.17) mit (2.24) erhalten wir die wichtige Beziehung

$$\Psi(\omega) = \frac{1}{\sqrt{2}} G(e^{j\frac{\omega}{2}}) \cdot \Phi(\frac{\omega}{2}) \tag{2.26}$$

Entsprechend der Behandlung von (2.19) erhalten wir daraus durch inverse Fourier-Transformation das folgende Ergebnis:

Das Wavelet $\psi(x)$ ist durch

$$\psi(x) = \sqrt{2} \sum_{k=-\infty}^{+\infty} g(k) \varphi(2x - k) \tag{2.27}$$

gegeben.

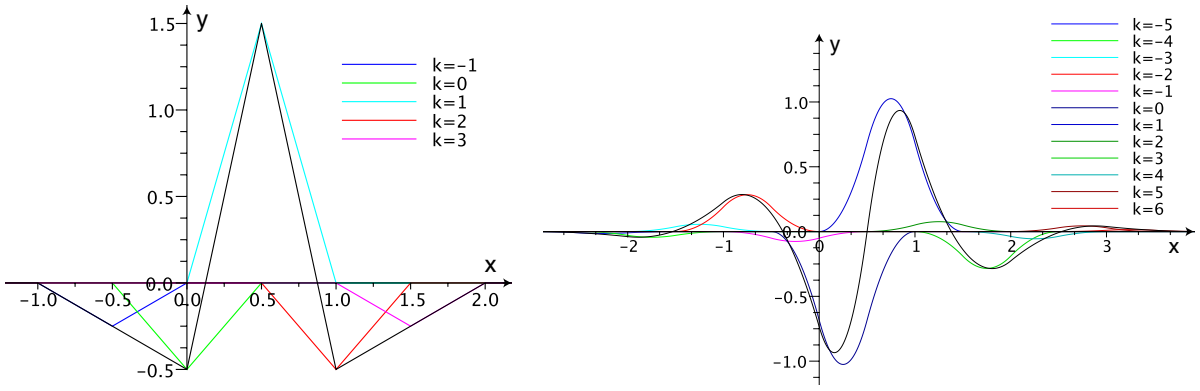


Abbildung 2.22: Beispiel zur Konstruktion von Wavelets (schwarz), links das Hut-Wavelet CDF(2,2), rechts das Wavelet CDF(3,5)) als Summe „gestauchter“ und verschobener Versionen der Skalierungsfunktion

Für das Haar-Wavelet war diese Gleichung trivial: $\psi(x) = \varphi(2x) - \varphi(2x - 1)$. Ein weniger triviales Beispiel ist das Hut-Wavelet (auch mit CDF(2,2) bezeichnet, die Filterkonstanten stehen in (2.125) bis (2.128). In Abb. 2.22 (links) ist illustriert, wie das Wavelet aus verschobenen Versionen der Skalierungsfunktion nach (2.27) aufgebaut wird.

Ein weiteres Beispiel ist das Wavelet CDF(3,5) (die Filterkonstanten $g(k)$ sind in der äußersten rechten Spalte der Tabelle 2.2 dargestellt). In Abb. 2.22 (rechts) ist sichtbar gemacht, wie das entsprechende Wavelet als Summe der verschobenen Versionen der Skalierungsfunktion gemäß (2.27) entsteht.

Das Wavelet für das hier betrachtete Filterbeispiel $D4$ ist in Abb. 2.23 dargestellt. Es beschreibt also den qualitativen Einfluß eines Einheitsimpulses (als „Vertreter“ eines Quantisierungsfehlers) beim Hochpass bei der Rekonstruktion.

Halten wir fest:

> Für die Rekonstruktionsfilter ist es wünschenswert, dass der Grenzwert (2.17) existiert und zu einer stetigen (oder, noch besser, differenzierbaren, n mal differenzierbaren) Funktion, der Skalierungsfunktion, führt. Diese erfüllt dann die Zwei-Skalen-Relation (2.20). Das zugehörige Wavelet ist dann durch (2.27) gegeben.

Auch im Frequenzbereich tritt durch den Übergang vom Haar-Wavelet zum Daubechies-Wavelet $D4$ eine Verbesserung ein, die in Abb. 2.24 deutlich sichtbar ist (gegenüber Abb. 1.8 und 1.9).

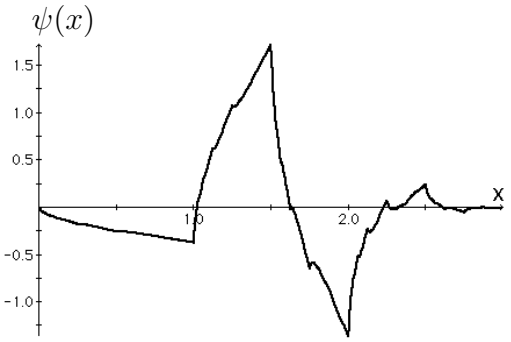


Abbildung 2.23: Daubechies-Wavelet $D4$

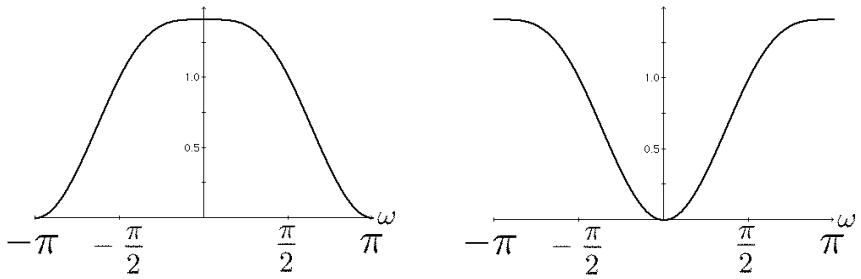


Abbildung 2.24: Absolutbetrag des Frequenzgangs für das Tiefpassfilter $D4$ (links) und für das zugehörige Hochpassfilter (rechts)

Abschließend schauen wir uns noch zwei typische Beispiele an, bei denen der Kaskade-Algorithmus (2.23) nicht konvergiert. Das erste ist durch die Analyse-Filter von CDF(3,1) gegeben (siehe (2.130)). Man kann hierfür zwar durch (2.17) eine Funktion $\tilde{\Phi}(\omega)$ definieren, sie ist in Abb. 2.25 gezeigt. Diese Funktion fällt jedoch für große Werte von $|\omega|$ nicht ab,

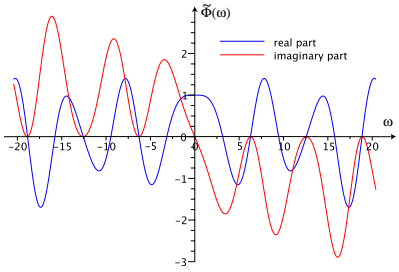


Abbildung 2.25: Ergebnis $\Phi(\omega)$ des Grenzübergangs (2.17) für das CDF(3,1)-Analyse-Filter (Realteil blau, Imaginärteil rot)

so dass die inverse Fourier-Transformation ist dann für diese Funktion nicht definert, denn das entsprechende uneigentliche Integral existiert nicht. Man hat also keine Skalierungsfunktion $\tilde{\varphi}(x)$ für die entsprechenden Filterkonstanten $\tilde{h}(k)$. Der Kaskade-Algorithmus divergiert, es entsthehen immer größere und immer schmalere „Impulse“. (In der Tat würde die inverse Fourier-Transformation eine Summe von Dirac-Impulsen liefern.) Dies ist an den ersten Schritten des Kaskade-Algorithmus sichtbar, die in Abb. 2.26 gezeigt sind.

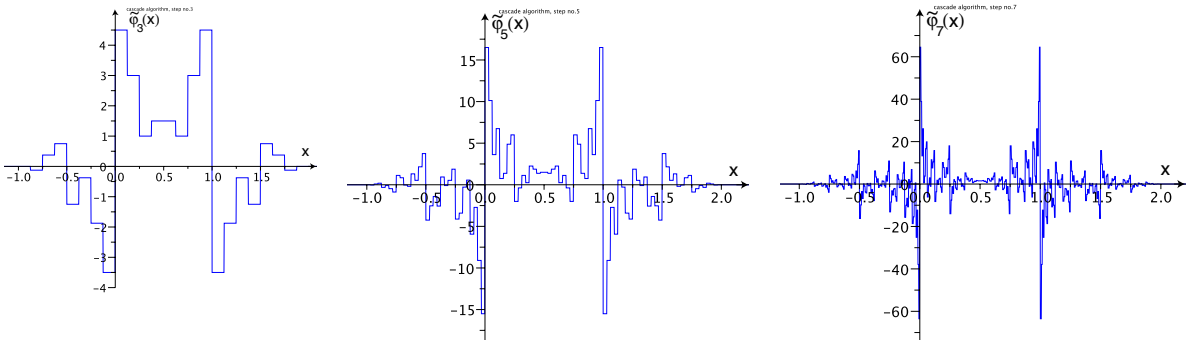


Abbildung 2.26: Erste Schritte des Kaskade-Algorithmus für das CDF(3,1)-Analyse-Filter

Weitere typische Beispiele entstehen, wenn man Nullen in Filter einfügt, für die der Kaskade-Algorithmus konvergiert. Wir nehmen hier das CDF(3,5)-Rekonstruktionsfilter (siehe Abschnitt 2.8.4)

$$H_0(z) = \frac{\sqrt{2}}{8}\left(z + 3 + 3z^{-1} + z^{-2}\right) \tag{2.28}$$

und machen daraus ein „schlechtes Filter“

$$H_1(z) = z \cdot H_0(z^3) = \frac{\sqrt{2}}{8}\left(z^4 + 3z + 3z^{-2} + z^{-5}\right) \tag{2.29}$$

Der Grenzwert (2.17) scheint kein Problem zu machen, das Ergebnis ist in Abb. 2.27 gezeigt. Die in Anhang D.1 durchgeführte Rechnung legt nahe, dass

$$\Phi_1(\omega) = e^{j\omega}\Phi_0(3\omega)$$

und aus den Rechenregeln für die Fourier-Transformation folgt

$$\tfrac{1}{3}\varphi_0\left(\tfrac{1}{3}(x+1)\right) = \varphi_1(x) \;\circ\!\!-\!\!\bullet\; \Phi_1(\omega)$$

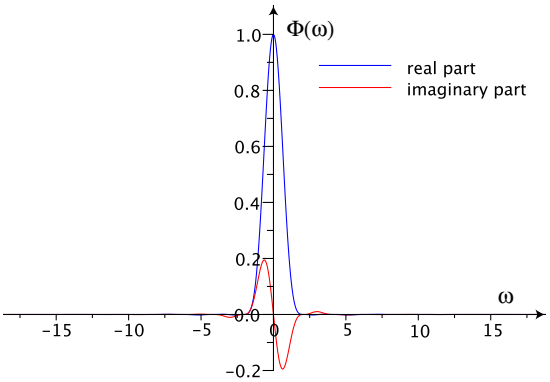


Abbildung 2.27: Ergebnis $\Phi_1(\omega)$ des Grenzübergangs (2.17) für das durch (2.29) gegebene Filter (Realteil blau, Imaginärteil rot)

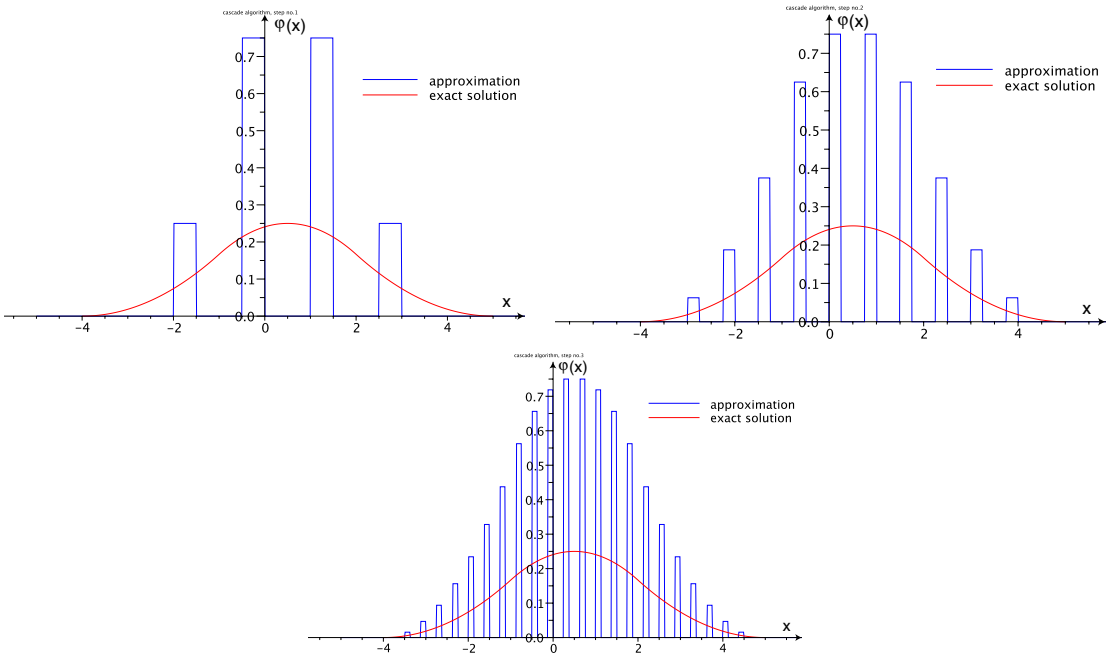


Abbildung 2.28: Erste Schritte des Kaskade-Algorithmus für das durch (2.29) gegebene Filter

(der Index bezieht sich immer auf das jeweilige Filter von (2.28) bzw. (2.29)). Der Kaskade-Algorithmus konvergiert jedoch nicht für das Filter $H_0(z)$. Hier wachsen die Funktionswerte nicht unbegrenzt an, sondern es entstehen immer schmalere Rechtecke, die auf einem Drittel des Intervalls das Dreifache des Funktionswerts approximieren. Dies ist in Abb. 2.28 gezeigt.

2.5 Biorthogonale Wavelets

2.5.1 Ausgangspunkt: geeignete Filter

Schauen wir uns nochmal eine Zwei-Kanal-Filterbank wie in Abb. 2.18 in der Wavelet-Notation an. Die Bedingungen der *perfekten Rekonstruktion* (1.46) und (1.47) lauten in der neuen Schreibweise

$$H(z)\tilde{H}(-z^{-1}) + G(z)\tilde{G}(-z^{-1}) = 0 \tag{2.30}$$
$$H(z)\tilde{H}(z^{-1}) + G(z)\tilde{G}(z^{-1}) = 2 \tag{2.31}$$

Ein Vorteil dieser neuen Schreibweise ist hier sofort sichtbar. Da diese Bedingungen für alle $z \neq 0$ gelten, kann man in der ersten Gleichung z durch $-z^{-1}$ und in der zweiten Gleichung z durch z^{-1} ersetzen und erhält

$$\tilde{H}(z)H(-z^{-1}) + \tilde{G}(z)G(-z^{-1}) = 0 \quad \text{und} \quad \tilde{H}(z)H(z^{-1}) + \tilde{G}(z)G(z^{-1}) = 2$$

Das sind die Bedingungen der perfekten Rekonstruktion, wobei Analyse- und Rekonstruktionsfilter vertauscht sind, also man kann $H(z)$ durch $\tilde{H}(z)$ und $G(z)$ durch $\tilde{G}(z)$ ersetzen, und die perfekte Rekonstruktion bleibt erhalten. Diese Symmetrieeigenschaft kann auch für die Impulsantwort formuliert werden: In einer Filterbank können Analyse- und Rekonstruktionsfilter vertauscht werden, wenn man bei allen Filtern in der Impulsantwort die Reihenfolge umkehrt.

k	$256\sqrt{2}\tilde{h}(k)$	$4\sqrt{2}\tilde{g}(k)$	$4\sqrt{2}h(k)$	$256\sqrt{2}g(k)$
−5	−5	0	0	−5
−4	15	0	0	−15
−3	19	0	0	19
−2	−97	0	0	97
−1	−26	1	1	−26
0	350	−3	3	−350
1	350	3	3	350
2	−26	−1	1	26
3	−97	0	0	−97
4	19	0	0	−19
5	15	0	0	15
6	−5	0	0	5

Tabelle 2.2: Beispiel für Filter, die der Bedingung der perfekten Rekonstruktion genügen (in der Literatur mit CDF(3,5) abgekürzt)

Beschränkt man sich auf FIR-Filter, so folgt aus den Bedingungen der perfekten Rekonstruktion — mit einer längeren Rechnung, die in Abschnitt 2.8 durchgeführt wird — dass zwei der vier Filter durch die andern beiden bis auf eine Konstante und eine Verschiebung gegeben sind. Hier sind die Möglichkeiten angegeben, zwei Filter durch die beiden andern auszudrücken:

$$H(z) = -cz^m\tilde{G}(-z^{-1}), \qquad G(z) = cz^m\tilde{H}(-z^{-1}) \tag{2.32}$$
$$\tilde{H}(z) = -\frac{1}{c}z^m G(-z^{-1}), \qquad \tilde{G}(z) = \frac{1}{c}z^m H(-z^{-1}) \tag{2.33}$$

dabei ist $c \neq 0$ und $m = 2k+1 \in \mathbb{Z}$ ungerade.

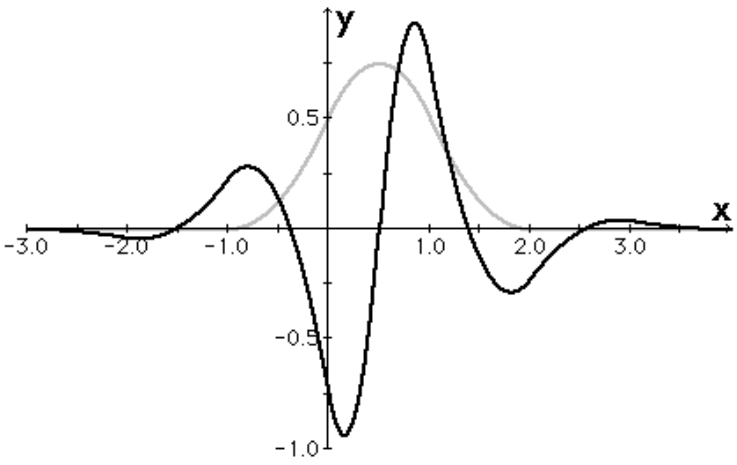


Abbildung 2.29: Skalierungsfunktion $\varphi(x)$ (grau) und Wavelet $\psi(x)$ (schwarz), die zu den Rekonstruktionsfiltern der Tabelle 2.2 gehören

Berechnet man beispielsweise die Hochpassfilter aus den Tiefpassfiltern, so hat man

$$g(k) = c \cdot (-1)^k \tilde{h}(-k+m) \tag{2.34}$$

$$\tilde{g}(k) = \frac{1}{c} \cdot (-1)^k h(-k+m) \tag{2.35}$$

Eine häufig vorgenommene Konvention ist $c = 1$ und $m = -1$, wir werden uns hier meistens an diese Konvention halten.

Ein Beispiel für Filterpaare, die den Bedingungen (2.30) und (2.31) genügen, ist in der Tabelle 2.2, die zu den Rekonstruktionsfiltern gehörende Skalierungsfunktion und das Wavelet sind in Abb. 2.29 gezeigt. Beide Funktionen sind stückweise aus quadratischen Polynomen zusammengesetzt und an den Übergangsstellen stetig differenzierbar. Dieses Wavelet wurde übrigens für Abb. 1.1 ausgewählt, seine Konstruktion aus der Skalierungsfunktion ist in Abb. 2.22 (rechts) gezeigt, die Skalierungsfunktion wurde zur Illustration der Zwei-Skalen-Relation in Abb. 2.20 dargestellt. Die hier nicht gezeigten Funktionen zu den Analyse-Filtern sind weniger regulär. Es ist darauf hinzuweisen, dass das Beispiel von Tabelle 2.2 zu *linearem* Phasenverhalten führt. Dies ist auch an der Symmetrie bzw. der Antisymmetrie (Punktsymmetrie) der zugehörigen Funktionen ersichtlich.

Die Transformationen der Filterbank von Abb. 2.18 können auch im Zeit- bzw. Ortsbereich ausgedrückt werden:

Man erhält für die Analyse

$$s_{m+1}(n) = \sum_{k=-\infty}^{+\infty} \tilde{h}(k-2n) s_m(k) = \sum_{k=-\infty}^{+\infty} \tilde{h}(k) s_m(k+2n) \tag{2.36}$$

$$d_{m+1}(n) = \sum_{k=-\infty}^{+\infty} \tilde{g}(k-2n) s_m(k) = \sum_{k=-\infty}^{+\infty} \tilde{g}(k) s_m(k+2n) \tag{2.37}$$

und für die Rekonstruktion

$$s_m(n) = \sum_{k=-\infty}^{+\infty} h(n-2k)s_{m+1}(k) + \sum_{k=-\infty}^{+\infty} g(n-2k)d_{m+1}(k) \tag{2.38}$$

Beim Vergleich mit Abschnitt 1.1 (siehe insbesondere (1.3)) ist zu beachten, dass hier die Reihenfolge der Numerierung des Skalenparameters m der Koeffizienten $d_m(k)$ umgekehrt ist ($d_m(k)$ enstpricht $d_{-m}(k)$ des Abschnitts 1.1). Dies erfolgte, um hier negative Indizes zu vermeiden. Weiterhin tritt – im Gegensatz zum Abschnitt 1.1 — hier die Skalierungsfunktion ins Spiel, da wir nicht beliebig feine Skalierungen zulassen (die feinste Skalierung entspricht $m = 0$). Dies berücksichtigt, dass eine Abtastung automatisch eine maximale Frequenz voraussetzt (die Abtastfrequenz ist hier stets zu 1 normiert).

Die Frage, woher man geeignete Filter bekommt, wird auf Abschnitt 2.8 verschoben. Häufig verwandte Filter sind im Anhang A aufgeführt. Es ist sinnvoll, sich zu überzeugen, dass der in Abschnitt 2.4 besprochene Grenzübergang eine stetige, möglichst auch differenzierbare Skalierungsfunktion liefert. Dabei ist es wichtig, für die Synthese (Rekonstruktion) das Filter mit der regulärereren Skalierungsfunktion auszuwählen. Für die Analyse kann man gegebenenfalls sogar auf die Existenz des entsprechenden Grenzwerts verzichten.

2.5.2 Eigenschaften der aus den Filtern konstruierten Funktionen

Es sollte nochmals betont werden, dass in der Praxis nur mit den Filterkonstanten gerechnet wird, also mit den durch (2.36) und (2.37) gegebenen Transformationen. Die praktische Bedeutung der zugehörigen Funktionen liegt ausschließlich in der qualitativen Form des Einflusses von Fehlern bei der Rekonstruktion, wie dies in Abschnitt 2.1 besprochen wurde.

Für ein tieferes Verständnis der Wavelet-Transformation ist die Beschäftigung mit Skalierungsfunktion und Wavelet als Funktionen jedoch sinnvoll. Wir wollen hier einsehen, warum die Filterbank-Transformationen der Gleichungen (2.36) und (2.37) tatsächlich — zumindest näherungsweise — die Koeffizienten einer Entwicklung des Signals mit Hilfe von skalierten und verschobenen Versionen des Wavelets berechnet. So war dies in der Einleitung im Abschnitt 1.1 behauptet worden.

Wir gehen also davon aus, dass wir Filterpaare $H(z)$, $G(z)$, $\tilde{H}(z)$, $\tilde{H}(z)$ ausgewählt haben, die den Bedingungen der perfekten Rekonstruktion (2.30) und (2.31) genügen und bei denen sowohl für die Rekonstruktion als auch für die Analyse (d.h. auch für $\tilde{H}(e^{j\omega})$) der Grenzwert (2.17) existiert und mit Hilfe von (2.18) zu einer stetigen Skalierungsfunktion führt.

Unter geeigneten technischen Bedingungen an die Filter (siehe beispielsweise [10, Theorem 8.3.1]), die von den meisten in der Praxis verwandten Filtern erfüllt sind, kann man dann garantieren, dass verschobene und skalierte Versionen der Skalierungsfunktionen $\varphi(x)$ und $\tilde{\varphi}(x)$ und der Wavelets $\psi(x)$ und $\tilde{\psi}(x)$ zueinander „biorthogonal“ sind, d.h. die

folgende Bedingungen für alle ganzen Zahlen m, n erfüllen:

$$\int_{-\infty}^{+\infty} \tilde{\varphi}(x-m)\varphi(x-n)dx = \begin{cases} 1 & \text{falls } m=n \\ 0 & \text{sonst} \end{cases} \tag{2.39}$$

$$\int_{-\infty}^{+\infty} \tilde{\psi}(x-m)\psi(x-n)dx = \begin{cases} 1 & \text{falls } m=n \\ 0 & \text{sonst} \end{cases} \tag{2.40}$$

$$\int_{-\infty}^{+\infty} \tilde{\varphi}(x-m)\psi(x-n)dx = 0 \tag{2.41}$$

$$\int_{-\infty}^{+\infty} \varphi(x-m)\tilde{\psi}(x-n)dx = 0 \tag{2.42}$$

Dies bedeutet, dass die Funktionen jeweils zu den entsprechenden Partnern der andern Familie orthogonal sind.

Wir wollen hier in diesem Abschnitt annehmen, dass die entsprechenden Bedingungen an die Filter erfüllt sind und diese Beziehungen tatsächlich gelten.

An dieser Stelle ist zum Vergleich der Hinweis hilfreich, dass für die Berechnung der Koeffizienten einer Fourier-Reihe die folgenden, für $\omega = \frac{2\pi}{T}$ und für alle $m, n \in \mathbb{N}$ gültigen Orthogonalitätsbeziehungen

$$\int_0^T \sin(m\omega t)\cos(n\omega t)dt = 0$$

$$\int_0^T \sin(m\omega t)\sin(n\omega t)dt = \begin{cases} \frac{1}{2}T & \text{falls } m=n \\ 0 & \text{falls } m \neq n \end{cases}$$

$$\int_0^T \cos(m\omega t)\cos(n\omega t)dt = \begin{cases} \frac{1}{2}T & \text{falls } m=n \\ 0 & \text{falls } m \neq n \end{cases}$$

wichtig sind.

Hier werden statt der trigonometrischen Funktionen skalierte und verschobene Versionen des Mutter-Wavelets verwendet. Es ist sinnvoll, auch entsprechende Versionen der Skalierungsfunktion hinzuzunehmen, d.h. wir werden folgende Funktionen als „Bausteine“ verwenden (vgl auch (2.4) und (2.6)):

$$\psi_{m,n}(x) := 2^{\frac{m}{2}}\psi(2^m x - n), \qquad m, n \in \mathbb{Z} \tag{2.43}$$

$$\varphi_{m,n}(x) := 2^{\frac{m}{2}}\varphi(2^m x - n), \qquad m, n \in \mathbb{Z} \tag{2.44}$$

und analog für die Analyse-Funktionen $\tilde{\psi}(x)$ und $\tilde{\varphi}(x)$.

2.5.3 Kontinuierliches Signal und Abtastwerte

Durch Linearkombination der durch (2.43) und (2.44) gegebenen Funktionen erhält man stets ein *kontinuierliches* Signal. In der Praxis hat man jedoch in der Regel nur *Abtastwerte*

$y(k)$ des Signals zur Verfügung. Für das Beispiel des Haar-Wavelets haben wir daraus mit Hilfe der Skalierungsfunktion durch (2.9) wieder ein kontinuierliches Signal gemacht. Entsprechend gehen wir hier davon aus, dass unsere Skalierungsfunktion durch

$$f_0(x) = \sum_{k=-\infty}^{+\infty} y(k)\varphi(x-k) \tag{2.45}$$

eine gute Näherung an das unbekannte ursprüngliche Signal $f(x)$ liefert, also $f(x) \approx f_0(x)$. (Zu beachten ist, dass bei Verwendung von FIR-Filtern die Skalierungsfunktion außerhalb eines Intervalls endlicher Länge verschwindet, so dass die formal unendliche Summe für jedes x tatsächlich nur aus endlich vielen von Null verschiedenen Summanden besteht.) Wir identifizieren also die Abtastwerte $y(k)$ mit den Entwicklungskoeffizienten der Skalierungsfunktion und bezeichnen diese hier mit $s_0(k)$. Wir können sie mit Hilfe der Analyse-Skalierungsfunktion wieder aus dem kontinuierlichen Signal zurückgewinnen:

$$y(k) = s_0(k) := \int_{-\infty}^{+\infty} \tilde{\varphi}(x-k) f_0(x) dx \tag{2.46}$$

Diese hier vorgenommene Identifikation der Koeffizienten $s_0(k)$ mit den Abtastwerten $y(k)$ ist allerdings nicht ganz unumstritten, sie wird in [31, Kapitel 7.1] sogar als „wavelet crime“ kritisiert. In der Tat ist im allgemeinen für beliebige Funktionen $f(x)$

$$f(k) \neq \int_{-\infty}^{+\infty} \tilde{\varphi}(x-k) f(x) dx$$

Wenn sich jedoch $f(x)$ in dem Intervall, in dem $\tilde{\varphi}(x)$ von Null verschieden ist (bei FIR-Filtern ein endliches Intervall) nur wenig ändert, dann ist die hier vorgenommene Näherung nicht so furchtbar schlecht. Man beachte, dass wir hier die Abtastfrequenz (bzw. den Pixelabstand bei Bildern) durch (2.45) zu 1 normiert haben! Durch eine entsprechend hohe Abtastfrequenz bzw. Auflösung können wir also häufig erreichen, dass die Näherung $f(x) \approx f_0(x)$ einigermaßen akzeptabel ist. Andere Abhilfemöglichkeiten sind in [31, Kapitel 7.1] angegeben.

2.5.4 Übereinstimmung von Waveletkoeffizienten und Ergebnis der Filterbanktransformationen

Der zentrale Punkt dieses Abschnitts ist es, zu zeigen, dass durch die Filterbankoperationen tatsächlich die Koeffizienten einer Waveletentwicklung berechnet werden. Für die hier vorgenommenen Betrachtungen sind die Zwei-Skalen-Relation (2.20) sowie die Identität (2.27) von fundamentaler Bedeutung (sie gelten auch für die Analyse-Funktionen mit den entsprechenden Filterkonstanten). Ersetzt man die Variable x durch $\frac{1}{2}x - n$, so erhält man

$$\varphi(\tfrac{1}{2}x - n) = \sqrt{2} \sum_{k=-\infty}^{+\infty} h(k)\varphi(x-k-2n) \tag{2.47}$$

und hieraus nach Division durch $\sqrt{2}$ in der Schreibweise von (2.44)

$$\varphi_{-1,n}(x) = \sum_{k=-\infty}^{+\infty} h(k)\varphi_{0,k+2n}(x) \tag{2.48}$$

Analog erhält man für die Wavelets aus (2.27)

$$\psi_{-1,n}(x) = \sum_{k=-\infty}^{+\infty} g(k)\varphi_{0,k+2n}(x) \tag{2.49}$$

Dies bedeutet, dass man Skalierungsfunktionen und Wavelets zur Skalierungsstufe -1 (gröbere Auflösung) als Summe von Skalierungsfunktionen zur Skalierungsstufe 0 („normale“ Auflösung) darstellen kann. Für die Analysefunktionen erhält man analoge Identitäten.

Wir wollen nun das ursprüngliche Signal $f(x)$ durch eine gröbere Approximation $f_1(x)$ als in (2.45) annähern, indem wir Skalierungsfunktionen der Skalierungsstufe -1 benutzen:

$$f_1(x) := \sum_{k=-\infty}^{+\infty} s_1(k)\varphi_{-1,k}(x) \approx f(x) \tag{2.50}$$

Aufgrund der Biorthogonalitätsrelation (2.39) erhalten wir

$$s_1(n) = \int_{-\infty}^{+\infty} \tilde{\varphi}_{-1,n}(x) f(x) dx \tag{2.51}$$

Setzt man (2.48) und $f(x) = f_0(x)$ ein, so erhält man

$$\begin{aligned} s_1(n) &= \int_{-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \tilde{h}(k)\tilde{\varphi}_{0,k+2n}(x) f_0(x) dx \\ &= \sum_{k=-\infty}^{+\infty} \tilde{h}(k) \int_{-\infty}^{+\infty} \tilde{\varphi}_{0,k+2n}(x) f_0(x) dx = \sum_{k=-\infty}^{+\infty} \tilde{h}(k-2n) s_0(k) \end{aligned} \tag{2.52}$$

Wir erhalten also die Koeffizienten der gröberen Näherung als Ergebnis einer Tiefpassfilterung mit anschließender Unterabtastung.

Wir versuchen nun, die Differenz zwischen der gröberen Näherung (Skalierungsstufe -1) und der „Standard“- Näherung (2.45) durch verschobene Versionen der Wavelets zur Auflösung -1 darzustellen, d.h. wir betrachten den Ansatz

$$f_0(x) = f_1(x) + \sum_{k=-\infty}^{+\infty} d_1(k)\psi_{-1,k}(x) \tag{2.53}$$

Wir verzichten hier auf den formalen Beweis, dass ein derartiger Ansatz möglich ist, d.h. die Differenz $f_0(x) - f_1(x)$ tatsächlich exakt durch eine Summe von Funktionen $\psi_{-1,k}(x)$ darstellbar ist. Im Beispiel des Haar-Wavelets war eine derartige Darstellung unmittelbar einzusehen (siehe (2.11)). Die Koeffizienten $d_1(k)$ ergeben sich aus den Biorthogonalitätsrelationen (2.39) bis(2.42): Durch eine Substitution kann man nämlich einsehen, dass entsprechende Biorthogonalitätsrelationen auch für die Skalierungsstufe -1 gelten. Multipliziert man (2.53) mit $\tilde{\psi}_{-1,n}(x)$ und integriert, so erhält man

$$d_1(n) = \int_{-\infty}^{+\infty} \tilde{\psi}_{-1,n}(x) f_0(x) dx \tag{2.54}$$

Unter Benutzung der Identität (2.49) für das Analysewavelet erhalten wir daraus

$$\begin{aligned} d_1(n) &= \int_{-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \tilde{g}(k)\tilde{\varphi}_{0,k+2n}(x) f_0(x) dx \\ &= \sum_{k=-\infty}^{+\infty} \tilde{g}(k) \int_{-\infty}^{+\infty} \tilde{\varphi}_{0,k+2n}(x) f_0(x) dx = \sum_{k=-\infty}^{+\infty} \tilde{g}(k-2n) s_0(k) \end{aligned} \tag{2.55}$$

Als Ergebnis erhalten wir, dass die Filterbank-Operationen tatsächlich die Koeffizienten der Wavelets (als Hochpass-Ausgang) und die Koeffizienten der Skalierungsfunktion (als Tiefpass-Ausgang) liefern. In der Tat haben wir in (2.52) und (2.55) gerade den Spezialfall von (2.36) und (2.37) für $m = 1$ erhalten. Eine entsprechende Übereinstimmung für $m = 2$ erhalten wir, indem wir die Vorgehensweise fortsetzen und eine noch gröbere Näherung des ursprünglichen Signals vornehmen:

$$f_2(x) := \sum_{k=-\infty}^{+\infty} s_2(k)\varphi_{-2,k}(x) \approx f(x) \tag{2.56}$$

Wir können dann die ganze Rechnung analog wiederholen. Entscheidend ist dabei, dass man durch Ersetzen von x durch $\frac{1}{2}x - n$ in (2.48) und (2.49) entsprechende Darstellungen von $\varphi_{-2,k}(x)$ und $\psi_{-2,k}(x)$ durch $\varphi_{-1,k}(x)$ erhält. Wiederholen wir diese Vorgehensweise M mal, so erhalten wir als Ergebnis folgende Darstellung unserer Näherung $f_0(x)$ des kontinuierlichen Signals $f(x)$:

$$f_0(x) = \sum_{k=-\infty}^{+\infty} y(k)\varphi(x-k) = \sum_{i=1}^{M} \sum_{k=-\infty}^{+\infty} d_i(k)\psi_{-i,k}(x) + \sum_{k=-\infty}^{+\infty} s_M(k)\varphi_{-M,k}(x) \tag{2.57}$$

mit

$$s_i(k) = \int_{-\infty}^{+\infty} \tilde{\varphi}_{-i,k}(x) f_0(x) dx \tag{2.58}$$

$$d_i(k) = \int_{-\infty}^{+\infty} \tilde{\psi}_{-i,k}(x) f_0(x) dx \tag{2.59}$$

Die durch die Filterbankoperationen berechneten Zahlen treten also in der Tat als Koeffizienten einer Reihenentwicklung des Signals auf — zumindest näherungsweise. Bei dieser Reihenentwicklung werden — wie dies in der Einführung (Abschnitt 1.1) behauptet wurde — verschobene und skalierte Versionen des Mutterwavelets $\psi(x)$ als „Bausteine“ benutzt. Die Filterbankoperationen liefern also tatsächlich Waveletkoeffizienten.

Wir sind hier von einer „feinsten Auflösung“ $f_0(x)$ ausgegangen, und das macht bei einem Bezug zu abgetasteten Signalen Sinn. Weiterhin haben wir die Zerlegung bei einer „gröbsten Auflösung“, die der Skalierungsstufe M entspricht, beendet. Dies wurde durch

den Gebrauch der Skalierungsfunktion ermöglicht. Daher unterscheidet sich die Reihe (2.57) von der Reihe (1.3) in Abschnitt 1.1. Die dortige Reihe enthält beliebig feine und beliebig grobe Skalierungen (und damit beliebig hohe Frequenzen). Dies kann man als Grenzwert zulassen, für die Praxis ist jedoch eine Reihe der in diesem Abschnitt betrachteten Form angemessener.

Die Koeffizienten erfüllen mit (2.58) und (2.59) ganz ähnliche Identitäten wie die Koeffizienten einer Fourier-Reihe. Es sollte nochmal ausdrücklich betont werden, dass sie in der Praxis *nicht* durch diese Integrale, sondern die Filterbank-Operationen (2.36) und (2.37) ausgerechnet werden. Gleichung (2.57) macht jedoch nochmal die Wichtigkeit der Regularität der Synthese-Funktionen $\varphi(x)$ und $\psi(x)$ deutlich: Fehler bei den Koeffizienten (die beispielsweise bei der Datenkompression für eine hohe Kompressionsrate unvermeidlich sind) führen bei unstetigen Synthese-Funktionen zu unerwünschten Sprüngen bzw. Kanten.

Anmerkungen:

(a) Hochpassfilter, die zu sinnvollen Wavelets führen, sollten $G(1) = 0$ erfüllen (siehe Abschnitt 2.7). Verlangt man dasselbe auch für das Analyse-Wavelet, so folgt aus (2.31) die Bedingung $H(1)\tilde{H}(1) = 2$. Eine symmetrische Normierung erreicht man durch die Forderung $H(1) = \tilde{H}(1) = \sqrt{2}$. Diese Normierung ist in Abschnitt 2.4 vorausgesetzt. Zuweilen werden auch andere Normierungen, z.B. $H(1) = 1$, $\tilde{H}(1) = 2$ benutzt (z.B. in Abb. 1.4). Also Vorsicht beim Übernehmen von Filterkonstanten und Formeln aus unterschiedlichen Büchern! Eine andere Normierung ist in einem andern Vorfaktor in der Zwei-Skalen-Relation (2.20) sichtbar. Auch hier ist in Kapitel 5 eine andere Normierung vorgenommen.

(b) Die Notation ist leider überhaupt nicht einheitlich. In einigen Büchern und Zeitschriftenartikeln haben die Rekonstruktionsfilter statt der Analysefilter eine Tilde. Die Zeitumkehr, die hier in der Änderung der Bezeichnungsweise von $H(z)$ zu $\tilde{H}(z^{-1})$ steckt, wird zuweilen von einer andern Notation „versteckt“.

(c) Die mathematischen Spielregeln, die hinter dem in diesem Abschnitt vorgenommenen Hin- und Herspielen zwischen verschiedenen Skalierungsstufen stecken, werden unter dem Stichwort „Multiresolutionsanalysis“ zusammengefaßt. Sie sind in jedem mathematisch anspruchsvollen Lehrbuch über Wavelets (beispielsweise [10]) ausführlich besprochen.

2.6 Orthogonale Wavelets

Wie aus Tabelle 1.2 ersichtlich ist, hat das hier mit $D4$ abgekürzte Filterbeispiel die angenehme Eigenschaft, daß die Filter für die Analyse mit denen für die Rekonstruktion übereinstimmen — bis auf die Reihenfolge. D.h. die Rekonstruktionsfilter entstehen aus den Analysefiltern durch Umkehrung der Reihenfolge. Dieser Sachverhalt drückt sich in der in Tabelle 2.1 eingeführten neuen Schreibweise ganz einfach aus durch

$$\tilde{H}(z) = H(z), \qquad \tilde{G}(z) = G(z) \tag{2.60}$$

Derartige Filter und die zugehörigen Wavelets und Skalierungsfunktionen wollen wir in diesem Abschnitt betrachten. Die Bedingungen der perfekten Rekonstruktion (1.46) und

(1.47) lauten im hier betrachteten Spezialfall:

$$H(z)H(-z^{-1}) + G(z)G(-z^{-1}) = 0 \tag{2.61}$$
$$H(z)H(z^{-1}) + G(z)G(z^{-1}) = 2 \tag{2.62}$$

Die Gleichungen (2.32) und (2.33) sind nur möglich, wenn $c = \frac{1}{c}$, und dies ist nur für $c = \pm 1$ erfüllt. Also haben wir hier

$$G(z) = \pm z^m H(-z^{-1}) \qquad \text{mit ungeradem } m \in \mathbb{Z} \tag{2.63}$$

Für die Filterkonstanten heißt dies

$$g(k) = \mp(-1)^k h(-k-m), \qquad m \in \mathbb{Z} \text{ ungerade} \tag{2.64}$$

Es genügt also, das Tiefpassfilter zu kennen. Im bisher betrachteten Beispiel $D4$ ist

$$G(z) = -z^{-3} H(-z^{-1})$$

Wir gehen — wie im Abschnitt 2.5 — hier davon aus, dass die Filter so gewählt sind, dass der Grenzwert (2.17) existiert und zu einer stetigen Skalierungsfunktion führt. Man beachte, dass wir hier nur eine Sorte von Funktionen erhalten, d.h. Analyse- und Synthesefunktionen stimmen überein. Unter weiteren technischen Bedingungen an die Filter, die wir hier als erfüllt annehmen, erhält man eine Skalierungsfunktion und ein Wavelet, das *orthogonal* zu seinen verschoben Versionen ist. Es gelten dann die Beziehungen

$$\int_{-\infty}^{+\infty} \varphi(x-m)\varphi(x-n)dx = \begin{cases} 1 & \text{falls } m = n \\ 0 & \text{sonst} \end{cases} \tag{2.65}$$

$$\int_{-\infty}^{+\infty} \psi(x-m)\psi(x-n)dx = \begin{cases} 1 & \text{falls m=n} \\ 0 & \text{sonst} \end{cases} \tag{2.66}$$

$$\int_{-\infty}^{+\infty} \varphi(x-m)\psi(x-n)dx = 0 \tag{2.67}$$

gilt. Diese Wavelets heißen daher *orthogonale* Wavelets.

Ingrid Daubechies hat zum ersten Mal eine Serie solcher Filter entwickelt, die zu orthogonalen Wavelets führen. Sie werden hier mit Dn abgekürzt, wobei n eine gerade natürliche Zahl ist und die Zahl der nichtverschwindenden Filterkoeffizienten angibt. $n = 2$ führt auf das Haar-Wavelet, und $n = 4$ wurde bisher als einziges weiteres Beispiel behandelt. Die zugehörigen Filter haben die Länge n, d.h. nur n Konstanten sind von Null verschieden. Für die Skalierungsfunktion und das Wavelet heißt dies, dass beide Funktionen außerhalb eines Intervalls endlicher Länge Null sind. Konkret hat die Skalierungsfunktion nur innerhalb des Intervall $[0, n-1]$ nicht verschwindende Werte. Beispiele für die entsprechenden Wavelets sind in Abb. 2.30 gezeigt, dabei ist sichtbar, dass die „Regularität“ der Funktionen mit der Filterlänge wächst. Zu betonen ist nochmal, daß in der Praxis nicht die Funktionen, sondern nur die Filterkonstanten verwandt werden.

Das Frequenzverhalten der Filter wird mit zunehmender Filterlänge besser und nähert sich dem eines idealen Tief- bzw. Hochpasses an. Dies ist in Abb. 2.31 für das Filter $D30$

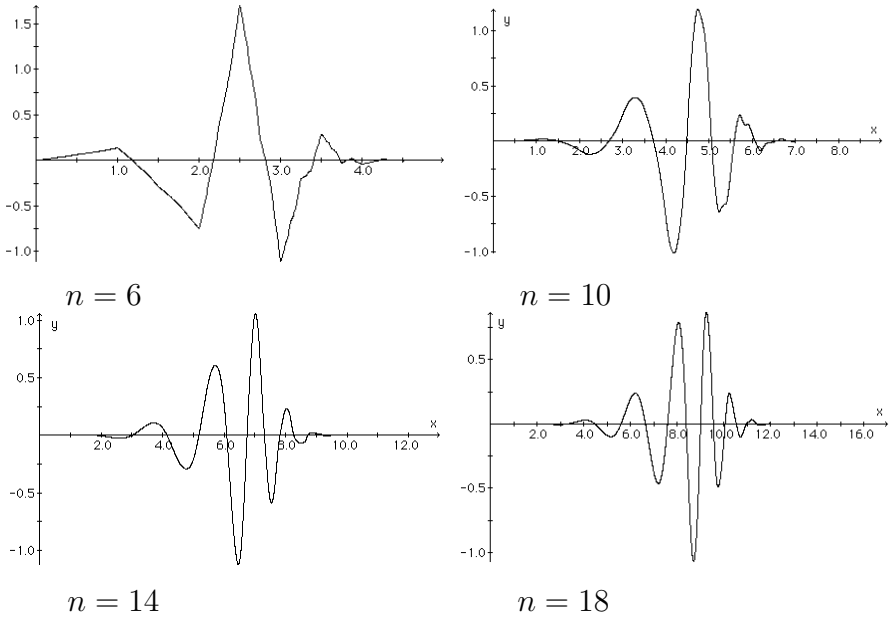


Abbildung 2.30: Daubechies-Wavelets Dn für verschiedene Filterlängen n

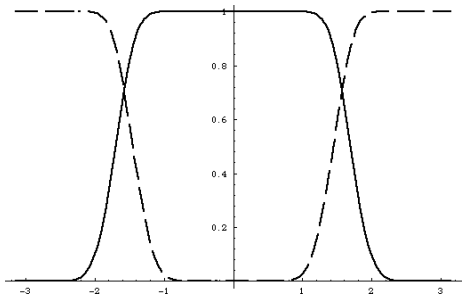


Abbildung 2.31: Absolutbetrag des Frequenzgangs (auf 1 normiert) für das Tiefpassfilter $|H(e^{j\omega})|$ (durchgezogen) sowie für das Hochpassfilter $|G(e^{j\omega})|$ (gestrichelt) zum Daubechies-Wavelet $D30$

deutlich gemacht (zu vergleichen mit Abb. 1.8 und Abb. 1.9). Zu beachten ist jedoch, dass auch die Rechenzeit mit der Filterlänge anwächst.

Ein wesentlicher Nachteil ist, dass alle Filter dieser Reihe — außer dem für das Haar-Wavelet — kein *lineares* Phasenverhalten haben. Dies ist mit der Bedingung (2.60) und einem Filter endlicher Länge nur mit dem Haar-Wavelet möglich. Daher werden in der Praxis häufig die allgemeineren, biorthogonalen Wavelets bevorzugt.

2.7 Regularitätsanforderungen an Filter und Wavelets

Wir sind bisher davon ausgegangen, dass die Filter $H(z)$ und $\tilde{H}(z)$ Tiefpassfilter sind. Als Mindestforderung ergibt sich daher, dass

$$H(-1) = 0 \qquad \text{und} \qquad \tilde{H}(-1) = 0 \tag{2.68}$$

erfüllt sein soll, denn $\omega = \pi$ und damit $z = e^{j\pi} = -1$ entspricht der höchsten möglichen Frequenz bei einer zu 1 normierten Abtastfrequenz. Da wir hier nur FIR-Filter betrachten, $H(z)$ und $\tilde{H}(z)$ also Polynome sind, ergibt sich sofort eine Faktorisierung

$$H(z) = (z+1)R(z) \tag{2.69}$$

mit einem LAURENT-Polynom $R(z)$. Eine analoge Faktorisierung ist für $\tilde{H}(z)$ mit einem $\tilde{R}(z)$ möglich. Aufgrund von (2.32) und (2.33) erhalten wir

$$\tilde{H}(-1) = 0 \implies G(1) = 0 \qquad \text{und} \qquad H(-1) = 0 \implies \tilde{G}(1) = 0 \tag{2.70}$$

Dies ermöglicht, $G(z)$ und $\tilde{G}(z)$ als Hochpassfilter anzusehen. Weiterhin erhalten wir damit aus der Bedingung der perfekten Rekonstruktion (2.31) für $z = 1$

$$H(1)\tilde{H}(1) + G(1)\tilde{G}(1) = H(1)\tilde{H}(1) = 2$$

Wir werden daher meist die Normierung

$$H(1) = \tilde{H}(1) = \sqrt{2} \tag{2.71}$$

wählen.

Es zeigt sich, dass es sinnvoll ist, Filter zu wählen, bei denen $z = -1$ eine *mehrfache* Nullstelle von $H(z)$ und $\tilde{H}(z)$ ist, die also

$$H(z) = (z+1)^p R(z) \qquad \text{und} \qquad \tilde{H}(z) = (z+1)^{\tilde{p}} \tilde{R}(z) \tag{2.72}$$

mit $p > 1$ und $\tilde{p} > 1$ erfüllen. Ein größeres p führt zu einer größeren Regularität von Skalierungsfunktion und Wavelet. Im Spezialfall $H(z) = (z+1)^p$ ist die Skalierungsfunktion gerade $(p-2)$ mal stetig differenzierbar. Im allgemeinen ist dieser Zusammenhang jedoch komplizierter und die Zahl der existierenden Ableitungen erheblich kleiner als $p-2$. Im Fall der orthogonalen Daubechies-Wavelets Dn benötigt man für die Existenz von m Ableitungen — asymptotisch für große m — etwa $p \approx 5m$ und damit $n \approx 10m$ nichtverschwindende Filterkoeffizienten. Es war bereits darauf hingewiesen worden, dass die Regularität für die Synthesefunktionen (Skalierungsfunktion und Wavelet) wichtiger ist als für die Analysefunktionen. Man wird also in der Regel die Filter so wählen, dass $p \geq \tilde{p}$.

Weil $H(z) = \sum_{k=-\infty}^{+\infty} h(k) z^{-k}$, bedeutet unsere Normierung (2.71) im Zeitbereich, dass

$$\sum_{k=-\infty}^{+\infty} h(k) = \sqrt{2} \tag{2.73}$$

Für $\omega = 0$ folgt dann aus (2.17), weil $\Phi(\omega) = \int_{-\infty}^{+\infty} \varphi(x) e^{-j\omega x} dx$

$$\Phi(0) = \int_{-\infty}^{+\infty} \varphi(x) dx = 1 \tag{2.74}$$

Für die Erläuterung von hinreichenden Bedingungen an die Filter für die Existenz des Grenzwerts (2.17) wird auf [31, 7.2] verwiesen.

Aus (2.24) und $G(1) = 0$ folgt für $\omega = 0$, dass $\Psi(0) = 0$. Da $\Psi(\omega) = \int_{-\infty}^{+\infty} \psi(x) e^{-j\omega x} dx$, heißt dies

$$\int_{-\infty}^{+\infty} \psi(x) dx = 0 \tag{2.75}$$

was im allgemeinen als eine Mindestanforderung an eine Funktion angesehen wird, dass sie als Wavelet dient.

Aufgrund von (2.32) und (2.33) hat eine p-fache Nullstelle von $H(z)$ in $z = -1$ eine p-fache Nullstelle von $\tilde{G}(z)$ in $z = 1$ zur Folge. Entsprechend verursacht eine $\tilde{p}$-fache Nullstelle von $\tilde{H}(z)$ in $z = -1$ eine Nullstelle derselben Vielfachheit von $G(z)$ in $z = 1$. Man beachte, dass hier das Analyse-Tiefpassfilter mit dem Synthese-Hochpassfilter und umgekehrt „gekoppelt“ ist.

Eine $\tilde{p}$-fache Nullstelle von $G(z)$ in $z = 1$ hat aufgrund von (2.24) eine Nullstelle derselben Vielfachheit der Fouriertransformierten des Synthesewavelets $\Psi(\omega)$ in $\omega = 0$ zur Folge, d.h. es gilt für die Ableitungen

$$\Psi^{(k)}(\omega) = \frac{d^k \Psi}{d\omega^k}(\omega) = 0 \qquad \text{für } \omega = 0 \text{ und } k = 0, 1, 2, \ldots \tilde{p} - 1 \tag{2.76}$$

Andererseits gilt

$$\Psi^{(k)}(\omega) = \frac{d^k}{d\omega^k} \int_{-\infty}^{+\infty} \psi(x) e^{-j\omega x} dx = \int_{-\infty}^{+\infty} \psi(x) (-j)^k x^k e^{-j\omega x} dx \tag{2.77}$$

Damit folgt insgesamt

$$\int_{-\infty}^{+\infty} x^k \psi(x) dx = 0 \qquad \text{für } k = 0, 1, 2, \ldots \tilde{p} - 1 \tag{2.78}$$

Man sagt, das Wavelet hat $\tilde{p}$ *verschwindende Momente*. Entsprechend hat eine p-fache Nullstelle von $\tilde{G}(z)$ zur Folge, dass das Analysewavelet $\tilde{\psi}(x)$ p verschwindende Momente hat:

$$\int_{-\infty}^{+\infty} x^k \tilde{\psi}(x) dx = 0 \qquad \text{für } k = 0, 1, 2, \ldots p - 1 \tag{2.79}$$

Wir leiten nun Folgerungen aus der Regularitätsforderung (2.72) im Zeitbereich her. Wenn $H(z)$ eine p-fache Nullstelle in $z = -1$ hat, dann haben wir aufgrund von (2.33) mit der hier vorgenommenen Konvention $c = 1$ und $m = -1$

$$\tilde{G}(z) = z^{-1} H(-z^{-1}) = (-z^{-1} + 1)^p R(-z^{-1}) \tag{2.80}$$

Benutzen wir die Ableitungen

$$\tilde{G}^{(n)}(z) = \frac{d^n}{dz^n} \tilde{G}(z) \quad \text{mit} \quad \tilde{G}^{(0)}(z) = \tilde{G}(z)$$

dann heißt dies

$$\tilde{G}^{(n)}(z) = 0 \quad \text{für} \quad z = 1 \quad \text{und} \quad n = 0, 1, 2, 3 \ldots p - 1$$

Weil

$$\tilde{G}(z) = \sum_{k=-\infty}^{+\infty} \tilde{g}(k) z^{-k}$$

bedeutet die Mindestanforderung an unser Hochpassfilter $\tilde{G}(-1) = 0$, dass

$$\sum_{k=-\infty}^{+\infty} \tilde{g}(k) = 0$$

Wenn $p > 1$, haben wir

$$\tilde{G}'(z) = \sum_{k=-\infty}^{+\infty} (-k)\tilde{g}(k) z^{-k-1} \qquad \text{und daher} \qquad -\tilde{G}^{(1)}(1) = \sum_{k=-\infty}^{+\infty} k\tilde{g}(k) = 0$$

Analog erhalten wir

$$\tilde{G}^{(2)}(z) = \sum_{k=-\infty}^{+\infty} (-k)(-k-1)\tilde{g}(k) z^{-k-2} = \sum_{k=-\infty}^{+\infty} (k^2+k)\tilde{g}(k) z^{-k-2}$$

Wenn wir $z = 1$ setzen und das vorherige Ergebnis benutzen, bekommen wir für $p > 2$

$$\sum_{k=-\infty}^{+\infty} k^2 \tilde{g}(k) = 0$$

Wenn wir diese Vorgehensweise fortsetzen, erhalten wir das Ergebnis

$$\sum_{k=-\infty}^{+\infty} k^n \tilde{g}(k) = 0 \qquad \text{für} \quad n = 0, 1, 2, 3 \ldots p-1 \tag{2.81}$$

und aus dieser Summenregel folgt auch umgekehrt, dass $\tilde{G}(z)$ eine p-fache Nullstelle in $z = -1$ hat.

Für den Eingang $X(z)$ beim Analyse-Hochpassfilter erhalten wir den Ausgang $Y(z) = \tilde{G}(z^{-1})X(z)$ oder im „Zeit“-Bereich

$$y(k) = \sum_{l=-\infty}^{+\infty} \tilde{g}(-l)x(k-l) = \sum_{l=-\infty}^{+\infty} \tilde{g}(l)x(k+l)$$

Unterabtastung liefert für den Eingang $x(k) = s_0(k)$ den Hochpass-Analysis-Ausgang der Filterbank

$$d_1(k) = \sum_{l=-\infty}^{+\infty} \tilde{g}(l) s_0(l+2k)$$

Dies hat zur Folge, dass wir für alle Eingangssignale der Form

$$x(k) = s_0(k) = k^n, \qquad \text{mit} \quad n = 0, 1, 2, 3 \ldots p-1$$

das Ausgangssignal

$$d_1(k) = \sum_{l=-\infty}^{+\infty} \tilde{g}(l)(l+2k)^n = 0$$

erhalten (dabei wurde der binomische Lehrsatz benutzt).

Wir haben somit die für die Praxis (Datenkompression!) wichtige Folgerung bekommen:

> Wenn $H(z) = (z+1)^p R(z)$, dann haben alle Eingangssignale $x(k)$ von der Form eines Polynoms mit einem Grad, der kleiner als p ist, als Hochpass-Ausgang der Analysis-Filterbank **verschwindende** Koeffizienten $d(k)$.

Entsprechend ist die Summenregel

$$\sum_{k=-\infty}^{+\infty} k^n g(k) = 0 \qquad \text{für} \quad n = 0, 1, 2, 3 \ldots \tilde{p} - 1 \tag{2.82}$$

äquivalent dazu, dass $G(z)$ eine $\tilde{p}$-fache Nullstelle in $z = 1$ hat. Dies hat jedoch keine direkte praktische Bedeutung.

Wenden wir uns nun den Folgerungen aus der Regularität von $\tilde{H}(z)$ zu. Eine intuitiv einleuchtende Folgerung aus der Mindestforderung (2.68) ergibt sich im Zeitbereich:

$$\sum_{k=-\infty}^{+\infty} (-1)^k \tilde{h}(k) = 0 \tag{2.83}$$

Dies hat zur Konsequenz, dass bei Tiefpassfilterung und Unterabtastung für ein Signal der Form

$$\ldots, 1, -1, 1, -1, 1, -1, 1, -1, \ldots$$

als Ergebnis Null herauskommt. D.h. für $x(k) = s_0(k) = (-1)^k$ erhalten wir $s_1(k) = 0$. Ein solches Signal enthält nur die höchste mögliche Frequenz, und eine Tiefpassfilterung sollte in der Tat Null ergeben.

Wenn $\tilde{H}(z)$ eine $\tilde{p}$-fache Nullstelle in $z = -1$ hat, dann gilt für die Ableitungen

$$\tilde{H}^{(n)}(-1) = 0 \qquad \text{für} \quad n = 0, 1, 2, 3 \ldots \tilde{p} \qquad \text{wobei} \quad \tilde{H}^{(n)}(z) = \frac{d^n}{dz^n} \tilde{H}(z) \tag{2.84}$$

Wie üblich vereinbaren wir $\tilde{H}^{(0)}(z) = \tilde{H}(z)$. Im Zeitbereich erhält man daraus

$$\sum_{k=-\infty}^{+\infty} (-1)^k k^n \cdot \tilde{h}(k) = 0 \qquad \text{für} \quad n = 0, 1, 2, 3 \ldots \tilde{p} - 1 \tag{2.85}$$

Für das Eingangssignal $x(k) = s_0(k) = (-1)^k \cdot k^n$ erhält man also $s_1(k) = 0$, falls $n < \tilde{p}$ (dies folgt aus einer kleinen Rechnung mit Hilfe des binomischen Lehrsatzes und (2.36)). Umgekehrt folgt aus Bedingung (2.85), dass $\tilde{H}(z)$ eine $\tilde{p}$-fache Nullstelle in $z = -1$ hat. Analog sind p entsprechende Summationsregeln der Form (2.85) für $h(k)$ äquivalent dazu, dass $H(z)$ eine p-fache Nullstelle in $z = -1$ hat.

Für eine ausführlichere Diskussion von Regularitätsbedingungen wird auf [31, Kapitel 7] verwiesen.

2.8 Zur Konstruktion geeigneter Filter

Eine häufig gestellte Frage lautet: „Wie kommt man zu Filtern?“ Für den Praktiker ist ist dies nicht wesentlich, denn er kann die Filterkonstanten aus der Literatur entnehmen. Dieser Abschnitt dient daher der Ergänzung und kann ohne Verständnisschwierigkeiten in den weiteren Kapiteln übersprungen werden.

2.8.1 Folgerungen aus den Bedingungen der perfekten Rekonstruktion

Ausgangspunkt sind die Bedingungen der perfekten Rekonstruktion (2.30) und (2.31):

$$\begin{aligned} H(z)\tilde{H}(-z^{-1}) + G(z)\tilde{G}(-z^{-1}) &= 0 \\ H(z)\tilde{H}(z^{-1}) + G(z)\tilde{G}(z^{-1}) &= 2 \end{aligned}$$

Wir nehmen hier grundsätzlich an, dass alle auftretenden Filter FIR-Filter sind. Die *Modulationsmatrix* wird durch

$$\mathbf{M}(z) := \begin{pmatrix} H(z) & H(-z) \\ G(z) & G(-z) \end{pmatrix} \tag{2.86}$$

definiert. Eine analoge Definition mit Hilfe der Analysisfilter liefert $\tilde{\mathbf{M}}(z)$. Damit erhalten wir für das Produkt $\tilde{\mathbf{M}}(z^{-1})^T\mathbf{M}(z)$ (durch ^{T} ist die transponierte Matrix bezeichnet):

$$\begin{pmatrix} \tilde{H}(z^{-1})H(z) + \tilde{G}(z^{-1})G(z) & \tilde{H}(z^{-1})H(-z) + \tilde{G}(z^{-1})G(-z) \\ \tilde{H}(-z^{-1})H(z) + \tilde{G}(-z^{-1})G(z) & \tilde{H}(-z^{-1})H(-z) + \tilde{G}(-z^{-1})G(-z) \end{pmatrix} \tag{2.87}$$

Ein Vergleich ergibt, dass die Bedingungen der perfekten Rekonstruktion äquivalent sind zu

$$\tilde{\mathbf{M}}(z^{-1})^T\mathbf{M}(z) = 2\mathbf{E} \tag{2.88}$$

wobei durch $\mathbf{E}$ die (2×2)-Einheitsmatrix bezeichnet ist.

Aus den allgemeinen Regeln $\det(\mathbf{AB}) = \det\mathbf{A} \cdot \det\mathbf{B}$ und $\det(\mathbf{A}^T) = \det\mathbf{A}$ erhält man sofort die Folgerung

$$\det\tilde{\mathbf{M}}(z^{-1}) \cdot \det\mathbf{M}(z) = 4 \tag{2.89}$$

Da alle Matrixelemente Laurent-Polynome sind, gilt dies auch für die Determinanten. Die einzigen Laurent-Polynome, deren Kehrwerte ebenfalls Laurent-Polynome sind, sind Potenzen. Bedingung (2.89) ist also nur möglich, wenn

$$\det\mathbf{M}(z) = a \cdot z^m \quad \text{und} \quad \det\tilde{\mathbf{M}}(z) = \tilde{a} \cdot z^m \quad \text{mit} \quad m \in \mathbb{Z} \tag{2.90}$$

Nun ist $\det\mathbf{M}(z) = H(z)G(-z) - G(Z)H(-z)$ ungerade (d.h. $\det\mathbf{M}(-z) = -\det\mathbf{M}(z)$). Also muss m ungerade sein. Aus der Bedingung der perfekten Rekonstruktion in der Form (2.88) folgt

$$\mathbf{M}(z)^{-1} = \frac{1}{2}\tilde{\mathbf{M}}(z^{-1})^T \tag{2.91}$$

Mit Hilfe der allgemeinen Regel

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}^{-1} = \frac{1}{a_{11}a_{22} - a_{21}a_{12}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \tag{2.92}$$

erhält man daraus durch Vergleich der einzelnen Matrixelemente und mit $c = -\frac{a}{2}$ die Beziehungen (2.32) und (2.33), die es erlauben, zwei der vier Filter jeweils durch die beiden andern auszudrücken. Wir werden hier die schon früher erwähnte Konvention $c = 1$ and damit $a = -2$ sowie $m = -1$ benutzen. Meist werden die Tiefpassfilter zuerst konstruiert und anschließend dann aus (2.32) und (2.33) die Hochpassfilter berechnet. Wir erhalten damit

$$G(z) = z^{-1}\tilde{H}(-z^{-1}) \qquad \text{und} \qquad \tilde{G}(z) = z^{-1}H(-z^{-1}) \tag{2.93}$$

Wenn wir diese Beziehungen benutzen, um die beiden Filter $G(z)$ und $\tilde{G}(z)$ in der Gleichung (2.31) durch $H(z)$ und $\tilde{H}(z)$ auszudrücken, so bekommen wir

$$H(z)\tilde{H}(z^{-1}) + H(-z)\tilde{H}(-z^{-1}) = 2 \tag{2.94}$$

Für das Produkt der beiden Filter

$$P(z) := H(z)\tilde{H}(z^{-1}) \tag{2.95}$$

erhalten wir damit die „Halbband-Bedingung“

$$P(z) + P(-z) = 2 \tag{2.96}$$

Notwendig und hinreichend hierfür ist, dass im Polynom $P(z)$ außer $z^0 = 1$ keine Potenzen mit *geradem* Exponenten vorkommen und der Koeffizient von $z^0 = 1$ den Wert 1 hat. Also gilt

$$P(z) = \sum_{k=-\infty}^{+\infty} p(k)z^{-k} \qquad \text{mit} \quad p(2k) = 0 \quad \text{für} \quad k \neq 0 \quad \text{und} \quad p(0) = 1 \tag{2.97}$$

Bevor wir uns der Konstruktion entsprechender Polynome widmen, ist es sinnvoll, einige Folgerungen aus dieser Bedingung zu ziehen. Diese Folgerungen werden benötigt, wenn man zeigen möchte, dass die Skalierungsfunktionen und Wavelets, die aus den Filtern konstruiert werden, tatsächlich biorthogonal sind. Für $F(z) = \tilde{H}(z^{-1})$ gilt $f(k) = \tilde{h}(-k)$ und damit haben wir wegen $P(z) = H(z)F(z) = H(z)\tilde{H}(z^{-1})$

$$\begin{aligned} p(n) &= \sum_{k=-\infty}^{+\infty} h(n-k)f(k) = \sum_{k=-\infty}^{+\infty} h(k)f(n-k) = \sum_{k=-\infty}^{+\infty} h(n-k)\tilde{h}(-k) \\ &= \sum_{k=-\infty}^{+\infty} h(n+k)\tilde{h}(k) = \sum_{k=-\infty}^{+\infty} h(k)\tilde{h}(k-n) \end{aligned} \tag{2.98}$$

Für $n = 0$ erhalten wir daraus

$$\sum_{k=-\infty}^{+\infty} h(k)\tilde{h}(k) = 1 \tag{2.99}$$

und

$$\sum_{k=-\infty}^{+\infty} h(k)\tilde{h}(k-2n) = \sum_{k=-\infty}^{+\infty} h(k+2n)\tilde{h}(k) = 0 \qquad \text{für alle} \quad n \neq 0,\ n \in \mathbb{Z} \tag{2.100}$$

Die um $2n$ gegeneinander verschobenen Filterkonstanten von h und $\tilde{h}$ sind also zueinander *orthogonal*, wenn $n \neq 0$.

Nach (2.32) und (2.33) können wir auch die Tiefpassfilter durch die Hochpassfilter ausdrücken. Mit der hier vorgenommenen Konvention erhalten wir

$$H(z) = -z^{-1}\tilde{G}(-z^{-1}) \qquad \text{und} \qquad \tilde{H}(z) = -z^{-1}G(-z^{-1}) \tag{2.101}$$

Damit bekommt man

$$P(z) = H(z)\tilde{H}(z^{-1}) = (-z^{-1})\tilde{G}(-z^{-1}) \cdot (-z)G(-z) = \tilde{G}(-z^{-1})G(-z)$$

und (wenn man z durch $-z$ ersetzt)

$$G(z)\tilde{G}(z^{-1}) = P(-z) = \sum_{k=-\infty}^{+\infty} (-1)^k p(k) z^{-k}$$

Eine zu (2.98) analoge Rechnung liefert

$$\sum_{k=-\infty}^{+\infty} g(k)\tilde{g}(k) = 1 \quad \text{und} \quad \sum_{k=-\infty}^{+\infty} g(k)\tilde{g}(k-2n) = 0 \quad \text{für alle} \quad n \neq 0,\ n \in \mathbb{Z} \tag{2.102}$$

Ein ähnliches Ergebnis erhält man für $\tilde{h}(k)$ und $g(k)$. Wir wählen hier die Bezeichnungsweise

$$f(n) \ \circ\!\!-\!\!\bullet \ F(z) \qquad \text{wenn} \qquad F(z) = \sum_{k=-\infty}^{+\infty} f(k) z^{-k}$$

Da die Faltung der Multiplikation der z-Transformierten entspricht, gilt

$$\sum_{k=-\infty}^{+\infty} \tilde{h}g(k-n) \ \circ\!\!-\!\!\bullet \ \tilde{H}(z)G(z^{-1})$$

Unterabtastung liefert

$$\sum_{k=-\infty}^{+\infty} \tilde{h}g(k-2n) \ \circ\!\!-\!\!\bullet \ \frac{1}{2}\Big(\tilde{H}(z^{\frac{1}{2}})G(z^{-\frac{1}{2}}) + \tilde{H}(-z^{\frac{1}{2}})G(-z^{-\frac{1}{2}})\Big)$$

Andererseits erhalten wir durch Benutzung von (2.93)

$$\tilde{H}(z)G(z^{-1}) + \tilde{H}(-z)G(-z^{-1}) = \tilde{H}(z)z\tilde{H}(-z) + \tilde{H}(-z)(-z)\tilde{H}(z) = 0$$

Ersetzt man hier z durch $z^{\frac{1}{2}}$, erhält man das Ergebnis

$$\sum_{k=-\infty}^{+\infty} \tilde{h}(k)g(k-2n) = 0 \qquad \text{für alle} \quad n \in \mathbb{Z} \tag{2.103}$$

Ein entsprechendes Ergebnis erhält man für $h(k)$ und $\tilde{g}(k)$

2.8.2 Berücksichtigung von Symmetrie- und Regularitätsforderungen

Unser Ziel ist es, geeignete Polynome $P(z)$ zu konstruieren, die der Halbband-Bedingung (2.96) genügen, und durch Zerlegung in zwei Faktoren $H(z)$ und $\tilde{H}(z^{-1})$ zu gewinnen. Vorher ist festzulegen, welche Bedingungen an die Filter gestellt werden.

Wir werden verlangen, dass die Filter *symmetrisch* sind, d.h.

$$H(z^{-1}) = H(z) \qquad \text{bzw.} \qquad h(-k) = h(k) \qquad \text{bei Symmetrie bzgl. } 0 \tag{2.104}$$

oder

$$H(z^{-1}) = zH(z) \qquad \text{bzw.} \qquad h(-k) = h(k+1) \qquad \text{bei Symmetrie bzgl. } \frac{1}{2} \tag{2.105}$$

erfüllen. Dies garantiert *lineares* Phasenverhalten, was insbesondere in der Bildverarbeitung erwünscht ist. Symmetrie *eines* Filters (z.B. für die Synthese, wie oben formuliert) erlaubt es, das *andere* Filter symmetrisch (mit demselben Symmetrietyp) zu machen, ohne Bedingung (2.94) zu verletzen: Im Fall der Symmetrie bzgl. 0 hat man

$$\tilde{H}(z) \qquad \text{durch} \qquad \frac{1}{2}\Big(\tilde{H}(z) + \tilde{H}(z^{-1})\Big)$$

und bei Symmetrie bzgl. $\frac{1}{2}$

$$\tilde{H}(z) \qquad \text{durch} \qquad \frac{1}{2}\Big(\tilde{H}(z) + z^{-1}\tilde{H}(z^{-1})\Big)$$

zu ersetzen.

Wenn wir *orthogonale* Wavelets haben wollen, müssen wir auf Symmetrieforderungen verzichten, denn die Bedingung $H(z) = \tilde{H}(z)$ und Symmetrie ist *nur* durch das Haar-Wavelet mit dem Filter $H(z) = (1 + z^{-1})/\sqrt{2}$ erfüllbar.

Wir gehen also im weiteren davon aus, dass sowohl $H(z)$ als auch $\tilde{H}(z)$ derselben Symmetriebedingung (2.104) oder (2.105) genügen oder die Orthogonalitätsbedingung $H(z) = \tilde{H}(z)$ erfüllt ist. Dies hat zur Folge, dass $P(z)$ außer der Halbbandbedingung auch der *Symmetriebedingung*

$$P(z) = P(z^{-1}) \tag{2.106}$$

genügt.

Außer Symmetrie *oder* Orthogonalität wollen wir, daß $H(z)$ und $\tilde{H}(z)$ mit den daraus konstruierten Filtern $G(z)$ und $\tilde{G}(z)$ eine vernünftige Filterbank liefert. Wie im Abschnitt 2.7 besprochen wurde, ist eine Mindestforderung hierfür, dass $H(z)$ und $\tilde{H}(z)$ eine Nullstelle in $z = -1$ haben, also mindestens einen Faktor $(1 + z)$ enthalten. Besser ist, sogar eine *mehrfache* Nullstelle zu verlangen.

Aufgrund der Symmetriebedingung (2.106) gehört zu jeder Nullstelle z_0 von $P(z)$ auch eine Nullstelle z_0^{-1}, und die Anzahl der Nullstellen muss *gerade* sein (ihrer Vielfachheit nach gezählt). Damit muss die Vielfachheit einer Nullstelle in $z = -1$ ebenfalls *gerade* sein, denn $(-1)^{-1} = -1$. Aus unserer Regularitätsannahme $H(z) = (z + 1)^p R(z)$ und $\tilde{H}(z) = (z + 1)^{\tilde{p}}\tilde{R}(z)$ sowie $P(z) = H(z)\tilde{H}(z^{-1})$ erhalten wir somit

$$\begin{aligned} P(z) &= H(z)\tilde{H}(z^{-1}) = (z+1)^p(z^{-1}+1)^{\tilde{p}}R(z)\tilde{R}(z^{-1}) \\ &= (z+1)^p z^{\tilde{p}}(z^{-1}+1)^{\tilde{p}}z^{-\tilde{p}}R(z)\tilde{R}(z^{-1}) \\ &= (z+1)^{p+\tilde{p}}z^{-\tilde{p}}R(z)\tilde{R}(z^{-1}) = (z+1)^{2q}z^{-\tilde{p}}R(z)\tilde{R}(z^{-1}) \end{aligned}$$

Da die Vielfachheit der Nullstelle in $z = -1$ gerade ist, muss $p + \tilde{p}$ gerade sein und wir haben $q := \frac{1}{2}(p + \tilde{p})$ eingeführt. Wir schreiben nun $P(z)$ symmetrischer

$$\begin{aligned} P(z) &= (z+1)^{2q} z^{-\tilde{p}} R(z)\tilde{R}(z^{-1}) = (1+z)^q (1+z)^q z^{-q} z^q \cdot z^{-\tilde{p}} R(z)\tilde{R}(z^{-1}) \\ &= (1+z)^q (1+z^{-1})^q z^q z^{-\tilde{p}} R(z)\tilde{R}(z^{-1}) \\ &= \left(\frac{1+z}{2}\right)^q \left(\frac{1+z^{-1}}{2}\right)^q \cdot 2^{2q} \cdot z^q z^{-\tilde{p}} R(z)\tilde{R}(z^{-1}) \\ &= \left(\frac{1+z}{2}\right)^q \left(\frac{1+z^{-1}}{2}\right)^q Q(z) \end{aligned}$$

Wir gehen also im weiteren davon aus, dass das Halbbandfilter $P(z)$ die folgende Form hat:

$$P(z) = \left(\frac{1+z}{2}\right)^q \left(\frac{1+z^{-1}}{2}\right)^q Q(z) \qquad \text{mit} \quad Q(z^{-1}) = Q(z) \quad \text{und} \quad Q(1) \neq 0 \tag{2.107}$$

wobei $q \in \mathbb{N}$ und $Q(z)$ ein LAURENT-Polynom ist. Wir versuchen nun, $Q(z)$ für ein gegebenes q so zu konstruieren, daß $P(z)$ der Halbband-Bedingung genügt und der *Grad* von $P(z)$ minimal wird. Dies wird möglichst kurze Filter liefern. Derartige Filter werden in der Literatur *maximal flache* Filter genannt (englisch kurz „maxflat“).

2.8.3 Berechnung einer Familie von maximal flachen Filtern

Die Symmetriebedingung (2.106) sowie der Ansatz (2.107) legen es nahe, eine neue Variable anstelle von z einzuführen. Wenn man sich auf den Einheitskreis beschränkt, d.h. $z = e^{j\omega}$ und $z^{-1} = e^{-j\omega}$ betrachtet, so führt die Symmetrie zunächst auf $\cos\omega = \frac{1}{2}(z+z^{-1})$. Wir haben auf dem Einheitskreis

$$\left(\frac{1+z}{2}\right)\left(\frac{1+z^{-1}}{2}\right) = \frac{1}{2}(1+\cos\omega) = \cos^2\frac{\omega}{2} \tag{2.108}$$

Es hat sich als nützlich herausgestellt, statt $\cos\omega$ die neue Variable

$$y := \frac{1}{2}(1-\cos\omega) = \sin^2\frac{\omega}{2}$$

einzuführen. In $\mathbb{C}$ führt dies auf die Transformation

$$y = J(z) := \frac{1}{2} - \frac{1}{4}(z+z^{-1}) = \left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right) \tag{2.109}$$

Sie erfüllt

$$J(z) = J(z^{-1}) \qquad \text{und} \qquad J(-z) = 1 - J(z) \tag{2.110}$$

Wir drücken nun $P(z)$ und $Q(z)$ durch die neue Variable aus, d.h. wir schreiben

$$P(z) = \tilde{P}(J(z)) = \tilde{P}(y) \qquad \text{und} \qquad Q(z) = \tilde{Q}(J(z)) = \tilde{Q}(y) \tag{2.111}$$

Die Symmetriebedingung (2.106) ist damit automatisch erfüllt. Der entscheidende Vorteil dieser neuen Variablen y zeigt sich nun darin, dass aufgrund von (2.110) die Halbband-Bedingung übergeht in

$$\tilde{P}(y) + \tilde{P}(1-y) = 2 \tag{2.112}$$

Mit Hilfe der Identität

$$\left(\frac{1+z}{2}\right)\left(\frac{1+z^{-1}}{2}\right) = 1-y \tag{2.113}$$

erhält unser Ansatz (2.107) die Form

$$\tilde{P}(y) = (1-y)^q \tilde{Q}(y) \tag{2.114}$$

Damit folgt aus der Halbband-Bedingung

$$(1-y)^q \tilde{Q}(y) + y^q \tilde{Q}(1-y) = 2 \tag{2.115}$$

Derartige Gleichungen sind aus der Algebra altbekannt. So kann beispielsweise der Algorithmus von EUKLID benutzt werden, um bei gegebenem q ein Polynom $\tilde{Q}(y)$ möglichst niederen Grades ($q-1$ oder kleiner) zu konstruieren.

Eine andere Methode liefert die *Partialbruchzerlegung*

$$\frac{1}{y^q(1-y)^q} = \sum_{k=1}^{q} \frac{a_k}{y^k} + \sum_{k=1}^{q} \frac{b_k}{(1-y)^k} \tag{2.116}$$

Aus Symmetriegründen müssen die Koeffizienten $a_k = b_k$ für $k = 1, 2 \ldots q$ erfüllen. Denn die linke Seite bleibt unverändert, wenn man y durch $1-y$ ersetzt. Also muss dies auch für die rechte Seite gelten. Die Koeffizienten a_k liefern das gesuchte Polynom, da aus (2.116) folgt

$$1 = (1-y)^q \sum_{k=1}^{q} a_k y^{q-k} + y^q \sum_{k=1}^{q} a_k (1-y)^{q-k} \tag{2.117}$$

Mit den Koeffizienten $c_k := a_{q-k}$ für $k = 0, 1, 2, \ldots q-1$ erhalten wir die Lösung

$$\tilde{Q}(y) = 2\sum_{k=1}^{q} a_k y^{q-k} = 2\sum_{k=0}^{q-1} a_{q-k} y^k = 2\sum_{k=0}^{q-1} c_k y^k \tag{2.118}$$

Wir haben also lediglich die Koeffizienten der Partialbruchzerlegung a_k bzw. c_k mit einer der üblichen Methoden zu berechnen: Aus (2.116) erhält man

$$\frac{1}{(1-y)^q} = \sum_{k=1}^{q} a_k y^{q-k} + y^q \sum_{k=1}^{q} \frac{a_k}{(1-y)^k} \tag{2.119}$$

Setzt man $y = 0$ so erhält man $a_q = c_0 = 1$. Sukzessives Ableiten liefert auf der linken Seite

$$\frac{d^k}{dy^k}(1-y)^{-q} = (-1)^k(-q)(-q-1)(-q-2)\cdots(-q-k+1)(1-y)^{-q-k}$$

und wenn wir $y = 0$ setzen, dann bleibt übrig:

$$(-1)^k(-q)(-q-1)(-q-2)\cdots(-q-k+1) = q(q+1)(q+2)\cdots(q+k-1)$$

Auf der rechten Seite erhalten wir

$$\frac{d^k}{dy^k}\sum_{l=1}^{q} a_l y^{q-l} = \frac{d^k}{dy^k}\sum_{l=0}^{q-1} c_l y^l = \sum_{l=k}^{q-1} c_l l(l-1)(l-2)\cdots(l-k+1)y^{l-k}$$

und für $y = 0$ bleibt nur der Term für $l = k$ übrig:

$$c_k k(k-1)(k-2)\cdots 1 = k! \cdot c_k$$

Der Term

$$\frac{d^k}{dy^k}\left(y^q \sum_{k=1}^{q} \frac{a_k}{(1-y)^k}\right)$$

trägt für $y = 0$ nichts bei, denn er enthält nur Potenzen y^{q-k} oder höhere Potenzen von y. Für $y = 0$ liefert die k. Ableitung unserer Gleichung (2.119)

$$q(q+1)(q+2)\cdots(q+k-1) = k! \cdot c_k$$

oder

$$c_k = \frac{q(q+1)(q+2)\cdots(q+k-1)}{k!} = \binom{q+k-1}{k}$$

Damit erhalten wir die Lösung

$$\tilde{P}(y) = 2(1-y)^q \sum_{k=0}^{q-1} \binom{q+k-1}{k} y^k \tag{2.120}$$

und mit Hilfe von (2.113) das Ergebnis:

Das Halbbandfilter minimaler Länge der Form (2.107) ist gegeben durch

$$P(z) = 2\left(\frac{1+z}{2}\right)^q \left(\frac{1+z^{-1}}{2}\right)^q \sum_{k=0}^{q-1} \binom{q+k-1}{k} \left(\frac{1-z}{2}\right)^k \left(\frac{1-z^{-1}}{2}\right)^k \tag{2.121}$$

Wir haben hier Filter *minimaler* Länge konstruiert, indem wir das Polynom minimalen Grades berechnet haben, das (2.115) löst. Die allgemeine Lösung von (2.115) lautet

$$\tilde{Q}(y) = 2\sum_{k=0}^{q-1} \binom{q+k-1}{k} y^k + y^q R(1-2y) \tag{2.122}$$

wobei $R(x)$ ein beliebiges **ungerades** Polynom ist. Denn beim Einsetzen in (2.115) entstehen die Zusatzterme

$$(1-y)^q y^q R(1-2y) + y^q (1-y)^q R\big(1-2(1-y)\big) = (1-y)^q y^q \left(R(1-2y) + R\big(-(1-2y)\big)\right) = 0$$

Aufgrund der Identität

$$1 - 2y = \frac{1}{2}(z + z^{-1})$$

erhalten wir also längere Filter $P(z)$, die der Halbband-Bedingung und unserer Symmetrieforderung genügen, wenn wir zu (2.121) noch ein Polynom der Form

$$\left(\frac{1+z}{2}\right)^q \left(\frac{1+z^{-1}}{2}\right)^q \left(\frac{1-z}{2}\right)^q \left(\frac{1-z^{-1}}{2}\right)^q R\left(\frac{1}{2}(z+z^{-1})\right) \tag{2.123}$$

mit $R(-x) = -R(x)$ addieren.

2.8.4 Faktorisierung der maximal flachen Filter

(a) Für $q = 1$ erhalten wir aus (2.121)

$$P(z) = \left(\frac{1+z}{2}\right)\left(\frac{1+z^{-1}}{2}\right)\cdot 2 = \frac{1}{2}(1+z^{-1})(1+z) = H(z)\tilde{H}(z^{-1})$$

und daraus das Haar-Filter

$$H(z) = \tilde{H}(z) = \frac{1}{\sqrt{2}}(1+z^{-1})$$

(b) Für $q = 2$ erhalten wir

$$Q(z) = 2\left(1 + \binom{2}{1}\frac{(1-z)}{2}\frac{(1-z^{-1})}{2}\right) = -z + 4 - z^{-1}$$

und damit

$$\begin{aligned} P(z) &= \left(\frac{1+z}{2}\right)^2\left(\frac{1+z^{-1}}{2}\right)^2(-z+4-z^{-1}) \\ &= 2^{-4}(1+z)^2(1+z^{-1})^2(-z+4-z^{-1}) = H(z)\tilde{H}(z^{-1}) \end{aligned} \tag{2.124}$$

- **1. Faktorisierung:**

$$H(z) = \frac{1}{4}\sqrt{2}(1+z)(1+z^{-1}) = \frac{1}{4}\sqrt{2}(z+2+z^{-1}) \tag{2.125}$$

$$\begin{aligned} \tilde{H}(z) &= \frac{1}{8}\sqrt{2}(1+z)(1+z^{-1})(-z+4-z^{-1}) \\ &= \frac{1}{4}\sqrt{2}\Big(3 + (z+z^{-1}) - \frac{1}{2}(z^2+z^{-2})\Big) \end{aligned} \tag{2.126}$$

Die Hochpassfilter ergeben sich mit der üblichen Konvention aus

$$G(z) = z^{-1}\tilde{H}(-z^{-1}) \qquad \tilde{G}(z) = z^{-1}H(-z^{-1})$$

und wir erhalten hier

$$G(z) = \frac{1}{4}\sqrt{2}(-\frac{1}{2}z - 1 + 3z^{-1} - z^{-2} - \frac{1}{2}z^{-3}) \tag{2.127}$$

$$\tilde{G}(z) = \frac{1}{4}\sqrt{2}(-1 + 2z^{-1} - z^{-2}) \tag{2.128}$$

Diese Filter werden in der Literatur mit CDF(2,2) bezeichnet, das Rekonstruktionswavelet heißt wegen seiner Form *Hut-Wavelet*.

- **2. Faktorisierung:** Da

$$(-z+4-z^{-1}) = -z^{-1}(z^2 - 4z + 1)$$

die Nullstellen $z_1 = 2+\sqrt{3}$ und $z_2 = 2-\sqrt{3} = z_1^{-1}$ hat, gilt

$$(-z+4-z^{-1}) = z_1^{-1}(z-z_1)(z^{-1}-z_1)$$

und damit

$$P(z) = 2^{-4}(1+z)^2(1+z^{-1})^2 \cdot z_1^{-1}(z-z_1)(z^{-1}-z_1)$$

Dies liefert die Faktorisierung mit

$$H(z) = \tilde{H}(z) = c(1+z^{-1})^2(z^{-1}-z_1)$$

und damit orthogonalen Wavelets, wobei die Normierungskonstante c aus unserer Normierungsbedingung $H(1) = \sqrt{2}$ erhältlich ist. Wir haben damit unser früheres Beispiel D4 mit $H(z) = \sum_{k=0}^{3} h(k)z^{-k}$ und den folgenden Filterkonstanten

$$\begin{aligned} h(0) &= \frac{1}{4\sqrt{2}}(1+\sqrt{3}), \qquad h(1) = \frac{1}{4\sqrt{2}}(3+\sqrt{3}) \\ h(2) &= \frac{1}{4\sqrt{2}}(3-\sqrt{3}), \qquad h(3) = \frac{1}{4\sqrt{2}}(1-\sqrt{3}) \end{aligned}$$

erhalten.

- **3. Faktorisierung:** Die entstehenden Filter werden in der Literatur mit CDF(3,1) bezeichnet.

$$\begin{aligned} H(z) &= \frac{\sqrt{2}}{8}(1+z)(1+z^{-1})^2 = \frac{\sqrt{2}}{8}(z+3+3z^{-1}+z^{-2}) \quad (2.129) \\ \tilde{H}(z) &= \frac{\sqrt{2}}{4}(1+z^{-1})(-z+4-z^{-1}) \\ &= \frac{\sqrt{2}}{4}\left(-z+3+3z^{-1}-z^{-2}\right) \quad (2.130) \end{aligned}$$

  Der Kaskade-Algorithmus konvergiert nicht für die Filterkonstanten $\tilde{h}(k)$ (dies ist am Ende von Abschnitt 2.4 besprochen).

(c) Wir überspringen den Fall $q = 3$ und erhalten für $q = 4$ aus (2.121)

$$P(z) = 2^{-11}(1+z)^4(1+z^{-1})^4 \cdot \Big(208-131(z+z^{-1})+40(z^2+z^{-2})-5(z^3+z^{-3})\Big) \quad (2.131)$$

Eine erste Faktorisierung $P(z) = H(z)\tilde{H}(z^{-1})$ ist

$$\begin{aligned} H(z) &= \frac{1}{8}\sqrt{2}(1+z)(1+z^{-1})^2 \\ \tilde{H}(z) &= \frac{1}{512}\sqrt{2}(1+z)^2(1+z^{-1})^3 \\ &\qquad \cdot\Big(208-131(z+z^{-1})+40(z^2+z^{-2})-5(z^3+z^{-3})\Big) \end{aligned}$$

Die resultierenden Filter tragen die Bezeichnung CDF(3,5), die entsprechenden Filterkonstanten sind bereits in Tabelle 2.2 angegeben worden.

Für weitere Faktorisierungen müssen die $2q-2$ Nullstellen des hinteren Faktors in (2.131) numerisch bestimmt werden, wobei die Symmetriebedingung (2.106) ausgenutzt werden kann. Das hier auftretende Polynom $P(z)$ hat außer einer achtfachen

Nullstelle in $z = -1$ die Nullstellen z_1, z_1^*, $1/z_1$, $1/z_1^*$, z_2 und $1/z_2$, wobei man als Ergebnis einer numerischen Näherung (siehe auch Anhang D.2) erhält:

$$z_1 = 2.0311355 + j \cdot 1.7389508; \qquad z_2 = 3.0406605$$

Wir betrachten hier noch zwei Faktorisierungen:

FBI-Filter: Es hat seinen Namen daher, dass es erfolgreich zur Kompression der Fingerabdrücke angewandt wird, die das FBI sammelt. Es hat sich auch für die Kompression von andern Bildern bewährt. Man fasst die Faktoren aus den komplexen Nullstellen mit der Hälfte der Nullstellen in $z = -1$ zu $\tilde{H}(z)$ zusammen, die Faktoren aus den beiden reellen Nullstellen bleiben dann für $H(z)$. Bis auf noch zu bestimmende konstante Faktoren c und $\tilde{c}$ erhalten wir also (näheres siehe Anhang D.2)

$$\tilde{H}(z) = \tilde{c}(1+z)^2(1+z^{-1})^2(z_1 - z)(z_1 - z^{-1})(z_1^* - z)(z_1^* - z^{-1}) \quad (2.132)$$
$$H(z) = c(1+z)^2(1+z^{-1})^2(z_2 - z)(z_2 - z^{-1}) \quad (2.133)$$

Die beiden Faktoren c und $\tilde{c}$ kann man aus der Normierungsbedingung $H(1) = \tilde{H}(1) = \sqrt{2}$ bestimmen:

$$\tilde{c} = \frac{\sqrt{2}}{16(z_1 - 1)^2(z_1^* - 1)^2} \approx 0,00529109 \quad (2.134)$$
$$c = \frac{\sqrt{2}}{16(z_2 - 1)^2} \approx 0.0212253 \quad (2.135)$$

Die Filterkonstanten $\tilde{h}(k)$ und $h(k)$ erhält man dann durch Ausmultiplizieren von (2.132) bzw. (2.133) und Koeffizientenvergleich. Die Zahlenwerte sind in Tabelle A.1 aufgeführt.

Symlets: Für ein lineares Phasenverhalten sind symmetrische oder antisymmetrische Filter notwendig. Dies ist mit orthogonalen Wavelets nur für das Haar-Wavelets möglich. Man kann jedoch durch eine geeignete Faktorisierung von $P(z)$ Filterkonstanten für orthogonale Wavelets so konstruieren, dass für eine gegebene Filterlänge eine möglichst geringe Abweichung vom linearen Phasenverhalten herauskommt. Die zugehörigen Wavelets werden zuweilen „Symlets“ genannt. Es gibt vier verschiedene Faktorisierungen des durch (2.131) gegebenen Halbbandfilters $P(z)$, die *orthogonale* Wavelets liefern, also $P(z) = H(z)H(z^{-1})$ erfüllen. Zwei davon liefern Filterkonstanten, die weniger asymmetrisch sind. Eine davon ist

$$H(z) = c(1+z^{-1})^4(z_2 - z)(z_1^{-1} - z)(z_1^{*-1} - z) \quad (2.136)$$

Auch hier kann die Konstante c aus der Normierungsbedingung $H(1) = \sqrt{2}$ bestimmt werden, man erhält $c \approx 0.0757657$. Die Filterkonstanten erhält man durch Ausmultiplizieren von $H(z)$ und Sortieren nach Potenzen von z. Die Zahlenwerte sind in Tabelle A.2 im Anhang angegeben. Als qualitatives Maß für die Asymmetrie wurde

$$a = \sum_{k=0}^{4} \Big(h(k) - h(-k+1) \Big)^2 \quad (2.137)$$

berechnet. Eine weitere Faktorisierung liefert dieselben Filterkonstanten $h(k)$ in umgekehrter Reihenfolge und damit denselben Zahlenwert für den Asymmetrieparameter a. Mit den beiden andern Faktorisierungen, die auf orthogonale Wavelets führen, erhält man einen höheren Wert von a.

2.9 Behandlung endlich vieler Daten, Randbedingungen

Die bisherige Darstellung setzte stillschweigend voraus, dass das zu verarbeitende Signal $s_0(k) = x(k)$ aus unendlich vielen Abtastwerten besteht ($k \in \mathbb{Z}$). Dies ist völlig unrealistisch. In der Praxis hat man N Werte, hier numeriert als $x(0)$, $x(1)$, $x(2), \ldots x(N-1)$. Durch die Unterabtastung halbiert sich die Zahl der Werte in jedem Kanal, so dass man erwartet, dass man bei geradem N in jedem Kanal $\frac{1}{2}N$ Werte erhält und die Gesamtzahl N der zu verarbeitenden Werte gleichbleibt.

Eine naheliegende Idee, daraus unendlich viele Werte zu machen, ist die Fortsetzung zu Null (englisch „zero padding“) durch

$$x(k) = 0 \qquad \text{falls } k < 0 \text{ oder } k \geq N$$

Dies führt jedoch zu folgenden Nachteilen:

(a) Durch die Filterung werden mehr Werte von Null verschieden, das Signal würde sich ausbreiten.

(b) Am Rand führt man in vielen Fällen einen Sprung ein. Dies äußert sich nach der Filterung in großen Werten für die Koeffizienten $d(k)$ und in unerwünschten Randeffekten für $s(k)$. Bei einem Bild mit einer weißem Fläche am Rand entsteht durch die Tiefpassfilterung ein grauer Randstreifen.

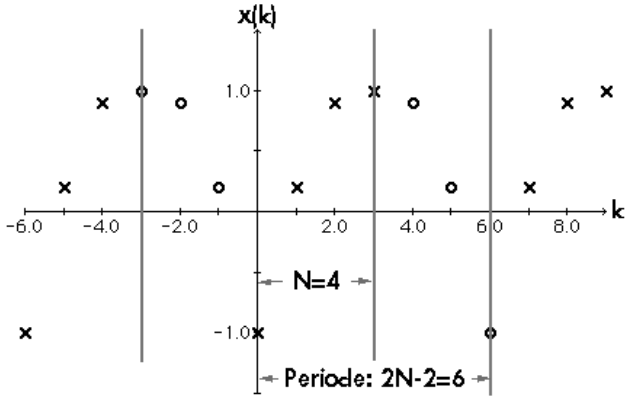


Abbildung 2.32: symmetrische Fortsetzung ohne Wiederholung des Randpunkts bei $N = 4$ Datenpunkten

Eine mathematisch *bequeme* Methode ist die *periodische Fortsetzung* mit der Periode N durch

$$x(k) = x(k \mod N)$$

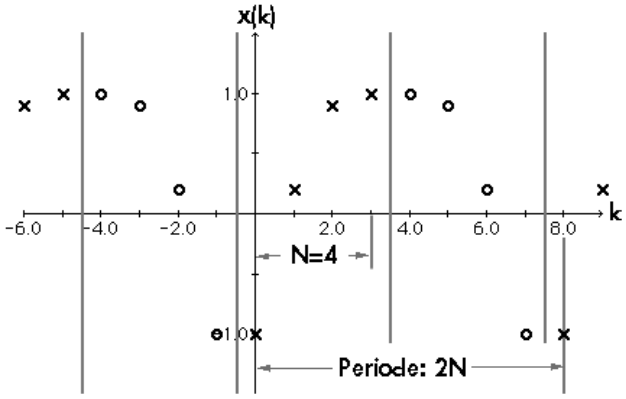


Abbildung 2.33: symmetrische Fortsetzung mit Wiederholung des Randpunkts bei $N = 4$ Datenpunkten

Dabei ist $k \mod N$ der ganzzahlige Rest, der bei Division von k durch N entsteht (in der Programmiersprache C durch `k % N` realisiert). Bei negativem k ist, damit ein Index zwischen 0 und $N-1$ entsteht, N dazuzuzählen (Achtung, das Ergebnis des %-Operators kann maschinenabhängig sein). Die periodische Fortsetzung ist mathematisch völlig unproblematisch, denn durch Filterung entsteht wieder eine Folge mit derselben Periode (als Übungsaufgabe zu überprüfen, siehe Abschnitt 1.3.2!) Falls N gerade ist, halbiert sich die Periode bei der Unterabtastung, so dass man, wie erwünscht, für jeden Kanal nur halb so viele Werte abzuspeichern hat. Damit dies in m Stufen fortgesetzt werden kann (siehe Abschnitt 1.7), muss N ein ganzahliges Vielfaches von 2^m sein. Falls N eine Zweierpotenz ist, d.h. $N = 2^m$, kann man die Wavelet-Transformation mit maximal m Stufen durchführen und erhält dann konstante Folgen $s_m(k)$ und $d_m(k)$ (Periode 1 heißt konstant). Man braucht also hierfür nur jeweils 1 Wert abzuspeichern. Da sich bei jeder Stufe die Zahl der abzuspeichenden Werte nicht ändert, bleibt sie insgesamt unverändert (zur Erinnerung: Datenkompression erreicht man über Quantisierung und Kodierung). Die Eigenschaft der perfekten Rekonstruktion der verwandten Filter garantiert, dass man wieder die ursprünglichen Werte zurückbekommt (bis auf die in der Praxis unvermeidlichen Rundungsfehler). Die Implementierung stellt keine großen Schwierigkeiten dar (Vorsicht jedoch mit den Indizes, falls nach mehreren Stufen die Periode kleiner als die Filterlänge geworden ist!).

Der durch die periodische Fortsetzung in vielen Fällen am Rand eingeführte Sprung hat dieselben unangenehmen Konsequenzen wie beim „zero padding“, die oben beschrieben wurden. Abhilfe kann hier die *symmetrische* Fortsetzung schaffen, wie sie in Abb. 2.32 illustriert ist. Sie führt zu einem unendlichen Signal der Periode $2N-2$. Aufgrund der Symmetrie braucht man nur die halbe Periode, also N Werte abspeichern. Aber bei der Implementierung muss man auf eine ganze Reihe technischer Einzelheiten aufpassen:

(a) Nur bei symmetrischen oder antisymmetrischen Filtern ist das gefilterte Signal wieder symmetrisch oder antisymmetrisch, und ohne Symmetrie müsste man die ganze Periode, also fast doppelt so viele Werte abspeichern (was nicht sehr sinnvoll für die Datenkompression wäre).

(b) Der Randpunkt wurde in Abb. 2.32 nicht wiederholt, die Folge ist symmetrisch in Bezug auf den Anfangspunkt Null. Bei der Unterabtastung treten dadurch je nach

Filterlänge und Symmetrieart des Filters Probleme auf. Denn bei einer geraden Filterlänge sind die Filterkonstanten symmetrisch bezüglich dem Mittelpunkt zwischen zwei Indizes, also formal bezüglich einem halbzahligen Index. Dann muss man bei den Daten zur symmetrischen Fortsetzung mit Wiederholung des Randpunkts übergehen, wie dies in Abb. 2.33 gezeigt ist. Dadurch wird auch das Signal symmetrisch in Bezug auf einen Zeitpunkt in der Mitte zwischen den Abtastwerten. Die Periode ist bei dieser Art der Fortsetzung $2N$.

(c) Bei der Implementierung hat man für die Rekonstruktion darauf zu achten, ob Symmetrie oder Antisymmetrie (Achsen- oder Punktsymmetrie) vorliegt. Dies ist meist bei Hoch- und Tiefpassfilter unterschiedlich.

Für die technischen Einzelheiten wird auf [31, Kapitel 8.2] verwiesen. Eine ziemlich vollzählige Darstellung von allem, was bei der symmetrischen Fortsetzung zu beachteten ist, findet man in [4].

Insgesamt sind sehr viele komplizierte Einzelheiten bei der korrekten Implementierung der symmetrischen bzw. antisymmetrischen Fortsetzung zu beachten. Eine wesentlich einfachere und flexiblere Möglichkeit ergibt sich durch das Lifting-Schema. Die dadurch mögliche Behandlung der Daten am Rand wird in Abschnitt 3.4 besprochen.

Kapitel 3

Das Lifting-Schema

3.1 Einfaches Beispiel, Vorteile des Lifting-Schemas

Das Lifting-Schema ermöglicht eine schnellere Berechnung der Wavelet-Koeffizienten (als die direkte Umsetzung der Filterbank) sowie eine intuitiv einfache Konstruktion von Filter-Paaren (und von neuen Wavelets). Es soll in diesem Abschnitt nur so vorgestellt werden, wie es in der Praxis (beispielsweise bei der Bildkompression) angewandt wird. An Beispielen soll plausibel gemacht werden, dass die Transformation, die durch das Lifting-Schema definiert ist, im Hinlick auf die Anwendungen sinnvoll ist. Außer für das ganz einfache Beispiel des Haar-Wavelets wird aber zunächst in keiner Weise dargelegt, daß diese neue Methode tatsächlich die früher behandelten Wavelet-Koeffizienten liefert. Erst in Abschnitt 3.2 wird aufgezeigt, dass das Ergebnis des Lifting tatsächlich mit dem bisher behandelten der Filterbank übereinstimmt.

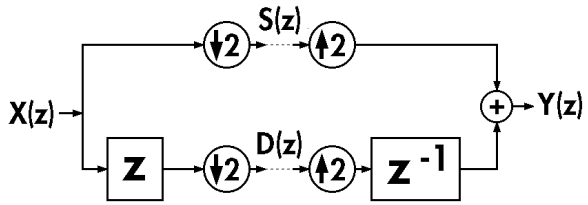


Abbildung 3.1: „Filterbank“ zum „Lazy“ Wavelet

Ausgangspunkt ist die zum „Lazy“-Wavelet gehörige Transformation (siehe Abb. 3.1, der Name sagt schon alles). Dies ist die einfachste Filterbank mit $H(z) = \tilde{H}(z) = 1$ und $G(z) = \tilde{G}(z)$. Für die Analyse erhält man:

$$s(k) = x(2k), \qquad d(k) = x(2k+1) \tag{3.1}$$

Diese Operation wird in der englischen Literatur auch mit dem Namen „split“ bezeichnet. Für eine Analyse oder Kompression der Daten hat man durch diese bloße Aufspaltung allein noch nichte gewonnen.

Hierzu fügt man dann „Lifting“-Stufen und „duale Lifting“-Stufen hinzu. Zur Datenkompression ist es beispielsweise sinnvoll, die Werte zu ungeraden Indizes (in der Abb. unten) durch die Differenz zu den Werten zu ersetzen, die sich als *Vorhersage* aus den Werten mit geraden Indizes ergeben. Und die einfachste Vorhersage für unbekannte Abtastwerte ist der Mittelwert der benachbarten bekannten Werte. Man wählt also statt der

„Lazy“-Transformation, die mit „alt“ gekennzeichnet ist

$$d_{\text{neu}}(k) = x(2k+1) - \frac{1}{2}\Big(x(2k) + x(2k+2)\Big) \tag{3.2}$$

bzw. durch die „alten“ Werte ausgedrückt (englische Bezeichnung der Operation „predict“)

$$d_{\text{neu}}(k) = d_{\text{alt}}(k) - \frac{1}{2}\Big(s_{\text{alt}}(k) + s_{\text{alt}}(k+1)\Big) \tag{3.3}$$

Überall dort, wo die ursprünglichen Werte *linear* vom Index abhängen, d.h. wo eine Bezieung der Form

$$x(k) = \alpha k + \beta \tag{3.4}$$

gilt, erhält man $d_{mn}(k) = 0$. Dies führt in der Praxis auf sehr *kleine* Werte für $d(k)$ (geringer Fehler beim Weglassen, nützlich für die Datenkompression). Ausnahmen treten an den Stellen auf, an denen sich die Signalwerte abrupt ändern. Bei Bildern sind das die Kanten.

Für die Rekonstruktion hat man einfach (3.3) wieder nach $d_{\text{alt}}(k)$ aufzulösen. Dies führt zum sehr einfachen Schema von Abb. 3.2. Ein wesentlicher Unterschied zu den bisher betrschteten Filterbänken (Abb. 1.10 oderAbb. 2.18) ist, daß die Unterabtastung *vor* dem Filtern und das Upsampling *nach* dem Filtern stattfindet.

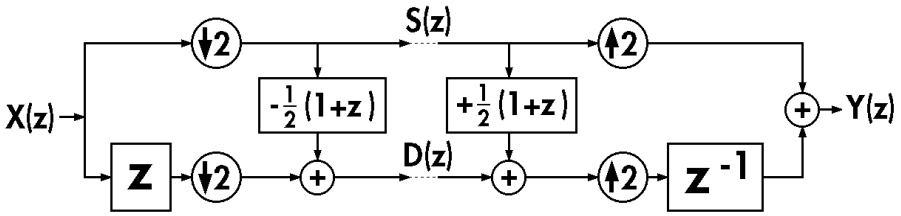


Abbildung 3.2: Lifting: einfachste Vorhersage jedes 2. Wertes durch den Mittelwert der Nachbarwerte

Im Frequenzbereich wäre diese Transformation noch sehr unbefriedigend, vor allem wenn man sie in mehreren Stufen hintereinander ausführt. Denn die Werte mit geradem (durch 4, 8, 16 . . . teilbarem) Index bleiben unverändert. Sie sollten dagegen etwas mit dem Mittelwert zu tun haben. Eine kleine Änderung bringt Abhilfe. Zusätzlich zu (3.3) führt man eine Korrektur ein (englische Bezeichnung „update“)

$$s_{\text{neu}}(k) = s_{\text{alt}}(k) + \frac{1}{4}\big(d_{\text{neu}}(k-1) + d_{\text{neu}}(k)\big) \tag{3.5}$$

Dies ist ebenfalls durch Auflösen nach $s_{\text{alt}}(k)$ leicht rückgängig zu machen, siehe hierzu Abb. 3.3. Mit dieser Änderung ist gewährleistet, dass der Mittelwert über die Werte $s_{\text{neu}}(k)$ bis auf einen Normierungsfaktor mit dem über das ursprüngliche Signal übereinstimmt, d.h.

$$\sum_{k=-\infty}^{+\infty} x(k) = 2 \sum_{k=-\infty}^{+\infty} s(k) \tag{3.6}$$

Dieser Normierungsfaktor berücksichtigt, dass nach der Unterabtastung nur noch halb so viele Signalwerte übrigbleiben. Durch den Korrekturschritt (3.5) wurde eine wesentliche

Verbesserung des Frequenzverhaltens der Transformation erreicht. Dies kann man daran sehen, dass ein Signal der größtmöglichen Frequenz der Form $x(k) = (-1)^k$ als Ergebnis $s(k) = 0$ liefert. Ohne diese Korrektur würden wir $s(k) = 1$ erhalten, was für eine Tiefpassfilterung nicht akzeptabel ist.

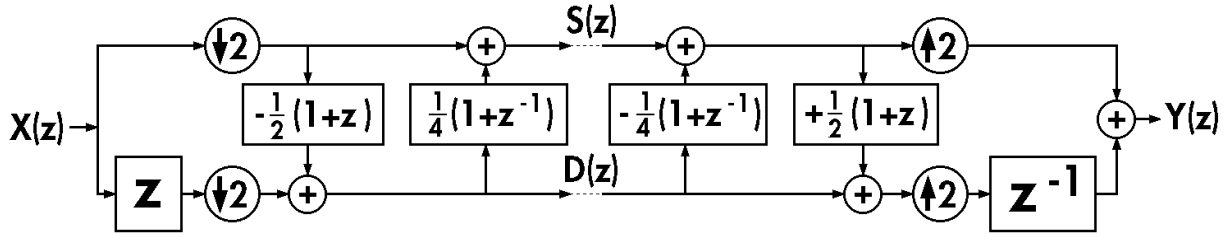


Abbildung 3.3: Lifting: Korrektur nach der Vorhersage durch den Mittelwert

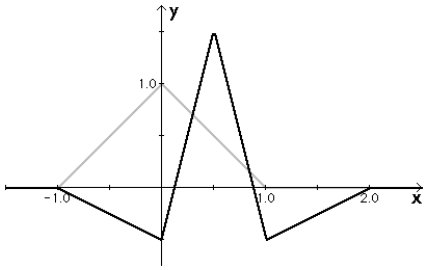


Abbildung 3.4: Skalierungsfunktion $\varphi(x)$ (grau) und Wavelet $\psi(x)$ (schwarz), die zu den Rekonstruktionsfiltern gehören, das zum Lifting-Schema von Abb. 3.3 äquivalent ist (in der Literatur mit CDF(2,2) bezeichnet).

Nun ist noch die Normierung verbesserungsbedürftig, denn (3.6) stimmt nicht mit der bisherigen Vereinbarung $H(1) = \tilde{H}(1) = \sqrt{2}$ überein, die dafür sorgt, dass ein konstantes Signal (was der niedrigsten Frequenz $\omega = 0$ und damit $z = 1$ entspricht) durch die Tiefpassfilterung gerade mit dem Faktor $\sqrt{2}$ multipliziert wird. Man kann durch die Normierung

$$s_{\text{norm}}(k) = \sqrt{2} \cdot s_{\text{neu}}(k), \qquad d_{\text{norm}}(k) = \frac{1}{2}\sqrt{2} \cdot d_{\text{neu}}(k) \tag{3.7}$$

für Übereinstimmung mit den bisherigen Konventionen sorgen.

Man kann zeigen, dass die durch (3.3), (3.5) und (3.7) beschriebene Transformation kann auch durch eine Filterbanktransformation wie in Abb. 2.18 durchgeführt werden kann, allerdings mit etwa doppeltem Rechenaufwand. Dies wird im einzelnen in Abschnitt 3.2 erläutert. Das zu den dabei auftretenden Rekonstruktionsfiltern gehörende Wavelet ist mit seiner Skalierungsfunktion in Abb. 3.4 gezeigt. Es wird in der Literatur mit CDF(2,2) bezeichnet. Gegenüber dem Haar-Wavelet zeichnet es sich durch Stetigkeit aus. Es ist vielleicht überraschend, dass trotz einer oberflächlichen Ähnlichkeit der auftretenden Filter hier nicht das Haar-Wavelet ins Spiel kommt.

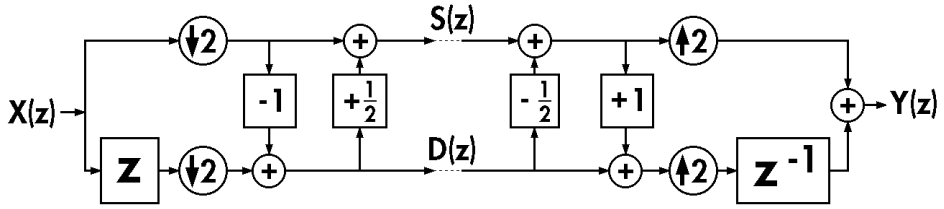


Abbildung 3.5: Lifting-Schema für das Haar-Wavelet

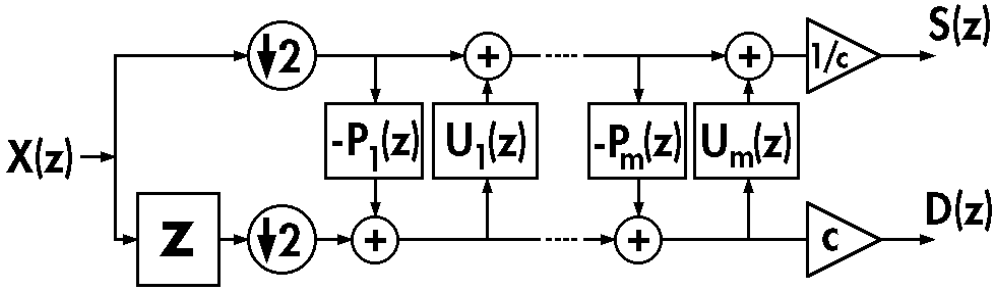


Abbildung 3.6: Lifting-Schema, allgemein: Analyse

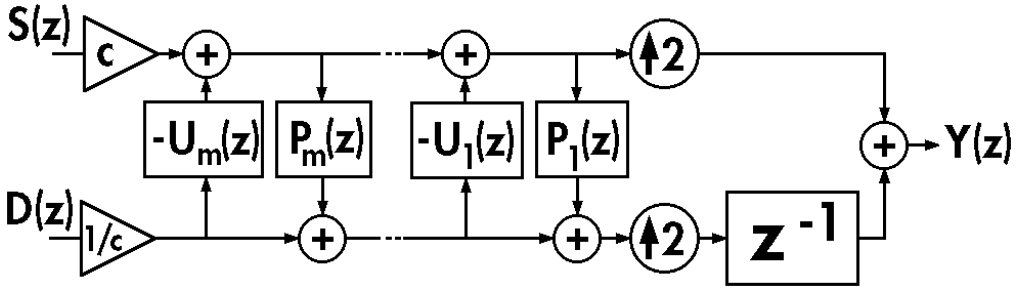


Abbildung 3.7: Lifting-Schema, allgemein: Rekonstruktion (Synthese)

Man kann in der Tat fast dieselbe Transformation von Abschnitt 1.2, die zum Haar-Wavelet gehört, durch folgende Lifting-Schritte erreichen (siehe auch Abb. 3.5):

$$\begin{aligned} s_{\text{alt}}(k) &= x(2k), \qquad d_{\text{alt}}(k) = x(2k+1) \\ d_{\text{neu}}(k) &= d_{\text{alt}}(k) - s_{\text{alt}}(k) \qquad (3.8) \\ s_{\text{neu}}(k) &= s_{\text{alt}}(k) + \frac{1}{2} d_{\text{neu}}(k) \qquad (3.9) \end{aligned}$$

Ausgedrückt durch das Eingangssignal $x(k)$ erhalten wir damit

$$\begin{aligned} d_{\text{neu}}(k) &= x(2k+1) - x(2k) \qquad (3.10) \\ s_{\text{neu}}(k) &= x(2k) + \frac{1}{2} x(2k+1) - \frac{1}{2} x(2k) = \frac{1}{2}\Big(x(2k) + x(2k+1)\Big) \qquad (3.11) \end{aligned}$$

in Übereinstimmung mit (1.5) für $s(k)$. Für $d(k)$ ergibt sich gerade das umgekehrte Vorzeichen gegenüber (1.6). Wir haben genau das Negative des Ausgangs des in Abschnitt 1.2 verwandten Hochpassfilters. Das negative Vorzeichen ändert das Frequenzverhalten in

keiner Weise. In Abschnitt 3.2.3 wird klar, warum hier nur bis auf ein Vorzeichen eine Übereinstimmung erzielt wird.

Rechenschritte analog zu (3.3) und (3.5) können mehrfach hintereinander ausgeführt werden. Die entsprechende Rekonstruktion ist durch einfaches Auflösen dieser Gleichungen und Durchführung der zugehörigen Rechenschritte in umgekehrter Richtung möglich. Es ist sinnvoll, noch einen Faktor zur Normierung am Schluß zuzulassen. Das zugehörige Schema ist in Abb. 3.6 zur Analyse und 3.7 zur Rekonstruktion dargestellt.

Es hat sich herausgestellt, dass jede Zwei-Kanal-Filterbank, deren Filter der Bedingung der perfekten Rekonstruktion genügen und die FIR-Filter sind, durch ein Schema dieser Art ersetzt werden kann. Die Berechnung der zugehörigen Lifting-Schritte, d.h. der Filter $P_k(z)$ und $U_k(z)$ aus den Filtern $H(z)$, $G(z)$, $\tilde{H}(z)$ und $\tilde{G}(z)$ der Filterbank ist in Abschnitt 3.2.5 erläutert. Umgekehrt kann man Lifting-Schritte nach Abb. 3.6 und 3.7 stets durch eine äquivalente Filterbank ersetzen. Dies ist ausführlich in Abschnitt 3.2.4 besprochen.

Beispiel: Lifting-Schritte für das Filterbeispiel CDF(3,5) aus Tabelle 2.2: (In jedem Schritt ist der links von „$\mapsto$“ stehende Ausdruck durch den rechts davon stehenden zu ersetzen, was in der Programmiersprache C einfach durch „=“ zu errechnen ist.)

$$s(k) = x(2k), \qquad d(k) = x(2k+1) \tag{3.12}$$

$$s(k) \mapsto s(k) - \frac{1}{3}d(k-1) \tag{3.13}$$

$$d(k) \mapsto d(k) - \frac{1}{8}\Big(9s(k) + 3s(k+1)\Big) \tag{3.14}$$

$$s(k) \mapsto s(k) - \frac{1}{288}\Big(5d(k-2) - 34d(k-1) - 128d(k) + 34d(k+1) - 5d(k+2)\Big) \tag{3.15}$$

$$s(k) \mapsto \frac{3\sqrt{2}}{2}s(k), \qquad d(k) \mapsto \frac{\sqrt{2}}{3}d(k) \tag{3.16}$$

Man erhält jedoch durch das Lifting-Schema auch Wavelet-Transformationen, deren zugehörige Filterpaare auf anderem Weg nur sehr aufwendig zu konstruieren sind, bei denen die Lifting-Schritte jedoch eine einfache, intuitiv einleuchtende Bedeutung haben. So kann man die Vorhersage der Werte $d(k)$ aus den Werten $s(k)$ gegenüber dem Beispiel (3.3) interpretieren als Ergebnis einer Interpolation durch ein Polynom ersten Grades (graphisch eine Gerade). Besteht das Signal aus einem Polynom ersten Grades, d.h. $x(k) = c \cdot k + d$, dann ist diese Vorhersage exakt und $d_{\text{neu}}(k) = 0$. Diese Idee kann auch für Interpolationspolynome höheren Grades verwirklichen. Dies ist in Abschnitt 3.3 ausführlich besprochen.

Bisher wurde alles anhand der ersten Skalierungsstufe besprochen, d.h. es wurde beschrieben, wie die Berechnung von $s_1(k)$ und $d_1(k)$ statt durch die Filterbankoperationen (bzw. (2.36) und (2.37) für $m = 0$) durch die hier angegebenen Lifting-Schritte erfolgen kann. Dies genügt aber, denn man kann die Lifting-Schritte mit dem Ausgang $s_1(k)$ der ersten Stufe als neuem Eingang wiederholen, also mit der Aufspaltung

$$s_2(k) = s_1(2k), \qquad d_2(k) = s_1(2k+1)$$

und anschließender Vorhersage, Korrektur und Normierung $s_2(k)$ und $d_2(k)$ berechnen und dies dann mit dem so berechneten $s_2(k)$ als neuem Eingang fortsetzen.

An dieser Stelle ist es nützlich, die Vorteile des Lifting-Schemas zusammenzustellen (teilweise sind sie schon genannt worden):

(a) Rechenzeitersparnis (etwa um den Faktor 2)

(b) Möglichkeit, Arbeitsspeicher zu sparen. Man benötigt keinen Hilfsspeicher, da die alten Werte überschrieben werden können, d.h. man kann nach entsprechender Modifikation beispielsweise die Ersetzungen in den Gleichungen (3.13) bis (3.16) in C durch ein „+ =“ vornehmen. Die dafür notwendige Adressierung ist im Anhang C besprochen.

(c) einfache, intuitiv einsichtige Konstruktion von Filtern (siehe Abschnitt 3.3)

(d) einfache Behandlung von Randbedingungen (siehe Abschnitt 3.4)

Abschließend wird noch ein weiterer Vorteil des Lifting-Schemas besprochen. Es ermöglicht, die Wavelet-Transformation so durchzuführen, dass als Ergebnis wieder ganze Zahlen auftreten, wenn die ursprünglichen Daten ganze Zahlen sind (was z.B. bei Bilddaten meist der Fall ist). Dies kann zur *verlustfreien* Datenkompression (z.B. für medizinische Röntgenaufnahmen) ausgenutzt werden.

Die Wavelet-Transformation, wie sie durch Gleichungen (2.36) und (2.37) gegeben ist, führt auf Fließkommazahlen, auch wenn man mit ganzen Zahlen $s_0(k)$ startet. Für die Datenkompression werden diese dann durch eine entsprechende Quantisierung (grob für eine hohe Kompressionsrate) in ganze Zahlen umgewandelt. Dies führt zu *Verlusten* bei der Rekonstruktion. In Gleichung (3.3) und (3.5) können jedoch die Werte, die auf der rechten Seite zu den alten addiert werden, vorher auf ganze Zahlen *gerundet* werden. Wenn man bei der Rekonstruktion die auf dieselbe Weise gerundeten Zahlen wieder abzieht, so erhält man die ursprünglichen Werte fehlerfrei zurück.

Für eine formelmäßige Beschreibung dieses Vorgangs führt man üblicherweise die Bezeichnungsweise $\lfloor x \rfloor$ für die größte ganze Zahl ein, die kleiner oder gleich x ist. Die durch *Runden* aus x entstehende Zahl ist somit $\lfloor x + \frac{1}{2} \rfloor$. Damit kann das Beispiel (3.3) und (3.5) für ganze Zahlen umgeschrieben werden zu

$$d_{\text{neu}}(k) = d_{\text{alt}}(k) - \left\lfloor \frac{1}{2}\big(s_{\text{alt}}(k) + s_{\text{alt}}(k+1)\big) + \frac{1}{2} \right\rfloor \tag{3.17}$$

$$s_{\text{neu}}(k) = s_{\text{alt}}(k) + \left\lfloor \frac{1}{4}\big(d_{\text{neu}}(k-1) + d_{\text{neu}}(k)\big) + \frac{1}{2} \right\rfloor \tag{3.18}$$

Die Normierung von $d(k)$ in (3.7) ist in ganzzahliger Arithmetik so nicht möglich. Man kann sie jedoch bei eindimensionalen Daten für $d(k)$ weglassen und für $s(k)$ bei nur bei jedem zweiten Schritt durch Multiplikation mit 2 vornehmen. Weil bei Bilddaten jeweils die Zeilen und die Spalten nacheinander transformiert werden, kann diese Normierung durch Multiplikation mit 2 beim HH-Subband und $\frac{1}{2}$ beim GG-Subband durchgeführt werden. Bei den HG- und GH-Subbändern heben sich die Faktoren $\sqrt{2}$ und $\frac{1}{\sqrt{2}}$ weg.

Dies ist mehrfach an verschiedenen Stellen beschrieben! An den Wiederholungsstellen ersetzen durch einen Verweis auf hier!

Analog können auch Gleichungen (3.13) bis (3.15) für ganze Zahlen umgeschrieben werden. Die abschließende Normierung durch (3.16) kann man dann weglassen oder nach mehreren Schritten durch eine sinnvolle ganze Zahl vornehmen. Oder man berücksichtigt, dass auch die Normierung durch mehrere Lifting-Schritte ausgedrückt werden kann (siehe Gleichung (A.23)).

3.2 Der Zusammenhang Filterbank-Lifting und die Polyphasenmatrix

Faßt man Abschnitt 3.1 zusammen, so wurde dort behauptet:

> Jede Filterbanktransformation (wie sie in Abb. 2.18 dargestellt ist), die den Bedingungen der perfekten Rekonstruktion genügt, läßt sich durch endlich viele Lifting-Schritte gemäß Abb. 3.6 und 3.7 ersetzen. Umgekehrt läßt sich zu jeder Transformation mit endlich vielen Lifting-Schritten gemäß Abb. 3.6 und 3.7 mit FIR-Filtern $P_k(z)$ und $U_k(z)$ eine zugehörige äquivalente Filterbank mit FIR-Filtern $H(z)$, $G(z)$, $\tilde{H}(z)$ und $\tilde{G}(z)$ berechnen, die dann den Bedingungen der perfekten Rekonstruktion genügen.

Diese Behauptung wurde in Abschnitt 3.1 nicht einmal plausibel gemacht, lediglich für die beiden Beispiele des Haar-Wavelets und für die Filter des Wavelets „CDF(3,5)“ (aus Tabelle 2.2) wurden die Lifting-Schritte angegeben. Beim ursprünglichen Beispiel wurde direkt das Lifting-Schema unter dem Gesichtspunkt der Nützlichkeit für die Datenkompression konstruiert und behauptet, dass es zu dem in Abb. 3.4 gezeigten Wavelet gehört.

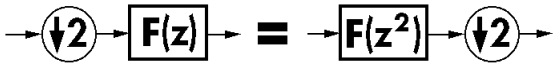
Für praktische Anwendungen kann man sich darauf beschränken, die im Abschnitt 3.1 besprochenen oder in der Literatur angegebenen Lifting-Schritte zu implementieren, ohne zu wissen, wie sie mit den Filterbänken oder gar mit Skalierungsfunktion und Wavelets zusammenhängen. Dieser Zusammenhang erschließt sich in der Tat nicht auf den ersten Blick, sondern bedarf einiger Vorbereitungen. Für den Praktiker, der nur an den Anwendungen der Wavelet-Transformation interessiert ist, ist dieser Abschnitt daher entbehrlich.

Es zeigt sich, dass es einfacher ist, die Berechnung der Filterbank aus den Lifting-Schritten vorzunehmen. Dies wird nach einigen Vorbereitungen in Abschnitt 3.2.4 erklärt. Der umgekehrte Weg kann hier nur angedeutet werden, seine Gangbarkeit wird in Abschnitt 3.2.5 plausibel gemacht.

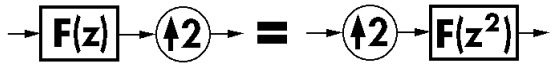
3.2.1 „Edle Gleichungen“

Vergleicht man die Filterbank-Transformation — wie sie beispielsweise in Abb. 2.18 dargestellt ist — mit dem Lifting-Schema, so fällt als ein wesentlicher Unterschied die Reihenfolge von Subsampling oder Upsampling und Filtern auf: Beim Lifting erfolgt das Subsampling bei der Analyse *vor*, das Upsampling bei der Rekonstruktion *nach* der Filter-Operation, bei der Filterbank ist dies gerade umgekehrt. Es ist naheliegend, dass die Reihenfolge beim Lifting vorteilhaft ist, weil die Werte, die weggelassen werden, gar nicht als Eingang bei den Filtern auftauchen.

Rechnerisch wird die Änderung der Reihenfolge Filter und Sub- bzw. Upsampling durch die „Edlen Gleichungen“ beschrieben (man beachte die unterschiedliche Reihenfolge bei Bild und Formel, beim Bild „fließen“ die Daten von links nach rechts, $F(z)(\downarrow 2)X(z)$ bedeutet *zuerst* Subsampling von $X(z)$, *danach* Filtern):



$$F(z)(\downarrow 2) = (\downarrow 2)F(z^2) \tag{3.19}$$



$$(\uparrow 2)F(z) = F(z^2)(\uparrow 2) \tag{3.20}$$

Beweis von (3.19): Man untersucht die Wirkung beider Seiten auf ein beliebiges Eingangssignal $X(z)$. Betrachtet man die linke Seite, so ist $Y_1(z) = (\downarrow 2)X(z)$ im Zeitbereich durch $y_1(n) = x(2n)$ gegeben. Anschließendes Filtern liefert für $Y_2(z) = F(z)Y_1(z) = F(z)(\downarrow 2)X(z)$ im Zeitbereich

$$y_2(n) = \sum_{k=-\infty}^{+\infty} f(k)y_1(n-k) = \sum_{k=-\infty}^{+\infty} f(k)x\Big(2(n-k)\Big)$$

Auf der rechten Seite hat man zunächst $Y_3(z) = F(z^2)X(z)$ zu bilden. Die Impulsantwort des Filters $G(z) = F(z^2)$ ist durch $g(k) = f(\frac{k}{2})$ falls k *gerade* und $g(k) = 0$ falls k *ungerade* gegeben. Damit erhält man

$$y_3(n) = \sum_{k=-\infty}^{+\infty} g(k)x(n-k) = \sum_{k \text{ gerade}} f(\tfrac{k}{2})x(n-k) = \sum_{k=-\infty}^{+\infty} f(k)x(n-2k)$$

Anschließende Unterabtastung liefert dann für $Y_4(z) = (\downarrow 2)Y_3(z) = (\downarrow 2)F(z^2)X(z)$ im Zeitbereich

$$y_4(n) = y_3(2n) = \sum_{k=-\infty}^{+\infty} f(k)x(2n-2k)$$

Damit haben wir die behauptete Übereinstimmung $y_2(n) = y_4(n)$ erhalten. □

Die zweite Identität (3.20) kann man durch ähnliche Überlegungen beweisen. Zur Benennung ist anzumerken, dass hier die in der Literatur übliche englische Bezeichnung „noble identities“ durch einen entsprechenden deutschen Namen ersetzt wurde, obwohl der deutsche Name vermutlich ungewöhnlich ist und in diesem Zusammenhang auch etwas merkwürdig klingt. Die dauernde Verwendung von englischen Begriffen, die nicht völlig in den Sprachgebrauch übergegangen sind, klingt jedoch recht holprig und wurde daher hier vermieden.

3.2.2 Polyphasendarstellung der Filter

Die „Edlen Gleichungen“ können noch nicht direkt auf unsere Filterbank angewandt werden, denn dort tauchen keine Filter der Form $F(z^2)$ auf. Hier hilft man sich mit einem Trick: der *Polyphasendarstellung* von Filtern. Hierunter versteht man die folgende, für jedes beliebige Filter $F(z)$ mögliche Zerlegung

$$F(z) = \sum_{k=-\infty}^{+\infty} f(2k)z^{-2k} + z^{-1} \sum_{k=-\infty}^{+\infty} f(2k+1)z^{-2k} = F_e(z^2) + z^{-1}F_o(z^2) \tag{3.21}$$

mit

$$F_e(z) := \sum_{k=-\infty}^{+\infty} f(2k)z^{-k} = \frac{1}{2}\Big(F(z^{\frac{1}{2}}) + F(-z^{\frac{1}{2}})\Big) \tag{3.22}$$

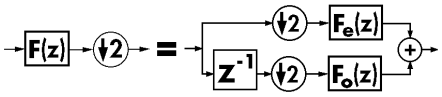
$$F_o(z) := \sum_{k=-\infty}^{+\infty} f(2k+1)z^{-k} = \frac{1}{2}z^{\frac{1}{2}}(F(z^{\frac{1}{2}}) - F(-z^{\frac{1}{2}})\Big) \tag{3.23}$$

Dabei steht der Index e für „even“ und o für „odd“. Die Impulsantwort des Filters $F_e(z)$ enthält nur die Werte der Impulsantwort von $F(z)$ mit *geradem* Index, für $F_o(z)$ nur die Werte mit *ungeradem* Index. Man kann sich also die Impulsantwort von $F_e(z)$ durch eine Unterabtastung der Impulsantwort von $F(z)$ enstanden denken. Zu beachten ist, dass in der Literatur die Konventionen leider unterschiedlich sind, was die Verschiebung anbetrifft, die für $F_o(z)$ notwendig ist, also Vorsicht beim Vergleich von Formeln.

Wir haben damit eine Möglichkeit gewonnen, die erste „noble identity“ zu benutzen, um die Unterabtastung hinter einem beliebigen Filter $F(z)$ in der Reihenfolge nach vorne zu bringen:

$$(\downarrow 2)F(z) = (\downarrow 2)F_e(z^2) + (\downarrow 2)\Big(F_o(z^2)z^{-1}\Big) = F_e(z)(\downarrow 2) + F_o(z)(\downarrow 2)z^{-1} \tag{3.24}$$

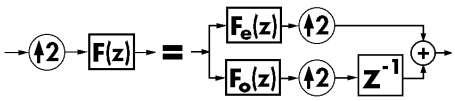
Bildlich dargestellt heißt dies



Bis auf einen Unterschied in der Richtung der Verschiebung taucht hier im linken Teil die Operation „Aufspaltung“ des Lifting-Schemas auf. Analog gilt für das Upsampling

$$F(z)(\uparrow 2) = F_e(z^2)(\uparrow 2) + z^{-1}F_o(z^2)(\uparrow 2) = (\uparrow 2)F_e(z) + z^{-1}(\uparrow 2)F_o(z) \tag{3.25}$$

bildlich dargestellt



Der rechte Teil stimmt mit dem letzten Teil der Rekonstruktion beim Lifting-Schema (bzw. der Synthese bei der „Lazy“-Transformation) überein.

3.2.3 Polyphasenmatrix

Bevor wir den Zusammenhang zwischen der Filterbank-Formulierung und dem Lifting-Schema sichtbar machen können, ist noch eine weitere Vorbereitung notwendig. Aufgrund der hier gewählten Konventionen (der „Wavelet-Bezeichnungsweise“ von Abschnitt 2.4) ist es einfacher, zunächst den Syntheseteil der Filterbank (also den rechten Teil von Abb. 2.18) umzuwandeln. Das Ergebnis ist in Abb. 3.8 dargestellt. Dabei ist (3.25) sowohl für $F(z) = H(z)$ als auch für $F(z) = G(z)$ ausgenutzt worden.

Das Ergebnis läßt sich weiter vereinfachen, wenn man berücksichtigt, dass sich Summation und Upsampling vertauschen lassen nach der Regel

$$(\uparrow 2)\Big(X_1(z) + X_2(z)\Big) = (\uparrow 2)X_1(z) + (\uparrow 2)X_2(z) \tag{3.26}$$

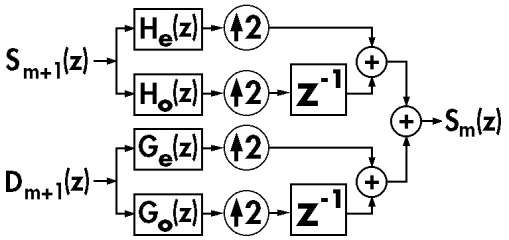


Abbildung 3.8: Umwandlung des Syntheseteils der Filterbank von Abb. 2.18 mit Hilfe der „Edlen Gleichungen“

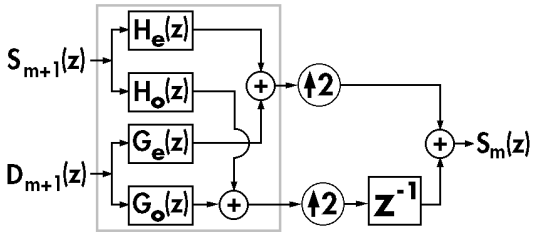


Abbildung 3.9: Vereinfachtes zu Abb. 3.8 äquivalentes Schema. Der grau eingerahmte Teil läßt sich zur Polyphasenmatrix (3.27) zusammenfassen.

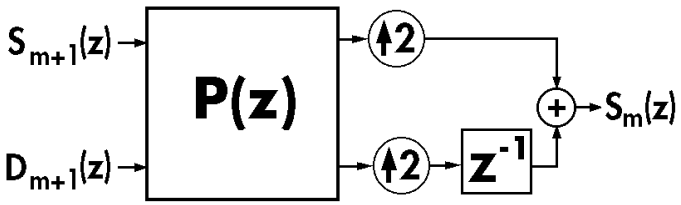


Abbildung 3.10: Zum Syntheseteil der Filterbank von Abb. 2.18 äquivalentes Schema mit der Polyphasenmatrix (3.27)

Ebenso lassen sich Summation und Verschiebung vertauschen. Damit erhalten wir das in Abb. 3.9 gezeigte Schema, das zum Syntheseteil der Filterbank äquivalent ist. Faßt man den Eingang als einem Vektor $\begin{pmatrix} S_{m+1}(z) \\ D_{m+1}(z) \end{pmatrix}$ auf, so kann man den eingerahmten Teil zur *Polyphasenmatrix* $\mathbf{P}(z)$ zusammenfassen:

$$\mathbf{P}(z) := \begin{pmatrix} H_e(z) & G_e(z) \\ H_o(z) & G_o(z) \end{pmatrix} \tag{3.27}$$

Der Syntheseteils der Filterbank von Abb. 2.18 läßt sich somit durch das in Abb. 3.10 gezeigte Schema mit Hilfe dieser Polyphasenmatrix ersetzen.

Für den Analyseteil erhält man aufgrund der Polyphasendarstellung, d.h. wenn man in (3.21) $F(z) = \tilde{H}(z)$ einsetzt und z durch z^{-1} ersetzt

$$\tilde{H}(z^{-1}) = \tilde{H}_e(z^{-2}) + z\tilde{H}_o(z^{-2}) \tag{3.28}$$

und eine analoge Gleichung für $\tilde{G}(z^{-1})$. Damit ergibt sich aus (3.19)

$$(\downarrow 2)\tilde{H}(z^{-1}) = (\downarrow 2)\tilde{H}_e(z^{-2}) + (\downarrow 2)\Big(\tilde{H}_o(z^{-2})z\Big) = \tilde{H}_e(z^{-1})(\downarrow 2) + \tilde{H}_o(z^{-1})(\downarrow 2)z \tag{3.29}$$

und eine entsprechende Gleichung für $(\downarrow 2)\tilde{G}(z^{-1})$. Das Ergebnis der Umwandlung des Analyseteils der Filterbank ist in Abb. 3.11 dargestellt.

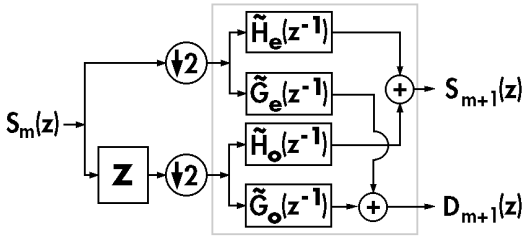


Abbildung 3.11: Umwandlung des Analyseteils der Filterbank von Abb. 2.18 mit Hilfe der „Edlen Gleichungen“. Der grau eingerahmte Teil läßt sich zur Polyphasenmatrix $\tilde{\mathbf{P}}(z^{-1})^T$ (siehe (3.30)) zusammenfassen.

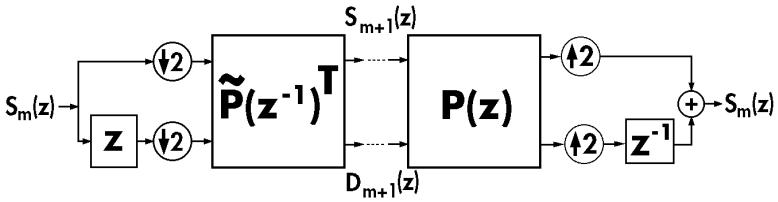


Abbildung 3.12: Zur gesamten Filterbank von Abb. 2.18 äquivalentes Schema mit den Matrizen $\tilde{\mathbf{P}}(z^{-1})^T$ (3.30) links und $\mathbf{P}(z)$ (3.27) rechts

Im Schema von Abb. 3.11 kann auch hier der grau eingerahmte Teil zu einer Matrix zusammengefaßt werden. Gegenüber der durch (3.27) definierten Polyphasenmatrix ist bei der hier auftretenden zu beobachten, dass außer einer Tilde und der Ersetzung von z durch z^{-1} die Matrix transponiert ist. Definiert man die Matrix $\tilde{\mathbf{P}}(z)$ analog zu (3.27), so kommt also bei der Analyse die Matrix

$$\tilde{\mathbf{P}}(z^{-1})^T = \begin{pmatrix} \tilde{H}_e(z^{-1}) & \tilde{H}_o(z^{-1}) \\ \tilde{G}_e(z^{-1}) & \tilde{G}_o(z^{-1}) \end{pmatrix} \tag{3.30}$$

ins Spiel, wobei wie üblich mit ^{T} die transponierte Matrix bezeichnet ist. Die gesamte Filterbank von Abb. 2.18 läßt sich somit durch das in Abb. 3.12 gezeigte Schema mit Hilfe der beiden Polyphasenmatrizen ersetzen.

Daraus ist sofort ersichtlich, dass perfekte Rekonstruktion bedeutet, dass

$$\tilde{\mathbf{P}}(z^{-1})^T \mathbf{P}(z) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.31}$$

und damit

$$\tilde{\mathbf{P}}(z^{-1})^T = \mathbf{P}(z)^{-1} \tag{3.32}$$

Notwendige Bedingung hierfür ist, dass

$$\det\Big(\tilde{\mathbf{P}}(z^{-1})\Big) \cdot \det\Big(\mathbf{P}(z)\Big) = 1 \tag{3.33}$$

Da alle Matrixelemente LAURENT-Polynome sind, müssen auch beide Determinanten LAURENT-Polynome sein. Ihr Produkt kann nur eins sein, wenn jede Determinante ein Vielfaches einer Potenz ist, also

$$\det\Big(\mathbf{P}(z)\Big) = \alpha z^l \tag{3.34}$$

Aus (3.32) kann man die Beziehungen (2.32) und (2.33) mit $m = 2l - 1$ und $\alpha = c$ herleiten. Die Konvention $c = 1$ und $m = -1$ in diesen Beziehungen führt auf $\alpha = 1$ und $l = 0$ und damit auf $\det\Big(\mathbf{P}(z)\Big) = \det\Big(\tilde{\mathbf{P}}(z)\Big) = 1$, was für Rechnungen mit der Polyphasenmatrix sehr bequem ist. Diese Konvention ist beim Beispiel des Haar-Wavelets in Abschnitt 1.2 nicht eingehalten. Daher kann man durch die zugehörigen Lifting-Schritte (3.8) den ursprünglichen Hochpass-Ausgang nur bis auf ein Vorzeichen erhalten.

3.2.4 Von den Lifting-Schritten zur Filterbank

Nach den umfangreichen Vorbereitungen können wir nun ausrechnen, auf welche Filter $H(z)$, $G(z)$, $\tilde{H}(z)$, $\tilde{G}(z)$ vorgegebene Lifting-Schritte führen. Nehmen wir als einfaches Beispiel die durch (3.3) und (3.5) gegebenen und in Abb. 3.3 gezeigten Operationen mit der anschließenden Normierung von (3.7). Wir gehen zu den z-Transformierten über, fassen die Koeffizienten zum Vektor $\begin{pmatrix} S(z) \\ D(z) \end{pmatrix}$ zusammen und drücken jeden der beiden Schritte durch eine Matrix aus. Wir erhalten so für Vorhersage, Korrektur und anschließende Normierung

$$\begin{pmatrix} S_{\text{neu}}(z) \\ D_{\text{neu}}(z) \end{pmatrix} = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{2}\sqrt{2} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{4}(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1}{2}(1+z) & 1 \end{pmatrix} \begin{pmatrix} S_{\text{alt}}(z) \\ D_{\text{alt}}(z) \end{pmatrix} \tag{3.35}$$

Die zuerst auszuführende Multiplikation mit der rechtsstehenden Matrix ändert nur $D_{\text{alt}}(z)$, die anschließende Multiplikation mit der mittleren Matrix ändert nur $S_{\text{alt}}(z)$. Für die Rekonstruktion hat man zu beachten, dass sich bei der Invertierung eines Produkts von Matrizen die Reihenfolge ändert. Die Inversen von Dreiecksmatrizen der hier auftretenden Form sind durch

$$\begin{pmatrix} 1 & 0 \\ t & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 \\ -t & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -t \\ 0 & 1 \end{pmatrix}$$

gegeben. Damit erhalten wir in Übereinstimmung mit Abb. 3.3 für die Rekonstruktion

$$\begin{pmatrix} S_{\text{alt}}(z) \\ D_{\text{alt}}(z) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{2}(1+z) & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{4}(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2}\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} S_{\text{neu}}(z) \\ D_{\text{neu}}(z) \end{pmatrix} \tag{3.36}$$

Ein Vergleich mit Abb. 3.12 ergibt sofort

$$\begin{aligned} \tilde{\mathbf{P}}(z^{-1})^T &= \begin{pmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{2}\sqrt{2} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{4}(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1}{2}(1+z) & 1 \end{pmatrix} \\ &= \frac{1}{2}\sqrt{2} \begin{pmatrix} \frac{3}{2} - \frac{1}{4}(z+z^{-1}) & \frac{1}{2}(1+z^{-1}) \\ -\frac{1}{2}(1+z) & 1 \end{pmatrix} \end{aligned} \tag{3.37}$$

Ein Vergleich von (3.36) mit Abb. 3.12 ergibt

$$\begin{aligned}\mathbf{P}(z) &= \begin{pmatrix} 1 & 0 \\ \frac{1}{2}(1+z) & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{4}(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2}\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix} \\ &= \frac{1}{2}\sqrt{2} \begin{pmatrix} 1 & -\frac{1}{2}(1+z^{-1}) \\ \frac{1}{2}(1+z) & \frac{3}{2} - \frac{1}{4}(z+z^{-1}) \end{pmatrix} \end{aligned} \tag{3.38}$$

Daraus erhält man durch Vergleich

$$H(z) = \frac{1}{2}\sqrt{2}\Big((1+\frac{1}{2}(z+z^{-1})\Big) \tag{3.39}$$

$$G(z) = \frac{1}{4}\sqrt{2}(-\frac{1}{2}z - 1 + 3z^{-1} - z^{-2} - \frac{1}{2}z^{-3}) \tag{3.40}$$

$$\tilde{H}(z) = \frac{1}{4}\sqrt{2}\Big(3 + (z+z^{-1}) - \frac{1}{2}(z^2+z^{-2})\Big) \tag{3.41}$$

$$\tilde{G}(z) = \frac{1}{4}\sqrt{2}(-1 + 2z^{-1} - z^{-2}) \tag{3.42}$$

Diese Filter werden in der Literatur mit CDF(2,2) bezeichnet (siehe beispielsweise [36], dort ist jedoch für $\tilde{G}(z)$ eine andere Vorzeichenkonvention vorgenommen). Wir haben diese Filter bereits in Abschnitt 2.8.4 berechnet. Die Synthesefunktionen sind stückweise durch Polynome 1. Grades gegeben und in Abb. 3.4 gezeigt.

Man sieht an diesem Beispiel, dass man lediglich die Dreiecksmatrizen von Vorhersage und Korrektur sowie die Diagonalmatrizen der Normierung miteinander zu multiplizieren hat, um die Polyphasenmatrix zu erhalten. Aus den einzelnen Matrixelementen der Polyphasenmatrix können dann die einzelnen Filter gemäß (3.21) gewonnen werden. So ist es beispielsweise etwas mühsam, aber nicht schwierig, aus den im Anhang A.1 angegebenen Lifting-Schritten (A.12) bis (A.15) entsprechende Dreiecksmatrizen sowie aus (A.16) eine Diagonalmatrix zu bestimmen und diese Matrizen miteinander zu multiplizieren, um aus der entstehenden Polyphasenmatrix die Filter der zugehörigen Filterbank zu erhalten (die in Tabelle A.1 angegeben sind).

3.2.5 Von der Filterbank zu den Lifting-Schritten

Es ist weitaus schwieriger, einzusehen, wie man von der Polyphasenmatrix zu den einzelnen Lifting-Schritten kommt. Von der Vorgehensweise im Abschnitt 3.2.4 her ist klar, dass man die Polyphasenmatrix in ein *Produkt* von Dreiecksmatrizen der Form

$$\begin{pmatrix} 1 & 0 \\ P(z) & 1 \end{pmatrix} \qquad \text{und} \qquad \begin{pmatrix} 1 & -U(z) \\ 0 & 1 \end{pmatrix}$$

und einer Diagonalmatrix zu zerlegen hat (siehe auch die Abbildungen 3.6 und 3.7). Dies ist mit Hilfe des Algorithmus von Euklid möglich. Dieser dient dazu, den größten gemeinsamen Teiler durch sukzessive Divisionen mit Rest zu berechnen. Dies ist nicht nur für ganze Zahlen möglich, sondern auch für Polynome, denn auch hierfür steht ein entsprechendes Divisionsverfahren zur Verfügung. Auf Einzelheiten kann jedoch hier nicht eingegangen werden, sondern es wird auf [11] verwiesen. Für die praktische Durchführung ist ein leistungsfähiges Computer-Algebra-Programm unentbehrlich, das es ermöglicht, Rechenoperationen mit Polynomen nicht-numerisch durchzuführen.

Festzuhalten ist, dass jede Filterbank mit FIR-Filtern, die den Bedingungen der perfekten Rekonstruktion genügt, durch endlich viele Lifting-Schritte ersetzt werden kann und die *konkrete* Berechnung dieser Schritte *möglich* ist. Für einige Filter sind entsprechende Ergebnisse in den Abschnitten A.1 – A.2 des Anhangs angegeben.

3.3 Deslauriers-Dubuc-Filter

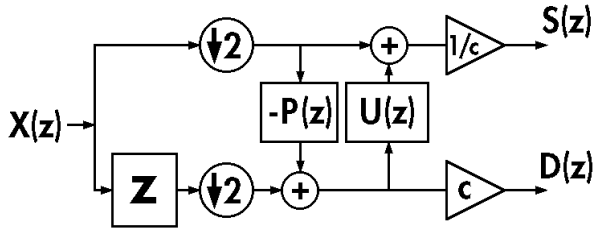


Abbildung 3.13: Lifting-Schema mit einem Vorhersage- und einem Korrekturschritt mit anschließender Normierung

In diesem Abschnitt soll das einführende Beispiel des Abschnitts 3.1 (siehe Abb. 3.3) verallgemeinert werden. Wir wollen uns dabei auf Transformationen beschränken, die mit einem Vorhersage- und einem Korrekturschritt mit anschließender Normierung auskommen, wie das in Abb. 3.13 dargestellt ist. Mit den Methoden des Abschnitts 3.2.4 erhalten wir hierfür die Polyphasenmatrizen

$$\mathbf{P}(z) = \begin{pmatrix} c & -\frac{1}{c}U(z) \\ cP(z) & \frac{1}{c}(1 - P(z)U(z)) \end{pmatrix} \tag{3.43}$$

und

$$\tilde{\mathbf{P}}(z) = \begin{pmatrix} \frac{1}{c}\Big(1 - P(z^{-1})U(z^{-1})\Big) & -cP(z^{-1}) \\ \frac{1}{c}U(z^{-1}) & c \end{pmatrix} \tag{3.44}$$

und damit aus (3.21) und (3.27)

$$H(z) = c + cz^{-1}P(z^2) \tag{3.45}$$

$$\tilde{H}(z) = \frac{1}{c}\Big(1 - P(z^{-2})U(z^{-2})\Big) + \frac{1}{c}z^{-1}U(z^{-2}) \tag{3.46}$$

Der Vorhersagewert einer Konstanten sollte dieselbe Konstante sein, also $P(1) = 1$. Damit ergibt sich aus der Normierungsbedingung $H(1) = 2c = \sqrt{2}$, also $c = \frac{1}{2}\sqrt{2}$. Das erwähnte Beispiel erhält man mit

$$P(z) = \frac{1}{2}(1 + z) \qquad \text{und} \qquad U(z) = \frac{1}{4}(1 + z^{-1}) \tag{3.47}$$

Die Vorhersage ist dabei eine *Interpolation* mit einem Polynom ersten Grades; man kann sie sich dadurch entstanden denken, dass man durch die Nachbarwerte eine Gerade (rechnerisch ein Polynom ersten Grades) legt und als Vorhersagewert den Wert auf der Gerade (also den Funktionswert des Polynoms) nimmt.

Dieses Beispiel wollen wir hier geeignet verallgemeinern. Es bietet sich an, mehr als zwei Nachbarwerte hinzuzuziehen. Geht man symmetrisch vor — und darauf beschränkt

man sich meist — so nimmt man eine gerade Anzahl, also insgesamt $2l$ Nachbarn (mit $l \in \mathbb{N}$). Durch zwei Werte ist eine Gerade und damit ein Polynom ersten Grades bestimmt, durch $2l$ Werte sind die $2l$ Koeffizienten eines Polynoms vom Grade $2l-1$ gegeben. Zur Berechnung der Konstanten des Vorhersageschritts ist es sinnvoll, sich nochmal die Numerierung der Daten vor und nach der Aufspaltung vor Augen zu halten und hierzu Tabelle 3.1 zu betrachten.

$x(-5)$	$x(-4)$	$x(-3)$	$x(-2)^{**}$	$x(-1)$	$x(0)^{*}$	$\underline{x(1)}$	$x(2)^{*}$	$x(3)$	$x(4)^{**}$	$x(5)$
	$s(-2)$		$s(-1)^{**}$		$s(0)^{*}$		$s(1)^{*}$		$s(2)^{**}$	
$d(-3)$		$d(-2)$		$d(-1)$		$\underline{d(0)}$		$d(1)$		$d(2)$

Tabelle 3.1: Änderung der Numerierung durch die Aufspaltung der Daten beim Lifting-Schema

Die Vorgehensweise soll hier am Beispiel $l = 2$ erläutert werden. Wir wollen also den Wert von $d(k)$ aufgrund der Werte der vier nächsten Nachbarn $s(k-1)$, $s(k)$, $s(k+1)$ und $s(k+2)$ bestimmen. Man kann sich klarmachen, dass die dabei auftretenden Koeffizienten nur davon abhängen, ob es sich um den nächsten oder den übernächsten Nachbarn handelt, jedoch vom Index des vorherzusagenden Wertes k unabhängig sind. Im Fall $l = 1$ ergaben sich die Koeffizienten $\frac{1}{2}$ für den rechten und für den linken Nachbarn. Hier, für $l = 2$, spielt allerdings auch der unterschiedliche Abstand der vier Nachbarn eine Rolle. Um diesen in einfacher Weise zu berücksichtigen, gehen wir zunächst von der ursprünglichen Numerierung der Daten $x(k)$ aus. Wir werden den Wert von $d(0) = x(1)$ aufgrund der von $s(1) = x(2)$, $s(2) = x(4)$, $s(0) = x(0)$ und $s(-1) = x(-2)$, vorhersagen. In der Tabelle sind diese Werte mit einem (nächste Nachbarn) bzw. zwei (übernächste Nachbarn) Sternen markiert. Hierzu bestimmen wir ein Polynom dritten Grades

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.48}$$

so, dass $f(t)$ mit den Werten dieser vier Nachbarn übereinstimmt, also

$$\begin{aligned}
f(-2) &= a_0 - 2a_1 + 4a_2 - 8a_3 = x(-2) = s(-1) && (3.49)\\
f(0) &= a_0 = x(0) = s(0) && (3.50)\\
f(2) &= a_0 + 2a_1 + 4a_2 + 8a_3 = x(2) = s(1) && (3.51)\\
f(4) &= a_0 + 4a_1 + 16a_2 + 64a_3 = x(4) = s(2) && (3.52)
\end{aligned}$$

Dies ist ein lineares Gleichungssystem in den vier Unbekannten a_0, a_1, a_2 und a_3. Die Lösung lautet

$$\begin{aligned}
a_0 &= s(0) && (3.53)\\
a_1 &= -\frac{1}{6}s(-1) - \frac{1}{4}s(0) + \frac{1}{2}s(1) - \frac{1}{12}s(2) && (3.54)\\
a_2 &= \frac{1}{8}s(-1) - \frac{1}{4}s(0) + \frac{1}{8}s(1) && (3.55)\\
a_3 &= -\frac{1}{48}s(-1) + \frac{1}{16}s(0) - \frac{1}{16}s(1) + \frac{1}{48}s(2) && (3.56)
\end{aligned}$$

Der Vorhersagewert von $d(0)$ ist damit

$$f(1) = a_0 + a_1 + a_2 + a_3 = \frac{1}{16}\left(9\Big(s(0) + s(1)\Big) - \Big(s(-1) + s(2)\Big)\right) \tag{3.57}$$

Es ist darauf hinzuweisen, dass wir hier mit der expliziten Lösung des linearen Gleichungssystems (3.49) bis (3.52) einen besonders umständlichen Weg zur Bestimmung des Polynoms $f(t)$ gewählt haben, das an den vier Argumenten $t = -2, 0, 2, 4$ mit den vorgegebenen Werten $x(-2)$, $x(0)$, $x(2)$ und $x(4)$ übereinstimmt. Die Lösung einer derartigen Aufgabenstellung heißt *Interpolation*. Das Interpolationsverfahren von LAGRANGE oder von NEWTON ermöglicht es, direkt das entsprechende Interpolationspolynom anzugeben. Wenn man jedoch nicht das Polynom, sondern —- wie hier — nur seinen Funktionswert an einer Zwischenstelle (hier $t = 1$) benötigt, dann kann man am einfachsten das Verfahren von NEVILLE anwenden, das diesen Funktionswert direkt liefert. Hierzu wird auf die Literatur verwiesen.

Da die Konstruktion der Konstanten nicht von der absoluten Position der Werte, sondern nur vom Abstand von den berücksichtigten Nachbarn abhängt, erhalten wir für $d(k)$ den Vorhersagewert

$$\frac{1}{16}\left(9\Big(s(k) + s(k+1)\Big) - \Big(s(k-1) + s(k+2)\Big)\right)$$

also

$$P(z) = \frac{1}{16}\Big(9(1+z) - (z^{-1} + z^2)\Big) \tag{3.58}$$

Zieht man von $d(k)$ diesen Vorhersagewert ab, so hat dies zur Folge, dass Eingangssignale der Form $x(k) = k^n$ für $n = 0, 1, 2, 3$ ein Ausgangssignal $d(k) = 0$ haben. Nach den Erläuterungen des Abschnitts 2.7 heißt dies, dass $H(z)$ in $z = -1$ eine vierfache Nullstelle hat.

Allgemein erhält man durch die Berücksichtigung von $p = 2l$ Nachbarn bei der Vorhersage eine p-fache Nullstelle des Synthesefilters $H(z)$, das die Regularität von Skalierungsfunktion und Wavelet für die Synthese bestimmt.

Durch $P(z)$ sind $H(z)$ und $\tilde{G}(z)$ festgelegt, $\tilde{H}(z)$ und $G(z)$ hängen jedoch auch vom Korrekturschritt, also von $U(z)$ ab. Ein Verzcht auf einen Korrekturschritt, d.h. $U(z) = 0$ würde auf $\tilde{H}(z) = \sqrt{2}$ führen. Wir verlangen aber, dass $\tilde{H}(z)$ eine $\tilde{p}$-fache Nullstelle in $z = -1$ hat mit $1 \le \tilde{p} \le p$, wobei p die Vielfachheit die Vielfachheit der Nullstelle des Synthesefilters ist.

Leider ist die Konstruktion eines Korrekturfilters $U(z)$ nicht so intuitiv durchsichtig, wie dies beim Vorhersagefilter $P(z)$ der Fall ist. Eine längere Überlegung zeigt, dass die geeigneten Korrekturfilter bis auf einen Faktor $\frac{1}{2}$ und die Reihenfolge mit den Vorhersagefiltern $P(z)$ übereinstimmen müssen. Die Wahl

$$U(z) = \frac{1}{2}P(z^{-1}) \tag{3.59}$$

garantiert, dass $\tilde{H}(z)$ eine $2\tilde{l}$-fache Nullstelle in $z = 1$ hat, wenn $P(z)$ ein Vorhersagefilter ist, das durch die Interpolation mit $2\tilde{l}$ Nachbarn entsteht. Dabei kann für den Korrekturschritt nach Gleichung (3.59) ein Vorhersagefilter mit weniger Nachbarn gewählt werden als zur eigentlichen Vorhersage, d.h. mit $\tilde{l} \le l$. Im Beispiel, das durch Abb. 3.3 charakterisiert ist, haben beide Filter $H(z)$ und $\tilde{H}(z)$ eine zweifache Nullstelle in $z = -1$. Wir können den entsprechenden Korrekturschritt für unser neues Beispiel verwenden und erhalten hierfür folgende Lifting-Schritte:

Aufspaltung:

$$s(k) = x(2k), \qquad d(k) = x(2k+1) \tag{3.60}$$

Vorhersage:

$$d(k) \mapsto d(k) - \frac{1}{16}\Big(9\Big(s(k) + s(k+1)\Big) - \Big(s(k-1) + s(k+2)\Big)\Big) \tag{3.61}$$

Korrektur:

$$s(k) \mapsto s(k) + \frac{1}{4}\Big(d(k) + d(k-1)\Big) \tag{3.62}$$

Normierung:

$$s(k) \mapsto \sqrt{2}s(k), \qquad d(k) \mapsto \frac{1}{\sqrt{2}}d(k) \tag{3.63}$$

Bei Verwendung ganzzahliger Arithmetik ist es auch hier sinnvoll, bei eindimensionalen Daten die Normierung nur bei jedem zweiten Schritt durch Multiplikation mit 2 bzw. $\frac{1}{2}$ vorzunehmen. Bei Bilddaten sind nach Transformation von Zeilen und Spalten die Koeffizienten des HH-Bandes mit 2, die des GG-Bandes mit $\frac{1}{2}$ zu multiplizieren, die des HG- und GH-Bandes bleiben unverändert.

Aus (3.27) und (3.21) kann man die zugehörigen Filter $H(z)$ und $\tilde{H}(z)$ erhalten. Die auftretenden Filterkonstanten sind in Tabelle 3.2 angegeben. Bemerkenswert ist, dass auch dabei bis auf einen Faktor $\sqrt{2}$ nur Brüche auftreten, in deren Nenner Zweierpotenzen stehen. Gilbert Strang hat diese auf andere Art berechnet und für die zugehörigen Wavelets den Namen „Binlets“ vorgeschlagen. $H(z)$ hat eine vierfache, $\tilde{H}(z)$ hat eine zweifache Nullstelle in $z = -1$.

k	$\frac{1}{\sqrt{2}}\tilde{h}(k)$	$\frac{1}{\sqrt{2}}h(k)$
0	$\frac{46}{64}$	$\frac{16}{32}$
±1	$\frac{16}{64}$	$\frac{9}{32}$
±2	$-\frac{8}{64}$	0
±3	0	$-\frac{1}{32}$
±4	$\frac{1}{64}$	0

Tabelle 3.2: (9/7)-„Binlet“-Filter von Strang (identisch mit DD(4,2))

Die hier vorgestellten, durch Interpolation entstehenden Filter wurden eingehend von Deslauriers und Dubuc untersucht. Die durch die entsprechenden Lifting-Schritte konstruierten Wavelets werden daher Deslauriers-Dubuc-Wavelets genannt. Als Kurzbezeichnung wird hier DD$(p, \tilde{p})$ vorgeschlagen. Man beschränkt sich in der Regel auf gerade p und $\tilde{p}$ mit $\tilde{p} \leq p$. Dabei gibt p die Länge des Vorhersagefilters und gleichzeitig die Vielfachheit der Nullstelle von $H(z)$ in $z = -1$ an; $\tilde{p}$ ist die Länge des Korrekturfilters und die Vielfachheit der Nullstelle von $\tilde{H}(z)$ in $z = -1$. Das durch Abb. 3.3 charakterisierte Beispiel erhält also die Bezeichnung DD(2,2). Es stimmt mit einem nach anderen Prinzipien berechneten Filter überein und wurde daher früher mit CDF(2,2) bezeichnet. Die Filter zu Strangs (9/7)-Binlet stimmen mit DD(4,2) überein.

An den Vorhersageschritt mit vier Nachbarn (3.61) kann man auch einen Korrekturschritt derselben Länge anschließen: das Filter DD(4,4) erhält man durch Vorhersage (3.61) und die folgende Korrektur

$$s(k) \mapsto s(k) + \frac{1}{32}\Big(9\Big(d(k) + d(k-1)\Big) - \Big(d(k-2) + d(k+1)\Big)\Big) \tag{3.64}$$

Die Normierung bleibt unverändert. Die zugehörigen Tiefpassfilter $H(z)$ und $\tilde{H}(z)$ sind in Tabelle 3.3 aufgeführt, auch hier treten bis auf die Normierung nur Zweierpotenzen im Nenner auf.

k	$\frac{1}{\sqrt{2}}\tilde{h}(k)$	$\frac{1}{\sqrt{2}}h(k)$
0	$\frac{348}{512}$	$\frac{16}{32}$
±1	$\frac{144}{512}$	$\frac{9}{32}$
±2	$-\frac{63}{512}$	0
±3	$-\frac{16}{512}$	$-\frac{1}{32}$
±4	$\frac{18}{512}$	0
±5	0	0
±6	$-\frac{1}{512}$	0

Tabelle 3.3: (13/7)-"Binlet"-Filter von Strang (übereinstimmend mit DD(4,4))

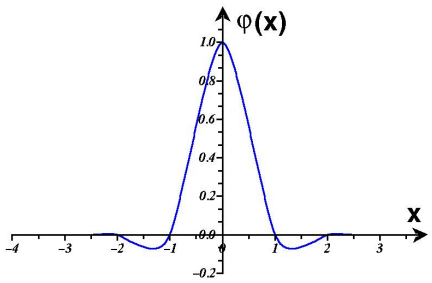


Abbildung 3.14: Synthese-Skalierungsfunktion zum DD(4,2)-Binlet

Wie aus Gleichung (3.45) ersichtlich ist, haben die Deslauriers-Dubuc-Filter für alle $k \in \mathbb{Z}$ die Eigenschaft

$$h(2k) = \begin{cases} 0 & \text{falls } k \neq 0 \\ \frac{1}{2}\sqrt{2} & \text{falls } k = 0 \end{cases} \tag{3.65}$$

und die zugehörigen Skalierungsfunktionen erfüllen für alle $k \in \mathbb{Z}$

$$\varphi(k) = \begin{cases} 0 & \text{falls } k \neq 0 \\ 1 & \text{falls } k = 0 \end{cases} \tag{3.66}$$

Skalierungsfunktionen, die diese Bedingung erfüllen, nennt man *interpolierend*. Hierfür ist Gleichung (3.65) notwendige, aber nicht hinreichende Bedingung. Die Synthese-Skalierungsfunktion zum Binlet ist interpolierend, wie auch aus Abb. 3.14 ersichtlich ist.

Wenn die Skalierungsfunktion interpolierend ist, dann hat dies für die Darstellung von kontinuierlichen Signalen eine bemerkenswerte Folge. Betrachten wir ein kontinuierliches Signal $f(x)$, das mit verschobenen Versionen der Skalierungsfunktion dargestellt werden kann, also

$$f(x) = \sum_{k=-\infty}^{+\infty} s_0(k)\varphi(x-k) \tag{3.67}$$

In Abschnitt 2.5.3 wurde ausdrücklich gewarnt, dass die Abtastwerte $f(k)$ (mit $k \in \mathbb{Z}$) eines solchen Signals im allgemeinen nicht exakt mit den Koeffizienten $s_0(k)$ dieser Entwicklung übereinstimmt. Die Annahme einer solchen Übereinstimmung ist allenfalls als Näherung zu rechtfertigen. Für interpolierende Skalierungsfunktionen ist diese Übereinstimmung jedoch exakt erfüllt, d.h. wenn φ der Bedingung (3.66) genügt, dann gilt

$$s_0(k) = f(k) \tag{3.68}$$

exakt. Dies kann man sich klar machen, wenn man in (3.67) $x = l \in \mathbb{Z}$ setzt und (3.66) benutzt. Die Deslauriers-Dubuc-Skalierungsfunktionen sind interpolierend. Da die Näherung

$$s_0(k) \approx f(k)$$

grundlegend dafür ist, dass die Filterbankoperationen tatsächlich die Koeffizienten einer Wavelet-Entwicklung liefern, ist nicht auszuschließen, daß interpolierende Skalierungsfunktionen bzw. die dazu gehörigen Filterbankoperationen oder Lifting-Schritte in manchen praktischen Anwendungen Vorteile haben gegenüber Skalierungsfunktionen, die diese Eigenschaft nicht haben.

Im Anhang A.3 sind weitere Deslauriers-Dubuc-Filter angegeben.

3.4 Einfache Behandlung der Daten am Rand

In Abschnitt 2.9 war klar geworden, dass bei einer korrekten Implementierung der symmetrischen bzw. antisymmetrischen Fortsezung am Rand eine Vielzahl verwickelter Bedingungen zu beachten ist. Das Lifting-Schema bietet eine sehr einfache Möglichkeit, ohne Beeinträchtigung der perfekten Rekonstruktion Randbedingungen festzulegen. Jede nur denkbare Fortsetzung ist möglich. Man hat einen einzigen Punkt zu beachten: Beim zugehörigen Schritt zur Rekonstruktion (bei dem in Abb. 3.7 gegenüber Abb. 3.6 $-S_i(z)$ oder $-T_i(z)$ durch $+S_i(z)$ bzw. $+T_i(z)$ ersetzt ist) hat man die Fortsetzung so vorzunehmen, wie dies beim entsprechenden Analyse-Schritt getan wurde. Dadurch werden bei der Rekonstruktion genau dieselben Terme wieder abgezogen, die bei der Analyse addiert wurden.

Ein sinnvolles Fortsetzungsprinzip ist die symmetrische Fortsetzung bei jedem einzelnen Schritt. Dabei kann nach Gutdünken die Fortsetzung ohne oder mit Wiederholung gewählt werden. Die für die Analyse getroffene Wahl muss lediglich bei der Rekonstruktion beibehalten werden. Es ist allerdings darauf hinzuweisen, dass das Ergebnis einer symmetrischen Fortsetzung bei jedem einzelnen Lifting-Schritt nicht mit dem einer einmaligen symmetrischen Fortsetzung bei der Filterbank-Transformation übereinstimmt.

Weiterhin kann man sich überlegen, welcher Sinn hinter den einzelnen Lifting-Schritten steht und diese Schritte am Rand sinngemäß abändern. Dies soll hier an einem Beispiel verdeutlicht werden. Wenn eine gerade Anzahl N von Daten $x(0)$, $x(1)$, $x(2) \ldots x(N-1)$ vorliegt, dann kann zunächst die Lazy-Wavelet-Transformation (3.1) für $k = 0, 1, 2, \ldots \frac{N}{2} - 1$ unverändert übernommen werden. Die lineare Vorhersage aufgrund der benachbarten Werte in Gleichung (3.3) ist jedoch nur für $k = 0, 1, 2, \ldots \frac{N}{2} - 2$ möglich, für den letzten Wert $k = \frac{N}{2} - 1$ ist eine Sonderbehandlung notwendig, denn $d(\frac{N}{2} - 1) = x(N-1)$, und es gibt keinen nachfolgenden Nachbarwert. Eine lineare Extrapolation aufgrund der beiden vorherigen Werte führt auf

$$d_{\text{neu}}(\frac{N}{2} - 1) = d_{\text{alt}}(\frac{N}{2} - 1) - \frac{3}{2} s_{\text{alt}}(\frac{N}{2} - 1) + \frac{1}{2} s_{\text{alt}}(\frac{N}{2} - 2)\big) \tag{3.69}$$

Für endlich viele Daten läßt sich die Bedingung (3.6) nicht vollständig erfüllen. Als Ersatz hierfür ist es sinnvoll, zu verlangen, dass bei der Tiefpassfilterung für ein Signal der Form

$$\ldots, 1, -1, 1, -1, 1, -1, 1, -1, \ldots$$

als Ergebnis Null herauskommt. D.h. für $x(k) = (-1)^k$ verlangen wir $s_{\text{neu}}(k) = 0$. Dies entspricht der Forderung nach einer Nullstelle von $\tilde{H}(z)$ in $z = 1$ (siehe Abschnitt 2.7). Es trägt zu einer Verbesserung der Regularität der Analyse-Skalierungsfunktion bei, wenn man zusätzlich verlangt, daß die Tiefpassfilterung auch für Signale der Form

$$\ldots, 2, -1, 0, -1, 2, -3, 4, -5, 6 \ldots$$

auf Null führt, d.h. $s_{\text{neu}}(k) = 0$ auch für $x(k) = (-1)^k \cdot k$ (was der Forderung nach einer zweifachen Nullstelle von $\tilde{H}(z)$ in $z = 1$ entspricht). Diese beiden Forderungen führen für unendlich ausgedehnte Signale auf die Korrektur (3.5). Für endlich viele Daten liefern sie am linken Rand

$$s_{\text{neu}}(0) = s_{\text{alt}}(0) + \frac{3}{4} d_{\text{neu}}(0) - \frac{1}{4} d_{\text{neu}}(1) \tag{3.70}$$

Am rechten Rand ist keine Sonderbehandlung notwendig.

Für längere Filter kann man weitere Forderungen der Art stellen, dass die Hochpassfilterung von Signalen der Form $x(k) = k^n$ für $n = 1, 2, \ldots p-1$ mit möglichst großem p auch bei endlich vielen Daten auf das Ergebnis Null führt (was der Forderung nach einer p-fachen Nullstelle von $H(z)$ in $z = 1$ entspricht). Zudem kann man fordern, dass die Tiefpassfilterung von Signalen der Form $x(k) = (-1)^k \cdot k^n$ für $n = 1, 2, \ldots \tilde{p} - 1$ auch am Rand auf Null führt (was der Forderung nach einer $\tilde{p}$-fachen Nullstelle von $\tilde{H}(z)$ in $z = 1$ entspricht). Bei den im Abschnitt 3.3 behandelten Deslauriers-Dubuc-Filtern ist dies besonders einfach durchzuführen. Die Vorhersage durch Interpolation wird am Rand durch eine Vorhersage durch Extrapolation ersetzt. Analog wird die Korrektur am Rand modifiziert. Beispiele für Randfilter, die nach diesem Prinzip berechnet wurden, sind im Anhang A.3 aufgeführt.

Diese Randfilter haben jedoch erhebliche Nachteile in der Praxis. Beim Binlet führen bei der Bildkompression Quantisierungsfehler am rechten oder unteren Bildrand zu extrem hohen Abweichungen vom ursprünglichen Bild bei der Rekonstruktion. Dies scheint in der Literatur kaum diskutiert. Demgegenüber hat die symmetrische Fortsetzung mit Wiederholung bei Vorhersage und Korrektur am Rand keine derartigen Nachteile.

3.5 Beliebige Filterbank als Startpunkt für das Lifting-Schema

Bisheriger Ausgangspunkt für das Lifting-Schema war ausschließlich die zum „Lazy"-Wavelet gehörige Filterbank-Transformation (siehe Abb. 3.1) mit den trivialen Filtern $H(z) = \tilde{H}(z) = 1$ und $G(z) = \tilde{G}(z) = z^{-1}$, die selbstverständlich den Bedingungen der perfekten Rekonstruktion (2.30) und (2.31) genügen. Dies läßt sich verallgemeinern: Man kann eine beliebige FIR-Filterbank als Startpunkt wählen, deren Filter den Bedingungen der perfekten Rekonstruktion genügt. Man kann die Filterbank durch einen Lifting-Schritt nach dem Subsampling abändern. Wenn man diesen durch einen entsprechenden Schritt mit umgekehrtem Vorzeichen wieder kompensiert, dann bleibt die perfekte Rekonstruktion erhalten. Man erhält also damit eine neue FIR-Filterbank. Dies ist in Abb 3.15 illustriert.

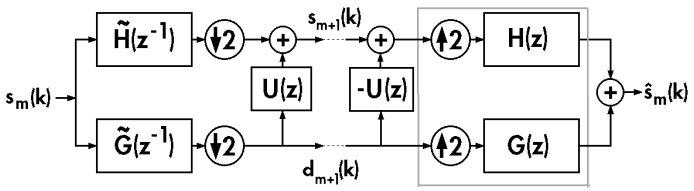


Abbildung 3.15: Lifting-Schema: Beliebige FIR-Filterbank als Ausgangspunkt für das Lifting-Schema

Nur der Tiefpassausgang $s_{m+1}(k)$ wird dadurch geändert. Das Filter $U(z)$ ist ein völlig beliebiges FIR-Filter.

Es erhebt sich nun die Frage, welcher neuen Filterbank das dargestellte Schema (alte Filterbank und Lifting-Schritt) entspricht. Diese kann mit den Methoden des Abschnitts 3.2 beantwortet werden. Der grau eingerahmte Teil der Filterbank (der Rekonstruktionsteil der alten Filterbank) wird durch die Polyphasenmatrix beschrieben. Zusammen mit dem Rekonstruktionsteil des Lifting-Schritts erhalten wir die neue Polyphasenmatrix

$$\mathbf{P}_{\text{neu}}(z) = \mathbf{P}(z) \begin{pmatrix} 1 & -U(z) \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} H_e(z) & G_e(z) \\ H_o(z) & G_o(z) \end{pmatrix} \begin{pmatrix} 1 & -U(z) \\ 0 & 1 \end{pmatrix} \tag{3.71}$$

Ausrechnen und Vergleich mit (3.27) liefert die neuen Filter

$$G_{\text{neu}}(z) = G(z) - H(z)U(z^2), \qquad \tilde{H}_{\text{neu}}(z) = \tilde{H}(z) + \tilde{G}(z)U(z^{-2}) \tag{3.72}$$

$H(z)$ und $\tilde{G}(z)$ bleiben unverändert. Man kann sich sogar davon überzeugen (dies ist allerdings ein klein wenig schwieriger), daß alle FIR-Filter mit unverändertem $H(z)$ und $\tilde{G}(z)$, die der Bedingung der perfekten Rekonstruktion genügen, auf diese Weise aus einer gegebenen Filterbank erzeugt werden können. Dies ist die Aussage des Lemmas von Vetterli und Herley.

Der in Abb 3.15 gezeigte Lifting-Schritt entspricht beim Start mit der trivialen Filterbank des „Lazy“ Wavelets einem Korrektur-Schritt (update). Man kann nun analog auch einen Vorhersage-Schritt (predict) nach einer beliebigen Filterbank vornehmen. Dies ist in Abb. 3.16 gezeigt. Nur der Hochpassausgang $d_{m+1}(k)$ wird geändert. Man spricht dann von einem dualen Lifting-Schritt. Dabei ist $T(z)$ ein völlig beliebiges FIR-Filter.

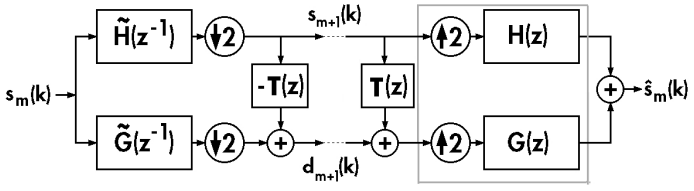


Abbildung 3.16: duales Lifting mit einer beliebigen FIR-Filterbank als Ausgangspunkt

Auch hier kann man sich fragen, welchen neuen Filtern dieses Schema entspricht. Der grau eingerahmte Teil entspricht hier wieder der alten Polyphasenmatrix $\mathbf{P}(z)$. Für die neue Polyphasenmatrix erhält man

$$\mathbf{P}_{\text{neu}}(z) = \mathbf{P}(z) \begin{pmatrix} 1 & 0 \\ T(z) & 1 \end{pmatrix} = \begin{pmatrix} H_e(z) & G_e(z) \\ H_o(z) & G_o(z) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ T(z) & 1 \end{pmatrix} \tag{3.73}$$

und für die neuen Filter heißt dies

$$H_{\text{neu}}(z) = H(z) + G(z)T(z^2), \qquad \tilde{G}_{\text{neu}}(z) = \tilde{G}(z) - \tilde{H}(z)T(z^{-2}) \tag{3.74}$$

und $G(z)$ und $\tilde{H}(z)$ bleiben unverändert. Man erhält in der Tat alle FIR-Filter mit perfekter Rekonstruktion und unverändertem $G(z)$ und $\tilde{H}(z)$ durch eine derartige Konstruktion.

Bei der trivialen Filterbank des „Lazy“ Wavelets haben wir mit $H(z) = 1$ keine Funktion als Skalierungsfunktion und damit auch keine Wavelet-Funktion. Man könnte nun auch Distributionen als Wavelets zulassen, dies würde aber den hier vorgegebenen mathematischen Rahmen sprengen. In diesem Abschnitt ist jedoch eine allgemeinere Filterbank als Startpunkt für das Lifting-Schema zugelassen. Wir können nun eine solche Filterbank betrachten, bei der (durch den in Abschnitt 2.4 besprochenen Grenzübergang) durch $H(z)$ eine Skalierungsfunktion und daraus durch $G(z)$ ein Wavelet gegeben ist. Es erhebt sich die Frage, wie sich Skalierungsfunktionen und Wavelet ändern, wenn wir einen Lifting-Schritt oder einen dualen Lifting-Schritt vornehmen.

Betrachten wir zunächst einen Lifting-Schritt (nach Abb. 3.15). Da $H(z)$ sich nicht ändert, bleibt auch die Skalierungsfunktion $\varphi(x)$ unverändert. Die Änderung von $G(z)$ verursacht folgende Änderung der Fourier-Transformierten des Wavelets (siehe (2.24)):

$$\begin{aligned}
\Psi_{\text{neu}}(\omega) &= \frac{1}{\sqrt{2}} G_{\text{neu}}(e^{j\frac{\omega}{2}})\Phi(\frac{\omega}{2}) \\
&= \frac{1}{\sqrt{2}} G(e^{j\frac{\omega}{2}})\Phi(\frac{\omega}{2}) - \frac{1}{\sqrt{2}} H(e^{j\frac{\omega}{2}})U(e^{j\omega})\Phi(\frac{\omega}{2}) \\
&= \Psi(\omega) - U(e^{j\omega})\Phi(\omega)
\end{aligned} \tag{3.75}$$

Für das Wavelet selbst erhalten wir damit

$$\psi_{\text{neu}}(x) = \psi(x) - \sum_{k=-\infty}^{+\infty} u(k)\varphi(x-k) \quad \text{wobei} \quad U(z) = \sum_{k=-\infty}^{+\infty} u(k)z^{-k} \tag{3.76}$$

Bemerkenswert ist, daß die Änderung des Wavelets durch die Skalierungsfunktion der ursprünglichen Skalierungsstufe $\varphi(x-k)$ beschrieben wird. Man beachte, daß das Wavelet ursprünglich durch die Skalierungsfunktion der nächst feineren Sklaierungsstufe $\varphi(2x-k)$ erzeugt wird (siehe (2.27)).

Für die Änderung des Wavelets beim dualen Lifting erhalten wir entsprechend

$$\begin{aligned}
\tilde{\Psi}_{\text{neu}}(\omega) &= \frac{1}{\sqrt{2}} \tilde{G}(e^{j\frac{\omega}{2}})\tilde{\Phi}(\frac{\omega}{2}) - \frac{1}{\sqrt{2}} \tilde{H}(e^{j\frac{\omega}{2}})T(e^{-j\omega})\tilde{\Phi}(\frac{\omega}{2}) \\
&= \tilde{\Psi}(\omega) - T(e^{-j\omega})\tilde{\Phi}(\omega)
\end{aligned} \tag{3.77}$$

und damit

$$\tilde{\psi}_{\text{neu}}(x) = \tilde{\psi}(x) - \sum_{k=-\infty}^{+\infty} t(-k)\tilde{\varphi}(x-k) \quad \text{wobei} \quad T(z) = \sum_{k=-\infty}^{+\infty} t(k)z^{-k} \tag{3.78}$$

3.6 „Schachbrett“-Abtastung bei der Bildtransformation

3.6.1 Abtastschema und einfache Lifting-Schritte

Es wurde bereits in Abschnitt 1.6 darauf hingewiesen, dass die bisherige unsymmetrische zeilen- und spaltenweise Transformation zweidimensionaler Daten Nachteile mit sich

bringt. Das Lifting-Schema erlaubt es, ohne großen Aufwand geeignete Wavelettransformationen ohne diesen Nachteil durchzuführen. Zunächst ist die Operation „Aufspaltung“ zu verallgemeinern. Bei eindimensionalen Daten hatten wir in Werte mit geradem Index und ungeradem Index aufgeteilt (bei Unterabtastung die mit geradem Index übrigbehalten). Bei Bildern kann man entsprechend der Aufteilung eines Schachbretts in weiße und schwarze Felder vorgehen und zur Unterabtastung die Werte übrigbehalten, die zu den weißen Feldern gehören. Für die entsprechende Transformation wurde in [37] der Name „red-black wavelet transform“ gewählt, in der englischsprachigen Literatur ist sonst der Name „Quincunx-Abtastung“ üblich. Hier wird der bisher ungebräuchliche Name „Schachbrett-Abtastung“ vorgeschlagen.

Die „Lazy“ Transformation besteht also in der ersten Skalierungsstufe in einer Aufspaltung in die „weißen“ Werte $s_1(i,k)$ und die „schwarzen“ Werte $d_1(i,k)$. Zur Ersparnis von Speicherplatz führt man die Transformation sinnvollerweise im ursprünglichen Bild mit den Werten $x(i,k)$ (Helligkeit in der i. Zeile und k. Spalte) durch. Wir haben also („Aufspaltung“)

$$x(i,k) = \begin{cases} s_1(i,k) & \text{falls } i+k \text{ gerade} \\ d_1(i,k) & \text{falls } i+k \text{ ungerade} \end{cases} \tag{3.79}$$

Ein sehr einfacher Vorhersagewert ist der Mittelwert über die vier nächsten Nachbarn. In der ersten Sakalierungsstufe heißt dies für den Schritt „Vorhersage“ (nur falls $i+k$ ungerade!)

$$x(i,k) \mapsto x(i,k) - \frac{1}{4}\Big(x(i-1,k) + x(i+1,k) + x(i,k-1) + x(i,k+1)\Big) \tag{3.80}$$

Diese Vorhersage ist überall dort exakt, d.h. führt zu verschwindenden Koeffizienten, wo die ursprümglichen Intensitätswerte eine *lineare* Funktion der Bildkoordinaten ist, also eine Beziehung der Form

$$x(i,k) = \alpha \cdot i + \beta \cdot k + \gamma \tag{3.81}$$

Große Abweichungen vom Vorhersagewert und damit große Absolutbeträge für die Koeffizienten erwartet man an den *Kanten* des Bildes. Dies wird auch in den gezeigten Abbildungen deutlich.

Um zu erreichen, dass der Mittelwert über alle weißen Felder mit dem ursprünglichen über alle Felder übereinstimmt, ist ein Korrektur-Schritt („update“) notwendig (nur für die „weißen“ Felder, also wenn $i+k$ gerade)

$$x(i,k) \mapsto x(i,k) + \frac{1}{8}\Big(x(i-1,k) + x(i+1,k) + x(i,k-1) + x(i,k+1)\Big) \tag{3.82}$$

Diese Eigenschaft ist exakt nur für unendlich ausgedehnte Bilder erfüllt. In der Praxis setzt man das Bild symmetrisch am Rand fort. Es genügt, dass der Mittelwert näherungsweise erhalten ist. Man kann jedoch auch hier geeignete Randfilter berechnen (wie dies in Abschnitt 2.9 beschrieben ist).

Diese Korrektur gewährleistet, dass eine Intensitätsverteilung der höchstmöglichen Frequenz der Form $x(i,k) = (-1)^{i+k}$ ($+1$ auf den weißen, -1 auf den schwarzen Feldern) zum Ergebnis $s_1(i,k) = 0$ führt. Dies ist eine unerläßliche Eigenschaft für eine Tiefpassfilterung. Die Operationen der 1. Stufe sind in Abb 3.17 verdeutlicht. Das Ergebnis ihrer Anwendung auf das Testbild „Lena“ ist in Abb 3.18 gezeigt. Dabei sind die „weißen“ und die „schwarzen“ Pixel jeweils zu einem rechteckigen Bild zusammengeschoben, da die

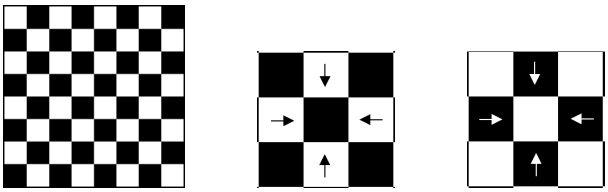


Abbildung 3.17: Zur Schachbrett-Transformation: Vorhersage der „schwarzen“ Werte (Mitte) und Korrektur der „weißen“ Werte (rechts) mit Hilfe des Mittelwerts über die nächsten Nachbarn

Abbildung 3.18: Anwendung der Schachbrett-Transformation auf das Testbild „Lena“. Die „weißen“ Pixel (oben) und die „schwarzen“ Pixel (unten) sind zu einem Rechteck zusammengeschoben. Der Kontrast für die Pixel $d_1(i,k)$ wurde verstärkt.

halbe Anzahl von Pixeln eines quadratischen Bildes nicht wieder ein quadratisches Bild ergibt.

Danach, in der zweiten Skalierungsstufe, bleiben die „schwarzen“ Werte unverändert, die „weißen“ werden der nächsten Transformation unterzogen. Dazu werden sie erneut aufgespaltet. Jeder zweite Wert wird den schwarzen der zweiten Stufe zugeschlagen und nur jeder zweite Wert bei den weißen gelassen. Man denkt sich dafür am einfachsten das „Schachbrett“ um 45^0 gedreht und ignoriert die schwarzen Felder. In der Abb. 3.19 ist dies verdeutlicht, die neuen „schwarzen“ Werte sind nur grau gefärbt. Auch von ihnen wird wieder der Vorhersagewert, der Mittelwert über die nächsten verbliebenen weißen Nachbarn, abgezogen. Zu beachten ist, dass diese jetzt in Diagonalrichtung liegen, da die alten schwarzen Werte nicht mehr benutzt werden. Anschließend erfolgt die Korrektur („update“), indem von den verbliebenen weißen Werten der halbe Mittelwert über die nächsten neuen schwarzen (also in der Abb. grauen) Werte abgezogen wird. Diese Nachbarn befinden sich ebenfalls in Diagonalrichtung. Für die Abbildung sind Zeilen und Spalten mit 0 beginnend numeriert. Das Ergebnis der Anwendung von zwei Skalierungsstufen auf das Testbild „Lena“ ist in Abb. 3.20 gezeigt.

Formelmäßig heißt dies für die zweite Stufe:

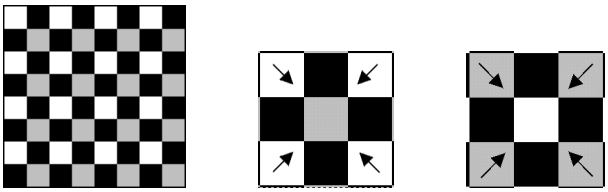


Abbildung 3.19: Schachbrett-Transformation in der zweiten Skalierungsstufe: Aufteilung der weißen Werte in weiße und graue. Bei der Vorhersage der grauen Werte (Mitte) und Korrektur der Werte (rechts) mit Hilfe des Mittelwerts über die nächsten Nachbarn bleiben die schwarzen Werte unberücksichtigt.

Abbildung 3.20: Anwendung der Schachbrett-Transformation mit zwei Skalierungsstufen auf das Testbild „Lena“. Die Zahl der „weißen“ Pixel (oben links) und die der „grauen“ Pixel (oben rechts) beträgt ein Viertel der Gesamtzahl des ursprünglichen Bildes. Die entsprechenden Koeffizienten können daher als verkleinerte Version des ursprünglichen quadratischen Bildes dargestellt werden. Der Kontrast für die Pixel $d_1(i,k)$ (unten) und $d_2(i,k)$ (oben rechts) wurde verstärkt.

Aufspaltung:

$$x(i,k) = \begin{cases} s_2(i,k) & \text{falls } i \text{ und } k \text{ gerade} \\ d_2(i,k) & \text{falls } i \text{ und } k \text{ ungerade} \end{cases} \tag{3.83}$$

Vorhersage: (nur falls i und k ungerade)

$$x(i,k) \mapsto x(i,k) - \frac{1}{4}\Big(x(i-1,k-1)+x(i+1,k-1)+x(i-1,k+1)+x(i+1,k+1)\Big) \tag{3.84}$$

Korrektur: (nur falls i gerade und k gerade)

$$x(i,k) \mapsto x(i,k) + \frac{1}{8}\Big(x(i-1,k-1)+x(i+1,k-1)+x(i-1,k+1)+x(i+1,k+1)\Big) \tag{3.85}$$

Für die dritte Skalierungsstufe müssen von den verbliebenen weißen Werten jeweils wieder die Hälfte den schwarzen zugeschlagen werden. Hierzu kann man die weißen Werte

in Gedanken zu einem neuen Schachbrett zusammensetzen und sich überlegen, welche Felder für ein Schachspiel schwarz eingefärbt werden müssen. Dies ist in Abb. 3.21 verdeutlicht (für diese Stufe hellgrau). „Vorhersage“ und „Korrektur“ erfolgen mit Hilfe eines Mittelwerts über die nächsten weißen Nachbarn. Diese sind jetzt wieder in horizontaler und vertikaler Richtung. Wir haben

Aufspaltung:

$$x(i,k) = \begin{cases} s_3(i,k) & \text{falls } i+k \text{ durch 4 teilbar} \\ d_3(i,k) & \text{falls } i+k \text{ nicht durch 4 teilbar und } x(i,k) = s_2(i,k) \end{cases} \tag{3.86}$$

Vorhersage: (nur falls $x(i,k) = d_3(i,k)$)

$$x(i,k) \mapsto x(i,k) + \frac{1}{4}\Big(x(i-2,k) + x(i+2,k) + x(i,k-2) + x(i,k+2)\Big) \tag{3.87}$$

Korrektur: (nur falls $x(i,k) = s_3(i,k)$)

$$x(i,k) \mapsto x(i,k) + \frac{1}{8}\Big(x(i-2,k) + x(i+2,k) + x(i,k-2) + x(i,k+2)\Big) \tag{3.88}$$

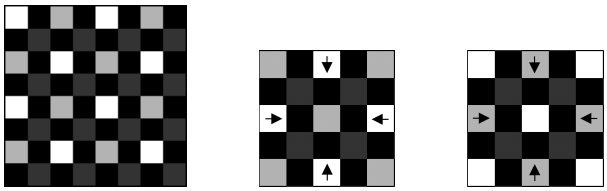


Abbildung 3.21: Schachbrett-Transformation in der dritten Skalierungsstufe: Aufteilung der weißen Werte in weiße und hellgraue (schwarze der vorherigen Stufe dunkelgrau). Vorhersage der hellgrauen Werte (Mitte) und Korrektur der Werte (rechts) mit Hilfe des Mittelwerts über die nächsten weißen bzw. hellgrauen Nachbarn.

Dies kann man weiter fortsetzen, in der vierten Skalierungsstufe liegen die nächsten weißen Nachbarn wieder in Diagonalrichtung. Das Ergebnis der Anwendung von vier Skalierungsstufen auf das Testbild „Lena“ ist in Abb. 3.22 gezeigt. Wir haben allgemein für die n. Skalierungsstufe ($n = 1, 2, 3, \ldots$) eine Fallunterscheidung zu treffen. Für den Start nehmen wir $s_0(i,k) = x(i,k)$.

Falls n *ungerade* ist, gilt mit $m := 2^{\frac{n-1}{2}}$

Aufspaltung:

$$x(i,k) = \begin{cases} s_n(i,k) & \text{falls } i+k \text{ durch } 2m \text{ teilbar} \\ d_n(i,k) & \text{falls } i+k \text{ nicht durch } 2m \text{ teilbar und } x(i,k) = s_{n-1}(i,k) \end{cases} \tag{3.89}$$

Vorhersage: (nur falls $x(i,k) = d_n(i,k)$)

$$x(i,k) \mapsto x(i,k) - \frac{1}{4}\Big(x(i-m,k) + x(i+m,k) + x(i,k-m) + x(i,k+m)\Big) \tag{3.90}$$

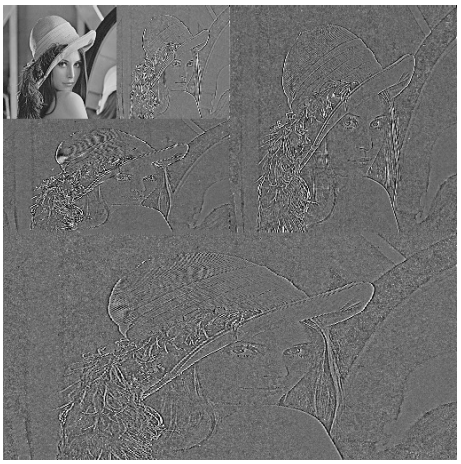


Abbildung 3.22: Anwendung der Schachbrett-Transformation mit vier Skalierungsstufen auf das Testbild „Lena“. Der Kontrast für die Pixel $d_m(i,k)$ wurde verstärkt.

Korrektur: (nur falls $x(i,k) = s_n(i,k)$))

$$x(i,k) \mapsto x(i,k) + \frac{1}{8}\Big(x(i-m,k) + x(i+m,k) + x(i,k-m) + x(i,k+m)\Big) \quad (3.91)$$

und falls n *gerade* ist, gilt mit $m := 2^{\frac{n}{2}-1}$

Aufspaltung:

$$x(i,k) = \begin{cases} s_n(i,k) & \text{falls } i \text{ und } k \text{ durch } 2m \text{ teilbar} \\ d_n(i,k) & \text{falls } i \text{ und } k \text{ nicht durch } 2m \text{ teilbar u. } x(i,k) = s_{n-1}(i,k) \end{cases} \quad (3.92)$$

Vorhersage: (nur falls $x(i,k) = d_n(i,k)$)

$$x(i,k) \mapsto x(i,k) - \frac{1}{4}\Big(x(i-m,k-m) + x(i+m,k-m) + x(i-m,k+m) + x(i+m,k+m)\Big) \quad (3.93)$$

Korrektur: (nur falls $x(i,k) = s_n(i,k)$)

$$x(i,k) \mapsto x(i,k) + \frac{1}{8}\Big(x(i-m,k-m) + x(i+m,k-m) + x(i-m,k+m) + x(i+m,k+m)\Big) \quad (3.94)$$

Die *Normierung* wurde bisher außer acht gelassen. Analog zu (3.7) ist es hier sinnvoll, nach jedem Schritt die Koeffizienten gemäß

$$s_n(i,k) \mapsto \sqrt{2} \cdot s_n(i,k), \qquad d_n(i,k) \mapsto \frac{1}{2}\sqrt{2} \cdot d_n(i,k) \quad (3.95)$$

zu normieren. Bei der Verwendung von ganzzahliger Arithmetik ist eine derartige Normierung nicht möglich. Dann empfiehlt es sich, die Normierung der Koeffizienten $d_n(i,k)$ ganz wegzulassen (sie werden ohnehin nicht weitervertransformiert) und die Normierung

von $s_n(i,k)$ nur bei jedem zweiten Schritt durch Multiplikation mit dem Faktor 2 durchzuführen. Dies kann für positive Zahlen im Rechner durch eine Verschiebung realisiert werden (jedoch nicht für negative Zahlen aufgrundder völlig unterschiedlichen rechnerinternen Darstellung!). D.h. bei der Verwendung ganzzahliger Arithmetik ist die Normierung (3.95) durch

$$s_n(i,k) \mapsto 2 \cdot s_n(i,k) \qquad \text{nur falls } n \text{ gerade} \tag{3.96}$$

zu ersetzen.

Schließlich ist darauf hinzuweisen, dass das hier dargestellte Schema eines der einfachsten seiner Art ist. Bessere Ergebnisse erzielt man sicherlich, indem man ein aufwendigeres Verfahren zur Vorhersage als die Mittelwertbildung über die vier nächsten Nachbarn benutzt. Die bisher betrachtete Mittelwertbildung über vier Nachbarn entsteht durch Polynominterpolation mit diesen Stützpunkten: man sucht ein Polynom möglichst niedrigen Grades, dessen Funktionswerte mit den Bildwerten dieser vier Nachbarn übereinstimmt. Als Vorhersagewert nimmt man dann den Funktionswert des Polynoms an der Stelle, an der der Bildwert vorherzusagen ist. Nun kann man mehr als vier Nachbarn zur Interpolation heranziehen.

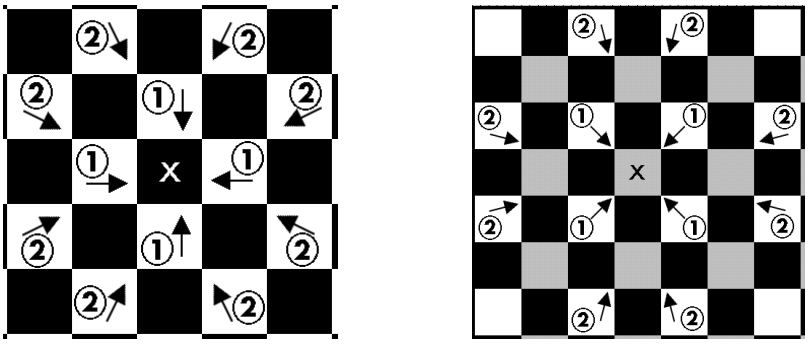


Abbildung 3.23: Verbesserung der Vorhersage (an der durch „×markierten Stelle) durch Interpolation über mehr Nachbarn: links für die erste, rechts für die zweite Skalierungsstufe

In Abb. 3.23 ist gezeigt, welche Nachbarn für die nächste Verbesserung der Vorhersage bei den ersten beiden Skalierungsstufen zu berücksichtigen sind. Als Ergebnis der Interpolation erhält man für die Vorhersage ein gewichtetes Mittel. Dabei gehen die vier nächsten Nachbarn (in der Abbildung mit ① gekennzeichnet) mit dem Gewichtsfaktor $\frac{5}{16}$ ein; die 8 weiter entfernten Nachbarn (in der Abbildung mit ② gekennzeichnet) tragen mit dem Gewichtsfaktor $-\frac{1}{32}$ bei. Bei der Korrektur ist das Vorzeichen umzudrehen, und die Gewichtsfaktoren sind mit $\frac{1}{2}$ zu multiplizieren.

Somit erhalten wir in der ersten Skalierungsstufe (die Aufspaltung bleibt dieselbe):

Vorhersage: (nur falls $i+k$ ungerade)

$$\begin{aligned} x(i,k) \;\mapsto\; x(i,k) &- \frac{5}{16}\Big(x(i-1,k)+x(i+1,k)+x(i,k-1)+x(i,k+1)\Big) \\ &+\frac{1}{32}\Big(x(i-2,k-1)+x(i-2,k+1)+x(i-1,k-2) \\ &\quad +x(i-1,k+2)+x(i+2,k-1)+x(i+2,k+1) \\ &\quad +x(i+1,k-2)+x(i+1,k+2)\Big) \end{aligned} \tag{3.97}$$

Korrektur: (nur falls $i+k$ gerade)

$$\begin{aligned} x(i,k) \;\mapsto\; x(i,k) &+ \frac{5}{32}\Big(x(i-1,k)+x(i+1,k)+x(i,k-1)+x(i,k+1)\Big) \\ &-\frac{1}{64}\Big(x(i-2,k-1)+x(i-2,k+1)+x(i-1,k-2) \\ &\quad +x(i-1,k+2)+x(i+2,k-1)+x(i+2,k+1) \\ &\quad +x(i+1,k-2)+x(i+1,k+2)\Big) \end{aligned} \tag{3.98}$$

Bemerkenswert ist, dass auch hier im Nenner nur Zweierpotenzen auftauchen, die sich bei Verwendung ganzzahliger Arithmetik durch eine Verschiebung (shift) realisieren lassen.

Für die zweite Skalierungsstufe erhalten wir entsprechend Abb. 3.23, rechte Seite:

Vorhersage: (nur falls i und k ungerade)

$$\begin{aligned} x(i,k) \;\mapsto\; x(i,k) &- \frac{5}{16}\Big(x(i-1,k-1)+x(i+1,k-1) \\ &\qquad +x(i-1,k+1)+x(i+1,k+1)\Big) \\ &+\frac{1}{32}\Big(x(i-3,k-1)+x(i-3,k+1)+x(i-1,k-3) \\ &\quad +x(i-1,k+3)+x(i+1,k-3)+x(i+1,k+3) \\ &\quad +x(i+3,k-1)+x(i+3,k+1)\Big) \end{aligned} \tag{3.99}$$

Korrektur: (nur falls $i+k$ gerade)

$$\begin{aligned} x(i,k) \;\mapsto\; x(i,k) &+ \frac{5}{32}\Big(x(i-1,k-1)+x(i+1,k-1) \\ &\qquad +x(i-1,k+1)+x(i+1,k+1)\Big) \\ &-\frac{1}{64}\Big(x(i-3,k-1)+x(i-3,k+1)+x(i-1,k-3) \\ &\quad +x(i-1,k+3)+x(i+1,k-3)+x(i+1,k+3) \\ &\quad +x(i+3,k-1)+x(i+3,k+1)\Big) \end{aligned} \tag{3.100}$$

Für die n. Skalierungsstufe muss wieder eine Fallunterscheidung getroffen werden:

Falls n *ungerade* ist, sind die Entfernungen in (3.97) und (3.98) mit $2^{\frac{n-1}{2}}$ zu multiplizieren, also beispielsweise in (3.97) ist

$$x(i-1,k+2) \quad \text{zu ersetzen durch} \quad x(i-2^{\frac{n-1}{2}}, k+2\cdot 2^{\frac{n-1}{2}})$$

Falls n *gerade* ist, hat man von (3.99) und (3.100) auszugehen und dort die Entfernungen mit $2^{\frac{n}{2}-1}$ zu multiplizieren, also beispielsweise in (3.99)

$$x(i+3,k+1) \quad \text{zu ersetzen durch} \quad x(i+3\cdot 2^{\frac{n}{2}}, k+2^{\frac{n}{2}})$$

Für die *Normierung* gilt dasselbe wie bei der Berücksichtigung von nur vier nächsten Nachbarn (siehe (3.95) oder (3.96)).

Die vollständige Vorschrift ist im Anhang B angegeben. Ebenso sind dort die Gewichtsfaktoren aufgeführt, die sich ergeben, wenn man weitere Nachbarn zur Interpolation und damit zur Vorhersage heranzieht. Es ist zu erwarten, dass eine Berücksichtigung von mehr Nachbarn zunächst zu einer Verbesserung, aber auch einer Vergrößerung der Rechenzeit führt. Bei sehr vielen berücksichtigten Nachbarn ist mit Artefakten zu rechnen. Wo das Optimum liegt, scheint noch nicht bekannt und hängt vermutlich vom Bild und den Einzelheiten der Kodierung ab. Die entsprechenden Filter werden in [20] „Quincunx Neville Filters“ genannt.

Auch hier kann man durch Berechnung einer entsprechenden Polyphasenmatrix analog zur Vorgehensweise in Abschnitt 3.2.4 aus den angegebenen Lifting-Schritten die zur üblichen Filterbank-Formulierung gehörenden Filter berechnen. Dies ist allerdings etwas mühsamer als im eindimensionalen Fall. Die näheren Einzelheiten sind im Abschnitt 3.6.2 beschrieben. Aus dem Tiefpassfilter erhält man dann durch eine Grenzwertbildung die Skalierungsfunktion. Ein qualilatives Bild der Skalierungsfunktion kann man sich machen, indem man alle Koeffizienten Null setzt bis auf einen möglichst in der Mitte des zur Verfügung stehenden Bereichs auf einem Pixel, das zu einem „weißen“ Wert gehört, der den Wert Eins erhält. Führt man nun eine Reihe von Transformationen zur Rekonstruktion durch, so erhält man bis auf eine Skalierung und Normierung eine Näherung an die Skalierungsfunktion, wie im eindimensionalen Beispiel des Abschnitts 2.1.

Beschränkt man sich auf je einen Lifting-Schritt zur Vorhersage und zur Korrektur, so ergibt sich ein Vorhersageschritt der hier vorgestellten „Bauart“, nämlich durch Polynominterpolation, allein aus der üblichen Regularitätsforderung, dass das Synthese-Tiefpassfilter eine mehrfache Nullstelle in $z_1 = z_2 = 1$ hat. Die Vielfachheit der Nullstelle stimmt dabei mit dem Grad des Interpolationspolynoms überein. Dies ist in [20] gezeigt.

Die hier beschriebenen Vorhersageschritte haben eine intuitiv einleuchtende Bedeutung. Dies kann von den Korrekturschritten nicht gesagt werden. Allein aus der Forderung nach einer mehrfachen Nullstelle des Analyse-Tiefpassfilters in $z_1 = z_2 = 1$ folgt, dass die für die Korrektur verwandten Filter bis auf die Reihenfolge (z_1 und z_2 sind durch z_1^{-1} und z_2^{-1} zu ersetzen) und einen Faktor $\frac{1}{2}$ mit denen der Vorhersage übereinstimmen. Dies ist ebenfalls in [20] erläutert.

Literatur: [20]

3.6.2 Vom Lifting-Schema zur Filterbank für die Schachbrett-Abtastung

Für die Schachbrett-Abtastung sind wir den umgekehrten Weg gegangen wie im eindimensionalen Fall: wir haben das Lifting-Schema als Ausgangspunkt genommen. Hier soll nun zunächst die Abtastung ohne Rückgriff auf die Operation „Aufspaltung“ des Lifting-Schemas, sondern als Unterabtastung formuliert werden. Durch eine geschickte Notation wird es dann möglich, den Weg von zur „normalen“ Filterbank-Formulierung mit Hilfe der Polyphasenmatrix aus dem eindimensionalen Fall einfach zu übernehmen. Die Erläuterungen dieses Abschnitts sind für ein intuitives Verständnis und für eine Implementierung der im Abschnitt 3.6.1 besprochenen Transformationen nicht notwendig. Auch sind sie für das Verständnis der folgenden Kapitel entbehrlich und können daher übersprungen werden.

Beschreibung der Schachbrettabtastung mit der Dilatationsmatrix

Wir gehen hier davon aus, dass das Bild unendlich ausgedehnt ist, die Pixel also durch beliebige ganzzahlige Indexpaare (k_1 für die Zeilennummer und k_2 für die Spaltennummer) numeriert sind. Wir fassen diese Indexpaare zu einem Vektor $\mathbf{k}$ zusammen, wie dies bereits im Abschnitt 1.6 erfolgt ist. Die Menge derartiger Indexpaare wird üblicherweise mit $\mathbb{Z}^2$ bezeichnet.

Die Dilatationsmatrix ist durch

$$\mathbf{D} := \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.101}$$

gegeben (man kann die Schachbrettabtastung auch mit anderen derartigen Matrizen formulieren, aber diese Form scheint die gebräuchlichste zu sein). Da alle Matrixelemente ganzzahlig sind, ist $\mathbf{Dk}$ wieder ein Bildpixel, falls $\mathbf{k} \in \mathbb{Z}^2$. Diese Matrix besitzt die bemerkenswerte Eigenschaft, dass

$$\mathbf{D}^2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \tag{3.102}$$

d.h. zweifache Hintereinanderanwendung multipliziert alle Pixelnummern mit zwei. Diese Matrix hat also tatsächlich etwas mit einer *Skalierung* zu tun. Sie erfüllt weiterhin $\det(\mathbf{D}) = -2$, ist also invertierbar und aufgrund von (3.102) gilt

$$\mathbf{D}^{-1} = \frac{1}{2}\mathbf{D} \tag{3.103}$$

Man beachte, dass $\mathbf{D}^{-1}\mathbf{k}$ nur dann ganzzahlige Komponenten hat, wenn $k = \mathbf{Dn}$ mit $\mathbf{n} \in \mathbb{Z}^2$.

Die „weißen“ Indexpaare der ersten Skalierungsstufe haben die Form $\mathbf{Dk}$ mit $\mathbf{k} \in \mathbb{Z}^2$. Durch Verschiebung, beispielsweise um ein „Feld“ nach rechts, erhält man alle „schwarzen“ Indexpaare. Man kann diese Verschiebung durch Addition des Vektors

$$\mathbf{t}_1 := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{3.104}$$

erreichen, d.h. man erhält alle „schwarzen“ Indizes durch $\mathbf{Dk} + \mathbf{t}_1$ mit $\mathbf{k} \in \mathbb{Z}^2$. dass man gerade um ein Feld nach rechts verschiebt und diesen Vektor $\mathbf{t}_1$ nennt, ist eine gebräuchliche, aber nicht überall eingehaltene Konvention. Die Zerlegung aller Indexpaare in „weiße“ und „schwarze“ kann also symbolisch durch

$$\mathbb{Z}^2 = \mathbf{D}\mathbb{Z}^2 \cup (\mathbf{D}\mathbb{Z}^2 + \mathbf{t}_1) \tag{3.105}$$

beschrieben werden. Sie entspricht der Zerlegung aller ganzen Zahlen in gerade und ungerade im eindimensionalen Fall. Statt des Vektors $\mathbf{t}_1$ tritt dort der Skalar 1 auf. Diese Zerlegung ist die Grundlage für die Unterabtastung. Es sei als Übungsaufgabe empfohlen, sich zu überzeugen, dass diese Aufspaltung tatsächlich mit der durch die Fallunterscheidung in Gleichung (3.79) gegebenen übereinstimmt.

Schachbrett-Unterabtastung bedeutet nun, dass aus dem Eingangssignal $x(\mathbf{k})$ das Ausgangssignal

$$y(\mathbf{k}) = (\downarrow \mathbf{D})x(\mathbf{k}) = x(\mathbf{Dk}) \tag{3.106}$$

wird. Für das entsprechende Upsampling wird aus dem Eingangssignal $x(\mathbf{k})$ das Ausgangssignal

$$y(\mathbf{k}) = (\uparrow \mathbf{D})x(\mathbf{k}) = \begin{cases} x(\mathbf{D}^{-1}\mathbf{k}) = x(\mathbf{n}) & \text{falls } \mathbf{k} = \mathbf{D}\mathbf{n} \text{ mit } \mathbf{n} \in \mathbb{Z}^2 \\ 0 & \text{sonst} \end{cases} \tag{3.107}$$

Man beachte, dass gegenüber den Formeln des eindimensionalen Falles (siehe Abschnitt 1.4) einfach die Multiplikation mit 2 durch die Multiplikation mit der Matrix $\mathbf{D}$ ersetzt wurde.

Für die Beschreibung der Schachbrett-Unterabtastung mit Hilfe der z-Transformation bedarf es noch einiger Tricks, um die gewünschte Ähnlichkeit der Formeln mit dem eindimensionalen Fall zu erreichen. Zunächst ist daran zu erinnern, dass wir bereits in Abschnitt 1.6 die Potenz für Vektoren als Basis und Vektoren im Exponenten durch $\mathbf{z}^{\mathbf{k}} := z_1^{k_1} \cdot z_2^{k_2}$ definiert haben (siehe (1.54)). In Bezug auf diese Potenz übernimmt $\mathbf{D}\mathbb{Z}^2$ die Rolle der geraden und $\mathbf{D}\mathbb{Z}^2 + \mathbf{t}_1$ die Rolle der ungeraden Zahlen in der folgenden Rechenregel:

$$(-\mathbf{z})^{\mathbf{n}} = \begin{cases} \mathbf{z}^{\mathbf{n}} & \text{falls } \mathbf{n} = \mathbf{D}\mathbf{k} \text{ mit } \mathbf{k} \in \mathbb{Z}^2 \\ -\mathbf{z}^{\mathbf{n}} & \text{falls } \mathbf{n} = \mathbf{D}\mathbf{k} + \mathbf{t}_1 \text{ mit } \mathbf{k} \in \mathbb{Z}^2 \end{cases} \tag{3.108}$$

Es sei auch hier als Übungsaufgabe empfohlen, dies durch Nachrechnen zu überprüfen.

Da wir für die Unterabtastung die Zahl 2 durch die Matrix $\mathbf{D}$ zu ersetzen haben, müssen wir noch festlegen, was es bedeuten soll, wenn eine Matrix als Exponent bei einem Vektor vorkommt. Wir *definieren* also für Matrizen $\mathbf{A}$

$$\mathbf{z}^{\mathbf{A}} := \begin{pmatrix} \mathbf{z}^{\mathbf{A}\mathbf{e}_1} \\ \mathbf{z}^{\mathbf{A}\mathbf{e}_2} \end{pmatrix} = \begin{pmatrix} z_1^{a_{11}} \cdot z_2^{a_{21}} \\ z_1^{a_{12}} \cdot z_2^{a_{22}} \end{pmatrix} \quad \text{wenn} \quad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{3.109}$$

Dabei sind wie üblich $\mathbf{e}_1$ und $\mathbf{e}_2$ die Einheitsvektoren

$$\mathbf{e}_1 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \mathbf{e}_2 := \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.110}$$

und $\mathbf{A}\mathbf{e}_1$ sowie $\mathbf{A}\mathbf{e}_2$ sind die Spaltenvektoren der Matrix $\mathbf{A}$. Die üblichen Potenzrechenregeln bleiben hierfür erfüllt, d.h. es gilt für (2×2)- Matrizen $\mathbf{A}$ und $\mathbf{B}$ sowie für Vektoren $\mathbf{n} \in \mathbb{Z}^2$

$$\left(\mathbf{z}^{\mathbf{A}}\right)^{\mathbf{n}} = \mathbf{z}^{\mathbf{A}\mathbf{n}} \qquad \text{und} \qquad \left(\mathbf{z}^{\mathbf{A}}\right)^{\mathbf{B}} = \mathbf{z}^{\mathbf{A}\mathbf{B}} \tag{3.111}$$

Die so definierte Potenz ist also mit der Matrixmultiplikation verträglich (es wird empfohlen, dies als Übungsaufgabe nachzurechnen!).

Damit kann die Schachbrett-Unterabtastung für die z-Transformierten formuliert werden: das Eingangssignal $X(\mathbf{z})$ geht über in

$$Y(\mathbf{z}) = (\downarrow \mathbf{D})X(\mathbf{z}) = \frac{1}{2}\Big(X(\mathbf{z}^{\mathbf{D}^{-1}}) + X(-\mathbf{z}^{\mathbf{D}^{-1}})\Big) \tag{3.112}$$

Zum Beweis sei angemerkt, dass es aufgrund der Linearität der Vorschrift genügt, die Übereinstimmung mit (3.106) für $X(\mathbf{z}) = \mathbf{z}^{\mathbf{k}}$ für ein beliebiges $\mathbf{k} \in \mathbb{Z}^2$ zu zeigen. Dies ist aber mit Hilfe der Rechenregel (3.108) leicht durchzuführen.

Entsprechend wird beim Upsampling aus dem Eingangssignal $X(\mathbf{z})$ das Ausgangssignal

$$Y(\mathbf{z}) = (\uparrow \mathbf{D})X(\mathbf{z}) = X(\mathbf{z}^{\mathbf{D}}) \tag{3.113}$$

Für Filterbänke ist noch die Beschreibung der Hintereinanderausführung von Unterabtastung (Subsampling) und Upsampling nützlich:

$$(\uparrow \mathbf{D})(\downarrow \mathbf{D})X(\mathbf{z}) = \frac{1}{2}\Big(X(\mathbf{z}) + X(-\mathbf{z})\Big) \tag{3.114}$$

Weiterhin lassen sich die „Edlen Gleichungen“ durch analoge Änderungen auf den zweidimensionalen Fall übertragen:

$$F(\mathbf{z})(\downarrow \mathbf{D}) = (\downarrow \mathbf{D})F(\mathbf{z}^{\mathbf{D}}) \tag{3.115}$$

$$(\uparrow \mathbf{D})F(\mathbf{z}) = F(\mathbf{z}^{\mathbf{D}})(\uparrow \mathbf{D}) \tag{3.116}$$

Wir haben nun genügend Vorbereitungen getroffen, um eine zweidimensionale Filterbank mit Schachbrettabtastung beschreiben zu können. Lediglich die Umkehr der Reihenfolge bei den Analyse-Filtern, die im eindimensionalen Fall durch den Übergang von z zu z^{-1} ausgedrückt wird, haben wir noch geschickt zu berücksichtigen. Man kann sich leicht klar machen, dass im zweidimensionalen die Reihenfolge sowohl für die Zeilen als auch für die Spaltenindizes zu ändern ist: z_1 ist durch z_1^{-1} und z_2 durch z_2^{-1} zu ersetzen. Dies kann mit Hilfe der (2×2)-Einheitsmatrix

$$\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.117}$$

ausgedrückt werden: Man ersetzt z^{-1} durch

$$\mathbf{z^{-1}} = \begin{pmatrix} z_1^{-1} \\ z_2^{-1} \end{pmatrix} \tag{3.118}$$

Mit dieser Änderungsvorschrift können die Bedingungen der perfekten Rekonstruktion übernommen werden.

Polyphasendarstellung und Polyphasenmatrix

Die Definition (3.109) macht es nun einfach, die Polyphasendarstellung von Filtern (siehe (3.21)) auf den hier behandelten zweidimensionalen Fall zu verallgemeinern:

$$F(\mathbf{z}) = \sum_{\mathbf{k}\in\mathbb{Z}^2} f(\mathbf{k})\mathbf{z}^{-\mathbf{k}} \tag{3.119}$$

$$= \sum_{\mathbf{k}\in\mathbb{Z}^2} f(\mathbf{Dk})\mathbf{z}^{-\mathbf{Dk}} + \sum_{\mathbf{k}\in\mathbb{Z}^2} f(\mathbf{Dk}+\mathbf{t}_1)\mathbf{z}^{-\mathbf{Dk}-\mathbf{t}_1}$$

$$= \sum_{\mathbf{k}\in\mathbb{Z}^2} f(\mathbf{Dk})\mathbf{z}^{-\mathbf{Dk}} + \mathbf{z}^{-\mathbf{t}_1}\sum_{\mathbf{k}\in\mathbb{Z}^2} f(\mathbf{Dk}+\mathbf{t}_1)\mathbf{z}^{-\mathbf{Dk}}$$

$$= F_e(\mathbf{z}^{\mathbf{D}}) + \mathbf{z}^{-\mathbf{t}_1}F_o(\mathbf{z}^{\mathbf{D}}) \tag{3.120}$$

Dabei ist der *gerade* Anteil $F_e(\mathbf{z})$ auch durch

$$F_e(\mathbf{z}) = \frac{1}{2}\Big(F\mathbf{z}^{\mathbf{D}^{-1}}) + F(-\mathbf{z}^{\mathbf{D}^{-1}})\Big) \tag{3.121}$$

und der ungerade Anteil (der Index ist ein „o“ von *odd*) auch durch

$$F_o(\mathbf{z}) = \frac{1}{2}(\mathbf{z}^{\mathbf{D}^{-1}})^{\mathbf{t}_1}\Big(F(\mathbf{z}^{\mathbf{D}^{-1}}) - F(-\mathbf{z}^{\mathbf{D}^{-1}})\Big) \tag{3.122}$$

gegeben.

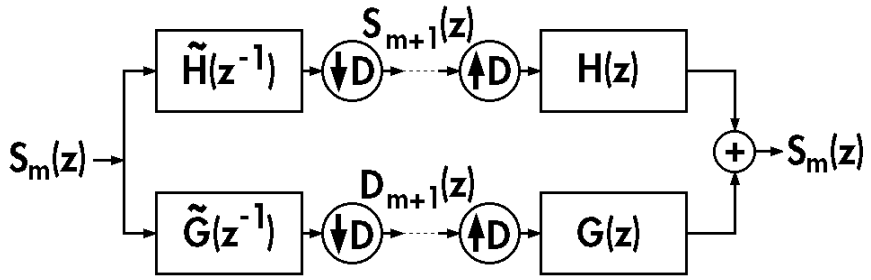


Abbildung 3.24: Filterbank zur Schachbrettabtastung. $\mathbf{z}$ ist als Vektor mit den Komponenten z_1 und z_2 aufzufassen.

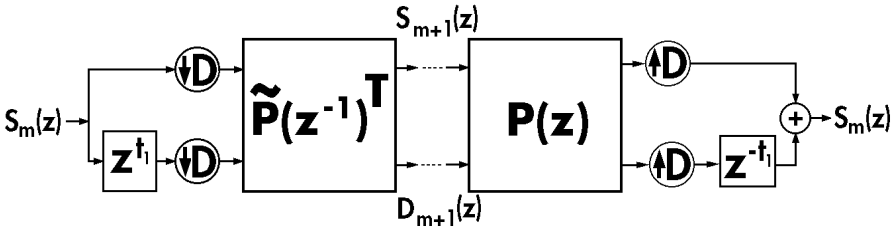


Abbildung 3.25: Polyphasenmatrizen statt der Filterbank. Auch hier ist $\mathbf{z}$ ein Vektor.

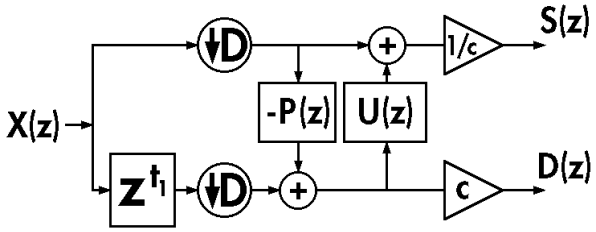


Abbildung 3.26: Lifting-Schema mit nur einem Vorhersage- und Korrekturschritt

Filterbank mit Schachbrettabtastung

Betrachten wir eine Zweikanal-Filterbank mit Schachbrettabtastung in Abb. 3.24. Nach den umfangreichen Vorbereitungen ist klar, dass die Filterbank durch die in Abb. 3.25 gezeigten Polyphasenmatrizen ersetzt werden kann. Wir haben in Abschnitt 3.6.1 Transformationen angegegeben, die nur aus einem Vorhersage- und einem Korrekturschritt bestehen. Wir haben uns also auf Transformationen beschränkt, wie sie in Abb. 3.26 gezeigt sind. Im dort gezeigten Schema ist die *Aufspaltung* explizit durch

$$s(\mathbf{k}) = x(\mathbf{Dk}), \qquad d(\mathbf{k}) = x(\mathbf{Dk} + \mathbf{t}_1) \tag{3.123}$$

Damit weichen wir von der speicherplatzsparenden Darstellung von Abschnitt 3.6.1 ab, wo die ursprüngliche Numerierung der Daten beibehalten wurde. Hier werden „schwarze“ und „weiße“ Felder getrennt numeriert. So hat beispielsweise das schwarze Feld $d(0,0) =$

$x(\mathbf{t}_1) = x(1,0)$ folgende nächste Nachbarn:

$$s(0,0) = x(\mathbf{D}\begin{pmatrix}0\\0\end{pmatrix}) = x(0,0), \quad s(0,1) = x(\mathbf{D}\begin{pmatrix}0\\1\end{pmatrix}) = x(1,-1), \tag{3.124}$$

$$s(1,0) = x(\mathbf{D}\begin{pmatrix}1\\0\end{pmatrix}) = x(1,1) \quad s(1,1) = x(\mathbf{D}\begin{pmatrix}1\\1\end{pmatrix}) = x(2,0) \tag{3.125}$$

Das einfachste Beispiel der Vorhersage mit Hilfe der vier nächsten Nachbarn erhalten wir mit

$$P(\mathbf{z}) = \frac{1}{4}(1 + z_1 + z_2 + z_1 \cdot z_2) \tag{3.126}$$

$$U(\mathbf{z}) = \frac{1}{8}(1 + z_1^{-1} + z_2^{-1} + z_1^{-1} \cdot z_2^{-1}) \tag{3.127}$$

Ähnlich zur Vorgehensweise in Abschnitt 3.2.4 kann man hieraus die Filter der Filterbank berechnen. Beispielsweise erhält man analog zu (3.45)

$$H(\mathbf{z}) = c + c\mathbf{z}^{-\mathbf{t}_1} P(\mathbf{z}^{\mathbf{D}}) \tag{3.128}$$

$$= \frac{1}{2}\sqrt{2}\Big(1 + \frac{1}{4}(z_1 + z_2 + z_1^{-1} + z_2{-}1)\Big) \tag{3.129}$$

Zwei-Skalen-Relation im Fall der Schachbrettabtastung

Auch die Zwei-Skalen-Relation kann mit Hilfe der Dilatationsmatrix $\mathbf{D}$ ausgedrückt werden:

$$\varphi(\mathbf{x}) = \frac{1}{\sqrt{|\det \mathbf{D}|}} \sum_{\mathbf{k}\in\mathbb{Z}^2} h(\mathbf{k})\varphi(\mathbf{D}\mathbf{x} - \mathbf{k}) \tag{3.130}$$

Entsprechend kann durch

$$\psi(\mathbf{x}) = \frac{1}{\sqrt{|\det \mathbf{D}|}} \sum_{\mathbf{k}\in\mathbb{Z}^2} g(\mathbf{k})\varphi(\mathbf{D}\mathbf{x} - \mathbf{k}) \tag{3.131}$$

aus der Skalierungsfunktion ein Wavelet konstruiert werden. Analoge Relationen gelten für die Analyse-Funktionen $\tilde{\varphi}(\mathbf{x})$ und $\tilde{\psi}(\mathbf{x})$. Skalierte und verschobene Versionen dieser Funktionen erhält man ganz analog zum eindimensionalen Fall durch

$$\varphi_{\mathbf{mn}}(\mathbf{x}) = (\sqrt{|\det \mathbf{D}|})^m \varphi(\mathbf{D}^m\mathbf{x} - \mathbf{n}) \tag{3.132}$$

für beliebige $\mathbf{m}, \mathbf{n} \in \mathbb{Z}^2$. Entsprechend erhält man $\psi_{\mathbf{mn}}(\mathbf{x})$ und die Analysefunktionen.

Kapitel 4

Verallgemeinerungen der Wavelet-Transformation

4.1 Wavelet-Pakete

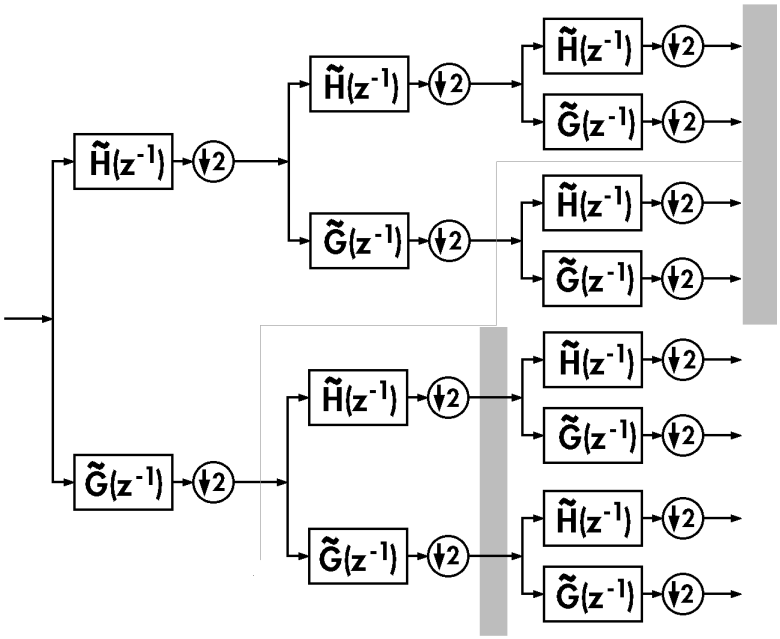


Abbildung 4.1: Filterbank zur Wavelet-Paket-Transformation

Betrachtet man die dreistufige Filterbank in Abb. 1.12 genauer, so fällt auf, dass die Koeffizienten unsymmetrisch weiterbehandelt werden: nur die Ausgänge der Tiefpassfilter $s_i(k)$ werden als Eingang auf eine neue Filterbank gegeben. Es ist naheliegend, diese Unsymmetrie aufzugeben und das in Abb. 4.1 gezeigte allgemeine Filterbankschema anzuwenden (diesmal in der Waveletnotation, die in Tabelle 2.1 dargestellt ist). Zu beachten ist, dass sich die Operationen aus jeder einzelnen Analysefilterbank durch die entsprechende Rekonstruktionsfilterbank rückgängig machen lassen. So ermöglichen nicht nur die Koeffizienten, die als Ausgang am rechten Rand der Abb. 4.1 auftreten, eine Rekonstruktion der ursprünglichen Daten. Sondern beispielsweise auch aus den Koeffizienten,

die als Ausgang an den grau unterlegten Stellen auftreten, können die ursprünglichen Daten zurückgewonnen werden.

Man hat damit die Struktur eines „binären Baumes“ erhalten. Dies ermöglicht eine *adaptive* Behandlung der Daten: Man wählt Koeffizienten, die eine perfekte Rekonstruktion ermöglichen, so aus, dass möglichst wenige wesentlich von Null verschieden sind. Dies wird in der Literatur als „Suche nach der besten Basis“ bezeichnet und ermöglicht bessere Ergebnisse bei der Datenkompression. Der Nachteil ist allerdings ein erhöhter Rechenaufwand.

Im Frequenzbereich wird der Nutzen dieser Methode sichtbar, siehe hierzu Abb. 4.2. Man kann eine sehr feine Unterteilung des gesamten Frequenzbereichs erreichen; in der Abbildung ist vereinfacht angenommen, dass die Filter den Frequenzbereich einfach halbieren, was nur bei idealen Filtern der Fall wäre. Die Koeffizienten, die als Ergebnis der „normalen“ Wavelettransformation auftreten, sind durch einen Stern „$*$“ gekennzeichnet (die Koeffizienten, die zur jeweils oberen Hälfte des auftretenden Frequenzbandes gehören, werden dabei nicht weiter bearbeitet). Führt man die Transformation in n Stufen bis zum Ende durch (in Abb. 4.1 bis zum rechten, in Abb. 4.2 bis zum oberen Rand), so gehören die entsprechenden Koeffizienten zu einem Frequenzbereich, dessen Länge das 2^{-n}-fache des urprünglichen beträgt (im Beispiel von 3 Stufen also $\frac{1}{8}$ der ursprünglichen Breite. Die hier besprochene Verallgemeinerung ermöglicht also eine sehr viel bessere Frequenzanalyse.

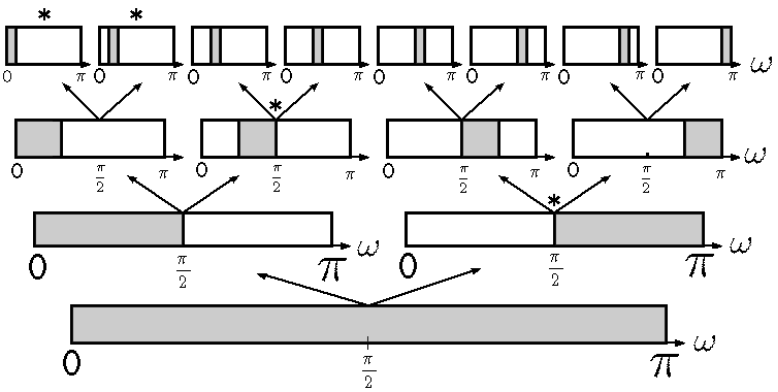


Abbildung 4.2: Frequenzverhalten der Waveletpakettransformation (grob vereinfacht, es ist ein ideales Tief-und Hochpassverhalten angenommen). Der zu den jeweiligen Koeffizienten gehörende Frequenzbereich ist grau unterlegt. Für einen Vergleich ist Abb. 4.1 um 90^0 entgegen dem Uhrzeigersinn zu drehen.

Man kann sich auch hier überlegen, welche Funktionen zu dieser Filterbankmethode gehören. Man erhält eine Folge rekursiv definierter Funktionen, wobei die „Startfunktion“ $W_0(x) = \varphi(x)$ die übliche Sklalierungsfunktion sowie $W_1(x) = \psi(x)$ das gewöhnliche Wavelet ist:

$$W_{2n}(x) = \sqrt{2} \sum_{k=-\infty}^{+\infty} h(k) W_n(2x-k) \tag{4.1}$$

$$W_{2n+1}(x) = \sqrt{2} \sum_{k=-\infty}^{+\infty} g(k) W_n(2x-k) \tag{4.2}$$

Diese Funktionen heißen „Wavelet-Pakete“. Für den Spezialfall des Haar-Wavelets erhält man durch dieses Vorgehen die Walsh-Funktionen.

Literatur: [39, chapter 7]

4.2 Mehrkanal-Filterbänke

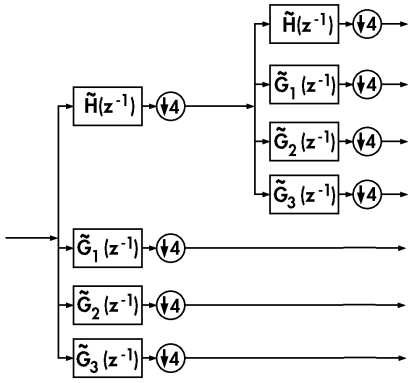


Abbildung 4.3: Vierkanal-Filterbank: Analyse

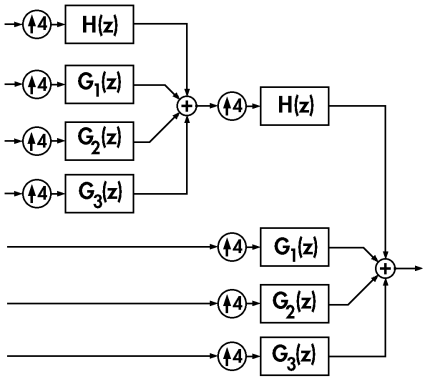


Abbildung 4.4: Vierkanal-Filterbank: Synthese (Rekonstruktion)

Eine andere Möglichkeit der Verallgemeinerung ergibt sich, wenn man zu mehr als zwei Kanälen in der Filterbank übergeht. Dies ist für 4 Kanäle in Abb. 4.3 für die Analyse und in Abb. 4.4 für die Rekonstruktion (Synthese) gezeigt. Bei der „Unterabtastung“ wird nur noch jeder 4. Wert behalten.

Der Vorteil einer derartigen Verallgemeinerung liegt in der größeren Flexibilität hinsichtlich der Auswahl der Filter. Die Bedingung für die perfekte Rekonstruktion ändert sich natürlich gegenüber dem bisher ausschließlich betrachteten Fall von 2 Kanälen. Für weitere Einzelheiten wird auf die Literatur (siehe z.B. [5, chapter 7.2]) verwiesen.

Kapitel 5

Wavelet-Transformation ohne Unterabtastung

5.1 Definition der Koeffizienten und perfekte Rekonstruktion

Unterabtastung hat zur Folge, dass eine zeitliche Verschiebung des Signals $s_0(k)$ keineswegs zu einer entsprechenden Verschiebung der Koeffizienten $d_1(k)$ und $s_1(k)$ führt, sondern zu einer verwickelten Änderung (siehe die Diskussion des Signalbeispiels (1.34) in Abschnitt 1.4). Für viele praktische Beispiele ist es daher sinnvoll, auf die Unterabtastung zu verzichten. Wir erhalten damit eine Filterbank der in Abb. 5.1 dargestellten Form. Zu Unterscheidung gegenüber den bisher benutzten Koeffizienten werden in diesem Abschnitt der Ausgang des Hochpassfilters mit $R_1(Z)$ (bzw. im Zeitbereich $r_1(k)$) bezeichnet. Der Buchstabe „r“ steht dabei für „redundant“, da mehr Koeffizienten berechnet werden, als zur Rekonstruktion des Signals notwendig ist. Zu beachten ist, dass auch $S_1(z)$ bzw. $s_1(k)$ sich geändert haben.

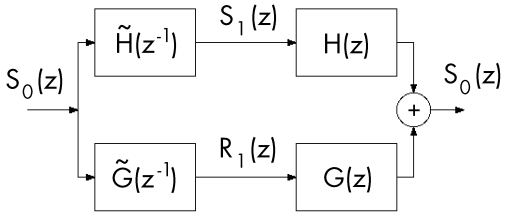


Abbildung 5.1: Filterbank ohne Unterabtastung

Die Bedingung für die *perfekte Rekonstruktion* lautet hier lediglich

$$H(z)\tilde{H}(z^{-1}) + G(z)\tilde{G}(z^{-1}) = 1 \tag{5.1}$$

Dies hat auch eine Änderung der Normierung zur Folge. Da ein sinnvolles Hochpassfilter $\tilde{G}(1) = 0$ erfüllt, erhalten wir $H(1)\tilde{H}(1) = 1$. Wir gehen daher in diesem Abschnitt von der Normierungsbedingung

$$\tilde{H}(1) = H(1) = 1 \quad \text{bzw.} \sum_{k=-\infty}^{+\infty} \tilde{h}(k) = \sum_{k=-\infty}^{+\infty} h(k) = 1 \tag{5.2}$$

aus. In dieser neuen Normierung lautet die Zwei-Skalen-Relation

$$\varphi(x) = 2 \sum_{k=-\infty}^{+\infty} h(k)\varphi(2x-k) \tag{5.3}$$

und das Wavelet ist durch

$$\psi(x) = 2 \sum_{k=-\infty}^{+\infty} g(k)\varphi(2x-k) \tag{5.4}$$

gegeben.

In Bezug auf die „Zeit“ k formuliert, ist der Ausgang der Filterbank durch

$$s_1(k) = \sum_{l=-\infty}^{+\infty} \tilde{h}(l)s_0(k+l) = \sum_{l=-\infty}^{+\infty} \tilde{h}(-(k-l))s_0(l) \tag{5.5}$$

$$r_1(k) = \sum_{l=-\infty}^{+\infty} \tilde{g}(l)s_0(k+l) = \sum_{l=-\infty}^{+\infty} \tilde{g}(-(k-l))s_0(l) \tag{5.6}$$

gegeben. Der Zusammenhang zu Skalierungsfunktion und Wavelet wird durch

$$r_1(k) = \frac{1}{2}\int_{-\infty}^{+\infty} f(x)\tilde{\psi}(\frac{1}{2}x - \frac{1}{2}k)dx \tag{5.7}$$

$$s_1(k) = \frac{1}{2}\int_{-\infty}^{+\infty} f(x)\tilde{\varphi}(\frac{1}{2}x - \frac{1}{2}k)dx \tag{5.8}$$

beschrieben. Die abgetasteten Werte des kontinuierlichen Signals werden dabei mit

$$s_0(k) = \int_{-\infty}^{+\infty} f(x)\tilde{\varphi}(x-k)dx \approx f(k) \tag{5.9}$$

identifiziert.

Auch hier wird man die durch die Filterbank von Abb. 5.1 bzw. durch (5.7) und (5.8) gegebene Transformation fortsetzen, indem man $s_1(k)$ erneut als Eingang auf eine Filterbank gibt. Dabei hat man aber hier zu berücksichtigen, dass der Frequenzbereich nicht durch die Unterabtastung halbiert wurde. Wir gehen zunächst von den Funktionen aus. Wir benötigen hier „mehr“ skalierte und verschobene Versionen von Wavelet und Skalierungsfunktion, als durch Gleichung (2.43) und (2.44) beschrieben. Wir benutzen stattdessen

$$\psi(2^m x - 2^m n) \qquad \text{und} \qquad \varphi(2^m x - 2^m n)$$

für negative m. Der Graph dieser Funktionen ist unabhängig vom Skalierungsparameter m, der für negative m eine Dehnung verursacht, stets um ganzzahlige Werte n verschoben. Dass hierdurch zusätzliche Funktionen definiert sind gegenüber früher, kann man sich leicht für den Fall $m = -1$ klar machen. Hier nehmen wir die Funktionen

$$\psi\Big(\frac{1}{2}(x-n)\Big)$$

Bei Unterabtastung sind dagegen durch (2.43) „nur“ die Funktionen

$$\psi_{-1,n}(x) = 2^{-\frac{1}{2}}\psi(\frac{1}{2}x - n) = 2^{-\frac{1}{2}}\psi\Big(\frac{1}{2}(x-2n)\Big)$$

gegeben. Wir definieren die Koeffizienten

$$s_i(k) := 2^{-i} \int_{-\infty}^{+\infty} f(x)\tilde{\varphi}(2^{-i}x - 2^{-i}k)dx \tag{5.10}$$

$$r_i(k) := 2^{-i} \int_{-\infty}^{+\infty} f(x)\tilde{\psi}(2^{-i}x - 2^{-i}k)dx \tag{5.11}$$

Dies liefert (mit Hilfe von (5.3) und (5.4) die Rekursionsformeln

$$s_i(k) = \sum_{l=-\infty}^{+\infty} \tilde{h}(l)s_{i-1}(k + 2^{i-1}l) \tag{5.12}$$

$$r_i(k) = \sum_{l=-\infty}^{+\infty} \tilde{g}(l)s_{i-1}(k + 2^{i-1}l) \tag{5.13}$$

Der Normierungsfaktor vor dem Integral wurde dabei so gewählt, daß ein lokal konstantes Signal zu lokal konstanten Koeffizienten $s_i(k)$ führt. Die obigen Rekursionsformeln können auch mit Hilfe der z-Transformierten ausgedrückt werden:

$$S_i(z) = \tilde{H}(z^{-2^{i-1}})S_{i-1}(z) \tag{5.14}$$

$$R_i(z) = \tilde{G}(z^{-2^{i-1}})R_{i-1}(z) \tag{5.15}$$

Die Bedingung der perfekten Rekonstruktion (5.1) garantiert, dass die ursprünglichen Daten Schritt für Schritt mit Hilfe von

$$S_{i-1}(z) = H(z^{2^{i-1}})S_i(z) + G(z^{2^{i-1}})R_i(z) \tag{5.16}$$

rekonstruiert werden können. Formuliert man diese Formel in Bezug auf die „Zeit“ k um, so erhält man

$$s_{i-1}(k) = \sum_{l=-\infty}^{+\infty} h(l)s_i(k - 2^{i-1}l) + \sum_{l=-\infty}^{+\infty} g(l)r_i(k - 2^{i-1}l) \tag{5.17}$$

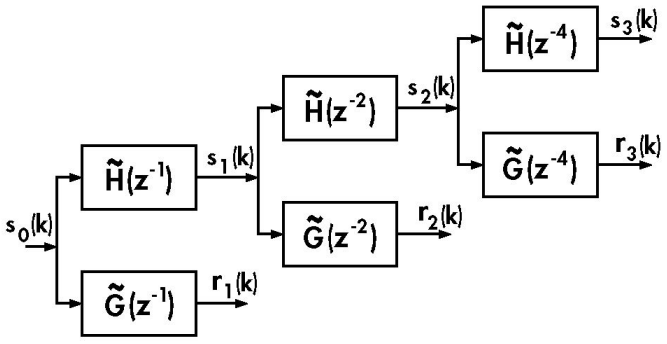


Abbildung 5.2: Dreistufige Filterbank ohne Unterabtastung: Analyseteil

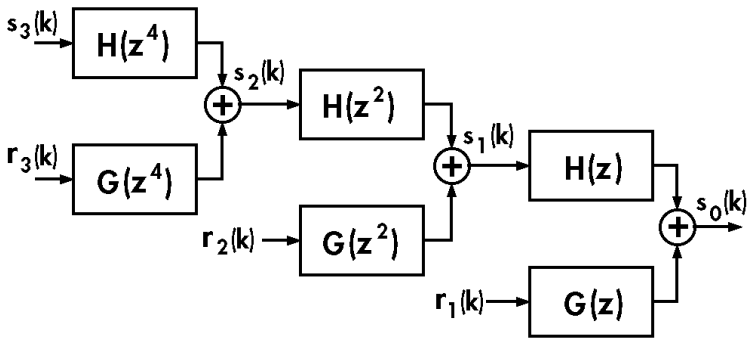


Abbildung 5.3: Dreistufige Filterbank ohne Unterabtastung: Syntheseteil

Das ursprüngliche Signal $s_0(k)$ wird also durch (5.12) und (5.13) transformiert in die Koeffizienten $r_1(k)$, $r_2(k)$, $r_3(k) \ldots r_I(k)$ und $s_I(k)$ (bis zu einem maximalen Index I) und kann aus diesen mit Hilfe von Gleichung (5.17) wieder zurückgewonnen werden. Für $I = 3$ ist dies in Abb. 5.2 und Abb. 5.3 dargestellt. Die Impulsantwort des Filters $H(z^2)$ entsteht aus der von $H(z)$, indem jeweils eine Null zwischen die Filterkoeffizienten $h(k)$ eingefügt wird. Analog hat man für $H(z^4)$ drei Nullen einzufügen. Das Weglassen der Unterabtastung wird hier im Frequenzbereich dadurch kompensiert, dass beim i-ten Skalierungsschritt z durch $z^{2^{i-1}}$ zu ersetzen ist. Dies entspricht im Zeitbereich dem Einfügen von $2^{i-1} - 1$ Nullen zwischen die Filterkoeffizienten. Daher rührt der französische Name „algorithme à trous“ für den durch Abb. 5.2 und Abb. 5.3 veranschaulichten Algorithmus. Bei der Implementierung sollte man tunlichst darauf achten, nicht durch Addition von Nullen Rechenzeit zu vergeuden, sondern die entsprechenden Indizes zu überspringen, wie dies aus (5.12) und (5.13) ersichtlich ist.

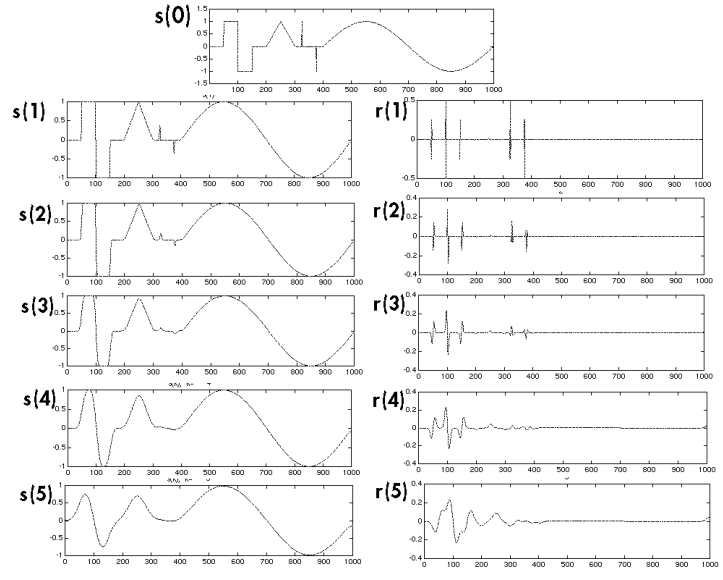


Abbildung 5.4: Wavelet-Transformation ohne Unterabtastung am Beispiel eines künstlich erzeugten Signals

Zwei Beispiele illustrieren, was qualitativ bei der redundanten Wavelet-Transformation geschieht. In Abb. 5.4 ist das Ausgangssignal aus Sprüngen, Rampen, Impulsen und dem

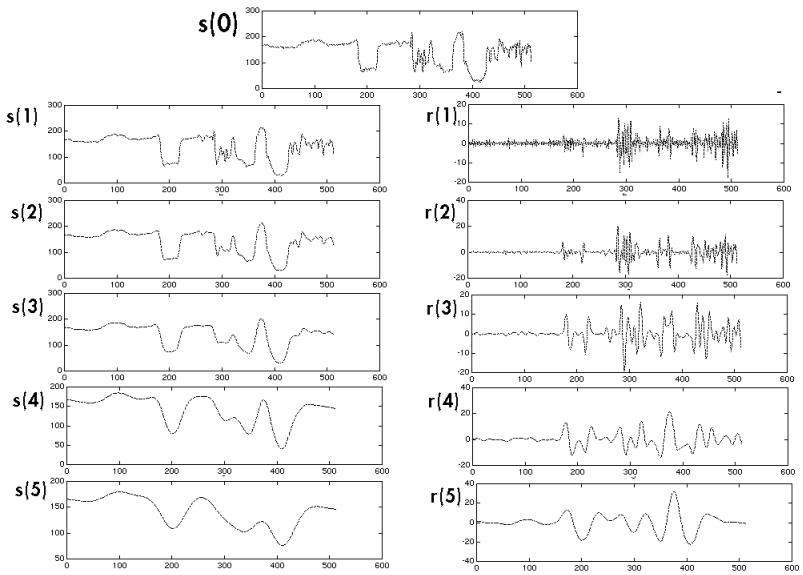


Abbildung 5.5: Wavelet-Transformation ohne Unterabtastung am Beispiel der 100. Spalte des Testbildes „boats“

Sinus zusammengesetzt, in Abb. 5.5 dienen die Intensitätswerte der 100. Spalte des Testbildes „boats“ als Ausgangspunkt. Für beide Beispiele wurde das Filter MZ(4,2) benutzt (siehe die folgenden Abschnitte 5.2 und 5.3). Für die Implementierung der Transformation wurde das Lifting-Schema nach (5.52) und (5.53) verwandt.

5.2 Auswahl von Filtern für die Wavelet-Transformation ohne Unterabtastung

Man kann die bisher als Beispiel gegebenen Filterkoeffizienten (für das Haarwavelet, $D4$ und CDF(3,5)) durch $\sqrt{2}$ dividieren und erhält so Filterkonstanten, die die Normierungsbedingung (5.2) sowie die Bedingung der perfekten Rekonstruktion (5.1) erfüllen. Da hier die Bedingung der Alias-Auslöschung (2.30) entfällt, gibt es sehr viel mehr Möglichkeiten, Filter (und damit Skalierungsfunktionen und Wavelets) zu konstruieren. Hier werden einige Beispiele gegeben, die sich in den Anwendungen besonders bewährt haben.

Ein Wort ist zur Benennung der Filter zu sagen. In Lehrbüchern und einführenden Artikeln ist die Wavelet-Transformation ohne Unterabtastung sehr häufig nur kurz oder überhaupt nicht behandelt, so dass sich für die hier vorgestellten Filter noch keine Kurzbezeichnungen durchgesetzt haben, die allgemein als bekannt vorausgesetzt werden können (wie beispielsweise CDF(3,5)). Es ist jedoch recht umständlich, die Filter jedesmal, wenn sie erwähnt werden, nochmal aufzuführen oder die entsprechenden Gleichungsnummern zu zitieren. Dies wird noch verschärft dadurch, dass es hier nicht genügt, zwei der Filter anzugeben, da durch die Bedingung der perfekten Rekonstruktion (5.1) nur eines von vier Filtern aufgrund der drei übrigen gegeben ist, also stets mindestens drei Filter angegeben werden müssen. Es werden daher in diesem Abschnitt auch Kurzbezeichnungen für die vorgestellten Filter *vorgeschlagen*. Diese Bezeichnungen sind jedoch neu und bisher noch nirgends verwandt. Sie sollten daher niemals ohne ausführliche Erklärung und explizite Angabe der Filter verwandt werden.

5.2.1 Spline-Wavelets

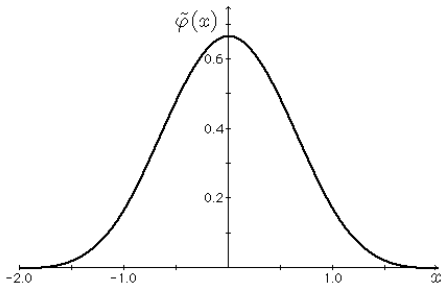


Abbildung 5.6: Skalierungsfunktion zum Tiefpassfilter (5.18) (kubischer B-Spline)

Die Skalierungsfunktion, die zum Tiefpassfilter

$$\tilde{H}(z) = 2^{-4}(z^{\frac{1}{2}} + z^{-\frac{1}{2}})^4 = 2^{-4}z^2(1+z^{-1})^4 \tag{5.18}$$

gehört, ist stückweise aus kubischen Polynomen zusammengesetzt, und zwar so, dass die 2. Ableitung existiert und stetig ist (sie ist in Abb. 5.6 graphisch dargestellt, die Stückelung ist nicht erkennbar). Derartige Funktionen werden „kubische Splines“ genannt. Die hier entstehende Skalierungsfunktion wird in der Literatur B-Spline der Ordnung 4 genannt. Durch

$$\tilde{G}^{(1)}(z) = \frac{1}{2} - \frac{1}{2}z^{-1} \qquad \text{d.h.} \qquad \tilde{g}^{(1)}(0) = \frac{1}{2},\ \tilde{g}^{(1)}(1) = -\frac{1}{2} \tag{5.19}$$

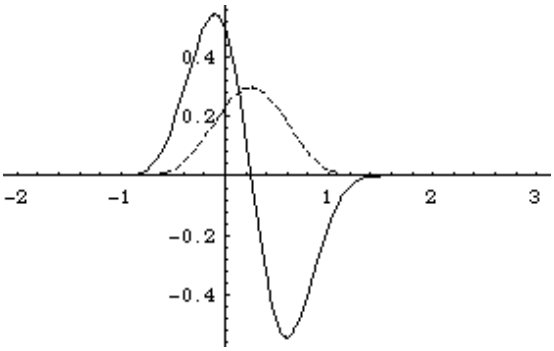


Abbildung 5.7: Wavelet $\tilde{\psi}^{(1)}$ (durchgezogene Linie, siehe (5.19)) und zugehörige Glättungsfunktion $\theta^{(1)}$ (gestrichelt) mit $\tilde{\psi}^{(1)}(x) = \frac{d}{dx}\theta^{(1)}(x)$

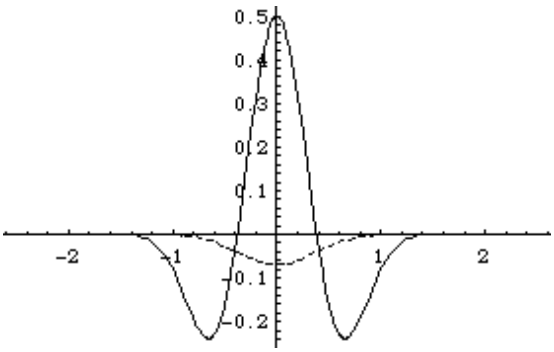


Abbildung 5.8: Wavelet $\tilde{\psi}^{(2)}$ (durchgezogene Linie, siehe (5.20)) und zugehörige Glättungsfunktion $\theta^{(2)}$ (gestrichelt) mit $\tilde{\psi}^{(2)}(x) = \frac{d^2}{dx^2}\theta^{(2)}(x)$

und

$$\tilde{G}^{(2)}(z) = \frac{1}{4}(-z + 2 - z^{-1}) \quad \text{d.h.} \quad \tilde{g}^{(2)}(0) = \frac{1}{2},\ \tilde{g}^{(2)}(1) = \tilde{g}^{(2)}(-1) = -\frac{1}{4} \tag{5.20}$$

sowie $g(k) = 0$ für alle anderen Indizes $k \in \mathbb{Z}$ werden mit Hilfe dieser Skalierungsfunktion und Gleichung (5.4) zwei Wavelets definiert, die in Abb. 5.7 und 5.8 gezeigt sind. Sie haben die für Anwendungen außerordentlich nützliche Eigenschaft, erste bzw. zweite Ableitung einer Glättungsfunktion zu sein. Darunter versteht man — grob gesprochen — eine Funktion, deren Graph so ähnlich aussieht wie eine Gaußsche Glockenkurve (gegebenfalls auch nach unten geklappt, wie in Abb. 5.8).

Für die Rekonstruktionsfilter ergeben sich sehr viel mehr Möglichkeiten, weil die Bedingung (2.30) nicht zu erfüllen ist. Meist fordert man, dass das Tiefpassfilter mit dem zur Analyse übereinstimmt, d.h.

$$H(z) = \tilde{H}(z) \tag{5.21}$$

was zur Folge hat, dass auch die entsprechenden Skalierungsfunktionen übereinstimmen, d.h. $\varphi(x) = \tilde{\varphi}(x)$. Man kann dann die Bedingung (5.1) nach $G(z)$ auflösen und erhält

$$G(z) = \frac{1 - H(z)H(z^{-1})}{\tilde{G}(z^{-1})} \tag{5.22}$$

In sehr vielen Beispielen ist diese Division ohne Rest durchführbar, und durch (5.22) wird ein FIR-Filter definiert. Dies liegt an der Normierungsbedingung $H(1) = 1$, was zur Folge

hat, dass der Zähler eine doppelte Nullstelle in $z = 1$ hat. In unserem, durch (5.18), (5.19) und (5.20) gegebenem Beispiel erhalten wir

$$G^{(1)}(z) = \frac{1}{128}\left(z^3 + 9\,z^2 + 37\,z + 93 - 93z^{-1} - 37z^{-2} - 9z^{-3} - z^{-4}\right) \tag{5.23}$$

und

$$G^{(2)}(z) = \frac{1}{64}\left(z^3 + 10\,z^2 + 47\,z + 140 + 47z^{-1} + 10z^{-2} + z^{-3}\right). \tag{5.24}$$

Dabei ist anzumerken, dass $G^{(2)}(z)$ kein Wavelet erzeugt, da $G^{(2)}(1) \neq 0$, was zur Folge hätte, daß $\int \psi^{(2)}(x)dx \neq 0$. Von einer Funktion wird aber gefordert, dass ihr Integral verschwindet, um als Wavelet „zugelassen“ zu werden. Dagegen wird durch (5.23) mit (5.4) ein Wavelet definiert, das ebenfalls stückweise durch kubische Ploynome gegeben ist, also ein kubischer Spline ist. Es ist in Abb. 5.9 gezeigt.

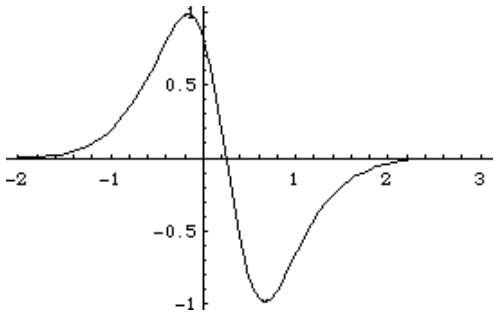


Abbildung 5.9: Wavelet $\psi^{(1)}(x)$, das durch das Rekonstruktionsfilter (5.23) gegeben ist.

Diese Konstruktion von Filtern wurde für den Fall von quadratischen Splines und das durch $^{(1)}$ gekennzeichnete Wavelet von Mallat und Zhong in [25] angegeben und benutzt. Es wird daher hier *vorgeschlagen*, die durch (5.18), (5.19) und (5.23) gegeben Filter mit MZ(4,1) und analog die durch (5.18), (5.20) und (5.24) gegeben Filter mit MZ(4,2) zu bezeichnen. Die in [25] verwandten Filter wären in dieser Bezeichnungsweise MZ(3,1). Die erste Ziffer gibt also die Ordnung der Splines an.

Verzichtet man auf die Forderung, dass durch die Rekonstruktionsfilter sinnvolle Skalierungsfunktionen und Wavelets erzeugt werden (wie dies in Abschnitt 2.4 erklärt ist), so kann man wesentlich kürzere Filter zur Rekonstruktion benutzen. Dies führt nicht in dem Maße zu den unerwünschten Effekten bei der Ausbreitung von Fehlern wie beim Fall mit Unterabtastung, da hier mit einer redundanten Entwicklung gearbeitet wird. Die folgenden Filter ergeben eine exakte Rekonstruktion für die Analysefilter (5.18) und (5.19) sowie (5.20).

$$H(z) = 1, \qquad G^{(1)}(z) = \frac{1}{8}(-z^{-2} - 5z^{-1} + 5 + z) \tag{5.25}$$

$$G^{(2)}(z) = \frac{1}{4}(z^{-1} + 6 + z). \tag{5.26}$$

Sie sind aber bisher noch wenig in der Praxis getestet worden, könnten also zu unerwarteten Nachteilen führen, sind aber kürzer und können dadurch eventuell Rechenzeit sparen. Sie wurden mit rein algebraischen Methoden (dem Algorithmus von Euklid) berechnet. Der Vorschlag für eine Bezeichnungsweise wäre hier „Binomialfilter mit minimaler Länge“ mit den Abkürzungen BML(4,1) und BML(4,2).

5.2.2 Filter mit $H(z) = \tilde{H}(z)$ und $G(z) = \tilde{G}(z)$

Man kann auch hier fordern, dass Rekonstruktions- und Analysefilter bis auf die Umkehr der Reihenfolge (die in der z-Transformierten am Auftreten von z^{-1} sichtbar ist) übereinstimmen (im Fall mit Unterabtastung führt dies auf orthogonale Wavelets). Die Bedingung der perfekten Rekonstruktion lautet mit dieser Zusatzforderung

$$H(z)H(z^{-1}) + G(z)G(z^{-1}) = 1 \tag{5.27}$$

Filterpaare, die (5.27) genügen, werden in der Literatur als „power complementary filter“ bezeichnet. Durch diese Forderung wird es unmöglich, symmetrische (oder antisymmetrische) Filter zu erhalten, außer denen des Haar-Wavelets. D.h. man kann dann kein lineares Phasenverhalten bekommen. Aber man kann näherungsweise symmetrische Filter bzw. Wavelets bekommen. Ein Beispiel ist durch dasselbe Tiefpassfilter (und damit dieselbe Skalierungsfunktion wie im Unterabschnitt 5.2.1, also

$$\tilde{H}(z) = H(z) = 2^{-4}z^2(1 + z^{-1})^4 \tag{5.28}$$

sowie die näherungsweise antisymmetrischen Filterkonstanten $g(k) = \tilde{g}(k)$ gegeben, die in Tabelle 5.1 aufgeführt sind. Das zugehörige Wavelet ist in Abb. 5.10 gezeigt. Als Kurzbezeichnungsweise wird für dieses Filter ALPC(4) vorgeschlagen, dabei steht „AL“ für angenähert lineares Phasenverhalten oder „almost linear phase“ und „PC“ für „power complementary filter“. Die Zahl gibt die Ordnung des B-Spline wie bei den Filtern des Unterabschnitts 5.2.1 an.

k	$g(k)$
0	0.119386232570
1	0.597894333070
2	-0.586666798211
3	-0.097894333070
4	-0.032719434359

Tabelle 5.1: Näherungsweise antisymmetrische Filterkoeffizienten zum Tiefpassfilter $H(z) = 2^{-4}z^2(1 + z^{-1})^4$

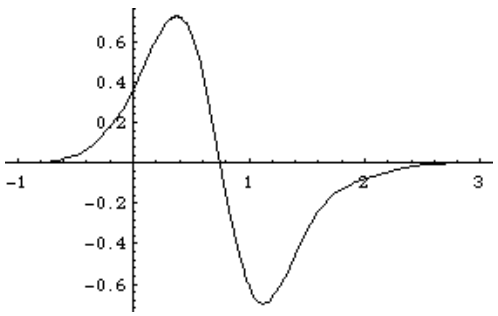


Abbildung 5.10: Näherungsweise antisymmetrisches Wavelet zum Filter der Tabelle 5.1.

5.2.3 Filter mit $H(z) = G(z) = 1$

Ein völlig anderer Standpunkt ist es, zu verlangen, dass die Rekonstruktionsfilter trivial sind, d.h. $H(z) = G(z) = 1$. Diese Filter führen nicht zu einer „vernünftigen“ Skalierungsfunktion und einem Wavelet, aber da wir hier eine redundante Entwicklung betrachten, ist es möglich (und in der Praxis nicht unüblich), die entsprechenden Filter zu benutzen. Die Bedingung der perfekten Rekonstruktion lautet in diesem Fall (nur noch eine Bedingung an die Analyse-Filter)

$$\tilde{H}(z) + \tilde{G}(z) = 1 \tag{5.29}$$

Derartige Filter kann man durch $\tilde{H}(z) = \frac{1}{2}P(z)$ und $\tilde{G}(z) = \tilde{H}(-z)$ erhalten, wobei $P(z)$ durch Gleichung (2.121) explizit gegeben ist. In diesem Fall sind die zugehörigen Skalierungsfunktionen und Wavelets die Autokorrelationsfunktionen der orthogonalen Daubechies-Wavelets. Ein Beispiel für derartige Filterkoeffizienten mit $p = 2$ in (2.121) ist in Tabelle 5.2 gegeben. Das Tiefpassfilter stimmt für beliebige p mit einer von Deslauriers und Dubuc untersuchten Filterfamilie überein (siehe beispielsweise [33]). Hochpass und Rekonstruktionsfilter dieses Unterabschnitts unterscheiden sich jedoch von den nach Deslauriers und Dubuc benannten Filtern (siehe Abschnitt 3.1 und 3.3). Die hier behandelten Filter sind in der Dissertation von Saito [28] ausführlich unter dem Stichwort „autocorrelation shell“ untersucht worden (allerdings mit anderer Normierung als hier). Es wird daher vorgeschlagen, sie „Autocorrelation-Shell-Filter“ zu nennen. Als Abkürzung wird ACS(p) vorgeschlagen, wobei p die Zahl ist, die das Filter gemäß Gleichung (2.121) bestimmt. Die Konstanten von Tabelle 5.2 gehören also zu den Filtern ACS(2).

k	-3	-1	0	1	3
$32 \cdot h(k)$	-1	9	16	9	-1
$32 \cdot g(k)$	1	-9	16	-9	1

Tabelle 5.2: Beispiel für Filterkoeffizienten, die Gleichung (5.29) genügen

5.3 Das Lifting-Schema für die Wavelet-Transformation ohne Unterabtastung

Das Lifting-Schema ist ohne Unterabtastung bisher noch kaum angewandt worden. Die Vorteile der Speicher- und Rechenzeitersparnis (durch Reduktion der Zahl der notwendigen Additionen und Multiplikationen) entfallen, es bleibt jedoch die hohe Flexibilität hinsichtlich Randbedingungen und die Möglichkeit der Konstruktion neuer Filter. Außerdem ermöglicht das Lifting-Schema auch hier die Verwendung ganzzahliger Arithmetik, was dann doch zu einer erheblichen Rechenzeitersparnis ausgenutzt werden kann.

Die „Lazy“-Wavelet-Transformation von Abb. 3.1 ist hier durch folgende Transformation

$$s_1(k) = s_0(k) = x(k); \qquad r_1(k) = s_0(k) = x(k) \tag{5.30}$$

mit der Rekonstruktion

$$s_0(k) = \frac{1}{2}(s_1(k) + r_1(k)) \tag{5.31}$$

zu ersetzen, wie dies in Abb. 5.11 dargestellt ist.

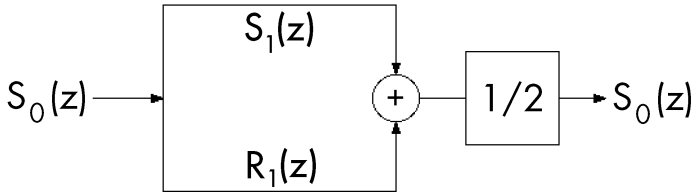


Abbildung 5.11: „Filterbank“ zum „Lazy“ Wavelet

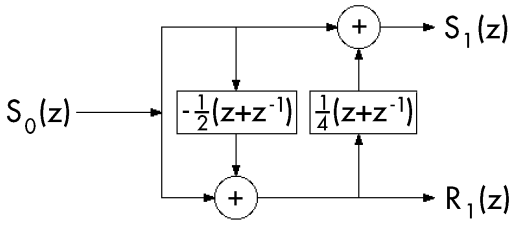


Abbildung 5.12: Beispiel für das Lifting-Schema ohne Unterabtastung (siehe Gleichungen (5.32) und (5.33))

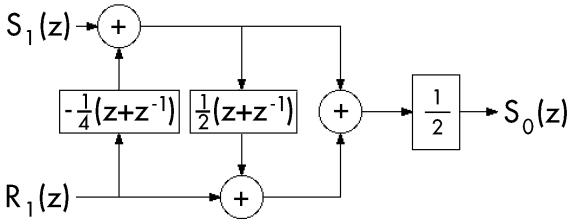


Abbildung 5.13: Rekonstruktion nach der Analyse von Abb. 5.12

Die Wavelettransformation ohne Unterabtastung, die dem Beispiel der Abb. 3.3 entspricht, ist für die erste Skalierungsstufe durch

$$\begin{aligned} s_1(k) &= x(k) = s_0(k); \qquad r_1(k) = x(k) = s_0(k) \\ r_1(k) &\mapsto r_1(k) - \frac{1}{2}\Big(s_1(k-1) + s_1(k+1)\Big) \qquad (5.32) \\ s_1(k) &\mapsto s_1(k) + \frac{1}{4}\Big(r_1(k-1) + r_1(k+1)\Big) \qquad (5.33) \end{aligned}$$

gegeben. Dies ist in Abb. 5.12 dargestellt. Dabei wurde berücksichtigt, dass hier die Operation „Aufspaltung“ einfach eine Verdoppelung der Daten darstellt. Die Indizes sind dadurch gegenüber dem Fall mit Unterabtastung verschoben. Die Rekonstruktion ergibt sich auch hier durch Wiederholung derselben Operationen in umgekehrter Reihenfolge und mit umgekehrtem Vorzeichen (siehe auch Abb. 5.13):

$$\begin{aligned} s_1(k) &\mapsto s_1(k) - \frac{1}{4}\Big(r_1(k-1) + r_1(k+1)\Big) \qquad (5.34) \\ r_1(k) &\mapsto r_1(k) + \frac{1}{2}\Big(s_1(k-1) + s_1(k+1)\Big) \qquad (5.35) \\ s_0(k) &= \frac{1}{2}(s_1(k) + r_1(k)) \end{aligned}$$

Die Analysefilter lassen sich direkt aus der Abb. 5.12 ablesen:

$$\tilde{G}(z^{-1}) = 1 - \frac{1}{2}(z + z^{-1}) \tag{5.36}$$

$$\begin{aligned}\tilde{H}(z^{-1}) &= \Big(1 - \frac{1}{2}(z + z^{-1})\Big)\frac{1}{4}(z + z^{-1}) + 1 \\ &= \frac{3}{4} + \frac{1}{4}(z + z^{-1}) - \frac{1}{8}(z^2 + z^{-2})\end{aligned} \tag{5.37}$$

Bis auf die in diesem Kapitel anders vorgenommene Normierung und eine Verschiebung von $\tilde{G}(z)$ stimmt dies mit den in (3.41) und (3.42) angegebenen Filtern CDF(2,2) überein. Analog kann man entsprechend der Abb. 5.13 durch Vergleich mit

$$H(z)S_1(z) + G(z)R_1(z) = S_0(z)$$

die Rekonstruktionsfilter direkt bestimmen. Man erhält

$$H(z) = \frac{1}{2} + \frac{1}{4}(z + z^{-1}) \tag{5.38}$$

$$G(z) = \frac{3}{8} - \frac{1}{8}(z + z^{-1}) - \frac{1}{16}(z^2 + z^{-2}) \tag{5.39}$$

Auch hier stimmt dies bis auf die unterschiedliche Normierung und eine Verschiebung von $G(z)$ mit den in (3.39) und (3.40) angegebenen Filtern überein.

Man erhält jedoch hier eine Besonderheit, die darauf beruht, daß man auf die Unterabtastung verzichtet hat und die Aufspaltung bei der Analyse eine Verdoppelung der Daten darstellt. Da man bei der Rekonstruktion Schrit für Schritt die Operationen der Analyse rückgängig macht, erhält man als Eingang des letzten Summenzeichens der Abb. 5.13 jeweils denselben Summanden $S_0(z)$. Man könnte also auf die letzte Summation verzichten, da jeder Summand bereits mit den ursprünglichen Daten übereinstimmt. Die Rekonstruktion in Abb. 5.13 stellt also die Bildung des arithmetischen Mittels von zwei verschiedenen Rekonstruktionen dar! Aus dem oberen Zweig entnimmt man

$$S_0(z) = S_1(z) - \frac{1}{4}(z + z^{-1})R_1(z) \tag{5.40}$$

Dies liefert die Rekonstruktionsfilter

$$H_0(z) = 1, \qquad G_0(z) = -\frac{1}{4}(z + z^{-1}) \tag{5.41}$$

Analog erhält man aus dem unteren Zweig

$$S_0(z) = \frac{1}{2}(z + z^{-1})S_1(z) + \Big(1 - \frac{1}{4}(z + z^{-1})\frac{1}{2}(z + z^{-1})\Big)R_1(z) \tag{5.42}$$

die Rekonstruktionsfilter

$$H_1(z) = \frac{1}{2}(z + z^{-1}), \qquad G_1(z) = \frac{3}{4} - \frac{1}{8}(z^2 + z^{-2}) \tag{5.43}$$

Die Rekonstruktionsfilter (5.38) und (5.39) ergeben sich aus diesen beiden Filtern durch

$$H(z) = \frac{1}{2}(H_0(z) + H_1(z)) \qquad \text{und} \qquad G(z) = \frac{1}{2}(G_0(z) + G_1(z)) \tag{5.44}$$

Es bleibt darauf hinzuweisen, daß zum Filter $H_0(z)$ keine „vernünftige“ Skalierungsfunktion gehört, während die Skalierungsfunktion zum Filter $H(z)$ die in Abb. 3.4 gezeigte „Hutfunktion“ ist. Es dürfte daher in der Praxis durchaus sinnvoll sein, die in Abb. 5.13 dargestellte Mittelung auch durchzuführen, weil dadurch eventuelle Nachteile der einzelnen Filter ausgeglichen werden können.

Entsprechende Überlegungen lassen sich auch bei einem allgemeineren Lifting-Schema ohne Unterabtastung anstellen. Die Rekonstruktionsfilter sind durch das Schema der Analyse eindeutig vorgegeben. Diese lassen sich darstellen als halbe Summe von je zwei einzelnen Filtern, wobei jedes dieser einzelnen Filterpaare für sich die Bedingung der perfekten Rekonstruktion (5.1) erfüllt.

Bis jetzt wurde nur die Wavelettransformation der ersten Skalierungsstufe besprochen. Der allgemeine Fall, d.h. die Berechnung der Koeffizienten von Stufe i aus denen der Stufe $i-1$ ergibt sich durch die Ersetzung von z durch $z^{2^{i-1}}$ (siehe Gleichungen (5.14) und (5.15) bzw. (5.12) und (5.13)). Für unser Beispiel erhalten wir für die Analyse

$$\begin{aligned} s_i(k) &= s_{i-1}(k); \qquad r_i(k) = s_{i-1}(k) && (5.45)\\ r_i(k) &\mapsto r_i(k) - \frac{1}{2}\Big(s_i(k-2^{i-1}) + s_i(k+2^{i-1})\Big) && (5.46)\\ s_i(k) &\mapsto s_i(k) + \frac{1}{4}\Big(r_i(k-2^{i-1}) + r_i(k+2^{i-1})\Big) && (5.47) \end{aligned}$$

Hinsichtlich Randbedingungen gilt dasselbe, was im Fall der Unterabtastung gesagt wurde (siehe Kapitel 2.9); symmetrische Fortsetzung dürfte akzeptable Ergebnisse liefern. Ebenso ermöglicht das Lifting-Schema auch hier, durch Runden oder Abschneiden als Ergebnis der Transformation ganze Zahlen zu bekommen (wie in Kapitel 3.1 beschrieben).

Lifting-Schritte für weitere Filterbeispiele:

(a) Die Analyse mit dem bisher besprochenen Filterpaar (5.18) und (5.19) (d.h. mit dem durch $^{(1)}$ gekennzeichneten Wavelet, also in der hier vorgeschlagenen Bezeichnungsweise mit den Analysefiltern von MZ(4,1), die mit den Analysefiltern von BML(4,1) übereinstimmen) kann durch folgende Lifting-Schritte ersetzt werden:

$$\begin{aligned} s_i(k) &= s_{i-1}(k); \qquad r_i(k) = s_{i-1}(k)\\ r_i(k) &\mapsto r_i(k) - \frac{1}{2}\Big(s_i(k) + s_i(k+2^{i-1})\Big) && (5.48)\\ s_i(k) &\mapsto s_i(k)\\ &\quad + \frac{1}{8}\Big(r_i(k-2\cdot 2^{i-1}) + 5r_i(k-2^{i-1}) - 5r_i(k) - r_i(k+2^{i-1})\Big) && (5.49) \end{aligned}$$

Die Rekonstruktionsfilter sind dadurch festgelegt. Sie sind entsprechend (5.44) darstellbar mit den Rekonstruktinsfiltern von (5.25) als $H_0(z)$ und $G_0(z)$ (in der hier vorgeschlagenen Bezeichnungsweise die Rekonstruktionsfilter von BML(4,1)) sowie den folgenden Filtern

$$\begin{aligned} H_1(z) &= \frac{1}{2}(z+1), && (5.50)\\ G_1(z) &= \frac{1}{16}(z^2 + 6z + 16 - 6z^{-1} - z^{-2}). && (5.51) \end{aligned}$$

Man beachte, dass $H_1(z)$ zur Haar-Skalierungsfunktion führt, aber $G_1(1) = 1$, also die entsprechende Funktion die Mindestforderung für ein Wavelet nicht erfüllt.

(b) Für die Analyse mit (5.18) und (5.20) (mit dem durch $^{(2)}$ gekennzeichneten Wavelet, also in der hier vorgeschlagenen Bezeichnungsweise mit den Analysefiltern von MZ(4,2), die mit den Analysefiltern von BML(4,2) übereinstimmen) ergeben sich folgende Lifting-Schritte:

$$\begin{aligned} s_i(k) &= s_{i-1}(k); \qquad r_i(k) = s_{i-1}(k) \\ r_i(k) &\mapsto r_i(k) - \frac{1}{4}\Big(s_i(k-2^{i-1}) + s_i(k+2^{i-1})\Big) - \frac{1}{2}s_i(k) \qquad (5.52) \\ s_i(k) &\mapsto s_i(k) - \frac{1}{4}\Big(r_i(k-2^{i-1}) + r_i(k+2^{i-1})\Big) - \frac{3}{2}r_i(k) \qquad (5.53) \end{aligned}$$

Die hierdurch festgelegten Rekonstruktionsfilter sind entsprechend (5.44) darstellbar mit $H_0(z) = 1$, dem durch (5.26) festgelegten $G_0(z)$ (in der hier vorgeschlagenen Bezeichnungsweise die Rekonstruktionsfilter von BML(4,2)) sowie den folgenden Filtern

$$\begin{aligned} H_1(z) &= \frac{1}{4}(z + 2 + z^{-1}) \qquad (5.54) \\ G_1(z) &= \frac{1}{16}(z^2 + 8z + 30 + 8z^{-1} + z^{-2}) \qquad (5.55) \end{aligned}$$

Hier führt $H_1(z)$ zur „Hutfunktion“ als Skalierungsfunktion, aber $G_1(z)$ nicht zu einer als Wavelet „zugelassenen“ Funktion.

(c) Für das Filterpaar aus Tabelle 5.2 (also in der hier vorgeschlagenen Bezeichnungsweise das Analysefilterpaar von ACS(2)) erhält man

$$\begin{aligned} s_i(k) &= s_{i-1}(k); \qquad r_i(k) = s_{i-1}(k) \\ r_i(k) &\mapsto r_i(k) - \frac{1}{2}s_i(k) + \frac{1}{32}\Big(s_i(k-3\cdot 2^{i-1}) + s_i(k+3\cdot 2^{i-1})\Big) \\ &\qquad -\frac{9}{32}\Big(s_i(k-2^{i-1}) + s_i(k+2^{i-1})\Big) \qquad (5.56) \\ s_i(k) &\mapsto s_i(k) - r_i(k) \qquad (5.57) \end{aligned}$$

Die Analyse-Lifting-Schritte für die Filter, die (5.29) genügen (also in der hier vorgeschlagenen Bezeichnungsweise die Analysefilter von ACS(p)) ergeben sich generell in sehr einfacher Weise aus den Filtern (siehe hierzu [30]). Für die Rekonstruktionsfilter, die durch diese Lifting-Schritte bestimmt sind, gilt die Summendarstellung (5.44) mit

$$H_0(z) = G_0(z) = 1, \quad H_1(z) = \tilde{H}(z^{-1}) \quad \text{und} \quad G_1(z) = \tilde{H}(z^{-1}) + 1 \qquad (5.58)$$

Die Rekonstruktion ist also durch das Lifting-Schema nicht mehr trivial!

(d) Weitere Beispiele kann man auch hier dadurch erhalten, dass man die einfache Vorhersage in (5.32) durch Interpolation mit einem Polynom ersten Grades durch eine aufwendigere mit Hilfe eines Polynoms höheren Grades ersetzt. Zahlenbeispiele hierfür sind im Anhang 3.3 angegeben.

5.4 Behandlung von Bilddaten

5.4.1 Transformation mit vier Subbändern

Man kann bei der Behandlung von Bildern genauso vorgehen, wie dies in Abschnitt 1.6 beschrieben ist, und lediglich auf die Unterabtastung verzichten. Für die erste Transformationsstufe erhält man damit ein Schema, wie es in Abb. 5.14 dargestellt ist. Benutzt man Filter, die eine Implementierung mit Hilfe des Lifting-Schemas mit einem Vorhersage- und einem Korrekturschritt erlauben (wie die meisten hier behandelten Beispiele), dann erhält man das in Abb. 5.15 gezeigte Schema. Das Ergebnis für das Testbild „boats“ ist in Abb. 5.16 gezeigt. Dabei wurde das Deslauriers-Dubuc-Filter DD(8,8) verwendet (siehe Anhang A.3.2).

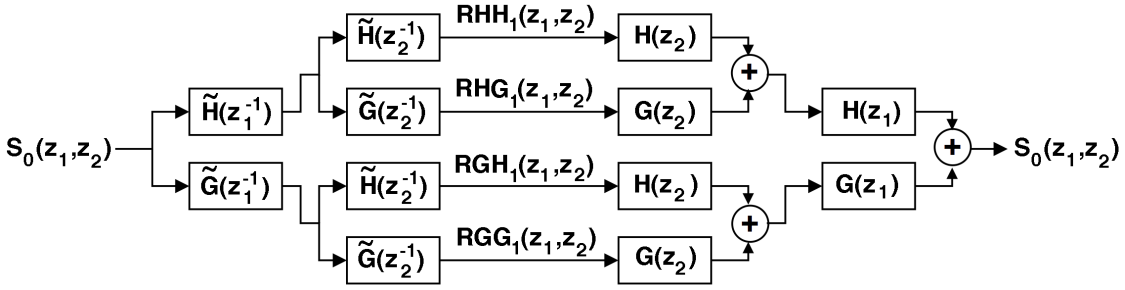


Abbildung 5.14: Transformation von Bilddaten ohne Unterabtastung mit 4 Subbändern, allgemeines Schema

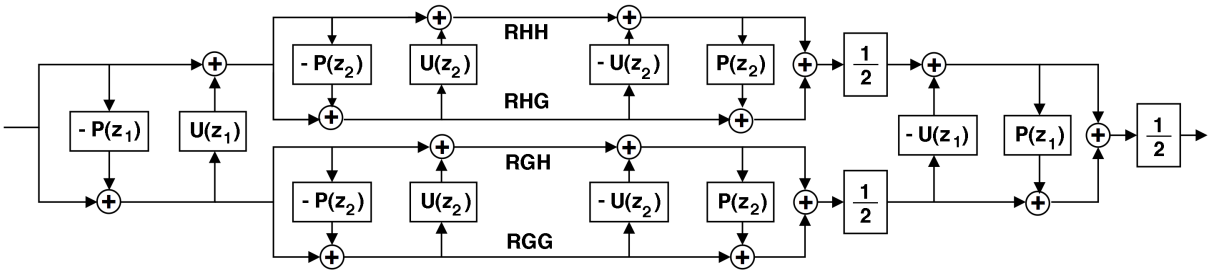


Abbildung 5.15: Transformation von Bilddaten ohne Unterabtastung mit 4 Subbändern, Implementierung mit dem Lifting-Schema

Das HH-Band (hier mit RHH bezeichnet) wird dann weiter transformiert. Bei der k. Skalierungsstufe hat man analog zu eindimensionalen Signalen z_1^{-1} und z_2^{-1} durch $z_1^{-2^{k-1}}$ und $z_2^{-2^{k-1}}$ zu ersetzen. Bei jeder Skalierungsstufe erhält man vier neue Subbänder. Bei M Skalierungsstufen erhält man damit $(3M+1)$ Subbänder, die alle dieselbe Größe wie das ursprüngliche Bild haben. Für das Testbild „boats“ und das Deslauriers-Dubuc-Filter DD(8,8) sind die Ergebnisse von der zweiten bis zur vierten Skalierungsstufe in der Abb. 5.17 gezeigt.

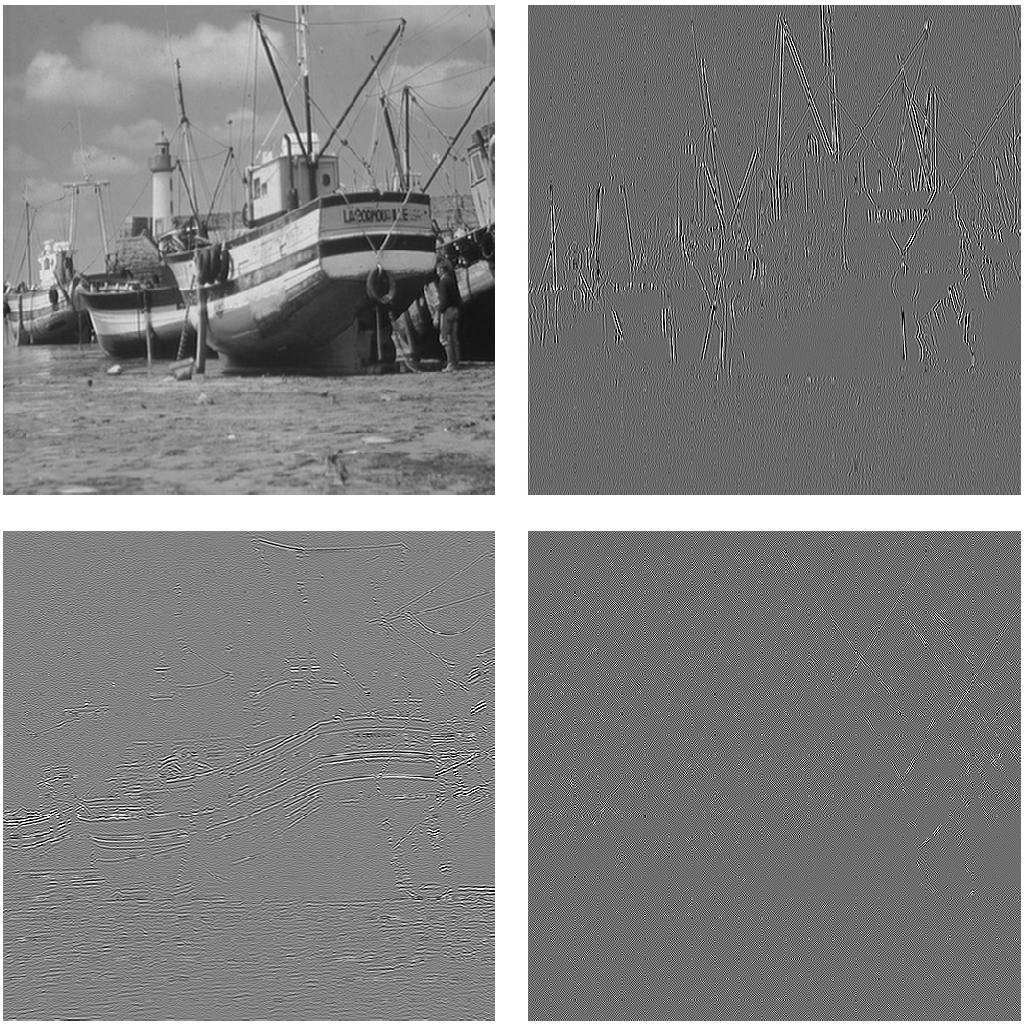


Abbildung 5.16: Transformation von Bilddaten ohne Unterabtastung mit vier Subbändern, Ergebnisse der 1. Skalierungsstufe für das Testbild „boats“. Die Anordnung der Subbänder entspricht der von Abschnitt 1.6

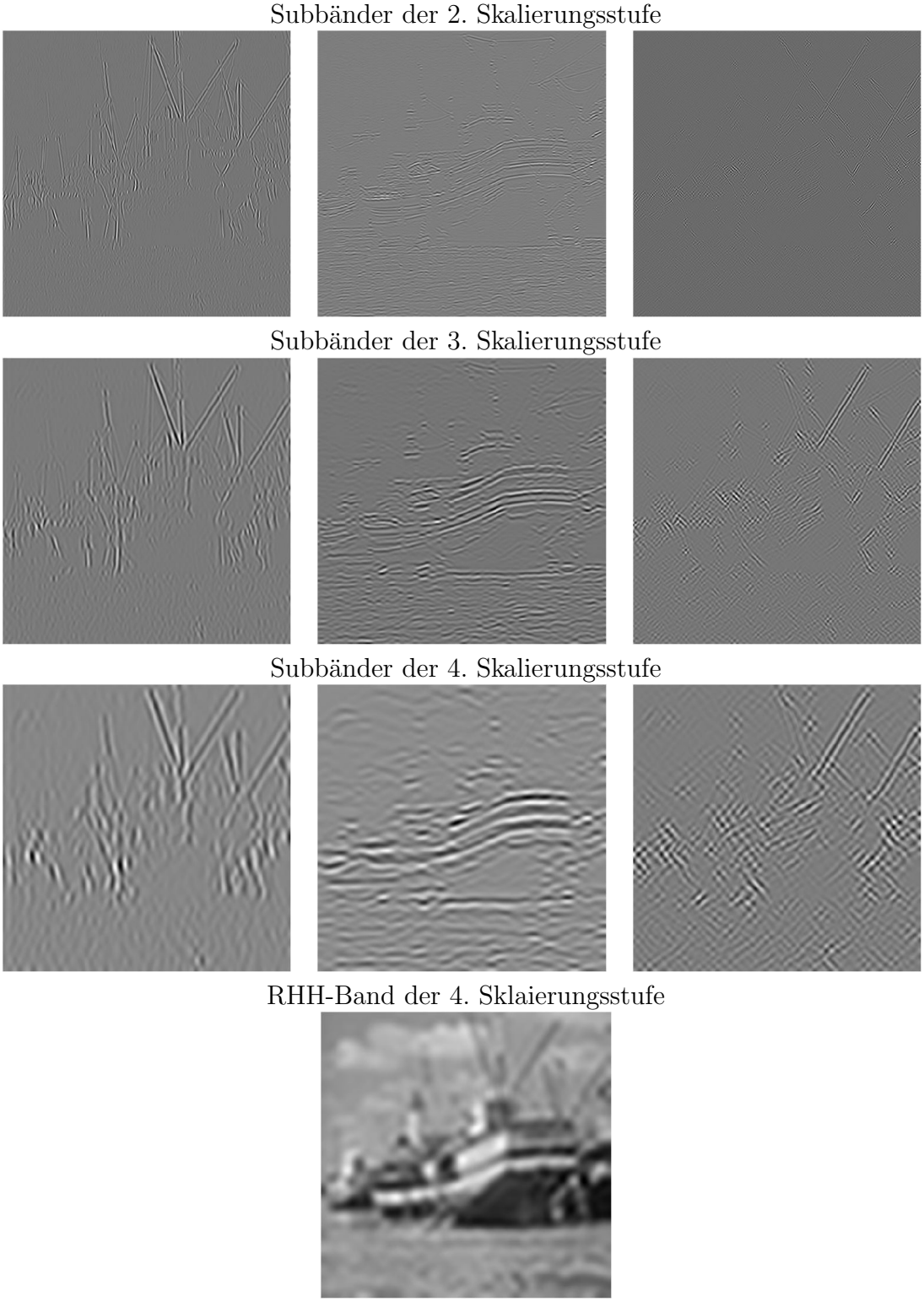


Abbildung 5.17: Transformation von Bilddaten ohne Unterabtastung mit vier Subbändern, Ergebnisse der 2. bis zur 4. Skalierungsstufe für das Testbild „boats“. Die Subbänder sind in der Reihenfolge RHG, RGH, RGG angeordnet

5.4.2 Transformation mit drei Subbändern

Die $(3M+1)$ Subbänder bei M Skalierungsstufen benötigen sehr viel Speicherplatz. Die entsprechende Transformation wurde daher bisher offensichtlich wenig in der Praxis angewandt. In [21] wird eine Anwendung auf das Entrauschen von Bildern beschrieben. In [9, 24, 25] werden nur drei Subbänder berechnet. Diese Möglichkeit soll in diesem Abschnitt näher besprochen werden. Ein Subband entsteht durch einer Hochpassfilterung der Zeilen, ein zweites durch Hochpassfilterung der Spalten. Tiefpassfilterung von Zeilen und Spalten liefert das HH-Subband als drittes Subband. Es stimmt überein mit dem, das man mit der Transformation von Abschnitt 5.4.1 erhält und das dort mit RHH bezeichnet wurde. In diesem Abschnitt werden die Koeffizienten des HH-Subbandes mit $s_m(n_1, n_2)$ bezeichnet (m steht für die Skalierungsstufe). Dieses Subband wird dann weiter transformiert. Wir erhalten damit bei M Skalierungsstufen nur $(2M+1)$ Subbänder.

In der in Abschnitt 1.6 eingeführten Bezeichnungsweise mit der z-Transformatierten haben wir für die erste Skalierungsstufe

$$R_1^{(1)}(z_1, z_2) = \tilde{G}(z_1^{-1}) S_0(z_1, z_2) \tag{5.59}$$

$$R_1^{(2)}(z_1, z_2) = \tilde{G}(z_2^{-1}) S_0(z_1, z_2) \tag{5.60}$$

$$S_1(z_1, z_2) = \tilde{H}(z_1^{-1}) \tilde{H}(z_2^{-1}) S_0(z_1, z_2) \tag{5.61}$$

Dies ist in Abb. 5.18 verdeutlicht. Dabei gehen wir wie üblich davon aus, dass die Werte $s_0(k_1, k_2)$ unsere ursprünglichen abgetasteten Daten (also die Helligkeit in der k_1. Zeile und k_2. Spalte) darstellen. Im Ortsbereich haben wir also

$$r_1^{(1)}(n_1, n_2) = \sum_{k=-\infty}^{\infty} \tilde{g}(k) s_0(n_1 + k, n_2) \tag{5.62}$$

$$r_1^{(2)}(n_1, n_2) = \sum_{k=-\infty}^{\infty} \tilde{g}(k) s_0(n_1, n_2 + k) \tag{5.63}$$

$$s_1(n_1, n_2) = \sum_{k_1, k_2=-\infty}^{\infty} \tilde{h}(k_1) \tilde{h}(k_2) s_0(n_1 + k_1, n_2 + k_2) \tag{5.64}$$

Diese Transformation ist in Abb. 5.18 schematisch dargestellt. Die qualitative Wirkung ist in Abb. 5.19 anhand des Testbilds „boats“ illustriert. Dabei wurde als Filter MZ(4,2) ausgewählt und die Transformation gemäß (5.52) sowie (5.53) durchgeführt.

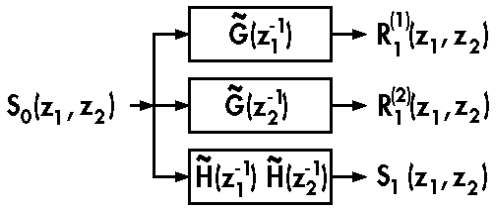


Abbildung 5.18: Transformation von Bilddaten ohne Unterabtastung mit drei Subbändern, allgemeines Schema

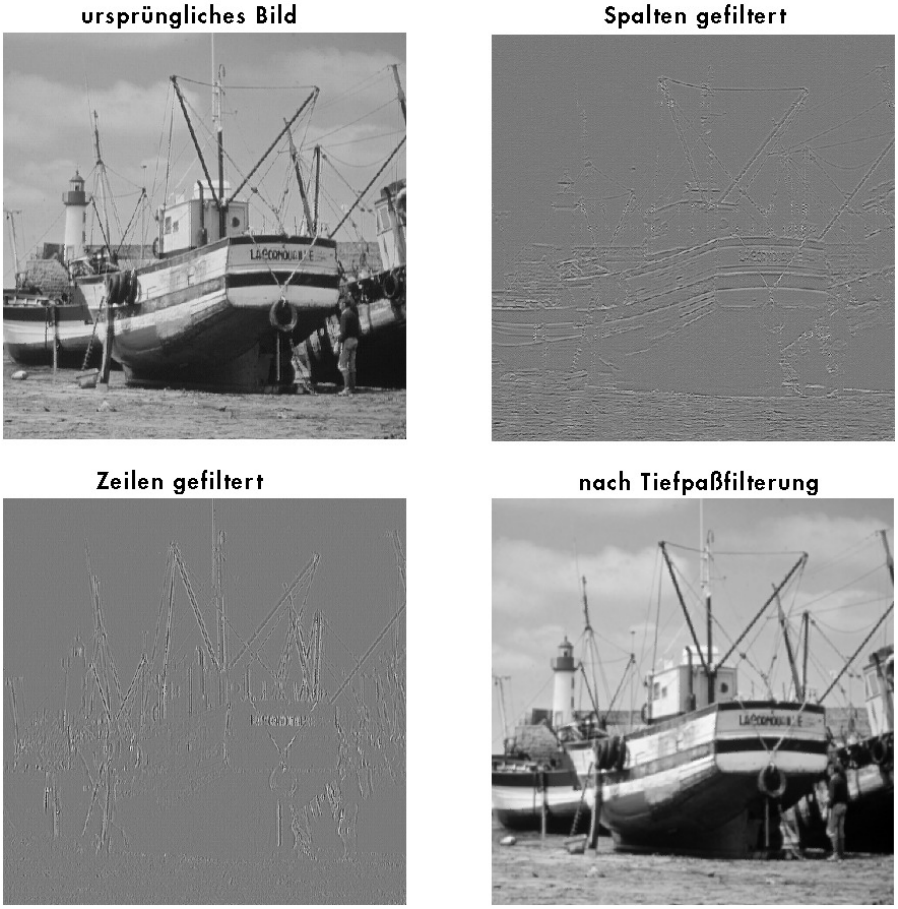


Abbildung 5.19: Transformation von Bilddaten ohne Unterabtastung mit drei Subbändern angewandt auf das Testbild „boats“

Für die i. Skalierungsstufe ist z^{-1} durch $z^{-2^{i-1}}$ bzw. $n+k$ durch $n+2^{i-1}k$ zu ersetzen. Damit erhält man

$$r_i^{(1)}(n_1,n_2) = \sum_{k=-\infty}^{\infty} \tilde{g}(k) s_{i-1}(n_1+2^{i-1}k, n_2) \tag{5.65}$$

$$r_i^{(2)}(n_1,n_2) = \sum_{k=-\infty}^{\infty} \tilde{g}(k) s_{i-1}(n_1, n_2+2^{i-1}k) \tag{5.66}$$

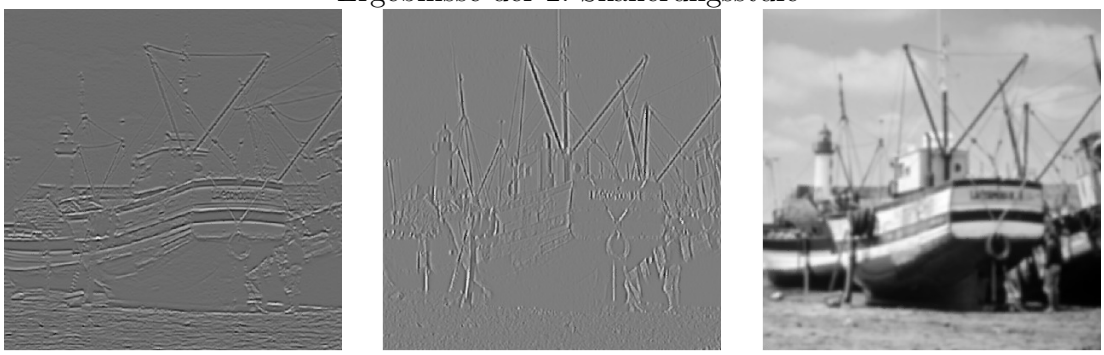
$$s_i(n_1,n_2) = \sum_{k_1,k_2=-\infty}^{\infty} \tilde{h}(k_1)\tilde{h}(k_2) s_{i-1}(n_1+2^{i-1}k_1, n_2+2^{i-1}k_2) \tag{5.67}$$

Das qualitative Ergebnis dieser Transformationen ist in Abb. 5.20 dargestellt. Auch hier wurde das Filter MZ(4,2) bzw. Gleichung (5.52) und (5.53) benutzt.

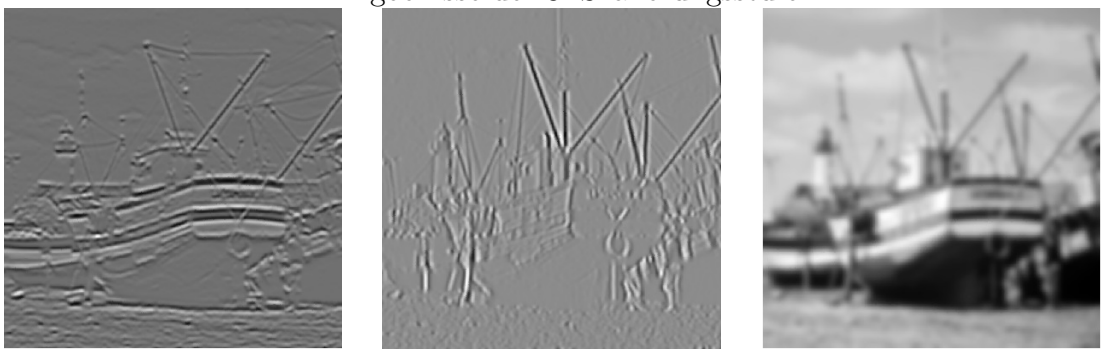
Die mit $^{(1)}$ gekennzeichneten Koeffizienten sind vor allem entlang der *horizontalen* Kanten des Bildes betragsmäßig besonders groß, entsprechen also den GH-Bändern. Entlang der *vertikalen* Kanten des Bildes sind die mit $^{(2)}$ bezeichneten Koeffizienten betragsmäßig groß — wie bei den HG-Bändern. Die Koeffizienten s_i entsprechen den HH-Bändern. Lediglich zu den GG-Bändern wird hier nichts Entsprechendes benutzt.

Für die Rekonstruktion muss man eine Tiefpassfilterung nachholen, die man bei der Analyse weggelassen hat. Dabei ergeben sich die beiden Möglichkeiten, die in Abb. 5.21

Ergebnisse der 2. Skalierungsstufe



Ergebnisse der 3. Skalierungsstufe



Ergebnisse der 4. Skalierungsstufe

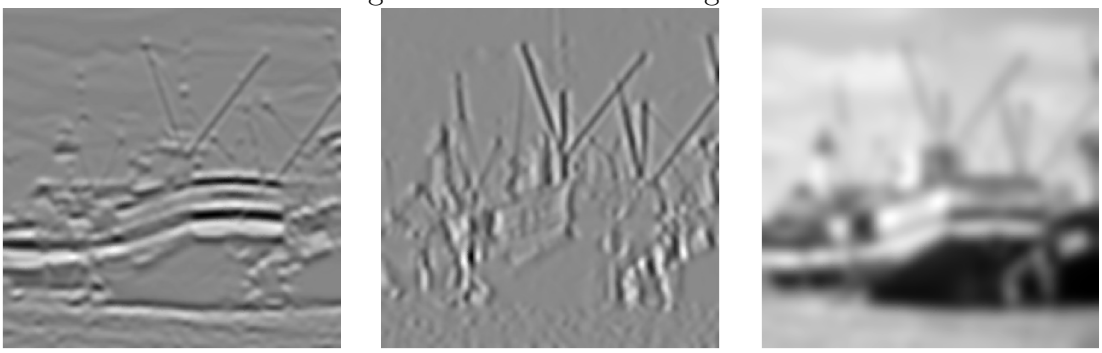


Abbildung 5.20: Transformation von Bilddaten ohne Unterabtastung mit drei Subbändern angewandt auf das Testbild „boats“, weitere Skalierungsstufen. Links ist das Ergebnis der Hochpassfilterung der Spalten, in der Mitte das der Zeilen, rechts das Ergebnis der Tiefpassfilterung.

und 5.22 gezeigt sind (als kleine Übungsaufgabe ist nachzurechnen, daß diese Filter tatsächlich eine perfekte Rekonstruktion gewährleisten, wenn die Bedingung (5.1) erfüllt ist).

Es ist sinnvoll, symmetrisch vorzugehen und nicht Zeilen oder Spalten einseitig zu behandeln. Man nimmt also den arithmetischen Mittelwert der beiden Rekonstruktionsmöglichkeiten. Dies kann man etwas vereinfachen, wenn man zur Abkürzung das Filter

$$L(z) := \frac{1}{2}\Big(1 + H(z)\tilde{H}(z^{-1})\Big) \tag{5.68}$$

definiert. Die Rekonstruktion erfolgt dann in der ersten Skalierungsstufe nach

$$S_0(z_1, z_2) = G(z_1)L(z_2)R_1^{(1)}(z_1, z_2) + G(z_2)L(z_1)R_1^{(2)}(z_1, z_2) + H(z_1)H(z_2)S_1(z_1, z_2) \tag{5.69}$$

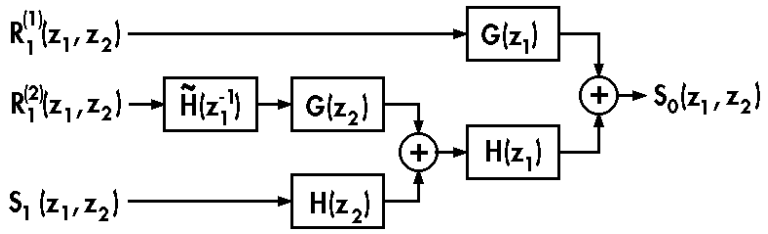


Abbildung 5.21: Möglichkeit zur Rekonstruktion

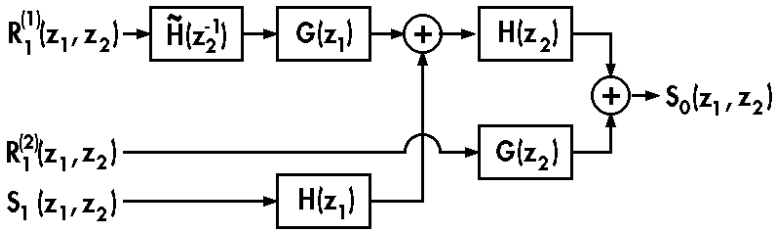


Abbildung 5.22: Weitere Möglichkeit zur Rekonstruktion

Dies ist in Abb. 5.23 dargestellt. Für die i. Skalierungsstufe ist z durch $z^{2^{i-1}}$ zu ersetzen.

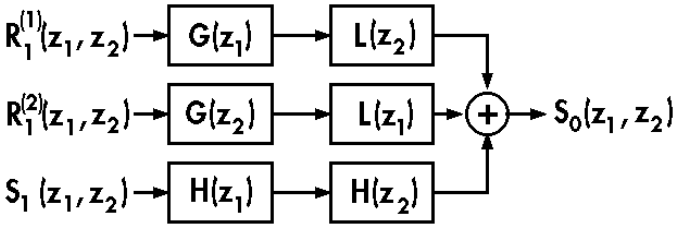


Abbildung 5.23: Symmetrische Rekonstruktion mit Filter (5.68)

Für das gebräuchliche Beispiel des Tiefpassfilters (5.18) sind die Koeffizienten von $L(z)$ in Tabelle 5.3 aufgeführt.

k	$2^9 \cdot l(k)$
0	326
± 1	56
± 2	28
± 3	8
± 4	1

Tabelle 5.3: Koeffizienten (Impulsantwort) des Filters (5.68) für das Beispiel (5.18)

Man kann die Transformation mit dem Lifting-Schema implementieren. Bei den meisten hier angeführten Filtern genügt dabei jeweils ein Vorhersage- und ein Korrekturschritt. Für diesen Fall erhält man für die in Abb. 5.21 gezeigte Rekonstruktion das in Abb. 5.24 gezeigte Schema. Analog erhält man für die Rekonstruktionsmöglichkeit von Abb. 5.22 das Schema der Abb. 5.25. Es ist auch bei einer Implementierung mit dem

Lifting-Schema sinnvoll, den arithmetischen Mittelwert der Werte zu bilden, die man mit den beiden Schemata erhält.

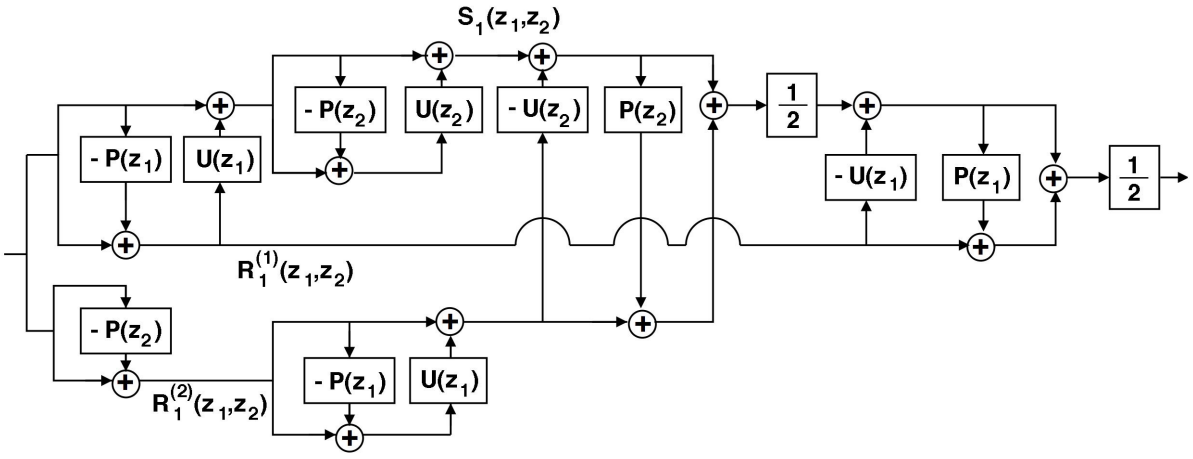


Abbildung 5.24: Lifting-Schema, das der Rekonstruktion nach Abb. 5.21 entspricht.

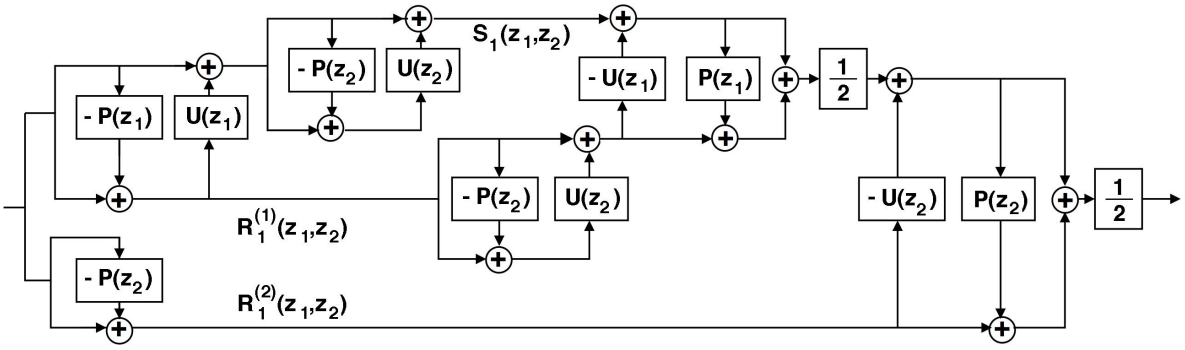


Abbildung 5.25: Lifting-Schema, das der Rekonstruktion nach Abb. 5.22 entspricht.

5.5 Rekonstruktion mit den Extremwerten oder Nulldurchgängen der Waveletkoeffizienten

Wenn auf die Unterabtastung verzichtet wird, hat man in den Koeffizienten eine große Redundanz. Die Rekonstruktion der ursprünglichen Daten ist, zumindest näherungsweise, auch mit andern Teilmengen möglich als mit denen, die durch Unterabtastung ausgewählt werden. Dabei kann man die Auswahl *adaptiv*, d.h. von den jeweiligen Daten abhängig, so vornehmen, dass das einfache Verhalten bei Translationen erhalten bleibt.

5.5.1 Rekonstruktion mit den Extremwerten

Diese Methode wurde zuerst von Mallat und Zhong in [25] beschrieben. Es wird hier jedoch eine einfachere Variante behandelt, die — bis auf eine geringfügige Abänderung — aus [9] stammt.

Die Rekonstruktion ist nur iterativ möglich, d.h. man erhält in mehreren Schritten eine jeweils bessere Näherung an die ursprünglichen Daten, wenn das Verfahren konvergiert. Ob dies der Fall ist, hängt entscheidend von den verwandten Filtern ab (für die Einzelheiten siehe [9]). In [25] wurde das Filter MZ(3,1) verwandt, das Verfahren sollte auch für MZ(4,1) sowie für ALPC(4) konvergieren, dagegen divergiert es für die BML-Filter (zur Bezeichnungsweise der Filter siehe Abschnitt 5.2).

Man führt für die vorgegebenen Daten $x(k) = s_0(k)$ die Wavelet-Transformation ohne Unterabtastung für eine maximale Anzahl M von Skalierungsstufen durch (gemäß (5.12) und (5.13)). Zu beachten ist dabei, daß $M \leq \log_2 N$, wobei N die Gesamtzahl der Daten ist. Denn bei weiteren Transformationen bliebe $s_i(k)$ konstant und $r_i(k)$ Null. Dann bestimmt man für $i = 1, \ldots M$ die Menge der Indizes, bei denen Extremwerte vorliegen durch

$$\begin{aligned} E_i^+ &:= \{k | r_i(k+1) \leq r_i(k) \text{ und } r_i(k-1) \leq r_i(k)\} && (5.70)\\ E_i^- &:= \{k | r_i(k+1) \geq r_i(k) \text{ und } r_i(k-1) \geq r_i(k)\} && (5.71)\\ E_{M+1}^+ &:= \{k | s_M(k+1) \leq s_M(k) \text{ und } s_M(k-1) \leq s_M(k)\} && (5.72)\\ E_{M+1}^- &:= \{k | s_M(k+1) \geq s_M(k) \text{ und } s_M(k-1) \geq s_M(k)\} && (5.73) \end{aligned}$$

Dabei wurde den Mengen, die aus den Indizes der Extremstellen der Koeffizienten $s_M(k)$ bestehen, einfach die Nummer $M+1$ zugeordnet. Sind mehr als zwei aufeinanderfolgende Koeffizienten gleich, dann wählt man nur einen Koeffizienten als Extremstelle aus, beispielsweise den mit dem niedrigsten Index. Die Extremwerte sind die Vereinigung der lokalen Maxima und Minima:

$$E_i := E_i^+ \cup E_i^-, \qquad i = 1, 2, \ldots M+1 \tag{5.74}$$

Man speichert dann nur die Werte (zusammen mit der Information, aus welcher Skalierungsstufe i sie stammen bzw. ob es sich um die Koeffizienten $s_M(k)$ handelt):

$$\{(k, r_i(k)) \mid k \in E_i,\ i = 1, 2, \ldots M\} \qquad \text{und} \qquad \{(k, s_M(k)) \mid k \in E_{M+1}\}$$

Für die *Rekonstruktion* wählt man als Startwerte die Koeffizienten, die durch lineare Interpolation zwischen den abgespeicherten Extremwerten entstehen: Da man nur die Extremwerte der graphischen Darstellung der Koeffizienten kennt, verbindet man diese einfach durch Geraden. Hierfür führt man dann die inverse Wavelet-Transformation durch

(sukzessive Anwendung der Rekonstruktionsfilter gemäß (5.17)) und gewinnt so eine ganz grobe erste Näherung $s_0^{(1)}(k)$ an die ursprünglichen Daten $s_0(k)$.

Führt man für die so gewonnene Näherung erneut die Wavelet-Transformation (durch Anwendung der Analyse-Filter gemäß (5.12) und (5.13)) durch, so erhält man nicht die durch lineare Extrapolation aus den abgespeicherten Extremwerten gewonnenen Startwerte zurück. Dies mag überraschend sein und bedarf einer kleinen Erklärung, die hier anhand eines ganz einfachen Zahlenbeispiels vorgenommen wird.

Wir betrachten $N = 4$ Datenpunkte, das Haar-Wavelet mit periodischen Randbedingungen sowie nur eine Skalierungsstufe, also $M = 1$. Da wir hier keine Unterabtastung vornehmen, sind die Normierungsfaktoren völlig anders zu wählen als im Beispiel des Abschnitts 1.2. Wir haben

$$\tilde{H}(z) = 1 + z^{-1}; \quad \tilde{G}(z) = 1 - z^{-1}; \quad H(z) = \frac{1}{4}(1 + z^{-1}); \quad G(z) = \frac{1}{4}(1 - z^{-1}) \tag{5.75}$$

Wir fassen die ursprünglichen Daten sowie die Wavelet-Koeffizienten zu Vektoren zusammen und beschreiben die Wavelettransformation (Analyse) durch eine Matrix $\mathbf{W}$:

$$\begin{pmatrix} s_1(0) \\ s_1(1) \\ s_1(2) \\ s_1(3) \\ r_1(0) \\ r_1(1) \\ r_1(2) \\ r_1(3) \end{pmatrix} = \mathbf{W} \begin{pmatrix} s_0(0) \\ s_0(1) \\ s_0(2) \\ s_0(3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} s_0(0) \\ s_0(1) \\ s_0(2) \\ s_0(3) \end{pmatrix} \tag{5.76}$$

Die Rücktransformation (Synthese) wird durch die Matrix $\mathbf{R}$ beschrieben:

$$\begin{pmatrix} s_0(0) \\ s_0(1) \\ s_0(2) \\ s_0(3) \end{pmatrix} = \mathbf{R} \begin{pmatrix} s_1(0) \\ s_1(1) \\ s_1(2) \\ s_1(3) \\ r_1(0) \\ r_1(1) \\ r_1(2) \\ r_1(3) \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} s_1(0) \\ s_1(1) \\ s_1(2) \\ s_1(3) \\ r_1(0) \\ r_1(1) \\ r_1(2) \\ r_1(3) \end{pmatrix} \tag{5.77}$$

Wir haben $\mathbf{R} = \frac{1}{4}\mathbf{W}^T$. Mit geeigneter Software kann man sich nun leicht überzeugen, dass

$$\mathbf{RW} = \mathbf{E} \tag{5.78}$$

wobei $\mathbf{E}$ die (4×4)-Einheitsmatrix ist. Aber

$$\mathbf{P} := \mathbf{WR} = \frac{1}{4} \begin{pmatrix} 2 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 2 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 2 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 & -1 & 2 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 2 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 & 2 \end{pmatrix} \tag{5.79}$$

ist keine Einheitsmatrix! Als Folge von (5.78) haben wir lediglich

$$\mathbf{P}^2 = (\mathbf{WR})^2 = \mathbf{W}(\mathbf{RW})\mathbf{R} = \mathbf{WER} = \mathbf{WR} \tag{5.80}$$

Lineare Abbildungen, die durch Matrizen $\mathbf{A}$ beschrieben werden, die $\mathbf{A}^2 = \mathbf{A}$ erfüllen, heißen *Projektionen*. Denn zwei- oder mehrmals projeziert ergibt dasselbe wie einmal projeziert. $\mathbf{WR}$ ist eine Projektion in den Bildraum der Wavelet-Transformation. Die durch lineare Interpolation der Extremwerte enstandenen Zahlen sind nämlich i.a. gar keine Waveletkoeffizienten eines Signals $x(0), x(1), \ldots x(N-1)$. Erst nach Anwendung von $\mathbf{WR}$ erhält man Waveletkoeffizienten eines Signals.

Kehren wir nun zum allgemeinen Fall zurück. Wir behalten die Bezeichnungsweise $\mathbf{W}$ für die Wavelet-Transformation sowie $\mathbf{R}$ für die Rücktransformation bei. Wir haben das Problem, daß die neuen Koeffizienten nach Anwendung von $\mathbf{WR}$ i.a. nicht mehr mit den abgespeicherten Extremwerten verträglich sind. Ihre Extrema liegen woanders, und es werden andere Werte angenommen. Wir müssen nun durch eine erneute, diesmal nichtlineare Projektion eine Übereinstimmung mit den abgespeicherten Bedingungen erreichen. Dies soll hier an einem Beispiel erläutert werden. Abb. 5.26 zeigt die Koeffizienten nach Anwendung von $\mathbf{WR}$ sowie die abgespeicherten Extremwerte (markiert durch $\bullet$).

Als erstes ersetzt man an den Extremstellen die durch Anwendung von $\mathbf{WR}$ gewonnenen Werte wieder durch die abgespeicherten Extremwerte. Dann muss man die Zwischenwerte so verändern, dass daraus auch wieder Extremstellen werden. Also müssen beispielsweise zwischen einem Maximum und Minimum die Werte monoton *abfallen*.

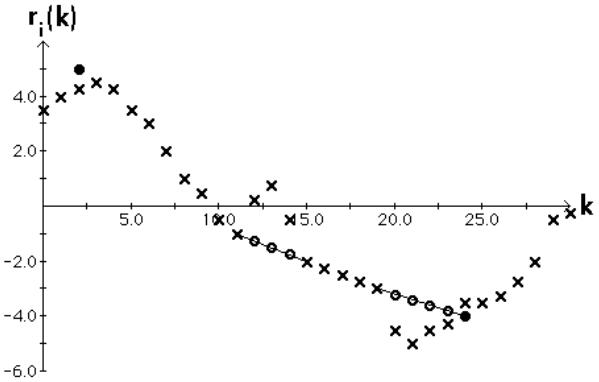


Abbildung 5.26: Anpassung der Näherung der Waveletkoeffizienten an die abgespeicherten Extremwerte (markiert durch $\bullet$). Die ursprünglichen Werte sind durch $\times$, die korrigierten durch $\circ$ markiert. Die Korrektur erfolgt durch lineare Interpolation für die Werte, die nicht mit den vorgegebenen Extremwerten verträglich sind.

Von der ersten Stelle k, an der diese Bedingung verletzt ist, d.h. an der $r_i(k+1) > r_i(k)$, geht man soweit (bis $k+l$), bis die Bedingung wieder erfüllt ist (also $r_i(k+l) \leq r_i(k)$ und ersetzt die dazwischen liegenden Werte durch lineare Interpolation zwischen $r_i(k)$ und $r_i(k+l)$ (graphisch ausgedrückt durch die Werte auf der Verbindungsgerade). Dies ist in Abb. 5.26 demonstriert. Entsprechend verfährt man, wenn Werte außerhalb des durch das Maximum und Minimum gegebenen Intervalls gelegen sind. Zwischen einem Minimum und einem Maximum verfährt man analog. Damit erhält man eine neue Näherung an Koeffizienten, die dieselben Extremstellen haben wie die der ursprünglichen Daten. Diese korrigierten Koeffizienten haben aber wiederum den Nachteil, nicht Wavelet-Koeffizienten

eines Signals zu sein. Dies erreicht man erst durch erneute Anwendung von **WR**, was allerdings wieder die Verträglichkeit mit den abgespeicherten Extremwerten zerstört. Man stellt sie durch das oben beschriebene Verfahren wieder her und kann erneut **WR** anwenden.

Zusammenfassend erhalten wir also folgendes iteratives Rekonstruktionsverfahren:
Start:

Lineare Interpolation zwischen den Extremstellen

Dann **Wiederholung** der Schritte

(a) Rücktransformation und anschließende Wavelet-Transformation: **WR**

(b) Anpassung an die vorgegebenen Extremwerte

bis sich die Koeffizienten dabei nur noch unwesentlich ändern. Durch abschließende Rücktransformation **R** erhält man dann die gewünschte Näherung an die ursprünglichen Daten.

5.5.2 Rekonstruktion mit den Nulldurchgängen

Für dieses Verfahren soll nur geschildert werden, was sich gegenüber der Rekonstruktion aus den Extremwerten ändert. Es wurde zuerst in [23] beschrieben, wir betrachten hier die in [9] angegebene Variante.

Man speichert die Stellen ab, an denen Nulldurchgänge auftreten:

$$Z_i := \{k \mid r_i(k) \cdot r_i(k+1) \leq 0\}, \quad i = 1, 2, \ldots M \tag{5.81}$$

$$Z_{M+1} := \{k \mid s_M(k) \cdot s_M(k+1) \leq 0\} \tag{5.82}$$

Sind mehr als zwei aufeinanderfolgende Koeffizienten gleich, dann wählt man nur einen Koeffizienten als Nullstelle aus, beispielsweise den mit dem niedrigsten Index. Mit $|Z_i|$ wird die Zahl der Nulldurchgänge der Koeffizienten der Skalierungsstufe i bezeichnet, die Nulldurchgänge werden mit k_l, $l = 1, 2, \ldots |Z_i|$ durchnumeriert. Zusätzlich zu den Nulldurchgängen sind die folgenden Summen der Koeffizienten zwischen den Nulldurchgängen abzuspeichern:

$$S_i(0) := \sum_{n=0}^{k_1} r_i(n);\ i = 1, 2, \ldots M \qquad S_{M+1}(0) := \sum_{n=0}^{k_1} s_M(n) \tag{5.83}$$

$$S_i(l) := \sum_{n=k_{l-1}}^{k_l - 1} r_i(n);\ i = 1, 2, \ldots M \qquad S_{M+1}(l) := \sum_{n=k_{l-1}}^{k_l - 1} s_M(n) \tag{5.84}$$

$$S_i(|Z_i|+1) := \sum_{n=k_{|Z_i|}}^{N-1} r_i(n);\ i = 1, 2, \ldots M \quad S_{M+1}(|Z_i|+1) := \sum_{n=k_{|Z_i|}}^{N-1} s_M(n) \tag{5.85}$$

Für diese Summen zählt also $r_i(0)$ und $r_i(N-1)$ als Nulldurchgang.

Als Startwerte für die Rekonstruktion bestimmt man zwischen den Nulldurchgängen Konstante so, dass deren Wert mit den abgespeicherten Summen verträglich ist. Dies bedeutet, dass die ursprünglichen Koeffizienten (deren Wert man ja nicht kennt) durch ihre Mittelwerte zwischen den Nulldurchgängen ersetzt werden. Diese Startwerte sind wiederum nicht Waveletkoeffizienten eines Signals $x(0), x(1), \ldots x(N-1)$. Man wendet daher die

lineare Projektion **WR** an und bekommt damit Koeffizienten, die nicht mehr mit den abgespeicherten Nulldurchgängen und Summen verträglich sind. Dies ist in Abbildung 5.27 illustriert.

Die Anpassung an Nulldurchgänge und Summen erfolgt hier zunächst, indem man die Koeffizienten an den Stellen, an denen sie ein falsches Vorzeichen haben (was nicht mit den abgespeicherten Nulldurchgängen verträglich ist) auf Null setzt (siehe Abb. 5.27). Anschließend verschiebt man die Werte zwischen den Nulldurchgängen so, dass die Summen die korrekten Werte annehmen (siehe Abb. 5.28). Die lineare Projektion **WR** und diese Anpassung werden dann so lange wiederholt, bis sich die Koeffizienten nicht mehr wesentlich ändern. Abschließende Rücktransformation **R** liefert die gewünschte Näherung an das ursprüngliche Signal. Konvergenz sollte für die Filter MZ(4,2) sowie ALPC(4) gewährleistet sein. Auch hier ist es unbekannt, ob das Verfahren bei Benutzung der Filter konvergiert, die durch die in Abschnitt 5.3 angegebenen Lifting-Schritte gegeben sind.

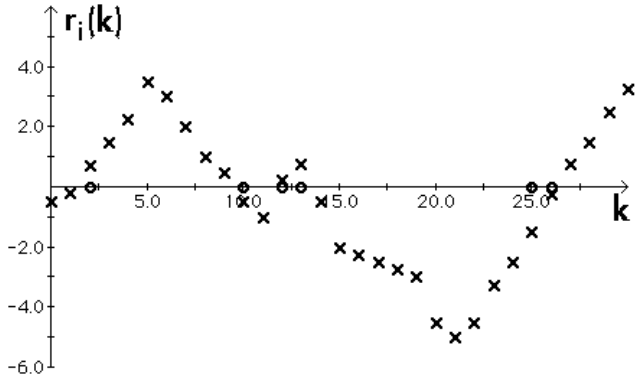


Abbildung 5.27: Anpassung der Näherung der Waveletkoeffizienten an die abgespeicherten Nulldurchgänge. Die ursprünglichen Werte sind durch $\times$, die korrigierten durch $\circ$ markiert.

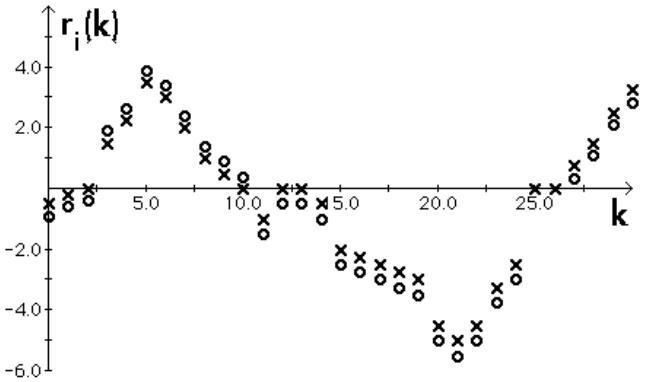


Abbildung 5.28: Anpassung der Näherung der Waveletkoeffizienten an die abgespeicherten Summen zwischen den Nulldurchgängen. Die ursprünglichen Werte sind durch $\times$, die korrigierten durch $\circ$ markiert.

Kapitel 6

Bildkompression mit Hilfe der Wavelet-Transformation

6.1 Überblick

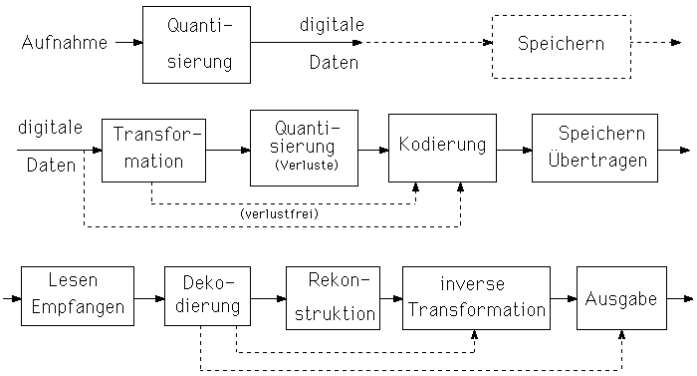


Abbildung 6.1: Übersicht über die einzelnen Schritte der Bilddatenkompression

Abbildung 6.1 gibt einen Überblick über die einzelnen Schritte einer Bilddatenkompression. In diesem Abschnitt wird die verlustbehaftete Kompression behandelt; die Möglichkeit einer verlustfreien Kompression ist in der Abbildung angedeutet; im Abschnitt 3.1 wurde auf eine derartige Möglichkeit kurz hingewiesen.

6.2 Warum ist die Wavelet-Transformation für die Datenkompression nützlich?

Die Bilddaten liegen in der Regel als positive ganze Zahlen zwischen 0 und 255 pro Pixel vor. Bei Graustufenbildern entspricht 0 Schwarz, 255 Weiß und 128 einem mittleren Grau. Farbbilder können als drei einzelne Bilder mit einer entsprechenden Darstellung behandelt werden.

Es gibt keinen Grund, weshalb eine bestimmte Graustufe (oder Farbintensität) statistisch sehr viel häufiger vorkommen sollte als andere. Abb. 6.2 zeigt die Häufigkeit der Grauwerte beim verbreiteten Testbild „Lena“. Derartige Darstellungen der Häufigkeit

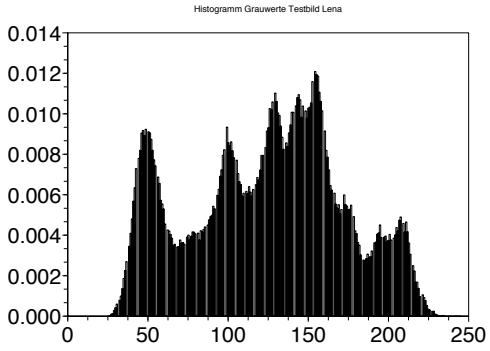


Abbildung 6.2: Histogramm (relative Häufigkeit der Grauwerte) für das Testbild „Lena“ (siehe Abb. 1.6)

werden auch „Histogramme“ genannt. Demgegenüber ergibt sich nach der Wavelet-Transformation eine geänderte Verteilung, wie sie für die einzelnen „Subbänder“ in Abb. 6.3 gezeigt ist.

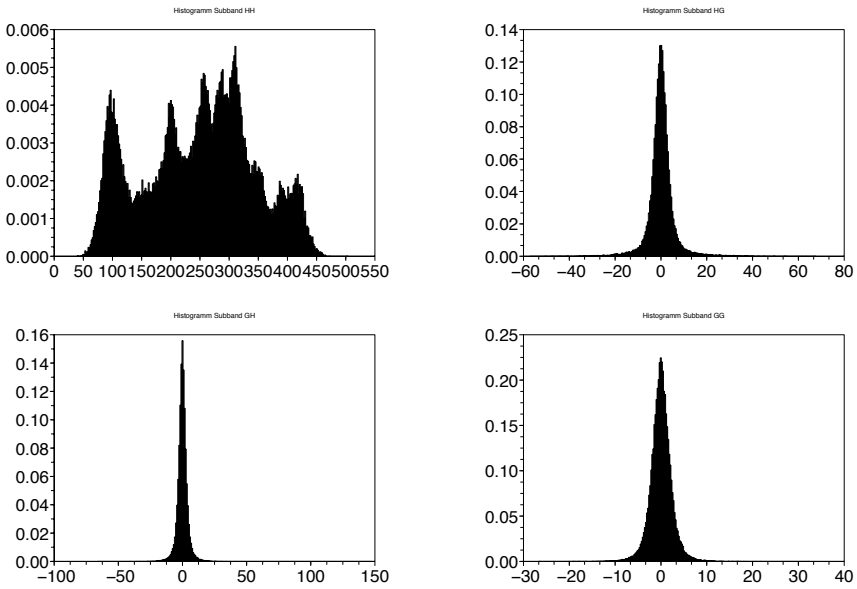


Abbildung 6.3: Histogramm (relative Häufigkeit der Grauwerte) für die Waveletkoeffizienten (hier mit dem Binlet, DD(4,2)) für das Testbild „Lena“. Die einzelnen Subbänder sind dabei wie in Abb. 1.7 angeordnet. Die meisten Koeffizienten liegen für alle Subbänder außer dem Subband „HH“ nahe dem Wert Wert 0.

Im Subband „HH“ (Tiefpassfilterung in Zeilen- und Spaltenrichtung, zur Bezeichnungsweise der Subbänder siehe Abschnitt 1.6) entspricht die Verteilung der ursprünglichen Verteilung. Bei den übrigen drei Subbändern sind die meisten Koeffizienten nahe dem Wert „Null“. Große Abweichungen ergeben sich nur an den Kanten des Bildes (für das Subband „HG“ an den vertikalen, für „GH“ an den horizontalen sowie für „GG“ an den diagonalen Kanten).

Häufigkeitsverteilungen, bei denen einige wenige Werte sehr häufig vorkommen, die

graphisch dargestellt ein ausgeprägtes Maximum besitzen, sind für die anschließende Kodierung sehr viel günstiger (d.h. ermöglichen eine bessere Kompression) als etwa eine Gleichverteilung. D.h. Verteilungen wie die für die Subbänder „HG“ , „GH“ und „GG“ ermöglichen eine gute Kompression, das Subband „HH“ wird in der Regel noch weiter transformiert, bis das entsprechende verbliebene Subband „HH“ so klein ist (die Zahl der Werte schrumpft bei jeder Stufe auf ein Viertel der ursprünglichen), dass es für die Kompression irrelevant ist (siehe beispielsweise Abb. 1.15). In der Praxis wird die Wavelettransformation in etwa drei bis fünf Stufen durchgeführt.

6.3 Quantisierung

Wie aus Gleichungen (2.36) und (2.37) ersichtlich ist, werden durch die Wavelet-Transformation die ursprünglich ganzzahligen Grauwerte des Bildes in reelle, im allgemeinen nicht ganzzahlige Koeffizienten umgewandelt. Zwar stehen für die Darstellung reeller Zahlen im Rechner auch nur endlich viele Werte zur Verfügung (z.B. für den Typ „float“ oder „double“ in der Programmiersprache C). Doch für eine effektive Kompression kann man nur ganzzahlige Werte aus einem nicht zu großen Wertevorrat gebrauchen. Die notwendige Umwandlung geschieht durch eine Vorschrift, die jedem *kontinuierlichen* Wert einen diskreten, d.h. *ganzzahligen Wert* zuordnet. Dieser Vorgang heißt *Quantisierung*. Hier treten die Verluste auf, da diese Abbildung nicht umkehrbar ist. Abb. 6.4 und 6.5 zeigen zwei verschiedene Beispiele, bei denen das Intervall $[0, 1]$ auf die Zahlen $\{0, 1, 2, \ldots 7\}$ bzw. das Intervall $[-1, +1]$ auf die Zahlen $\{-3, -2, \ldots + 3\}$ abgebildet wird. Im ersten Beispiel (Abb. 6.4) werden alle Zahlen des Intervalls $[0, \Delta h[$ auf den diskreten Wert 0 abgebildet. Als Rekonstruktionswert nimmt man im allgemeinen die Mitte eines solchen Intervalls. Dem quantisierten Wert 0 entsteht dann bei der Rekonstruktion der Wert $\frac{1}{2}\Delta h$ (in diesem Beispiel ist $\Delta h = \frac{1}{8}$).

Liegen die kontinuierlichen Werte symmetrisch zu 0 (wie im 2. Beispiel, Abb. 6.4), dann ist es sinnvoll, die Quantisierung so vorzunehmen, dass die entsprechenden Intervalle ebenfalls symmetrisch zu 0 liegen. Damit kommt 0 wieder als Rekonstruktionswert vor.

Die Zahl der durch die Quantisierung auftretenden diskreten Werte muss keinesfalls eine Zweierpotenz sein, da man zur Kompression meist sehr spezielle Kodierungen (siehe Abschnitt 6.4) einsetzt. Bei den beiden Beispielen handelt es sich um eine Quantisierung mit gleichförmiger Stufenhöhe, die also durch eine einzige Stufenhöhe (hier mit Δh bezeichnet) charakterisiert ist.

Eine Quantisierung durch ein Rechnerprogramm führt man durch eine Division durch anschließendes Runden durch. D.h. die Berechnung der quantisierten Koeffizienten q_{ij} aus den kontinuierlichen c_{ij} erfolgt durch

$$\textbf{Runden von} \quad \frac{c_{ij}}{\Delta h}$$

Die Rekonstruktion erfolgt durch

$$r_{ij} = q_{ij} \cdot \Delta h$$

Für die Waveletkoeffizienten hat sich eine spezielle Quantisierung mit ungleichförmiger Stufenhöhe als besonders nützlich herausgestellt. Da vom Betrag her kleine nichtverschwindende Waveletkoeffizienten häufig vom Rauschen stammen, ist es sinnvoll, alle Koeffizienten unterhalb einer bestimmten Schwelle zu Null zu quantisieren, wie es in Abb. 6.6

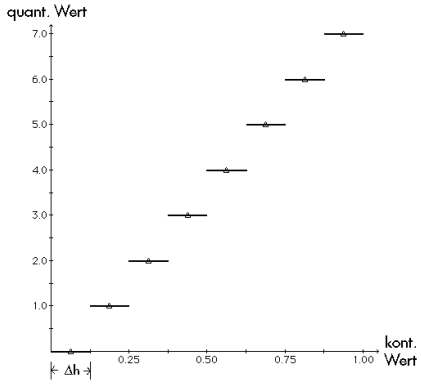


Abbildung 6.4: Quantisierung mit gleichförmiger Stufenhöhe, in diesem Beispiel ist die Stufenhöhe $\Delta h = \frac{1}{8}$. Die Rekonstruktionswerte sind durch $\triangle$ markiert.

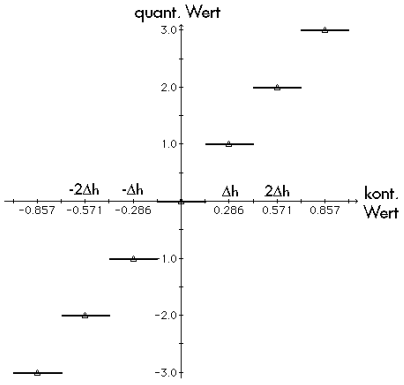


Abbildung 6.5: Symmetrische Quantisierung mit 0 als Rekonstruktionswert, hier ist die Stufenhöhe $\Delta h = \frac{2}{7}$ (Rekonstruktionswerte durch $\triangle$ markiert).

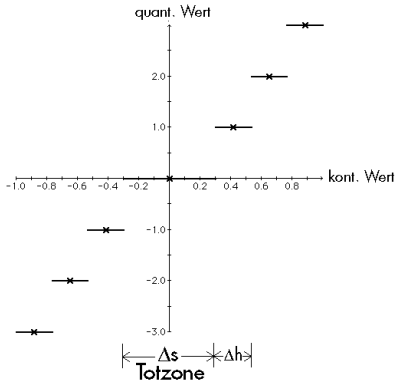


Abbildung 6.6: Totzonenquantisierung. In der Praxis hat sich die Wahl $\Delta s \approx 2\Delta h$ als sinnvoll erwiesen.

gezeigt ist. Dabei hat es sich in der Praxis bewährt, die Länge der Totzone Δs etwa doppelt so groß zu wählen wie die Quantisierungsstufenhöhe Δh.

Bei der skalaren Quantisierung wird die reelle Achse in Teilintervalle unterteilt und jedes Teilintervall auf ein Element einer diskreten Menge (meist eine Teilmenge von $\mathbb{Z}$)

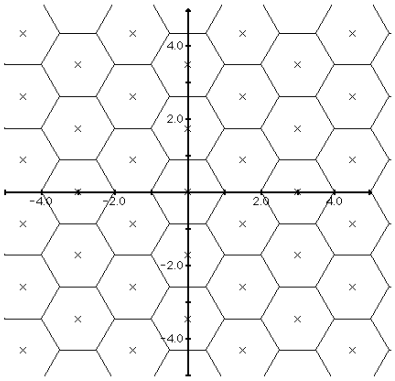


Abbildung 6.7: Beispiel zur Vektorquantisierung im Fall $n = 2$: Einteilung der Ebene in Sechsecke. Die Rekonstruktionswerte sind durch $\times$ markiert.

abgebildet. Alle bisher vorgestellten Quantisierungsmethoden fallen unter diese Kategorie. Bei der Vektorquantisierung faßt man je n kontinuierliche Datenwerte zu einem Vektor in $\mathbb{R}^n$ zusammen und unterteilt $\mathbb{R}^n$ in Teilmengen. Jeder solchen Teilmenge wird dann ein Element einer diskreten Menge zugeordnet. dass man durch die höhere Dimension sehr viel mehr Möglichkeiten der Unterteilung hat, kann man sich am einfachsten am Beispiel $n = 2$ klarmachen. Je 2 aufeinanderfolgende Datenwerte werden zu einem Vektor in $\mathbb{R}^2$ (darstellbar als Punkt der Ebene) zusammengefaßt. Eine Einteilung der Ebene in gleichgroße Quadrate entspricht dann der skalaren Quantisierung. Bei einer gleichförmigen Verteilung der Daten ist jedoch eine Einteilung der Ebene in Sechsecke (wie in Abb. 6.7 gezeigt) besonders günstig. Bei ungleichförmigen Verteilungen hat man sehr viel mehr Möglichkeiten, die Einteilung an die entsprechenden Daten anzupassen. Nachteile sind jedoch eine aufwendigere Programmierung sowie eine höhere Rechenzeit.

6.4 Allgemeine Methoden zur Kodierung

Nach der Quantisierung liegen die Koeffizienten nur noch als Elemente einer endlichen Teilmenge von $\mathbb{Z}$ (oder einer anderen endlichen Menge) vor. Diese Zahlen direkt als Binärzahlen abzuspeichern oder zu übertragen, würde noch keinen großen Erfolg für die Kompression bringen. Hier werden drei Methoden kurz vorgestellt, durch eine geschickte Kodierung eine bessere Kompression zu erreichen.

6.4.1 Lauflängenkodierung

Statt 17 mal hintereinander dasselbe Zeichen zu übertragen, kann man auch die Zahl 17 und dann das Zeichen übertragen, d.h. stets ein Paar (Zahl, Zeichen) übertragen oder abspeichern. Da nach Wavelettransformation und Quantisierung die Null besonders häufig und die übrigen Zahlen recht selten vorkommen, ist hier die Form (n_1, k_1), $(n_2, k_2), \ldots$ besonders effektiv, wobei die Lauflänge n_i die Zahl der Nullen bis zum nächsten von Null verschiedenen Zeichen k_i angibt. Sie wird insbesondere mit anschließender Huffman- oder arithmetischer Kodierung verwandt.

6.4.2 Huffmankodierung

Die Huffmankodierung verwendet kürzere Kodewörter für häufige Zeichen und längere Kodewörter für seltene Zeichen. Dieses Prinzip wurde auch beim Morse-Alphabet verwandt. Für den Rechner oder die moderne digitale Übertragungstechnik heißt dies: weniger Bits für häufige Zeichen. Das Schema zur Erstellung der entsprechenden Kodierungstabelle wird aus Tabelle 6.1 und Abb. 6.8 ersichtlich.

z	p	k	l	$p \cdot l$
0	0.01	000000	6	0.06
1	0.03	00001	5	0.15
2	0.14	001	3	0.42
3	0.28	10	2	0.56
4	0.29	11	2	0.58
5	0.16	01	2	0.32
6	0.07	0001	4	0.28
7	0.02	000001	6	0.12
mittl. Länge (gew.):				2.49

Tabelle 6.1: Beispiel für die Huffmann-Kodierung. Bedeutung der Spalten: z zu kodierendes Zeichen, p relative Häufigkeit, k Kodewort, l Länge des Kodeworts, $l \cdot p$ Produkt aus rel. Häufigkeit u. Länge

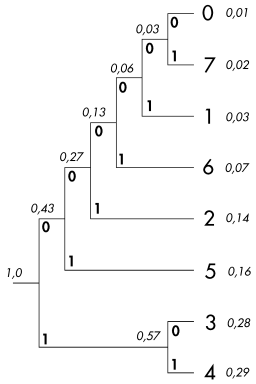


Abbildung 6.8: Beispiel zur Huffmankodierung (mit den relativen Häufigkeiten der Tabelle 6.1)

Man hat zunächst die relative Häufigkeit der einzelnen Zeichen (hier Zahlen) festzustellen, und zwar anhand der aktuellen Daten — oder anhand von Modellen oder Testdaten (z.B. Testbildern). Die Häufigkeiten des Beispiels sind einfach angenommen. Der Kode entsteht dadurch, daß man die beiden am wenigsten häufigen Zeichen willkürlich mit 0 und 1 versieht und dann zu einem einzigen Zeichen zusammenfaßt und die einzelnen Zeichen aus der Tabelle streicht. Die relative Häufigkeit dieses zusammengefaßten Zeichens ergibt sich als *Summe* der einzelnen Häufigkeiten. Dies setzt man fort, bis alle Zeichen zu einem einzigen der relativen Häufigkeit 1 zusammengefaßt sind. Im Beispiel der Häufigkeiten von Tabelle 6.1 entsteht dann der von rechts nach links der Baum von Abb. 6.8. Den Kode für ein einzelnes Zeichen erhält man dann, indem man von der Wurzel (links) zum Zeichen

(rechts) geht und die einzelnen Bits an den Verzweigungen „aufsammelt“. Umgekehrt weisen bei der Dekodierung die Bits den Weg von der Wurzel zum Zeichen.

Bildet man für jedes Zeichen das Produkt aus der relativen Häufigkeit und seiner Länge (in Bits) und addiert die entstehenden Werte, so erhält man die *mittlere Länge* eines Zeichens. Es handelt sich hierbei um den *gewichteten* Mittelwert, der mit dem „einfachen“ Mittelwert (addieren und durch die Gesamtzahl dividieren) nur übereinstimmt, wenn alle relativen Häufigkeiten gleich großsind. Der Huffmann-Kode liefert unter allen möglichen Kodes, die für jedes Zeichen eine feste Länge verwenden, das beste Ergebnis (d.h. die kürzeste mittlere Länge).

Man beachte, dass der Huffman-Kode ohne ein besonderes Trennzeichen auskommt. Dies ist möglich, weil kein Zeichen „Vorsilbe“ eines andern Zeichens ist (ein derartiges Prinzip wird auch bei Telefonnummern eingehalten). Solche Kodes heißen „Präfixkodes“.

Eine Verbesserung gegenüber dem Huffmankode erhält man, wenn man die Beschränkung einer festen Bitlänge je Zeichen aufgibt. Dies ist für den arithmetischen Kode der Fall.

6.4.3 Arithmetische Kodierung

Die Implementierung eines arithmetischen Kodierers ist recht verwickelt. Es kann daher hier nur auf das Prinzip eingegangen werden. Wir gehen von einem Beispiel mit demselben Zeichenvorrat und denselben relativen Häufigkeiten der einzelnen Zeichen aus wie in Unterabschnitt 6.4.2 (siehe Tabelle 6.1). Das Intervall $[0, 1[$ wird nun in Teilintervalle eingeteilt, deren Länge proportional zur relativen Häufigkeit der Zeichen ist (siehe Tabelle 6.2).

z	p	Intervall
0	0.01	[0.0 , 0.01[
1	0.03	[0.01, 0.04[
2	0.14	[0.04, 0.18[
3	0.28	[0.18, 0.46[
4	0.29	[0.46, 0.75[
5	0.16	[0.75, 0.91[
6	0.07	[0.91, 0.98[
7	0.02	[0.98, 1.0[

Tabelle 6.2: Beispiel für die arithmetische Kodierung. Bedeutung der Spalten: z zu kodierendes Zeichen, p relative Häufigkeit.

Jede Zahl aus dem betreffenden Teilintervall stellt das zugehörige Zeichen dar. Für das folgende Zeichen wird nun das jeweilige Teilintervall weiter unterteilt, jeweils wieder mit Teillängen entsprechend der relativen Häufigkeit, wie das in Abb. 6.9 dargestellt ist. Das folgende Kodierungsbeispiel soll dieses Prinzip verdeutlichen:

zu kodieren: 47110; Intervall jeweils $[a, b[$, Länge: d

4 liefert $a = 0.46$, $b = 0.75$, $d = 0.29$

7 liefert $a = 0.46 + 0.29 \cdot 0.98 = 0.7442$,
$b = 0.46 + 0.29 \cdot 1.0 = 0.75$, $d = 0.0058$

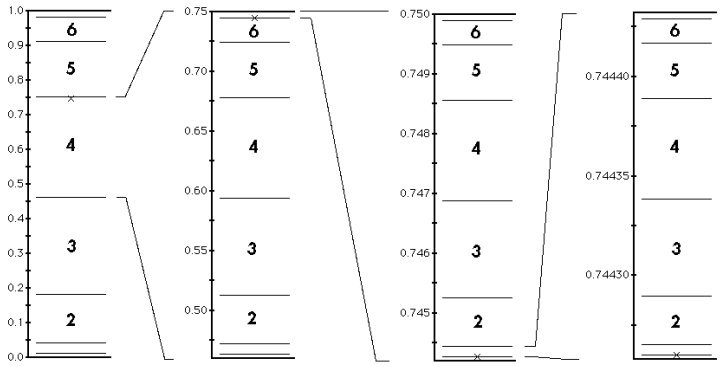


Abbildung 6.9: Zum Prinzip der arithmetischen Kodierung (siehe auch Tabelle 6.2).

1 liefert $a = 0.7442 + 0.0058 \cdot 0.01 = 0.744258$,
$b = 0.7442 + 0.0058 \cdot 0.04 = 0.744432$, $d = 0.000174$

1 liefert $a = 0.744258 + 0.000174 \cdot 0.01 = 0.74425974$,
$b = 0.744258 + 0.000174 \cdot 0.04 = 0.74426496$, $d = 5.22 \cdot 10^{-6}$

0 liefert $a = 0.74425974$, $b = 0.74425974 + 5.22 \cdot 10^{-6} \cdot 0.01 = 0.74426148$,
$d = 1.74 \cdot 10^{-6}$

gesendet: $a = x = 0.74426148$

Dekodierung:

$0.46 \leq x < 0.75 \implies$ 1. Zeichen: 4
$x_{\text{neu}} = (x_{\text{alt}} - 0.46)/0.29 = 0.980206$

$0.98 \leq x < 1.0 \implies$ 2. Zeichen: 7
$x_{\text{neu}} = (x_{\text{alt}} - 0.98)/0.02 = 0.0103$

$0.01 \leq x < 0.04 \implies$ 3. Zeichen 1
$x_{\text{neu}} = (x_{\text{alt}} - 0.01)/0.03 = 0.01$

$0.01 \leq x < 0.04 \implies$ 4. Zeichen 1
$x_{\text{neu}} = (x_{\text{alt}} - 0.01)/0.03 = 0$

$0 \leq x < 0.01 \implies$ 5. Zeichen 0

Wie aus diesem Beispiel deutlich wird, benötigt der Dekodierer einen Befehl zum Anhalten. Er würde sonst im Beispiel weitere Nullen anhängen. Entweder muss man vorher die Gesamtzahl der Zeichen übertragen oder eines der Zeichen als Stopzeichen vereinbaren, was dann die niederste relative Häufigkeit besitzt.

In der Praxis wird man nicht mit reellen Zahlen (bzw. Zahlen vom Typ float“ oder „double“) arbeiten, sondern mit ganzen Zahlen. Für die Einzelheiten der Implementierung wird auf die Literatur (siehe z.B. [2, 17, 41]) verwiesen.

6.5 Kodierung von Waveletkoeffizienten

Man könnte einfach die quantisierten Wavelet-Koeffizienten einem Programmodul zur Huffman- oder arithmetischen Kodierung zuführen. Dies liefert jedoch nicht optimale Ergebnisse. Zu einem erheblichen Fortschritt führte die von Shapiro [29] eingeführte Methode der „Zerotrees“. Diese wurde unter anderem von Said und Pearlman [27] weiterentwickelt und verbessert. Das Grundprinzip ist in Unterabschnitt 6.5.1 beschrieben.

Überraschenderweise bringt die später vorgeschlagene, viel einfacher zu implementierende Methode des „Stack-Run Coding“ [35] fast vergleichbare Ergebnisse. Sie ist in Unterabschnitt 6.5.2 erläutert.

6.5.1 Zerotrees

Betrachtet man Abb. 1.15 genauer, so stellt man fest, daß sich in allen Subbändern dasselbe Muster des ursprünglichen Bildes wiederfindet. Entsprechend den Histogrammen der Abb. 6.3 sind bis auf das Subband HH mit den tiefsten Frequenzen (links oben dargestellt) nur die Koeffizienten wesentlich von Null verschieden, die zu den Bereichen der Kanten des ursprünglichen Bildes gehören.

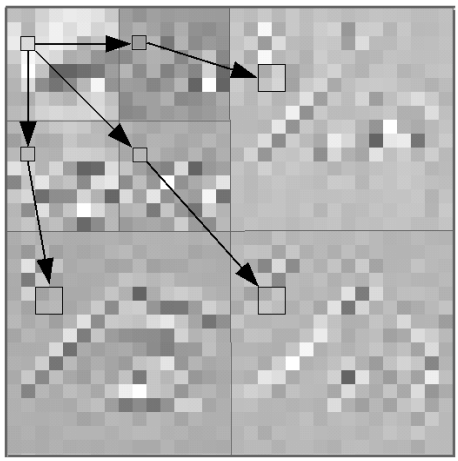


Abbildung 6.10: Zerotrees, baumartige Datenstruktur: Wavelet-Koeffizienten zu verschiedenen Skalierungsstufen (Subbändern), aber zur selben Bildregion, werden bei der Kodierung zusammen behandelt

Zur Kodierung wird daher die in Abb. 6.10 verdeutlichte Baumstruktur ausgenutzt. Jedem Koeffizienten des Subbandes des Subbandes HH mit den tiefsten Frequenzen werden drei, jedem Koeffizienten der mittleren Subbänder werden vier Tochterkoeffizienten“ zugeordnet (da das gerade folgende Subband die vierfache Anzahl an Koeffizienten enthält). Wenn die zugehörige Bildregion nicht gerade zu einer Kante gehört, so ist zu erwarten, dass die Werte aller Tochterkoeffizienten nahe Null liegen. Dieser Sachverhalt kann durch Übertragen oder Abspeichern eines einzigen Zeichens beschrieben werden. Er tritt vor allem deswegen so häufig auf, weil man zunächst einen sehr groben Maßstab dafür anlegt, was „nahe Null“ heißt und dies dann in späteren Durchgängen verfeinert.

Man stellt die Koeffizienten durch Betrag und Vorzeichen dar und ermittelt zuerst die maximale Bitzahl n der Beträge der Wavelet-Koeffizienten; d.h. die Beträge aller Koeffizienten liegen alle unterhalb 2^n und mindestens einer liegt oberhalb von $2^{n-1} - 1$.

Im ersten Durchgang werden alle Koeffizienten, bei deren Betrag im $n.$ (also im höchsten) Bit eine Eins steht, in eine spezielle Liste der signifikanten Koeffizienten übertragen und an ihrem ursprünglichen Platz durch eine Null ersetzt. Dies ist in der Regel nur bei wenigen Koeffizienten der Fall. Wenn dagegen beim Koeffizienten selbst und bei allen Töchtern im $n.$ Bit eine Null steht (dies tritt sehr häufig auf), dann wird dem arithmetischen Kodierer das Zeichen für „Zerotree“ zugeführt. Alle Töchter werden dann in diesem Durchgang übersprungen. In ganz seltenen Fällen steht beim Betrag des Koeffizienten zwar im $n.$ Bit eine Null, bei irgendeiner Tochter aber eine Eins. Dann wird an den arithmetischen Kodierer das Zeichen für eine „isolierte Null“ übergeben.

Für den nächsten Durchgang wird n um eins verringert, d.h. es wird nach dem nächstniederen Bit gefragt, aber nur bei den Koeffizienten, die bei den bisherigen Durchgängen noch nicht in die Liste der signifikanten Koeffizienten übertragen wurden, d.h. die an ihrem ursprünglichen Platz noch nicht durch Null ersetzt sind. Danach wird in einem Verfeinerungsgang jeweils das nächstniedere Bit der Koeffizienten dem arithmetischen Kodierer zugeführt, die in den vorangegangenen Durchgängen in die Liste der signifikanten Koeffizienten übertragen wurden.

Auf diese Weise wächst die Liste der signifikanten Koeffizienten, und am ursprünglichen Platz stehen zunehmend Nullen, was die überwiegende Häufigkeit des Zeichens „Zerotree“ auch bei den weiteren Durchgängen sicherstellt. Dies kann man fortsetzen, bis alle von Null verschiedenen Koeffizienten in die Liste der signifikanten Koeffizienten übertragen ist. Ist die Übertragungsrate oder der Speicherplatz erschöpft, so kann die weitere Kodierung abgebrochen werden. Die Methode garantiert, daß die für eine grobe Qualität wichtigen Informationen zuerst abgespeichert oder übertragen werden. Was danach kommt, dient der sukzessiven Verfeinerung des Bildes. Dieser Vorteil wird mit dem Stichwort „embedded code“ bezeichnet.

Die von Said und Pearlman [27] vorgeschlagene Verbesserung beruht auf demselben Prinzip. Der zugehörige Algorithmus ist in der Originalarbeit [27] in Abschnitt VI, „Coding Algorithm“ ausführlich beschrieben. Er führt zu einer weiteren Verbesserung der Bildqualität bei gleicher Kompressionsrate gegenüber dem hier geschilderten Verfahren.

6.5.2 Stack-Run Coding

Es ist verblüffend, dass eine so einfache Methode wie das in [35] beschriebene „Stack-Run Coding“ vergleichbare Ergebnisse wie die Zerotrees liefert. Sie benutzt eine geschickte Kombination von Lauflängen- und arithmetischer Kodierung. Da sehr viele Koeffizienten nach der Quantisierung Null sind, werden hier die Koeffizienten in der Form

$$(n_i, b_i, \sigma_i)$$

kodiert, wobei n_i die *Lauflänge* bis zum nächsten von Null verschiedenen Koeffizienten, b_i der um Eins erhöhte Betrag dieses Koeffizienten und σ_i sein Vorzeichen ist.

Hierzu müssen die Koeffizienten, die ursprünglich in Matrixform und in Subbändern angeordnet vorliegen, linear durchlaufen oder „gescannt“ werden. In der ursprünglichen Veröffentlichung [35] wird von einem zeilenweisen Scannen ausgegangen, Verbesserungen sind in [40] beschrieben. Um eine möglichst hohe Lauflänge von Nullen zu erreichen, ist es sinnvoll, die Subbänder HG vertikal, d.h. spaltenweise zu scannen. Denn sie enthalten fast ausschließlich an den vertikalen Kanten des Bildes von Null verschiedene Koeffizienten. Analog hat man die Subbänder GH horizontal, d.h. zeilenweise, zu scannen. Dies wird aus Abb. 6.11 deutlich. Die Kanten in Diagonalrichtung geben Ablaß zu von Null

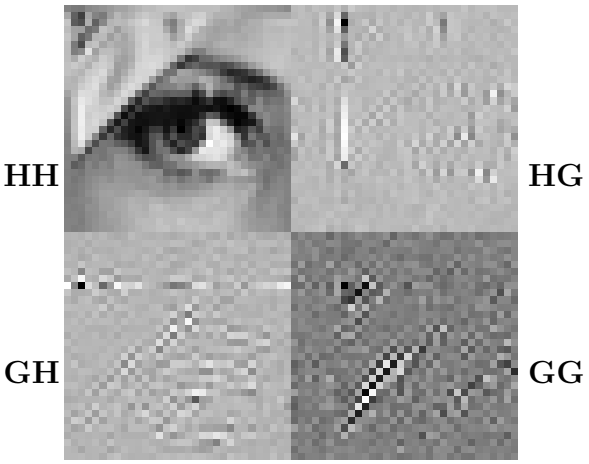


Abbildung 6.11: Vertikales Scannen in Subbändern HG, horizontales in GH sowie diagonales in GG (entsprechend Abb. 6.12) erzeugt eine hohe Lauflänge von Nullen

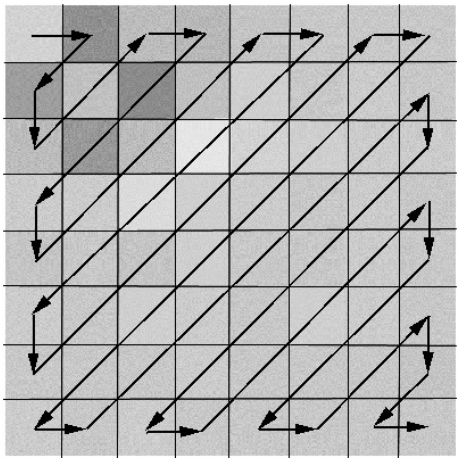


Abbildung 6.12: Scannen von Subbändern GG. In diesen Subbändern steckt die Information für Kanten in Diagonalrichtung.

verschiedenen Koeffizienten in Subbändern GG. Diese sind daher entsprechend Abb. 6.12 zu scannen. Alternativ zu dieser Möglichkeit, die Subbänder einzeln zu scannen, erscheint auch eine Anordnung der Koeffizienten, wie sie in Abb. 11.16 von Strang und Nguyen in [31] beschrieben ist, sinnvoll, weil sie bei den meisten Bildern zu einer hohen Lauflänge von Nullen führen dürfte.

Bei der „Schachbrett“-Abtastung (siehe Abschnitt 3.6.1) empfiehlt es sich, die verbliebenen „weißen“ Werte $s(i,k)$ der höchsten Skalierungsstufe getrennt zu scannen; ihre Position im Bild ist durch (3.92) bzw. (3.89) gegeben. Da die Werte alle an ihrem ursprünglichen Platz ohne Umspeichern berechnet werden, kann man alle „schwarzen“ Werte unabhängig von den Skalierungsstufen einfach zeilenweise scannen. Es könnte aber auch vorteilhaft sein, nach Skalierungsstufen getrennt zu scannen; vermutlich hängt dies vom Bild ab (ausprobieren!).

Der entscheidende „Trick“ des „Stack-Run Coding“ besteht nun darin, für die Bits der Lauflänge und der Beträge der Koeffizienten unterschiedliche Symbole zu verwenden.

Dies macht die Verwendung eines speziellen Trennzeichens überflüssig, und in den meisten Fällen kann zudem auf die Übertragung des höchstwertigen Bits (most significant bit, MSB) verzichtet werden. Für die Lauflänge wird also das Bit „0“ durch das Symbol „−“ und das Bit „1“ durch „+“ ersetzt werden. Zudem stehen die Symbole „+“ und „−“ für die Vorzeichen σ_i der Koeffizienten. Für beide „Sorten“ von Zahlen wird das am wenigsten bedeutende Bit (least significant bit, LSB) zuerst übertragen (also umgekehrt zu der üblichen Schreibreihenfolge bei Dezimalzahlen). Da durch die unterschiedlichen Symbole die Länge der einzelnen Zahlen klar ist, brauchen bei der normalen Zahlenreihenfolge führende“ und hier am Schluß kommende Nullen nicht aufgeführt werden. Dadurch wäre das MSB stets „1“ bzw. „+“ und daher in den meisten Fällen eine überflüssige Information.

Bei den Beträgen der Koeffizienten könnte nur eine einzelne „1“ nicht weggelassen werden, da sie aufgrund des fehlenden Trennzeichens in den benachbarten Zeichenketten von „+“ und „−“ verschwinden würde. Dieser Fall wurde durch den Übergang zu um Eins erhöhten Beträgen (der Koeffizient c_i führt auf $b_i = |c_i| + 1$) gerade *ausgeschlossen*. Für die Lauflängen wird vereinbart, dass das MSB „+“ immer dann wegzulassen ist, wenn die Lauflänge nicht ausschließlich durch Eins-Bits „+“ zu kodieren ist. D.h. immer dann, wenn die Lauflänge

$$n_i \neq 2^m - 1 \qquad \text{für alle} \quad m \in \mathbb{N}$$

erfüllt, wird das MSB „+“ weggelassen. Und diese Bedingung ist statistisch in den allermeisten Fällen erfüllt. Eine Lauflänge von Null kann einfach durch Weglassen des entsprechenden „−“ kodiert werden; dies ist für den Dekodierer an einem Fehlen eines Zeichens für die Lauflänge sichtbar. Die entstehende Zeichenkette von „+“, „−“, „0“, „1“ wird schließlich mit einem adaptiven arithmetischen Kodierer kodiert (wie er beispielsweise in [41] beschrieben ist).

Fassen wir die einzelnen Schritte zusammen:

(a) „Scannen“ der quantisierten Wavelet-Koeffizienten c_i.

(b) Darstellung in der Form

$$(n_1, b_1, \sigma_1), (n_2, b_2, \sigma_2), (n_3, b_3, \sigma_3), \ldots$$

wobei n_i die Zahl der Nullen bis zum nächsten Koeffizienten $c_i \neq 0$, $b_i = |c_i| + 1$ und σ_i das *Vorzeichen* von c_i ist.

(c) Binäre Kodierung, LSB stets zuerst:
n_i mit + statt 1 und − statt 0, abschließendes MSB + wegzulassen, außer wenn $n_i = 2^m - 1$ mit $m \in \mathbb{N}$; bei Lauflänge 0 Zeichen weglassen;
abschließendes MSB 1 bei b_i stets wegzulassen.

(d) adaptive arithmetische Kodierung der Zeichenkette aus Zeichen in $\{+, -, 0, 1\}$

Dies soll hier für das in [35] angegebene Beispiel durchgeführt werden. Ursprüngliches Subband:

$$\begin{array}{rrrr} 0 & 0 & 0 & 35 \\ 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -11 \end{array}$$

(a) nach dem Scannen:

$$0, 0, 0, 35, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -11$$

(b) Darstellung als

$$(3, 36, +), (0, 5, +), (10, 12, -)$$

(c) Kodierung: $(3, 36)$ ergibt $++00100+$ (das MSB „+“ von 3 wird nicht weggelassen, da $3 = 2^2 - 1$, das MSB 1 von 36 ist weggelassen.
$(0, 5, +)$ ergibt 10+ (kein Lauflängenzeichen, MSB 1 weggelassen)
$(10, 12, -)$ ergibt $-+-001-$ (MSB „+“ und 1 weggelassen)

(d) Die Zeichenkette

$$++00100+10+-+-001-$$

wird an den arithmetischen Kodierer übergeben.

6.6 Qualitätsvergleich von Kompressionsverfahren

Da die Quantisierung Verluste mit sich bringt, führt die Rekonstruktion eines komprimierten Bildes zu einem Intensitätswert $\hat{q}_{i,j}$, der sich vom ursprünglichen Wert $q_{i,j}$ in der i. Bildzeile und j. Spalte unterscheidet. Bezeichnet N die Gesamtzahl der Bildpixel, dann ist der *mittlere quadratische Fehler* (mean squared error)

$$\mathrm{MSE} = \frac{1}{N} \sum_{i,j} (q_{i,j} - \hat{q}_{i,j})^2 \tag{6.1}$$

wobei sich die Summe über alle Bildpixel erstreckt. Der Fehler in der Bildqualität wird jedoch meist anhand der Größe „peak signal to noise ratio“

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}} \right) \mathrm{dB} \tag{6.2}$$

beurteilt. Die Zahl 255 macht hierbei nur Sinn, wenn das ursprüngliche Bild eine Auflösung von 8 bpp (Bits pro Pixel), also einen maximalen Signalwert („peak“) von 255 hat.

Charakteristische Werte als Beispiel: Für das Testbild Lena (Abb. 1.6) mit 512×512 Pixeln und ursprünglich 8 bpp wird bei einer Kompression auf 0,5 bpp (also einer Komressionsrate von 16 : 1 mit dem 9/7-FBI-Filter und Stack-Run Coding in [35]

$$\mathrm{PSNR} = 36,78 \text{ dB}$$

angegeben. Für das verbesserte Zerotree-Verfahren erhielten Said und Pearlman [27] mit demselben Filter

$$\mathrm{PSNR} = 37,21 \text{ dB}$$

also eine nur geringfügig bessere Bildqualität.

Es bleibt anzumerken, dass der PSNR-Wert kein unumstrittenes Kriterium zur Beurteilung der Qualität des rekonstruierten Bildes darstellt. Er stimmt häufig nicht mit der subjektiv wahrgenommenen Bildqualität überein, da Bildfehler, die zum selben PSNR führen, vom Auge häufig sehr verschieden bewertet werden.
Literatur: Es wird der augezeichnete Übersichtsartikel [12] empfohlen.

Kapitel 7

Weitere Anwendungen in der Bild- und Signalverarbeitung

7.1 Vermindern von Rauschen

7.1.1 Vermindern von Rauschen mit Schwellwerten

Die Methode, mit Hilfe von Wavelets das Rauschen von Signalen zu vermindern (englisch „denoising“) geht auf Donoho [15] zurück. Manchmal wird auch vom „Entrauschen“ gesprochen. Es sollte jedoch klar sein, dass ein vollständiges Entrauschen eines verrauschten Signals unmöglich ist, wenn man das ursprüngliche, unverrauschte Signal nicht kennt. Ausgangspunkt ist ein *unbekanntes* ursprüngliches Signal $x(k)$, von dem nur eine verrauschte Version

$$y(k) = x(k) + r(k) \tag{7.1}$$

zur Verfügung steht. Es wird in der Theorie davon ausgegangen, dass die Rauschwerte $r(k)$ normalverteilt mit Erwartungswert 0 und gleicher Varianz σ^2 sowie stochastisch unabhängig voneinander sind (σ heißt Standardabweichung). D.h. bei vielen Messungen oder Übertragungen ist die relative Häufigkeit, dass $r(k)$ gerade zwischen u und $u + \Delta u$ liegt, annähernd gegeben durch

$$p(u) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{u^2}{2\sigma^2}} \cdot \Delta u \tag{7.2}$$

Wir gehen also davon aus, dass diese relative Häufigkeit unabhängig von k ist (im Fall eines Bildes das Rauschen gleichförmig über das Bild verteilt ist).

Zur Verminderung des Rauschanteils berechnet man die Wavelet-Transformatierte des verrauschten Signals $y(k)$ für eine bestimmte Anzahl M von Skalierungsstufen. Dann werden die Waveletkoeffizienten verändert. Betrachten wir zunächst ein Beispiel. Abb. 7.1 zeigt ein schon früher verwandtes Testsignal, zu dem ein errechnetes Rauschen addiert wurde. Dabei wurde ein Gaußsches Rauschen mit $\sigma = 0.2$ benutzt.

Für beide Signale (das ursprüngliche und das verrauschte) wurde die Wavelet-Transformation mit drei Skalierungsstufen durchgeführt. Hierfür wurden die Filter DD(4,2) (Strangs „9/7-Binlet“, siehe Abschnitt 3.3) benutzt. Das Ergebnis ist in Abb. 7.2 graphisch dargestellt. Das ursprüngliche Testsignal enthält 1000 Werte. Damit hat man ebenfalls 1000 Wavelet-Koeffizienten. In der Abbildung sind die ersten 125 Werte die Koeffizienten $s_3(k)$, gefolgt von den 125 Werten $d_3(k)$ (von Index 126 bis 250). An Indexposition 251

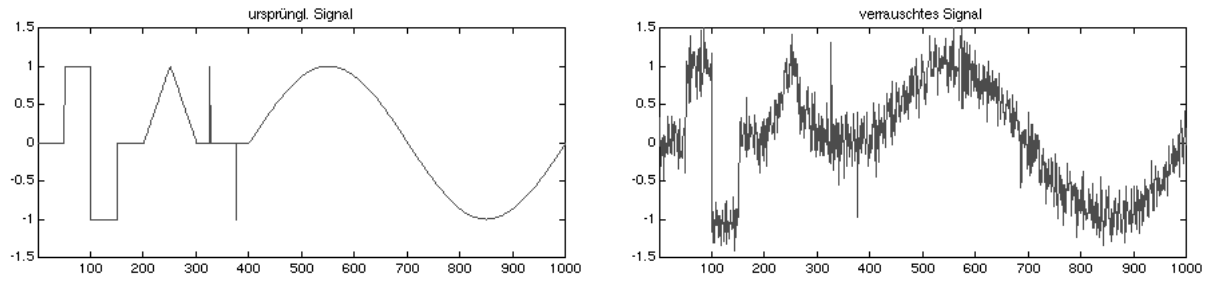


Abbildung 7.1: ursprüngliches Signal (links) sowie verrauschtes Signal (rechts, errechnetes Gaußsches Rauschen mit $\sigma = 0,2$)

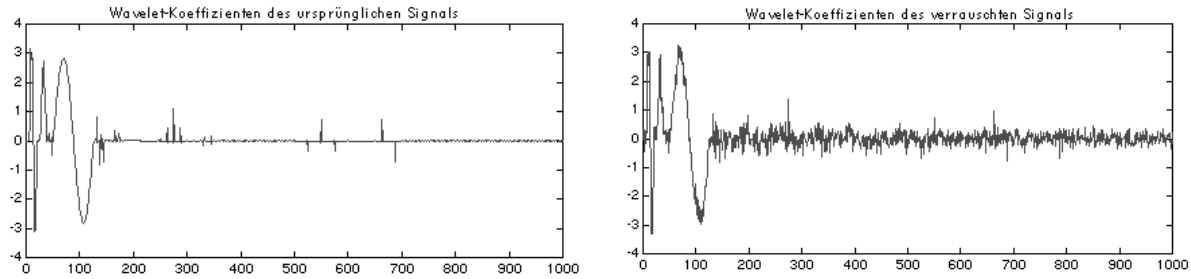


Abbildung 7.2: Wavelet-Koeffizienten des ursprünglichen Signal (links) sowie des verrauschten Signals (rechts). Es wurden die Filter DD(4,2) mit drei Skalierungsstufen gewählt.

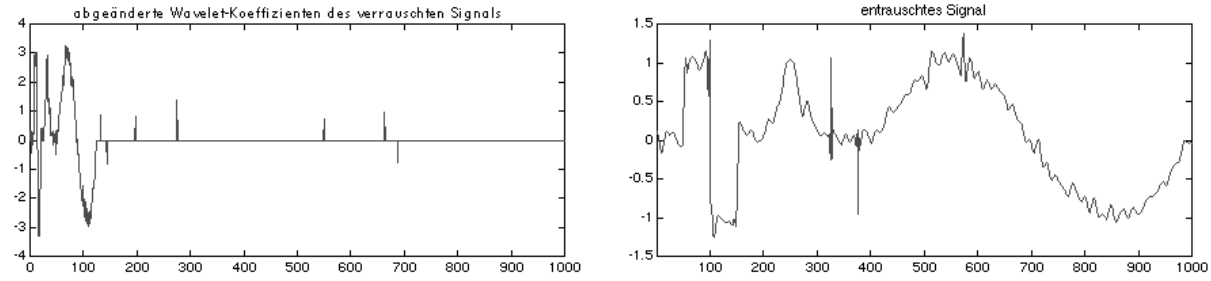
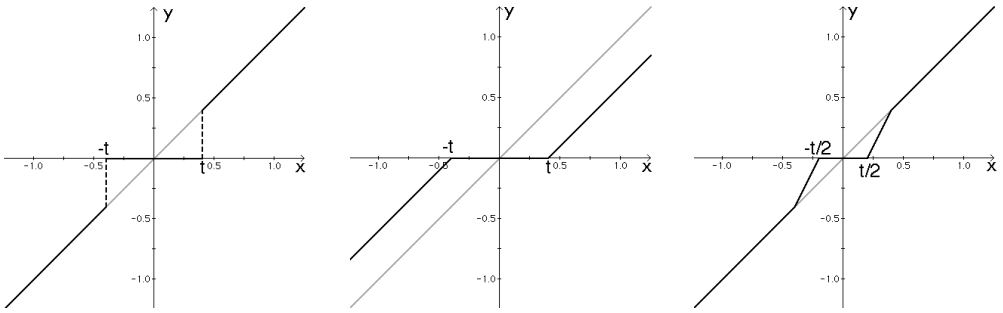


Abbildung 7.3: abgeänderte Wavelet-Koeffizienten des verrauschten Signals (links) sowie das durch Rücktransformation daraus gewonnene entrauschte Signal (rechts)

bis 500 stehen die Werte $d_2(k)$. An Indexposition 501 bis 1000 befinden sich die Werte $d_1(k)$.

Die meisten Wavelet-Koeffizienten des ursprünglichen Signals haben einen Betrag, der sehr nahe an Null liegt, wenn man von den Koeffizienten $s_3(k)$ der Skalierungsfunktion absieht. Nur dort, wo schnelle Änderungen im Signal auftreten, sind die Werte betragsmäßig groß. Ein Vergleich mit den Koeffizienten des verrauschten Signals legt eine einfache Abänderung nahe, die es erlaubt, eine Näherung an das (in der Praxis unbekannte) ursprüngliche Signal zu bekommen: Man setzt alle Koeffzienten Null, die betragsmäßig unterhalb einem Schwellwert liegen. Als Schwellwert wurde hier $t = 0,8$ gewählt. (Dieser Wert wurde rein experimentell bestimmt.) Für die so geänderten Koeffizienten (in Abb. 7.3 links graphisch dargestellt) führt man die inverse Wavelet-Transformation (Rücktransfor-

mation) durch. Man erhält ein Signal mit deutlich vermindertem Rauschanteil, das in Abb. 7.3 gezeigt ist.



harter Schwellwert sanfter Schwellwert Schwellwert mit Rampe

Abbildung 7.4: Verschiedene Möglichkeiten der Veränderung der Wavelet-Koeffizienten zum Entrauschen mit Hilfe eines Schwellwerts t (hier $t = 0.4$). Die ursprünglichen Werte der Koeffizienten, die *verändert* werden, sind grau markiert.

Für die Abänderung der Wavelet-Koeffizienten des verrauschten Signals stehen mehrere Varianten zur Verfügung (siehe hierzu auch Abb. 7.4):

(a) Harter Schwellwert (hard thresholding): Alle Waveletkoeffizienten, deren Betrag unter einer festen Schwelle t liegt, werden auf Null gesetzt, die übrigen werden unverändert gelassen. D.h. man führt die Ersetzung

$$d_m(k) \mapsto \begin{cases} 0 & \text{falls } |d_m(k)| < t \\ d_m(k) & \text{falls } |d_m(k)| \geq t \end{cases} \tag{7.3}$$

für $m = 1, \ldots M$ und alle k durch. Diese Variante wurde beim soeben besprochenen Beispiel (Abb. 7.3) benutzt.

(b) Sanfter oder weicher Schwellwert (soft thresholding): Hierbei werden auch die Koeffizienten mit großen Beträgen um den Schwellwert t verkleinert, also

$$d_m(k) \mapsto \begin{cases} 0 & \text{falls } |d_m(k)| < t \\ d_m(k) - t & \text{falls } d_m(k) \geq t \\ d_m(k) + t & \text{falls } d_m(k) \leq -t \end{cases} \tag{7.4}$$

Diese Variante vermeidet Sprünge von 0 auf den Schwellwert t oder $-t$ und verändert *alle* Koeffizienten.

(c) Schwellwert mit Rampe: Diese Variante vermeidet den Sprung auf den Schwellwert und läßt die Koeffitienten, deren Betrag größer als der Schwellwert ist, *unverändert*.

$$d_m(k) \mapsto \begin{cases} 0 & \text{falls } |d_m(k)| < \frac{t}{2} \\ d_m(k) & \text{falls } |d_m(k)| \geq t \\ 2d_m(k) + t & \text{falls } -t < d_m(k) \leq -\frac{t}{2} \\ 2d_m(k) - t & \text{falls } \frac{t}{2} < d_m(k) \leq t \end{cases} \tag{7.5}$$

Statt einer Rampe sind zahlreiche andere Funktionen möglich, die den Sprung des harten Schwellwerts vermeiden, und die Koeffizienten, deren Betrag größer als der Schwellwert ist, nicht oder annähernd nicht verändern.

Das Entrauschen durch die beschriebene Veränderung der Waveletkoeffizienten beruht darauf, dass die Waveletransformation eines reinen Rauschens mit Gauß-verteilten Werten wieder ein Rauschen mit derselben statistischen Verteilung ergibt. Dies gilt exakt nur bei der Verwendung von orthogonalen Wavelets und unendlich ausgedehnten Daten, wird aber von Randbedingungen nur geringfügig beeinträchtigt. Auch bei der Verwendung von biorthogonalen Wavelets ist dies näherungsweise erfüllt, da die Abweichung von einer orthogonalen Transformation bei den häufig verwandten Wavelets nicht sehr groß ist. Das ursprüngliche Signal wird jedoch in den meisten Fällen durch die Wavelettransformation dekorreliert, d.h. auf relativ wenige betragsmäßig große Koeffizienten konzentriert, der Rest ist betragsmäßig sehr klein oder Null (siehe hierzu beispielsweise die Häufigkeitsverteilung von Abb. 6.2 im Vergleich zu denen in Abb. 6.3). Beim verrauschten Signal enthalten die meisten Koeffizienten also nur noch das Rauschen. Diese werden durch die Anwendung des Schwellwerts auf Null gesetzt. Man kann also hoffen, dass dadurch ein großer Teil des Rauschens beseitigt wird. Ein vollständiges Entrauschen ist ohnehin ohne Kenntnis des ursprünglichen Signals nicht möglich.

Wir bezeichnen die „entrauschten“ Signalwerte, die wir durch Rücktransformation der mit einem Schwellwert veränderten Waveletkoeffizienten $d_m(k)$ (sowie der meist unveränderten Koeffizienten $s_M(k)$) gewinnen, mit $\hat{y}(k)$. Wir hoffen, dass der mittlere quadratische Fehler

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}(\hat{y}(k) - x(k))^2 \tag{7.6}$$

kleiner geworden ist gegenüber dem, den man erhält, wenn man die verrauschten Signalwerte $y(k)$ in diese Formel einsetzt. Doch nur, wenn man die unverrauschten Signalwerte $x(k)$ kennt (also bei künstlich verrauschten Testsignalen), ist diese Berechnung möglich. Aus dem mittleren quadratischen Fehler kann man dann auch den PSNR-Wert gemäß (6.2) berechnen und so verschiedene Verfahren zum Entrauschen testen.

Aus theoretischen Überlegungen erhält man für den Schwellwert

$$t = \sigma\sqrt{2\ln N} \tag{7.7}$$

Dabei ist N die Gesamtzahl der Signalwerte (z.B. der Bildpixel). Dieser Wert ist jedoch nur für *orthogonale* Wavelets richtig und muss strenggenommen bereits deswegen korrigiert werden, weil durch die Behandlung am Rand die Wavelettransformation auch im Fall orthogonaler Wavelets keine orthogonale lineare Transformation darstellt. Der Schwellwert muss in diesen Fällen im allgemeinen vergrößert werden.

Bei der Verwendung von nichtorthogonalen Wavelets ist es auch sinnvoll, verschieden große Schwellwerte für verschiedene Skalierungsstufen m zu benutzen. In [3] ist ein korrigierter Schwellwert in der Form

$$t = \sigma \cdot \sigma_m\sqrt{2(1+\delta_m)\ln N} \tag{7.8}$$

angegeben. Für das FBI-Filter weichen die Zahlen σ_m nur unwesentlich von 1 ab, $\delta_1 \approx 0,06$ und für $m > 1$ gilt $0,12 \le \delta_m \le 0,14$.

In der Praxis ist die Bestimmung der Varianz σ^2 für das Rauschen nicht einfach (ganz abgesehen davon, dass die Häufigkeitsverteilung eine ganz andere als die Gaußverteilung sein kann). Hat man ein reines Rauschsignal zur Verfügung (z.B. ein ursprünglich konstantes und dann verrauschtes Testbild), so erhält man durch die empirische Varianz

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\big(r(k) - \bar{r}\big)^2 \tag{7.9}$$

eine Schätzung von σ^2. Dabei ist

$$\bar{r} = \frac{1}{N} \sum_{i=1}^{N} r(k)$$

der empirische Mittelwert, der laut unserer Annahme nahe Null sein sollte.

In Abschnitt 7.1.2 wird eine Methode angegeben, wie σ ohne Kenntnis der Rauschwerte allein mit Hilfe der Waveletkoeffizienten $d_m(k)$ geschätzt werden kann.

Die Wavelet-Koeffizienten ändern sich in komplizierter Weise, wenn man das Signal verschiebt. Daher ist es nicht verwunderlich, dass man für das Beseitigen von Rauschen bessere Ergebnisse erzielt, wenn man die Wavelet-Transformation ohne Unterabtastung verwendet (siehe Abschnitt 5). Auch hierbei muss der Schwellwert t gegenüber (7.7) vergrößert werden und kann für die unterschiedlichen Skalierungsstufen unterschiedlich sein. Der Nachteil ist dann allerdings der höhere Speicherplatzbedarf und die größere Rechenzeit. Die theoretischen Untersuchungen der Verwendung der Wavelet-Transformation ohne Unterabtastung für das Entrauschen sind – im Gegensatz zur Verwendung von orthogonalen Wavelets — allerdings noch nicht weit fortgeschritten. Auch die Verwendung von Waveletpaketen (d.h. der entsprechenden Transformation, siehe Abschnitt 4.1) kann die Ergebnisse gegenüber der Verwendung der einfachen Wavelettransformation verbessern. Denn es ist möglich, die Koeffizienten übrigzubehalten, bei denen das ursprüngliche Signal auf möglichst wenig betragsmäßig große Koeffizienten konzentriert ist.

Die bisher angegebenen Empfehlungen für den Schwellwert enthalten alle einen Faktor $\sqrt{\ln N}$, wobei N die Gesamtzahl der Datenpunkte ist. Dies liegt daran, dass aufgrund des nur exponentiellen Abfalls der durch (7.2) gegebenen Häufigkeitsverteilung bei sehr vielen Daten durchaus einzelne Werte sehr hohe Abweichungen aufgrund des Rauschens enthalten können. Aufgrund der mathematisch üblichen Kriterien zur Optimierung des Schwellwertes (Minimierung der quadratischen Abweichung) werden solche „Ausreißer“ sehr hoch bewertet. Dies ist bei hohen Werten für N, insbesondere bei Bildern nicht in Übereinstimmung mit dem, was in der Praxis erwünscht ist. Hohe Fehler bei einzelnen Werten (Pixeln) werden akzeptiert, wenn der Gesamteindruck sonst einen guten Eindruck macht. Durch einen Schwellwert proportional zu $\sqrt{\ln N}$ werden dagegen sehr viele Werte beeinträchtigt zugunsten eines Versuchs, auch die „Ausreißer“ zu korrigieren.

Man sollte also in der Praxis bei großen Datensätzen, insbesondere bei Bildern, mit erheblich kleineren Schwellwerten arbeiten. Mallat berichtet in seinem Buch [26, chapter 10.2] über erfolgreiches Entrauschen mit Schwellwerten von $\frac{3}{2}\sigma$ und 3σ bei Bildern mit 512^2 Pixeln. In [38] ist generell die Verwendung des Schwellwerts $t = 4.5\sigma$ empfohlen. Abb. 7.6 zeigt ein Beispiel für das Entrauschen von Bildern mit „sanftem Schwellwert“. Als Wavelet wurde Strangs 9/7 Binlet (DD(4,2)) gewählt. Ein „harter Schwellwert“ liefert bei derselben Wahl der übrigen Parameter ein optisch schlechteres Ergebnis.

In Abschnitt 7.1.3 wird eine Methode vorgestellt, mit deren Hilfe ein optimaler Schwellwert bestimmt werden kann, der den mittleren quadratischen Fehler minimal macht.

Der Vorteil des Beseitigens von Rauschen mit Hilfe der Wavelet-Transformation gegenüber herkömmlichen Methoden mit Hilfe der Fourier-Transformation liegt darin, dass Spitzen und steile Rampen (d.h. die Kanten beim Bild) nicht so stark abgerundet bzw. abgeflacht werden. Man hat jedoch zu beachten, dass auch bei der Wavelet-Transformation das Gibbs-Phänomen auftritt: Durch die Schwellwertbildung wird ein „Überschwingen“ an Sprungstellen des ursprünglichen Signals verursacht. Dies ist in Abb. 7.5 verdeutlicht. Dort ist eine harte Schwellwertbildung auf die Wavelet-Koeffizienten eines unverrauschten

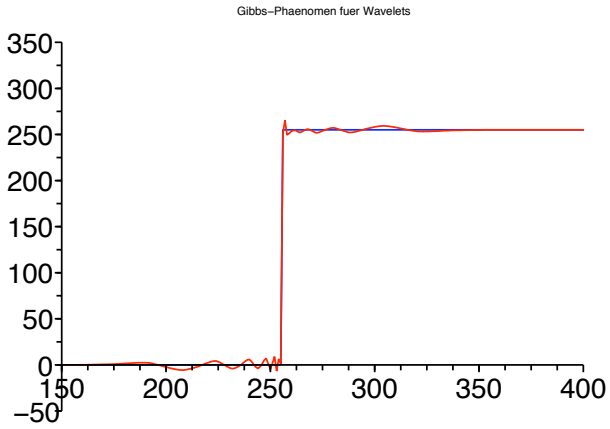


Abbildung 7.5: Gibbs-Pänomen bei der Wavelet-Transformation: harte Schwellwertbildung (Schwellwert 25) bei einer Binlet-Transformation (DD(4,2)) eines Sprungs mit 6 Skalierungsstufen

Sprungs von 0 auf 255 angewandt. Beim Entrauschen kann dies dazu führen, dass die rücktransformierten Signalwerte außerhalb des zulässigen Bereichs liegen (häufig von 0 bis 255). Dann müssen diese Werte erst wieder an den zulässigen Bereich angepasst werden. Dies geschieht zweckmäßigerweise durch Abschneiden, hier als Beispiel für den häufig vorkommenden Bereich von 0 bis 255:

$$y(k) \mapsto \begin{cases} 0 & \text{falls} \quad y(k) < 0 \\ y(k) & \text{falls} \quad 0 \leq y(k) \leq 255 \\ 255 & \text{falls} \quad y(k) > 255 \end{cases} \tag{7.10}$$

Literatur: Die Verwendung von Wavelets zum Entrauschen ist in zahlreichen Originalveröffentlichungen beschrieben, siehe hierzu beispielsweise [8, 15, 22]. Eine ausführliche theoretische Grundlegung erfolgt in [26, chapter 10]. Eine sehr verständliche Einführung steht in [38].

7.1.2 Schätzung der Standardabweichung mit Hilfe des Medians

Hat man kein reines Rauschsignal zur Verfügung, dann kann man mit Hilfe des Medians der Beträge der Waveletkoeffizienten eine Schätzung aus dem verrauschten Signal gewinnen. Der Median einer Verteilung ist der Wert m der zugehörigen Zufallsvariablen X, für die die Wahrscheinlichkeit dafür, dass $X \leq m$ gerade gleich $\frac{1}{2}$ ist. Dann ist die Wahrscheinlichkeit dafür, dass $X > m$ ist, ebenfalls $\frac{1}{2}$. Der Median zeigt also die „Mitte“ der Verteilung an. Er ist im allgemeinen vom Erwartungswert verschieden. Man kann den Median aufgrund von vorliegenden Messwerten $\{x_1, x_2, x_3, \ldots x_N\}$ der Zufallsvariablen (die Statistiker sprechen von einer „Stichprobe“) mit Hilfe des empirischen Medians schätzen. Hierzu sortiert man die Messwerte ihrer Größe nach aufsteigend. Dann ist der empirische Median bei einer ungeraden Anzahl N von Werten derjenige, der in der Mitte steht, also den Index $\frac{1}{2}(N+1)$ hat. Bei einer geraden Zahl N von Messwerten ist der empirische Median der Mittelwert der Werte mit dem Indizes $\frac{N}{2}$ und $\frac{N}{2}+1$ nach dem Sortieren. Der

ursprüngliches Bild (Testbild „boats“)

verrauschtes Bild

entrauschtes Bild

Abbildung 7.6: Entrauschen eines Bildes. Der Rauschanteil wurde als Gaußsches Rauschen mit $\sigma = 20$ errechnet und zum ursprünglichen Bild addiert. Für das Entrauschen wurde ein sanfter Schwellwert mit $t = 3\sigma$ gewählt.

allgemeine Vorteil des Medians gegenüber dem Erwartungswert besteht darin, dass der empirische Median sehr viel weniger empfindlich ist gegenüber „Ausreißern“, also Werten, die sehr weit vom Median entfernt sind und auch unwahrscheinlich sind, aber eben vorhanden sein können. Der empirische Mittelwert

$$\frac{1}{N}\sum_{k=1}^{N} x_k$$

ist dagegen außerordentlich empfindlich gegenüber solchen „Ausreißern“.

Wir gehen nun davon aus, dass die Beträge der Waveletkoeffizienten unseres *unverrauschten* Signals überall sehr klein sind außer an den wenigen Stellen, an denen sich das Signal stark ändert. Wir erwarten, dass der empirische Median der Beträge der Waveletkoeffizienten des *unverrauschten* Signals sehr klein ist, da die Zahl der „Ausreißer“ erheblich unter der Hälfte der Signalwerte liegen dürfte. Bilden wir den Median der Be-

träge der Waveletkoeffizienten des *verrauschten* Signals (ohne die Tiefpasswerte $s(k)$ bzw. das HH-Subband), dann stammt der entsprechende Wert im wesentlichen vom Rauschen. Da dieses laut Annahme normalverteilt ist, kennen wir auch die Verteilung der Beträge der Rauschwerte. Nach den Regeln der Wahrscheinlichkeitsrechnung kann man den Wert des Medians $|X|_m$ des Betrags $|X|$ einer normalverteilten Zufallsvariable X mit Erwartungswert 0 ausrechnen:

$$|X|_m = m_S \cdot \sigma \qquad \text{wobei} \qquad m_S = 0.67448975019608$$

Für Spezialisten: m_S ist der Median des Betrags einer standardnormalverteilten Zufallsvariable ($\sigma = 1$), der sich als das $\frac{3}{4}$-Quantil der Standardnormalverteilung berechnen lässt.

Man erhält damit die Schätzung der Standardabweichung mit Hilfe des empirischen Medians med durch

$$\sigma_e = \frac{1}{m_S}\text{med}\big(\{|d_m(k)|\}\big) \tag{7.11}$$

Dabei steht der Index m für die Skalierungsstufen oder die Subbänder und der Index k für die Nummern der einzelnen Koeffizienten (bei Bildern steht k für zwei Indizes). In der Praxis ist es meistens sinnvoll, für verschiedene Subbänder mit unterschiedlichen Schätzwerten für die Standardabweichung zu arbeiten. Dies führt dann zu unterschiedlichen Schwellwerten in den einzelnen Subbändern; ein größerer Wert für σ macht auch einen größeren Schwellwert notwendig. Wir schätzen also σ mit Hilfe des empirischen Medians in jedem Subband m durch

$$\sigma_m = \frac{1}{m_S}\text{med}\big(\{|d_m(1)|, |d_m(2)|, |d_m(3)|, \ldots |d_m(N)|\}\big) \tag{7.12}$$

7.1.3 Optimierung des Schwellwerts mit Hilfe von „SURE“

Wir gingen zu Beginn des Kapitels davon aus, dass unsere verrauschten Signalwerte sich aus einem unbekannten unverrauschten Signal und einem normalverteilten Rauschsignal zusammensetzen (siehe (7.1)). Dies hat zur Folge, dass die Waveletkoeffizienten $d_m(k)$ sich ebenfalls zusammensetzen aus den Koeffizienten des unbekannten unverrauschen Signals $u_m(k)$ und den Koeffizienten $w_m(k)$ der Rauschwerte, also

$$d_m(k) = u_m(k) + w_m(k) \tag{7.13}$$

Bei einer orthogonalen Transformation, wenn wir also orthogonale Wavelets verwenden (mit Unterabtastung), dann sind die Koeffizienten $w_m(k)$ ebenfalls unabhängig und normalverteilt mit Erwartungswert 0 und Varianz σ^2. Bei den hier bevorzugt dargestellten biorthogonalen Wavelets hoffen wir, dass dies näherungsweise immer noch zutrifft. Auch für den Fall der redundanten Wavelettransformation nehmen wir an, dass die Hochpasskoeffizienten diese Form mit näherungsweise unabhängigen normalverteilten Koeffizienten $w_m(k)$ haben. Es muss jedoch damit gerechnet werden, dass die Varianz von Subband zu Subband verschieden ist. Daher empfiehlt es sich, zur Verbesserung des Entrauschens in den einzelnen Subbändern unterschiedliche Schwellwerte zu wählen, die empfohlene Berechnung des optimalen Schwellwerts also für die einzelnen Subbänder (die hier mit m nummeriert sind) getrennt vorzunehmen. Wir werden aber zur Vereinfachung der Darstellung den Index m für das Subband bzw. die Skalierungsstufe weglassen.

Der mittlere quadratischen Fehler (MSE) für die Waveletkoeffizienten ist gegeben durch

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}\big(d(k) - u(k)\big)^2$$

und wir wollen ihn durch Veränderungen der Koeffizienten $d(k)$ verringern. Wir erwarten, dass nach Rücktransformation dadurch auch der mittlere quadratische Fehler für die ursprünglichen Signalwerte (in (7.6)) verringert wird. Wir bezeichnen die Veränderungen mit Hilfe eines Schwellwerts nach den durch (7.3) bis (7.5) gegebenen Zuordnungsvorschriften mit der Funktion η. Das Entrauschen erfolgt also durch die Ersetzung

$$d(k) \mapsto \eta\big(d(k)\big)$$

und wir wollen die Funktion η so wählen, dass

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}\big(\eta(d(k)) - u(k)\big)^2 \tag{7.14}$$

minimal wird. Das Problem ist, dass wir die Koeffizienten $u(k)$ nicht kennen. Hier ist ein Theorem von Charles M. Stein nützlich (siehe beispielsweise [22]). Angewandt auf unseren Fall besagt dieses, dass die Zufallsvariable

$$Y = \frac{1}{N}\sum_{k=1}^{N}\Big(\big(\eta(d(k))\big)^2 - 2d(k)\eta(d(k)) + 2\sigma^2\eta'(d(k))\Big) + \frac{1}{N}\sum_{k=1}^{N}u(k)^2 \tag{7.15}$$

denselben Erwartungswert hat wie die durch die rechte Seite von (7.14) gegebene Zufallsvariable. Beachten Sie, dass aufgrund von (7.13) die Koeffizienten $d(k)$ Zufallsvariable sind. Wir werden also die rechte Seite von (7.15) für unsere konkret vorliegenden Waveletkoeffizienten $d(k)$ minimieren und — aufgrund des Theorems — davon ausgehen, dass wir damit auch die rechte Seite von (7.14) minimieren. Wir erwarten, dass dies bei einer hinreichend großen Gesamtzahl N von Signalwerten gut erfüllt ist. Dieses Prinzip wird SURE-Schätzung oder SURE-Minimierung genannt. Dabei steht die Abkürzung für „Stein's unbiased risk estimate“. Der große Vorteil dieses Prinzips ist, dass die unbekannten Koeffizienten $u(k)$ in (7.15) nur im Term $\frac{1}{N}\sum_{k=1}^{N}u(k)^2$ stehen, also in der Form eines konstanten unbekannten Summanden eingehen. Dieser ist völlig unabhängig von der Funktion η. Wir können daher, statt (7.15) zu minimieren, auch die Funktion ohne diesen Summanden, also die Zufallsvariable

$$\tilde{Y} = \frac{1}{N}\sum_{k=1}^{N}\Big(\big(\eta(d(k))\big)^2 - 2d(k)\eta(d(k)) + 2\sigma^2\eta'(d(k))\Big) \tag{7.16}$$

minimieren. Dies wird zu derselben optimalen Funktion η führen.

Wir wollen nun die SURE-Schätzung anwenden, um ein Verfahren anzugeben, nach dem der optimale „weiche“ Schwellwert t berechnet werden kann. Die Koeffizienten werden mit Hilfe der Funktion

$$\eta(x) = \begin{cases} 0 & \text{falls } |x| < t \\ x - t & \text{falls } x \geq t \\ x + t & \text{falls } x \leq -t \end{cases}$$

verändert. Wir benötigen in (7.16) die Ableitung dieser Funktion. Sie existiert allerdings in den Punkten $x = \pm t$ nicht. Dies stört aber nicht, da die Zufallsvariablen $d(k)$ einen normalverteilten Rauschanteil enthalten und die Wahrscheinlichkeit, dass ein Koeffizient gerade $d(k) = \pm t$ exakt erfüllt, ohnehin 0 ist. Wir legen die Ableitung in diesen beiden Punkten willkürlich fest und haben damit

$$\eta'(x) = \begin{cases} 0 & \text{falls } |x| \le t \\ 1 & \text{falls } |x| > t \end{cases}$$

Für die Berechnung der Summe für $\tilde{Y}$ in (7.16) ist es nützlich, zu beachten, dass

$$\big(\eta(x)\big)^2 = \begin{cases} 0 & \text{falls } |x| \le t \\ \big(|x| - t\big)^2 & \text{falls } |x| > t \end{cases}$$

und

$$x \cdot \eta(x) = \begin{cases} 0 & \text{falls } |x| \le t \\ |x| \cdot \big(|x| - t\big) & \text{falls } |x| > t \end{cases}$$

(kleine Übungsaufgabe, schauen Sie sich Abb. 7.4 an). Dies reduziert die drei Fälle in der Definition von η in unserer Berechnung auf zwei zu unterscheidende Fälle. Damit haben wir

$$\tilde{Y} = \frac{1}{N} \sum_{\substack{k=1 \\ |d(k)| > t}}^{N} \Big(\big(|d(k)| - t\big)^2 - 2|d(k)|\big(|d(k)| - t\big) + 2\sigma^2 \Big)$$

Für die ersten Summanden gilt

$$\begin{aligned} \big(|d(k)| - t\big)^2 - 2|d(k)|\big(|d(k)| - t\big) &= |d(k)|^2 - 2t|d(k)| + t^2 - 2|d(k)|^2 + 2t|d(k)| \\ &= t^2 - |d(k)|^2 \end{aligned}$$

und der letzte Summand $2\sigma^2$ hängt gar nicht vom Summationsindex ab (wenn dieser $|d(k)| > t$ erfüllt). Üblicherweise bezeichnet man die Zahl der Indizes k, für die $|d(k)| \ge t$ gilt, mit $\#\{k \,|\, |d(k)| > t\}$. Dies ist die Zahl der Summanden in der Summe. Wir erhalten

$$\begin{aligned} \tilde{Y} &= \frac{2}{N}\sigma^2 \cdot \#\{k \,|\, |d(k)| > t\} + \frac{1}{N} \sum_{\substack{k=1 \\ |d(k)| > t}}^{N} \big(t^2 - |d(k)|^2\big) \\ &= \frac{1}{N}(2\sigma^2 + t^2) \cdot \#\{k \,|\, |d(k)| > t\} - \frac{1}{N} \sum_{\substack{k=1 \\ |d(k)| > t}}^{N} |d(k)|^2 \end{aligned} \tag{7.17}$$

In allen Intervallen, die durch zwei benachbarte Werte von $|d(k)|$ gegeben sind, ist $\tilde{Y}$ eine monoton steigende Funktion von t. Das Minimum wird also angenommen, wenn $t = |d(k)|$ für ein $k = 1, 2, \ldots N$. Man kann also den optimalen Schwellwert bestimmen, indem man $\tilde{Y}$ für $t = |d(k)|$ für alle $k = 1, 2, \ldots N$ berechnet. Hierzu kann man gegebenenfalls vom Rechner die Werte von $|d(k)|$ in aufsteigender Reihenfolge sortieren lassen.

In der Literatur (beispielsweise in [22]) ist statt $\tilde{Y}$ folgende Funktion angegeben, die zu minimieren ist:

$$\begin{aligned}
S &= \sigma^2 - \frac{2}{N}\sigma^2 \cdot \#\{k \,|\, |d(k)| \le t\} + \frac{1}{N}\sum_{k=1}^{N}\big(\min(|d(k)|, t)\big)^2 \\
&= \sigma^2 - \frac{2}{N}\sigma^2 \cdot \#\{k \,|\, |d(k)| \le t\} + \frac{1}{N}\sum_{\substack{k=1 \\ |d(k)| \le t}}^{N} |d(k)|^2 + \frac{t^2}{N} \cdot \#\{k \,|\, |d(k)| > t\} \\
&= \sigma^2 - \frac{2}{N}\sigma^2\Big(N - \#\{k \,|\, |d(k)| > t\}\Big) + \frac{1}{N}\sum_{\substack{k=1 \\ |d(k)| \le t}}^{N} |d(k)|^2 + \frac{1}{N}\sum_{\substack{k=1 \\ |d(k)| > t}}^{N} |d(k)|^2 \\
&\qquad -\frac{1}{N}\sum_{\substack{k=1 \\ |d(k)| > t}}^{N} |d(k)|^2 + \frac{t^2}{N} \cdot \#\{k \,|\, |d(k)| > t\} \\
&= -\sigma^2 + \frac{2}{N}\sigma^2 \cdot \#\{k \,|\, |d(k)| > t\} + \frac{1}{N}\sum_{k=1}^{N} |d(k)|^2 \\
&\qquad -\frac{1}{N}\sum_{\substack{k=1 \\ |d(k)| > t}}^{N} |d(k)|^2 + \frac{t^2}{N} \cdot \#\{k \,|\, |d(k)| > t\}
\end{aligned} \tag{7.18}$$

Ein Vergleich von (7.17) und (7.18) liefert

$$S = \tilde{Y} - \sigma^2 + \frac{1}{N}\sum_{k=1}^{N} |d(k)|^2$$

Dies bedeutet, dass sich S und $\tilde{Y}$ um eine Konstante unterscheiden, die von t unabhängig ist. Es kommt also auf dasselbe hinaus, ob wir S oder $\tilde{Y}$ minimieren.

7.2 Verbesserung der Schärfe von Bildern

In der englischsprachigen Literatur werden derartige Anwendungen unter dem Stichwort „enhancement“ und „sharpening“ beschrieben. Hierfür gibt es recht verschiedene Methoden, die Wavelets benutzen, aber die offensichtlich alle noch nicht sehr ausgiebig getestet sind und die Spielraum für weitere Varianten (und Tests) eröffnen. Dieser Abschnitt ist daher in erster Linie als Einladung zu Experimenten mit Wavelets gedacht.

7.2.1 Vergrößerung der Waveletkoeffizienten

In der Beschreibung zum Softwarepaket LIFTPACK [16] wird vorgeschlagen, alle Waveletkoeffizienten (d.h. alle Koeffizienten der Subbänder HG, GH und GG) mit einem Faktor β^m mit $\beta > 1$ zu multiplizieren, wobei m die Stufe der Wavelettransformation ist.

Dieses Verfahren kann leicht variiert werden. Es bietet sich an, die betragsmäßig kleinen Koeffizienten unverändert zu lassen (oder zu verkleinern) und nur die Beträge der betragsmäßig großen weiter zu vergrößern und dies umso mehr, je höher die Stufe der Wavelettransformation ist. Denn wichtige Kanten verursachen betragsmäßig große Waveletkoeffizienten für alle Skalierungen, während Rauschen und Textur in erster Linie die Koeffizienten der feineren Skalierungen (also insbesondere der ersten Skalierungsstufe) beeinflussen. Denkbar wäre, alle Beträge $|c_i|$ durch $|c_i|^{\alpha_m}$ zu ersetzen, wobei der Exponent $\alpha_m > 0$ gerade so gewählt ist, dass nur die betragsmäßig größeren Koeffizienten verstärkt werden. Dies kann in den einzelnen Skalierungsstufen (mit m numeriert) unterschiedlich erfolgen.

7.2.2 Differenzbildung zum geglätteten Bild

Diese Methode geht auf Mladen Victor Wickerhauser zurück und ist (vermutlich) bisher nicht veröffentlicht. Sie ist in erster Linie dazu gedacht, aus einem Bild mit schlechter Auflösung (wenigen Pixeln) ein scharfes Bild mit größerer Auflösung (der doppelten Pixeldichte) zu erzeugen, kann aber auch so abgeändert werden, daß man ein gleichgroßes schärferes Bild (mit derselben Pixeldichte) erhält.

Die zugrundeliegende Idee soll hier zunächst für ein eindimensionales Signal $x(n)$ erklärt werden und danach für Bildsignale verallgemeinert werden. Zunächst wählt man ein symmetrisches Tiefpassfilter, dessen Impulsantwort

$$\sum_{k=-\infty}^{+\infty} h(k) = 1$$

erfüllt, beispielsweise

$$h(0) = \frac{2}{3}, \quad h(\pm 1) = \frac{1}{6}, \quad h(k) = 0 \quad \text{falls} \quad |k| > 1$$

oder das durch (5.18) gegebene. Das hiermit gefilterte Signal (ohne Unterabtastung) sieht man als Tiefpassausgang einer Wavelettransformation (nach der Unterabtastung) an, also

$$s(n) := \sum_{k=-\infty}^{+\infty} h(k)x(n-k) \tag{7.19}$$

Als Hochpassausgang (nach der Unterabtastung) sieht man die Differenz zum ursprünglichen Signal an

$$d(n) := x(n) - s(n) \tag{7.20}$$

Das geschärfte Signal erhält man durch inverse Wavelettransformation, wobei darauf zu achten ist, dass man ein *symmetrisches* Synthesewavelet benutzt. Hierfür kommen also beispielsweise das (9/7)-„Binlet“-Filter von Strang und Sweldens (siehe Tabelle 3.2) oder auch die Filter CDF(4,x) aus [36] in Frage. Man erhält damit ein Signal mit der doppelten Zahl von Daten (d.h beim Bild bei derselben physikalischen Ausdehnung eine doppelte Auflösung, gemessen in dpi). Dies ist häufig in der Praxis erwünscht.

Wenn man dagegen ein verschärftes Bild derselben Größe möchte (also hier beim eindimensionalen Beispiel dieselbe Anzahl von Daten), so kann man durch Unterabtastung auch wieder auf dieselbe Anzahl kommen. In Grafik-Programmen findet zu einer Verkleinerung des Bildes häufig vor der Unterabtastung eine Tiefpassfilterung statt. Diese kann jedoch den Effekt der Erhöhung der Schärfe wieder rückgängig machen.

Andererseits ist es auch möglich, durch die inverse Wavelettransformation ohne Unterabtastung (siehe Abschnitt 5) direkt dieselbe Zahl von Daten zurückzuerhalten. Hinsichtlich der Filterauswahl kommen zunächst die durch (5.18), (5.21), (5.20) und (5.24) gegebenen Filter in Betracht. Da man hier nur die Rekonstruktion benötigt, könnte es sich als nützlich erweisen, die Rolle der Analysis- und Synthesefilter zu vertauschen (was aufgrund der Form von (5.1) möglich ist). D.h. man sollte auch testen, was herauskommt, wenn man das durch (5.20) (statt (5.24)) gebene Filter zur inversen Transformation benutzt. Dies ist aufgrund der Anmerkung im Anschluß an Gleichung (5.24) dringend zu empfehlen.

Eine technische Einzelheit bleibt noch anzumerken: In der Praxis hat man endliche Signale $x(n), n = 0, 1, \ldots N-1$. Gleichung (7.19) ist am Rand so abzuändern, dass ein konstantes Signal unverändert bleibt. Dies kann beispielsweise durch symmetrische Fortsetzung am Rand erreicht werden.

Für die Behandlung von zweidimensionalen Bildsignalen erzeugt man Koeffizienten eines Subbandes HG durch Anwendung von (7.19) und (7.20) auf die einzelnen *Bildzeilen*. Entsprechend erhält man ein Subband GH durch Anwendung dieser Operationen auf die *Bildspalten* und ein Subband GG durch hintereinanderausgeführte Anwendung dieser Operation auf Zeilen und Spalten (zur Bezeichnungsweise der Subbänder siehe Abschnitt 6.2).

7.2.3 Extrapolation von Extrema der Waveletkoeffizienten

Diese Methode ist in [7] beschrieben. Sie beruht darauf, dass die Wavelet-Koeffizienten in Abhängigkeit von der Skalierungsstufe m sich wie

$$c_n = K_n \cdot 2^{m \cdot \alpha_n}$$

verhalten. Dieses Verhalten (bereits sehr vereinfacht) gilt nur für kontinuierliche Signale d.h. deren (hier nicht behandelte) Waveletkoeffizienten. Dabei ist α_n eine Zahl, die die Regularität oder die Singularität des Signals an der durch n gekennzeichneten Stelle beschreibt ($\alpha = -1$ entspricht der Diracschen Deltafunktion, $\alpha = 0$ der Stufenfunktion. Ein α mit $n < \alpha < n+1$ mit $n \in \mathbb{N}$ entspricht einer n mal differenzierbaren Funktion. Kanten entsprechen also Singularitäten und führen zu betragsgrößeren Waveletkoeffizienten. Man kann versuchen, die Konstanten K_n anhand von Testbildern mit scharfen Kanten, also $\alpha = 0$ empirisch zu bestimmen.

Man faßt das zu verschärfende Bild als Ergebnis einer Tiefpassfilterung in Zeilen- und Spaltenrichtung auf, d.h. als das Subband HH eines unbekannten wavelettransformierten größeren Bildes mit ebenfalls unbekannten Subbändern HG, GH, GG. Man führt die Wavelettransformation um eine oder mehrere Stufen weiter, und versucht, die fehlenden Subbänder der ersten Skalierungsstufe aus den entsprechenden Subbändern der höheren Skalierungsstufen durch Extrapolation zu ermitteln. Dabei ist es sinnvoll, von den Maxima und Minima der Pixelwerte auszugehen, die man durch „Scannen“ senkrecht zu den Kanten erhält, die durch das jeweilige Band bevorzugt dargestellt werden (also horizontal für HG, vertikal für GH und diagonal für GG). Aus diesen Maxima und Minima bestimmt man die Maxima und Minima der unbekannten Bänder und erhält somit die Koeffizienten, an den Stellen, die Bildkanten beschreiben. Die restlichen Koeffizienten der unbekannten Subbänder erhält man durch lineare Interpolation.

Anschließend kann man die inverse Wavelettransformation durchführen. Wenn man dabei mit Unterabtastung arbeitet, so erhält man ein größeres Bild als das ursprüngliche.

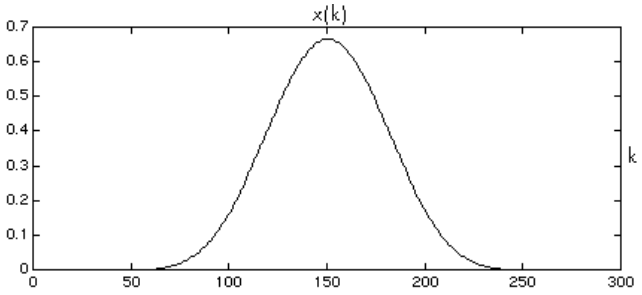
Man kann dann hoffen, dass dieses bei einer Darstellung mit höherer Auflösung (um dieselbe Ausdehnung wie das ursprüngliche zu erreichen) die gewünschte bessere Qualität besitzt. Daher wird in [7] die Wavelettransformation ohne Unterabtastung benutzt. Dies führt allerdings zu einem erheblich höheren Speicherbedarf, da die Wavelettransformation für mindestens zwei Skalierungsstufen zu berechnen ist. Je Skalierungsstufe erhält man dann drei neue „Subbänder", die gleich groß wie das ursprüngliche Bild sind (siehe Abschnitt 5.4).

Es ist vorstellbar, das in [7] vorgestellte Verfahren abzuändern, und einige Schritte durch ein mehr empirisches Vorgehen zu vereinfachen. Insbesondere ist es denkbar, auch die Wavelettransformation mit Unterabtastung zu benutzen. Dann erhält man aus den drei durch Extrapolation bestimmten Subbändern (die die gleiche Größe wie das vorhandene Bild haben) zusammen mit dem vorhandenen Bild ein neues Bild der doppelten Pixeldichte. Die Schwierigkeit beim Arbeiten mit Unterabtastung ist, dass zu jedem Pixel eines Subbandes einer Skalierungsstufe vier Pixel der entsprechenden Bildstelle der nächstniederen Skalierungsstufe gehören (wie aus Abb. 6.10 ersichtlich ist). Man hat also in diesem Fall die vierfache Zahl von Werten aus dem Subband zur nächstgröberen Skalierung durch Extrapolation zu bestimmen.

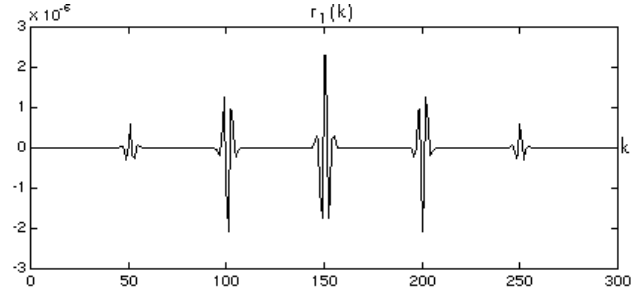
7.3 Analyse charakteristischer Merkmale von Daten

Bei vielen Anwendungen ist man daran interessiert, chakteristische Eigenschaften von Daten zu erkennen, um darauf — beispielsweise im Rahmen einer Regelung — angemessen zu reagieren. Dies wird in der englischsprachigen Literatur prägnant mit „feature detection" bezeichnet. Wir schauen uns hier ein mathematisches Beispiel an. In Abb. 7.7 ist oben ein Signal gezeigt, das stückweise aus kubischen Polynomen zusammengesetzt ist, und zwar so, dass sogar die zweite Ableitung überall stetig ist (das Signal ist ein skalierter und verschobener Bspline 4. Ordnung). Man kann in der grafischen Darstellung die Stellen, an denen „gestückelt" wurde, nicht erkennen. Die darunter abgebildeten Wavelet-Koeffizienten zeigen jedoch deutlich, dass an bestimmten Stellen offensichtlich eine qualitative Änderung im Signal vorliegt. Links und rechts von diesen Stellen ist das Signal durch unterschiedliche Polynome gegeben.

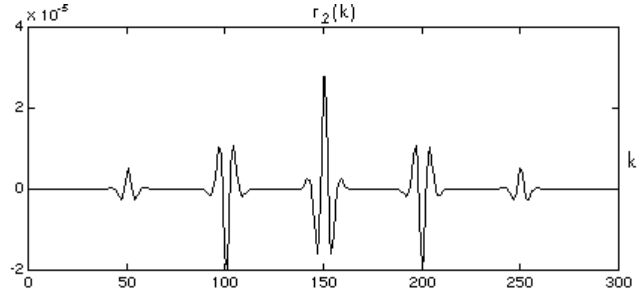
Zur Analyse chakteristischer Daten ist es sinnvoll, die Wavelet-Transformation ohne Unterabtastung zu nehmen. Denn man möchte charakteristische Änderungen in den Daten genauso erkennen, wenn sie zeitversetzt auftreten. Die Unterabtastung hat aber bekanntlich die Konsequenz, dass eine Translation des Signals zu einer komplizierten Änderung der Waveletkoeffizienten führt. Ohne Unterabtastung hat eine Translation des Signals eine entsprechende Translation der Koeffizienten zur Folge. Für das in der Abbildung gezeigte Beispiel wurden die Filter DD(8,8) sowie eine Implementierung mit dem Lifting-Schema gewählt (siehe (A.85) und (A.86)).



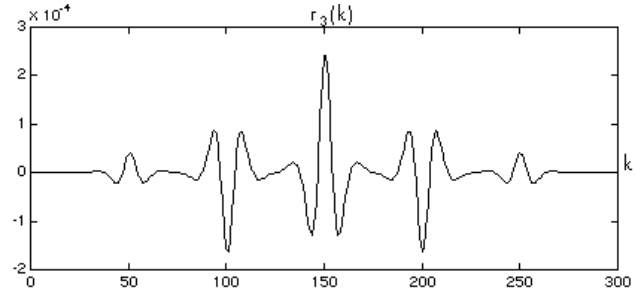
Signal (kubischer Spline)



Waveletkoeffizienten $r_1(k)$



Waveletkoeffizienten $r_2(k)$



Waveletkoeffizienten $r_3(k)$

Abbildung 7.7: Erkennung charakteristischer Eigenschaften eines Signals: die Sprungstellen der dritten Ableitung werden durch betragsgroße Waveletkoeffizienten sichtbar

Anhang A

Filter und Lifting-Konstanten für biorthogonale Wavelets

Hier sind einige Filter dokumentiert, die sich als nützlich herausgestellt haben. Weitere Filter findet man in [36] und [11].

A.1 FBI-Filter

k	$\tilde{h}(k)$	$h(k)$
0	0.8526986790094034193	0.7884856164056643978
±1	0.3774028556126537641	0.4180922732222122009
±2	-0.1106244044184234089	-0.04068941760955843672
±3	-0.02384946501938000189	-0.06453888262893843863
±4	0.03782845550699546 14	0.0

Tabelle A.1: Filterkonstanten für das FBI-Filter

Dieses Filter wird zur Kompression von Bildern von Fingerabdrücken benutzt (daher der Name) und hat sich auch für die Kompression von andern Bilddaten bewährt. Aufgrund der Länge wird es manchmal auch (9/7)-Filter genannt (in Abschnitt 3.3 ist ein anderes Filter dieser Länge beschrieben). Die Konstanten ergeben sich durch eine numerische Berechnung der Nullstellen eines Polynoms, wie dies in Abschnitt 2.8.4 beschrieben ist, daher die „krummen“ Zahlen, sie sind aus Tabelle A.1 ersichtlich. Es ist sinnvoll, die Konstanten c und d in Gleichung (2.32) und (2.33) so zu wählen, dass

$$G(z) = z^{-1}\tilde{H}(-z^{-1}) \text{ und } \tilde{G}(z) = z^{-1}H(-z^{-1}) \tag{A.1}$$

Dadurch sind die Hochpassfilter ebenfalls durch die Zahlen aus der Tabelle gegeben. Trotz der kleineren Länge führt das Rekonstruktionsfilter $H(z)$ zur regulärereren Skalierungsfunktion und damit zum regulärereren Wavelet.

Für die Implementierung des Lifting Schemas ist es sinnvoll, folgende Größen ein-

zuführen (Rechnung und Schreibweise — mit kleinen Änderungen — stammen aus [11])

$$
\begin{aligned}
r_0 &= \tilde{h}(0) - 2\frac{\tilde{h}(1)\tilde{h}(4)}{\tilde{h}(3)} && \text{(A.2)}\\
r_1 &= \tilde{h}(2) - \tilde{h}(4) - \frac{\tilde{h}(1)\tilde{h}(4)}{\tilde{h}(3)} && \text{(A.3)}\\
s_0 &= \tilde{h}(1) - \tilde{h}(3) - \tilde{h}(3)\frac{r_0}{r_1} && \text{(A.4)}\\
t_0 &= r_0 - 2r_1 && \text{(A.5)}\\
\alpha &= \frac{\tilde{h}(4)}{\tilde{h}(3)} \approx -1.58613434205992356 && \text{(A.6)}\\
\beta &= \frac{\tilde{h}(3)}{r_1} \approx -0.05298011857296141449 && \text{(A.7)}\\
\gamma &= \frac{r_1}{s_0} \approx 0.8829110755309332976 && \text{(A.8)}\\
\delta &= \frac{s_0}{t_0} \approx 0.4435068520439711518 && \text{(A.9)}\\
\zeta &= t_0 \approx 1.14960439886024116 && \text{(A.10)}
\end{aligned}
$$

Das Lifting Schema kann dann folgendermaßen implementiert werden:

$$
\begin{aligned}
s(k) &= x(2k), \qquad d(k) = x(2k+1) && \text{(A.11)}\\
d(k) &\mapsto d(k) + \alpha\Big(s(k) + s(k+1)\Big) && \text{(A.12)}\\
s(k) &\mapsto s(k) + \beta\Big(d(k) + d(k-1)\Big) && \text{(A.13)}\\
d(k) &\mapsto d(k) + \gamma\Big(s(k) + s(k+1)\Big) && \text{(A.14)}\\
s(k) &\mapsto s(k) + \delta\Big(d(k) + d(k-1)\Big) && \text{(A.15)}\\
s(k) &\mapsto \zeta s(k), \qquad d(k) \mapsto \frac{d(k)}{\zeta} && \text{(A.16)}
\end{aligned}
$$

Speichert man die Koeffizienten als ganzzahlige Werte ab, so bleibt die exakte Rekonstruktion gewährleistet (außer für (A.16)), wenn man die Änderung als reelle Zahl berechnet und vor der Addition rundet (oder die Nachkommastellen abschneidet). Bezeichnet man die Funktion des Rundens oder Abschneidens der Nachkommastellen mit $R(x)$, so erhält man beispielsweise für den Lifting-Schritt (A.12)

$$d(k) \mapsto d(k) + R\Big(\alpha\Big(s(k) + s(k+1)\Big)\Big) \qquad \text{(A.17)}$$

Für die Rücktransformation hat man dann

$$d(k) \mapsto d(k) - R\Big(\alpha\Big(s(k) + s(k+1)\Big)\Big) \qquad \text{(A.18)}$$

und der Rundungs- oder Abschneidefehler wird wieder abgezogen.

Dies gilt allerdings nicht für die Normierung in (A.16). Man kann diesen Normierungsschritt aber *ersetzen* durch folgende vier Lifting-Schritte

$$s(k) \;\mapsto\; s(k) + \left(\frac{1}{\zeta^2} - \frac{1}{\zeta}\right) d(k) \tag{A.19}$$

$$d(k) \;\mapsto\; d(k) + \zeta s(k) \tag{A.20}$$

$$s(k) \;\mapsto\; s(k) + \left(1 - \frac{1}{\zeta}\right) d(k) \tag{A.21}$$

$$d(k) \;\mapsto\; d(k) - s(k) \tag{A.22}$$

Dabei kann (A.19) mit (A.15) zu einem einheitlichen Liftingschritt zusammengefaßt werden.

Im Fall einer „exakten“ Arithmetik wird durch die Ersetzung der Normierung (A.16) durch die vier zusätzlichen Liftingschritte nichts geändert. Dies liegt an der folgenden Identität für Matrizen:

$$\begin{pmatrix} \zeta & 0 \\ 0 & \frac{1}{\zeta} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1-\frac{1}{\zeta} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \zeta & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{\zeta^2}-\frac{1}{\zeta} \\ 0 & 1 \end{pmatrix} \tag{A.23}$$

A.2 Filter zu orthogonalen Wavelets mit geringer Asymmetrie

k	$h(k)$
-3	-0.07576571478950221323
-2	-0.0296355276460024918
-1	0.49761866763277499
0	0.8037387518051320808
1	0.2978577956053060514
2	-0.09921954357663353258
3	-0.01260396726203130377
4	0.03222310060405146786

Tabelle A.2: Filterkonstanten für ein orthogonales Wavelet mit möglichst geringer Asymmetrie („Symlet“)

Tabelle A.2 zeigt die Filterkonstanten des in Abschnitt 2.8.4 konstruierten Beispiels, die Filterlänge entspricht der des FBI-Filters.

Aus diesen Filterkonstanten können folgende Hilfsgrößen berechnet werden:

$$r_0 = h(0) - \frac{h(-1)h(4)}{h(3)} \tag{A.24}$$

$$r_{-1} = h(-2) - \frac{h(-3)h(4))}{h(3)} \tag{A.25}$$

$$r_1 = h(2) - \frac{h(1)h(4)}{h(3)} \tag{A.26}$$

$$t_0 = h(1) - \frac{h(3)r_0}{r_1} - \frac{h(-3)r_1}{r_{-1}} \tag{A.27}$$

$$t_{-1} = h(-1) - \frac{h(-3)r_0}{r_{-1}} - \frac{h(3)r_{-1}}{r_1} \tag{A.28}$$

Aus den Filterkonstanten und diesen Hilfsgrößen lassen sich die bei den einzelnen Lifting-Schritten auftretenden Konstanten bestimmen:

$$\alpha = \frac{h(4)}{h(3)} \tag{A.29}$$

$$\beta_0 = \frac{h(3)}{r_1} \qquad \beta_1 = \frac{h(-3)}{r_{-1}} \tag{A.30}$$

$$\gamma_0 = \frac{r_{-1}}{t_{-1}} \qquad \gamma_1 = \frac{r_1}{t_0} \tag{A.31}$$

$$\xi = r_0 - \frac{r_1 t_{-1}}{t_0} - \frac{r_{-1}t_0}{t_{-1}} \tag{A.32}$$

$$\delta_0 = \frac{t_0}{\xi} \qquad \delta_1 = \frac{t_{-1}}{\xi} \tag{A.33}$$

$$\begin{aligned} \eta = -\xi\Big(& h(-1)\gamma_0\beta_1 + h(0)(\gamma_0 + \alpha\gamma_0\beta_1) \\ & + h(1)(1 + \beta_0\gamma_0 + \beta_1\gamma_1) \\ & + h(2)(\alpha + \alpha\beta_0\gamma_0 + \gamma_1 + \alpha\beta_1\gamma_1) \\ & + h(3)\beta_0\gamma_1 + h(4)\alpha\beta_0\gamma_1 \Big) \end{aligned} \tag{A.34}$$

Die numerischen Werte dieser Konstanten sind in Tabelle A.3 zusammengestellt.

Für die Implementierung des Lifting-Schemas ergibt sich damit:

$$s(k) = x(2k), \qquad d(k) = x(2k+1) \tag{A.35}$$

$$d(k) \mapsto d(k) + \alpha s(k+1) \tag{A.36}$$

$$s(k) \mapsto s(k) + \beta_0 d(k) + \beta_1 d(k-1) \tag{A.37}$$

$$d(k) \mapsto d(k) + \gamma_0 s(k) + \gamma_1 s(k+1) \tag{A.38}$$

$$s(k) \mapsto s(k) + \delta_0 d(k) + \delta_1 d(k-1) \tag{A.39}$$

$$d(k) \mapsto d(k) + \eta s(k) \tag{A.40}$$

$$s(k) \mapsto \xi s(k), \qquad d(k) \mapsto \xi^{-1} d(k) \tag{A.41}$$

Bei Abspeicherung der Koeffizienten als ganzzahlige Werte kann man wie beim FBI-Filter

Konstante	numerischer Wert
α	-2.556583965520256
β_0	-0.01903120704675512
β_1	0.3392439918648426
γ_0	1.05905726913757
γ_1	5.87692797927752
δ_0	0.03526079391665825
δ_1	-0.06598460741210342
ξ	3.195938969614102
η	-4.344017409218308

Tabelle A.3: numerische Werte der Konstanten für das Lifting-Schema für ein „Symlet“

die Normierung (A.41) durch folgende vier Liftingschritte ersetzen:

$$d(k) \mapsto d(k) + s(k) \tag{A.42}$$
$$s(k) \mapsto s(k) + (\xi - 1)d(k) \tag{A.43}$$
$$d(k) \mapsto d(k) - \frac{1}{\xi}s(k) \tag{A.44}$$
$$s(k) \mapsto s(k) + (\xi - \xi^2)d(k) \tag{A.45}$$

Dabei kann man (A.42) mit (A.40) zu einem einheitlichen Liftingschritt zusammenfassen.

Da für orthogonale Wavelets die Analysis- und Synthesefilter übereinstimmen, erhält man mit denselben Konstanten eine zweite Implementierungsmöglichkeit:

$$s(k) = x(2k), \qquad d(k) = x(2k+1) \tag{A.46}$$
$$s(k) \mapsto s(k) - \alpha d(k-1) \tag{A.47}$$
$$d(k) \mapsto d(k) - \beta_0 s(k) - \beta_1 s(k+1) \tag{A.48}$$
$$s(k) \mapsto s(k) - \gamma_0 d(k) - \gamma_1 d(k-1) \tag{A.49}$$
$$d(k) \mapsto d(k) - \delta_0 s(k) - \delta_1 s(k+1) \tag{A.50}$$
$$s(k) \mapsto s(k) - \eta d(k) \tag{A.51}$$
$$s(k) \mapsto \xi^{-1} s(k), \qquad d(k) \mapsto \xi d(k) \tag{A.52}$$

Bei der Verwendung von ganzzahliger Arithmetik können statt (A.52) folgende Schritte verwandt werden

$$s(k) \mapsto s(k) + (\xi^2 - \xi)d(k) \tag{A.53}$$
$$d(k) \mapsto d(k) + \frac{1}{\xi}s(k) \tag{A.54}$$
$$s(k) \mapsto s(k) + (1 - \xi)d(k) \tag{A.55}$$
$$d(k) \mapsto d(k) - s(k) \tag{A.56}$$

Auch hier können (A.51) und (A.53) zusammengefaßt werden.

A.3 Weitere Beispiele für Deslauriers-Dubuc-Filter

A.3.1 Filterbank mit Unterabtastung

Diese Filterfamilie wurde in Abschnitt 3.3 ausführlich besprochen. Hier sind weitere Beispiele, d.h. die Konstanten für das Lifing-Schema sowie spezielle Randfilter angegeben.

Randfilter für DD(4,2)

Für das Filter DD(4,2) wird hier anstelle von (3.61) und (3.62) für den linken Rand (d.h. für $k = 0$) vorgeschlagen

$$d(0) \;\mapsto\; d(0) - \frac{1}{16}\Big(5s(0) + 15s(1) - 5s(2) + s(3)\Big) \tag{A.57}$$

$$s(0) \;\mapsto\; s(0) + \frac{1}{4}\Big(3d(0) - d(1)\Big) \tag{A.58}$$

Die Gesamtzahl der ursprünglichen Daten sei mit N_0 bezeichnet, die Numerierung erfolge mit $k = 0, 1, 2, \ldots N_0 - 1$. Dann hat man $N_1 := \frac{1}{2}N_0$ Werte für die Koeffizienten $s(k)$ und $d(k)$; die Numerierung läuft hierfür von $k = 0$ bis $k = N_1 - 1$. Am rechten Rand ist die Operation (3.61) zu ersetzen durch

$$\begin{aligned} d(N_1 - 2) \;\mapsto\; & d(N_1 - 2) - \frac{1}{16}\Big(s(N_1 - 4) - 5s(N_1 - 3) \\ & \qquad +15s(N_1 - 2) + 5s(N_1 - 1)\Big) \end{aligned} \tag{A.59}$$

$$\begin{aligned} d(N_1 - 1) \;\mapsto\; & d(N_1 - 1) - \frac{1}{16}\Big(-5s(N_1 - 4) + 21s(N_1 - 3) \\ & \qquad -35s(N_1 - 2) + 35s(N_1 - 1)\Big) \end{aligned} \tag{A.60}$$

Für (3.62) ist am rechten Rand keine Abänderung notwendig.

Randfilter für DD(4,4)

Für das DD(4,4)-Filter erhält man die folgenden Änderungen für den linken Rand

$$d(0) \;\mapsto\; d(0) - \frac{1}{16}\Big(5s(0) + 15s(1) - 5s(2) + s(3)\Big) \tag{A.61}$$

$$s(0) \;\mapsto\; s(0) + \frac{1}{32}\Big(35d(0) - 35d(1) + 21d(2) - 5d(3)\Big) \tag{A.62}$$

$$s(1) \;\mapsto\; s(1) + \frac{1}{32}\Big(5d(0) + 15d(1) - 5d(2) + d(3)\Big) \tag{A.63}$$

sowie für den rechten Rand

$$\begin{aligned} d(N_1-2) &\mapsto d(N_1-2) - \frac{1}{16}\Big(-231s(N_1-4)+819s(N_1-3) \\ &\qquad -1001s(N_1-2)+429s(N_1-1)\Big) \end{aligned} \tag{A.64}$$

$$\begin{aligned} d(N_1-1) &\mapsto d(N_1-1) - \frac{1}{16}\Big(-429s(N_1-4)+1485s(N_1-3) \\ &\qquad -1755s(N_1-2)+715s(N_1-1)\Big) \end{aligned} \tag{A.65}$$

$$\begin{aligned} s(N_1-1) &\mapsto s(N_1-1) + \frac{1}{32}\Big(d(N_1-4)-5d(N_1-3) \\ &\qquad +15d(N_1-2)+5d(N_1-1)\Big) \end{aligned} \tag{A.66}$$

Für die Normierung sei an die Empfehlung in Abschnitt 3.3 erinnert.

Lifting-Schritte und Randfilter für DD(8,6)

Weiterhin sind hier die Lifting-Schritte für $N = 8$ und $\tilde{N} = 6$, d.h. für DD(8,6).

$$s(k) = x(2k), \qquad d(k) = x(2k+1) \tag{A.67}$$

$$\begin{aligned} d(k) &\mapsto d(k) + 2^{-11}\cdot\Big(5s(k-3)-49s(k-2)+245s(k-1)-1225s(k) \\ &\qquad -1225s(k+1)+245s(k+2)-49s(k+3)+5s(k+4)\Big) \end{aligned} \tag{A.68}$$

$$\begin{aligned} s(k) &\mapsto s(k) + 2^{-9}\cdot\Big(3d(k-3)-25d(k-2)+150d(k-1) \\ &\qquad +150d(k)-25d(k+1)+3d(k+2)\Big) \end{aligned} \tag{A.69}$$

Anschließend ist die Normierung

$$s(k) \mapsto \sqrt{2}s(k), \qquad d(k) \mapsto \frac{1}{2}\sqrt{2}d(k) \tag{A.70}$$

sinnvoll. Bei Verwendung ganzzahliger Arithmetik ist dies nicht möglich. Es ist in Abschnitt 3.3 beschrieben, wie die Multiplikation mit $\sqrt{2}$ in diesem Fall zu vermeiden ist.

Die folgenden Randfilter sind durch Extrapolation berechnet, wie dies in Abschnitt 3.4 erklärt wurde. Am linken Rand ($k = 0, 1, 2$) erhält man statt (A.68)

$$\begin{aligned} d(0) &\mapsto d(0) + 2^{-11}\cdot\Big(-429s(0)-3003s(1)+3003s(2)-3003s(3) \\ &\qquad +2145s(4)-1001s(5)+273s(6)-33s(7)\Big) \end{aligned} \tag{A.71}$$

$$\begin{aligned} d(1) &\mapsto d(1) + 2^{-11}\cdot\Big(+33s(0)-693s(1)-2079s(2)+1155s(3) \\ &\qquad -693s(4)+297s(5)-77s(6)+9s(7)\Big) \end{aligned} \tag{A.72}$$

$$\begin{aligned} d(2) &\mapsto d(2) + 2^{-11}\cdot\Big(-9s(0)+105s(1)-945s(2)-1575s(3) \\ &\qquad +525s(4)-189s(5)+45s(6)-5s(7)\Big) \end{aligned} \tag{A.73}$$

Die Operationen am rechten Rand sind bis auf einen zusätzlich auftretenden Fall symmetrisch zum linken Rand. Die Gesamtzahl der ursprünglichen Daten sei mit N_0 bezeichnet

(numeriert mit $k = 0, 1, 2, \ldots N_0 - 1$). Dann hat man $N_1 := \frac{1}{2}N_0$ Werte für die Koeffizienten $s(k)$ und $d(k)$; der Index k läuft hierfür also von $k = 0$ bis $k = N_1 - 1$. Man hat dann am rechten Rand statt (A.68) folgende Sonderfälle:

$$\begin{aligned} d(N_1-4) \;\mapsto\; & d(N_1-4) + 2^{-11} \cdot \Big(-5s(N_1-8) + 45s(N_1-7) - 189s(N_1-6) \\ & +525s(N_1-5) - 1575s(N_1-4) - 945s(N_1-3) \\ & +105s(N_1-2) - 9s(N_1-1) \Big) \end{aligned} \tag{A.74}$$

$$\begin{aligned} d(N_1-3) \;\mapsto\; & d(N_1-3) + 2^{-11} \cdot \Big(9s(N_1-8) - 77s(N_1-7) + 297s(N_1-6) \\ & -693s(N_1-5) + 1155s(N_1-4) - 2079s(N_1-3) \\ & -693s(N_1-2) + 33s(N_1-1)) \Big) \end{aligned} \tag{A.75}$$

$$\begin{aligned} d(N_1-2) \;\mapsto\; & d(N_1-2) + 2^{-11} \cdot \Big(-33s(N_1-8) + 273s(N_1-7) \\ & -1001s(N_1-6) + 2145s(N_1-5) - 3003s(N_1-4) \\ & +3003s(N_1-3) - 3003s(N_1-2) - 429s(N_1-1) \Big) \end{aligned} \tag{A.76}$$

$$\begin{aligned} d(N_1-1) \;\mapsto\; & d(N_1-1) + 2^{-11} \cdot \Big((429s(N_1-8) - 3465s(N_1-7) \\ & +12285s(N_1-6) - 25025s(N_1-5) + 32175s(N_1-4) \\ & -27027s(N_1-3) + 15015s(N_1-2) - 6435s(N_1-1) \Big) \end{aligned} \tag{A.77}$$

Auch für den Korrekturschritt kann man spezielle Randfilter verwenden. Für den linken Rand ergibt sich

$$\begin{aligned} s(0) \;\mapsto\; & s(0) + 2^{-9} \cdot \Big(693d(0) - 1155d(1) + 1386d(2) \\ & -990d(3) + 385d(4) - 63d(5) \Big) \end{aligned} \tag{A.78}$$

$$\begin{aligned} s(1) \;\mapsto\; & s(1) + 2^{-9} \cdot \Big(63d(0) + 315d(1) - 210d(2) \\ & +126d(3) - 45d(4) + 7d(5) \Big) \end{aligned} \tag{A.79}$$

$$\begin{aligned} s(2) \;\mapsto\; & s(2) + 2^{-9} \cdot \Big(-7d(0) + 105d(1) + 210d(2) \\ & -70d(3) + 21d(4) - 3d(5) \Big) \end{aligned} \tag{A.80}$$

und entsprechend für den rechten Rand

$$\begin{aligned} s(N_1-2) \;\mapsto\; & s(N_1-2) + 2^{-9} \cdot \Big(-3d(N_1-6) + 21d(N_1-5) - 70d(N_1-4) \\ & +210d(N_1-3) + 210d(N_1-2) - 112d(N_1-1) \Big) \end{aligned} \tag{A.81}$$

$$\begin{aligned} s(N_1-1) \;\mapsto\; & s(N_1-1) + 2^{-9} \cdot \Big(7d(N_1-6) - 45d(N_1-5) + 126d(N_1-4) \\ & -210d(N_1-3) + 630d(N_1-2) - 252d(N_1-1) \Big) \end{aligned} \tag{A.82}$$

A.3.2 Filterbank ohne Unterabtastung

Hierbei hat man zu beachten, dass die Operation „Aufspaltung“ einfach eine Verdopplung der Daten darstellt, was zu einer Verschiebung der entsprechenden Indizes führt.

Hier sind die Lifting-Schritte für zwei Testbeispiele angegeben, bei denen für die „Korrektur“-Schritte Filter derselben Länge gewählt sind (im Gegensatz zu den Beispielen in Abschnitt A.3.1). Bis auf ein Vorzeichen und einen Faktor $\frac{1}{2}$ tauchen dann dieselben Zahlen wieder auf. Für $N = 6$ erhält man

$$\begin{aligned}
s_i(k) &= s_{i-1}(k); \qquad r_i(k) = s_{i-1}(k) \\
r_i(k) &\mapsto r_i(k) - 2^{-8}\Bigg(150\Big(s_i(k-2^{i-1}) + s_i(k+2^{i-1})\Big) \\
&\qquad -25\Big(s_i(k-3\cdot 2^{i-1}) + s_i(k+3\cdot 2^{i-1})\Big) \\
&\qquad +3\Big(s_i(k-5\cdot 2^{i-1}) + s_i(k+5\cdot 2^{i-1})\Big)\Bigg) \qquad \text{(A.83)} \\
s_i(k) &\mapsto s_i(k) + 2^{-9}\Bigg(150\Big(r_i(k-2^{i-1}) + r_i(k+2^{i-1})\Big) \\
&\qquad -25\Big(r_i(k-3\cdot 2^{i-1}) + r_i(k+3\cdot 2^{i-1})\Big) \\
&\qquad +3\Big(r_i(k-5\cdot 2^{i-1}) + r_i(k+5\cdot 2^{i-1})\Big)\Bigg) \qquad \text{(A.84)}
\end{aligned}$$

sowie für $N = 8$

$$\begin{aligned}
s_i(k) &= s_{i-1}(k); \qquad r_i(k) = s_{i-1}(k) \\
r_i(k) &\mapsto r_i(k) - 2^{-11}\Bigg(1225\Big(s_i(k-2^{i-1}) + s_i(k+2^{i-1})\Big) \\
&\qquad -245\Big(s_i(k-3\cdot 2^{i-1}) + s_i(k+3\cdot 2^{i-1})\Big) \\
&\qquad +49\Big(s_i(k-5\cdot 2^{i-1}) + s_i(k+5\cdot 2^{i-1})\Big) \\
&\qquad -5\Big(s_i(k-7\cdot 2^{i-1}) + s_i(k+7\cdot 2^{i-1})\Big)\Bigg) \qquad \text{(A.85)} \\
s_i(k) &\mapsto s_i(k) + 2^{-12}\Bigg(1225\Big(r_i(k-2^{i-1}) + r_i(k+2^{i-1})\Big) \\
&\qquad -245\Big(r_i(k-3\cdot 2^{i-1}) + r_i(k+3\cdot 2^{i-1})\Big) \\
&\qquad +49\Big(r_i(k-5\cdot 2^{i-1}) + r_i(k+5\cdot 2^{i-1})\Big) \\
&\qquad -5\Big(r_i(k-7\cdot 2^{i-1}) + r_i(k+7\cdot 2^{i-1})\Big)\Bigg) \qquad \text{(A.86)}
\end{aligned}$$

Es sei warnend darauf hingewiesen, dass in der Literatur teilweise unterschiedliche Normierungen vorgenommen sind.

A.4 Filter aus der Mittelwert-Interpolation

Diese Filter werden durch eine „Average Interpolation“ oder genauer, durch eine „Average Interpolation Subdivision“ erzeugt. Dazu wird auf die Literatur [34, 14, 6] verwiesen. Hier wird für die entsprechenden Filter und die damit definierte Wavelet-Transformation die Abkürzung AI gefolgt von zwei Zahlen (in Anlehnung an [6]) benutzt, die erste Zahl gibt dabei die Vielfachheit von $z = -1$ als Nullstelle des Synthese-Tiefpassfilters an, die zweite die entsprechende Vielfachheit für das Analyse-Filter.

Die durch AI(3,1) gegebene Transformation läßt sich mit den folgenden Lifting-Schritten durchführen:

$$\begin{aligned}
s(k) &= x(2k), \qquad d(k) = x(2k+1) && \text{(A.87)}\\
s(k) &\mapsto s(k) + d(k) && \text{(A.88)}\\
d(k) &\mapsto d(k) + \frac{1}{16}(s(k-1) - \frac{1}{2}s(k) - \frac{1}{16}s(k+1) && \text{(A.89)}\\
s(k) &\mapsto \frac{1}{2}\sqrt{2}s(k), \qquad d(k) \mapsto \sqrt{2}d(k) && \text{(A.90)}
\end{aligned}$$

Die Transformation beginnt — im Unterschied zu den bisher hier behandelten Beispielen — mit einem Korrekturschritt. Dieser stimmt mit dem einer anderen (hier nicht besprochenen) Realisierung der zum Haar-Wavelet gehörenden Transformation überein. Die Normierung in (A.90) wurde in Übereinstimmung mit der hier üblichen und damit abweichend von der in [6] vorgenommen. Die Normierungsschritte sind jedoch gerade umgekehrt wie in den bisher behandelten Beispielen. Die Empfehlungen von Abschnitt 3.3 zur Implementierung der Normierung können dennoch entsprechend angewandt werden. Beachten Sie, dass beim Vorhersageschritt hier die Berichtigung eines offensichtlichen Tippfehlers in [6] vorgenommen wurde. Wenn man nach den Methoden von Abschnitt 3.2.4 die zugehörigen Filter berechnet, so erhält man als Analyse-Tiefpassfilter das des Haar-Wavelets mit einer unstetigen Skalierungsfunktion.

Eine Verbesserung erreicht man durch Übergang zur Transformation AI(3,3) durch Einfügen des folgenden Korrekturschritts vor der Normierung in (A.90):

$$s(k) \mapsto s(k) + \frac{1}{4}d(k-1) - \frac{1}{4}d(k+1) \qquad \text{(A.91)}$$

Dadurch erhält das Analyse-Tiefpassfilter eine dreifache Nullstelle in $z = -1$.

Die Transformation AI(5,1) erhält man durch folgende Lifting-Schritte (nur der Vorhersageschritt ist abgeändert)

$$\begin{aligned}
s(k) &= x(2k), \qquad d(k) = x(2k+1) && \text{(A.92)}\\
s(k) &\mapsto s(k) + d(k) && \text{(A.93)}\\
d(k) &\mapsto d(k) - \frac{3}{256}s(k-2) + \frac{11}{128}s(k-1) - \frac{1}{2}s(k)\\
&\qquad -\frac{11}{128}s(k+1) + \frac{3}{256}s(k+2) && \text{(A.94)}\\
s(k) &\mapsto \frac{1}{2}\sqrt{2}s(k), \qquad d(k) \mapsto \sqrt{2}d(k) && \text{(A.95)}
\end{aligned}$$

Sie kann durch denselben Korrekturschritt wie für AI(3,3), der durch (A.91) gegeben ist und vor der Normierung durchzuführen ist, zur Transformation AI(5,3) verbessert werden.

Durch den neuen Korrekturschritt

$$s(k) \mapsto s(k) - \frac{1}{32}d(k-2) + \frac{5}{16}d(k-1) - \frac{5}{16}d(k+1) + \frac{1}{32}d(k+2) \tag{A.96}$$

erhält man für Rekonstruktion und Analyse eine fünffache Nullstelle des Tiefpassfilters in $z = 1$. Bemerkenswert ist, dass auch hier bis auf die Normierung in den Nennern nur Zweierpotenzen auftreten. Bei Verwendung von ganzzahliger Arithmetik erlaubt dies eine sehr rechenzeitgünstige Implementierung der Transformation.

Es wäre von Interesse, diese Filter sorgfältig für die Bildkompression zu testen. Erste Versuche scheinen vielversprechend zu sein.

Anhang B

Filter für die Wavelet-Transformation mit Schachbrett-Abtastung

Für die Berücksichtigung der 12 nächsten Nachbarn hat man folgende Transformation für die n. Skalierungsstufe zu implementieren (die Aufspaltung ist dieselbe wie bei vier Nachbarn, ebenso die Fallunterscheidung):

Falls n *ungerade* ist, hat man mit $m := 2^{\frac{n-1}{2}}$ durchzuführen:

Vorhersage: (nur falls $x(i,k) = d_n(i,k)$)

$$\begin{aligned} x(i,k) \;\mapsto\; & x(i,k) - \frac{5}{16}\Big(x(i-m,k) + x(i+m,k) + x(i,k-m) + x(i,k+m)\Big) \\ & + \frac{1}{32}\Big(x(i-2m,k-m) + x(i-2m,k+m) + x(i-m,k-2m) \\ & \quad + x(i-m,k+2m) + x(i+2m,k-m) + x(i+2m,k+m) \\ & \quad + x(i+m,k-2m) + x(i+m,k+2m)\Big) \end{aligned} \tag{B.1}$$

Korrektur: (nur falls $x(i,k) = s_n(i,k)$)

$$\begin{aligned} x(i,k) \;\mapsto\; & x(i,k) + \frac{5}{32}\Big(x(i-m,k) + x(i+m,k) + x(i,k-m) + x(i,k+m)\Big) \\ & - \frac{1}{64}\Big(x(i-2m,k-m) + x(i-2m,k+m) + x(i-m,k-2m) \\ & \quad + x(i-m,k+2m) + x(i+2m,k-m) + x(i+2m,k+m) \\ & \quad + x(i+m,k-2m) + x(i+m,k+2m)\Big) \end{aligned} \tag{B.2}$$

Falls n *gerade* ist, setze $m := 2^{\frac{n}{2}-1}$. Dann ist durchzuführen:

Vorhersage: (nur falls $x(i,k) = d_n(i,k)$)

$$\begin{aligned} x(i,k) \;\mapsto\; x(i,k) - \frac{5}{16}\Big(&x(i-m,k-m) + x(i+m,k-m) \\ &+ x(i-m,k+m) + x(i+m,k+m)\Big) \\ +\frac{1}{32}\Big(&x(i-3m,k-m) + x(i-3m,k+m) + x(i-m,k-3m) \\ &+ x(i-m,k+3m) + x(i+m,k-3m) + x(i+m,k+3m) \\ &+ x(i+3m,k-m) + x(i+3,k+m)\Big) \end{aligned} \tag{B.3}$$

Korrektur: (nur falls $x(i,k) = s_n(i,k)$)

$$\begin{aligned} x(i,k) \;\mapsto\; x(i,k) + \frac{5}{32}\Big(&x(i-m,k-m) + x(i+m,k-m) \\ &+ x(i-m,k+m) + x(i+m,k+m)\Big) \\ -\frac{1}{64}\Big(&x(i-3m,k-m) + x(i-3m,k+m) + x(i-m,k-3m) \\ &+ x(i-m,k+3m) + x(i+m,k-3m) + x(i+m,k+3m) \\ &+ x(i+3m,k-m) + x(i+3m,k+m)\Big) \end{aligned} \tag{B.4}$$

Die Normierung ist gemäß (3.95) oder (3.96) durchzuführen.

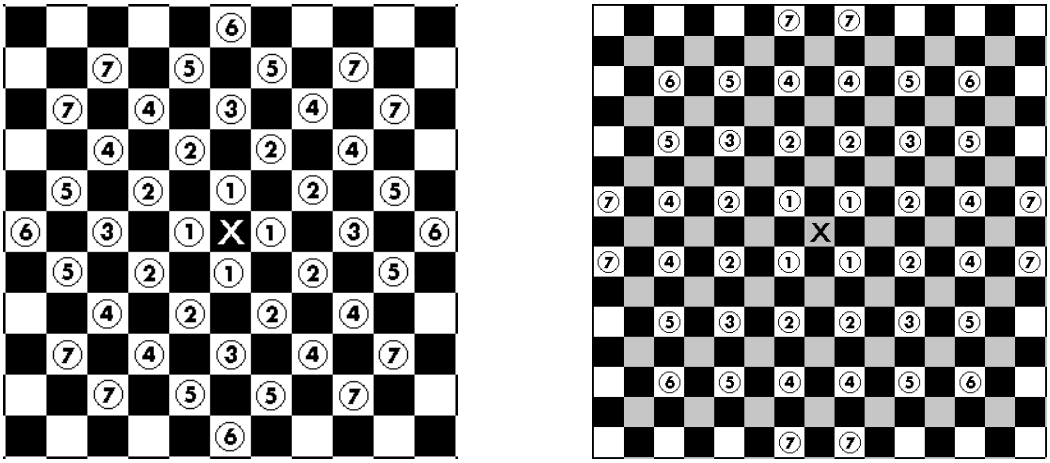


Abbildung B.1: Vorhersage (an der durch „×“ markierten Stelle) durch Interpolation, Einteilung der Nachbarn nach Entfernung und geometrischer Lage: links für die erste, rechts für die zweite Skalierungsstufe. Die Zahlen in Kreisen geben die Kategorie an, der zugehörige Gewichtsfaktor kann Tabelle B.1 entnommen werden.

Berücksichtigt man noch mehr Nachbarn bei der Interpolation, so ist es sinnvoll, diese nach Entfernung und geometrischer Lage zu der Stelle, an der die Vorhersage erfolgen soll, in Kategorien einzuteilen. Dies ist in Abb. B.1 gezeigt. Die Vorhersage erfolgt nun als gewichtetes Mittel über die Nachbarn, wobei die Gewichtsfaktoren der Tabelle B.1 zu entnehmen sind. Für die Korrektur sind diese Gewichtsfaktoren jeweils mit umgekehrtem Vorzeichen sowie einem zusätzlichen Faktor $\frac{1}{2}$ zu benutzen. Man macht sich die

Bedeutung der einzelnen Zahlen in der Tabelle am besten klar, indem man die schon angegebenen Transformationen bei Berücksichtigung der Nachbarn der ersten Kategorie (nur die nächsten Nachbarn) sowie der ersten und zweiten Kategorie (siehe z.B. (3.99)) mit der entsprechenden Zeile der Tabelle vergleicht. Zu beachten ist dabei, dass in einigen Fällen die Hinzunahme weiterer Nachbarn, beispielsweise in der vierten Zeile, den Grad des Interpolationspolynoms nicht erhöht. Die Nachbarn der 6. und 7. Kategorie haben den selben Abstand vom Punkt, an dem der Wert vorhergesagt werden soll. Daher wurde darauf verzichtet, die Filterkonstanten für die Interpolation mit den ersten sechs Kategorien zu berechnen. Die auftretenden Filter wurden von Kovačević und Sweldens in [20] „Quincunx Neville Filters“ genannt. Daher wird als Abkürzung hier QN(n) vorgeschlagen, wobei n die Zahl der berücksichtigten Nachbarn angibt.

Zähler							Nenner	Grad	Zahl
1							2^2	2	4
10	-1						2^5	4	12
81	-9	1					2^8	6	16
174	-27	2	3				2^9	6	24
1404	-231	34	27	-3			2^{12}	8	32
23300	-4470	625	850	-75	9	-80	2^{16}	8	44
1	2	3	4	5	6	7			
Kategorie									

Tabelle B.1: Gewichtsfaktoren für die Vorhersage durch Interpolation. Die Kategorie entspricht den Ziffern in den Kreisen in Abb. B.1. „Grad“ gibt den Grad des Interpolationspolynoms, „Zahl“ die Gesamtzahl der berücksichtigten Nachbarn an.

Literatur: Ein großer Teil der hier angegebenen Filter steht in [20]. Das Interpolationsverfahren für mehrere Variable, das bei der Filterberechnung benutzt wird, steht in [13].

Hinweis auf Normierung bzw. Verweis auf Stelle, wo die Normierung erklärt ist! Überprüfen, ob Normierung mehrfach erklärt ist (einmal richtig genügt!).

Anhang C

Einsparung von Speicherplatz durch das Lifting-Schema

Die Überlegungen dieses Abschnitts gelten nur für den Fall, daß man die Wavelettransformation zeilen- und spaltenweise durchführt, wie das in Abschnitt 1.6 beschrieben ist. Sie sind für die Schachbrett-Abtastung *nicht* zutreffend.

Für eine direkte Implementierung der durch (2.36) und (2.37) gegebenen Wavelet-Transformation benötigt man einen Arbeitsspeicher zusätzlich zum Speicher für die Daten $s_m(k)$, d.h. man kann die Speicherplätze für $s_m(k)$ nicht sofort überschreiben. Beim Lifting-Schema kann man diesen zusätzlichen Arbeitsspeicher einsparen, wenn man sich „merkt“, dass die Koeffizienten $s_{m+1}(k)$ an den Speicherplätzen von $s_m(k)$ mit *geradem* Index, $d_{m+1}(k)$ an denen mit *ungeradem* Index stehen.

Führt man die Wavelet-Transformation in mehreren Stufen (Skalierungen) durch, so ergibt sich durch fortgesetztes Überschreiben folgende Position der Koeffizienten der m. Stufe, wenn die ursprünglichen Daten mit $x(k)$ $(k = 0, \ldots N-1)$ bezeichnet werden:

$s_m(k)$	steht an der Position von	$x(2^m k)$
$d_m(k)$	steht an der Position von	$x(2^m k + 2^{m-1})$

Dabei läuft k von 0 bis $2^{-m}N - 1$. Falls N keine Zweierpotenz ist, so stellt dies beim Lifting-Schema kein Problem dar, man muß lediglich bei der Implementierung darauf achten, nicht über den Rand des Speicherbereiches zu geraten.

Etwas komplizierter wird die Speicheraufteilung bei der Transformation eines Bildes mit einer Breite von N_H und einer Höhe von N_V Pixeln (also N_V Zeilen und N_H Spalten. Die Bilddaten werden i.a. zeilenweise in einem Vektor $x(k)$ $(k = 0, \ldots N_V \cdot N_H - 1)$ abgespeichert, der Intensitätswert des Pixels der i. Zeile und j. Spalte ist also $x(i \cdot N_H + j)$. Die folgenden Überlegungen betreffen nur die übliche getrennte Transformation von Zeilen und Spalten, bei denen die vier Subbänder HH, HG, GH und GG entstehen, nicht aber die in Abschnitt 3.6.1 und B beschriebene Transformation mit Schachbrett-Abtastung.

Ordnet man die Koeffizienten nach der Transformation in den einzelnen Subbändern wie z.B. in Abb. 1.15 an, so ergeben sich folgende Speicherplatzpositionen für die Koeffizienten der Subbänder der m. Stufe zur Bildposition i. Zeile, j. Spalte:

Band	Koeff. steht in Pos. von
HH	$x(2^m \cdot i \cdot N_H + 2^m \cdot j)$
HG	$x(2^m \cdot i \cdot N_H + 2^m \cdot j + 2^{m-1})$
GH	$x\Big((2^m \cdot i + 2^{m-1})N_H + 2^m \cdot j\Big)$
GG	$x\Big((2^m \cdot i + 2^{m-1})N_H + 2^m \cdot j + 2^{m-1}\Big)$

Für die Durchführung des Lifting-Schemas in der m. Stufe ergeben sich folgende Speicherpositionen in der hierzu verwandten Notation:

Für die Transformation der Zeilen (Index i):

Koeff.	steht in Pos. von
$s(k)$	$x(2^{m-1} \cdot i \cdot N_H + 2^m \cdot k)$
$d(k)$	$x(2^{m-1} \cdot i \cdot N_H + 2^m \cdot k + 2^{m-1})$

und für die anschließende Transformation der Spalten (Index j):

Koeff.	steht in Pos. von
$s(k)$	$x(2^m \cdot k \cdot N_H + 2^{m-1} \cdot j)$
$d(k)$	$x\Big((2^m \cdot k + 2^{m-1}) \cdot N_H + 2^{m-1} \cdot j\Big)$

Anhang D

Mathematische Ergänzungen

D.1 Beispiel für die Nichtkonvergenz des Kadkade-Algorithmus

Hier werden einige Rechnungen ausführlich durchgeführt, deren Ergebnisse am Ende von Abschnitt 2.4 beim zweiten Beispiel benutzt wurden, für das der Kaskade-Algorithmus nicht existiert. Wir gehen vom Tiefpass-Rekonstruktionsfilter des Wavelets CDF(3,5) aus:

$$H_0(z) = \frac{\sqrt{2}}{8}\left(z + 3 + 3z^{-1} + z^{-2}\right) \tag{D.1}$$

Hierfür existiert der Grenzwert (2.17) und liefert die Funktion $\Phi_0(\omega)$. Durch inverse Fouriertransformation nach (2.18) erhalten wir daraus die Skalierungsfunktion $\varphi_0(x)$, die in Abb. 2.20 grafisch dargestellt ist. Von den Anwendungen her ist es natürlich nicht vielversprechend, in das Filter (D.1) Nullen einzufügen, was zum folgenden Filter führt

$$H_1(z) = \frac{\sqrt{2}}{8}\left(z^4 + 3z + 3z^{-2} + z^{-5}\right) \tag{D.2}$$

Wir haben dabei darauf geachtet, dass das neue Filter wieder symmetrisch zu $\frac{1}{2}$ ist. Zwischen beiden Filtern besteht die Beziehung

$$H_1(z) = zH_0(z^3) \qquad \text{bzw.} \qquad H_1(e^{j\omega}) = e^{j\omega}H_0(e^{j3\omega}) \tag{D.3}$$

Ein naiver Vergleich von

$$\Phi_0(\omega) = \frac{1}{\sqrt{2}}H_0(e^{j\frac{\omega}{2}}) \cdot \frac{1}{\sqrt{2}}H_0(e^{j\frac{\omega}{4}}) \cdot \frac{1}{\sqrt{2}}H_0(e^{j\frac{\omega}{8}}) \cdot \frac{1}{\sqrt{2}}H_0(e^{j\frac{\omega}{16}}) \cdot \cdots$$

und

$$\begin{aligned}
\Phi_1(\omega) &= \frac{1}{\sqrt{2}}H_1(e^{j\frac{\omega}{2}}) \cdot \frac{1}{\sqrt{2}}H_1(e^{j\frac{\omega}{4}}) \cdot \frac{1}{\sqrt{2}}H_1(e^{j\frac{\omega}{8}}) \cdot \frac{1}{\sqrt{2}}H_1(e^{j\frac{\omega}{16}}) \cdot \cdots \\
&= \frac{1}{\sqrt{2}}e^{j\frac{\omega}{2}}H_0(e^{j\frac{3\omega}{2}}) \cdot \frac{1}{\sqrt{2}}e^{j\frac{\omega}{4}}H_0(e^{j\frac{3\omega}{4}}) \cdot \frac{1}{\sqrt{2}}e^{j\frac{\omega}{8}}H_0(e^{j\frac{3\omega}{8}}) \cdot \frac{1}{\sqrt{2}}e^{j\frac{\omega}{16}}H_0(e^{j\frac{3\omega}{16}}) \cdot \cdots \\
&= e^{j\omega(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots)}\Phi_0(3\omega)
\end{aligned} \tag{D.4}$$

liefert

$$\Phi_1(\omega) = e^{j\omega}\Phi_0(3\omega) \tag{D.5}$$

Dabei wurde der bekannte Grenzwert der geometrischen Reihe

$$\sum_{k=1}^{\infty}\left(\frac{1}{2}\right)^k = \sum_{k=0}^{\infty}\left(\frac{1}{2}\right)^k - 1 = \frac{1}{1-\frac{1}{2}} - 1 = 1$$

benutzt. „Ein naiver Vergleich“ bedeutet hier, dass wir vorausgesetzt haben, dass wir hier aus dem unendlichenProdukt unendlich viele Faktoren herausziehen können und die entstehenden Grenzwerte getrennt berechnen können. Tatsächlich handelt es sich hier um den Grenzwert von Funktionen und es kommt genau darauf an, welcher Grenzwert durch die Pünktchen $\cdots$ gemeint ist. Für einzelne (beliebige) Werte von ω ist die hier vorgenommene Vorgehensweise gerechtfertigt.

Mit Hilfe der Rechenregeln für die Fourier-Transformation für beliebige $a > 0$

$$af(ax) \;\circ\!\!-\!\!\bullet\; F\left(\tfrac{\omega}{a}\right) \qquad \text{bzw.} \qquad \tfrac{1}{a}f\left(\tfrac{1}{a}x\right) \;\circ\!\!-\!\!\bullet\; F(a\omega)$$

erhalten wir

$$\tfrac{1}{3}\varphi_0\left(\tfrac{1}{3}x\right) \;\circ\!\!-\!\!\bullet\; \Phi_0(3\omega) = e^{-j\omega}\Phi_1(j\omega)$$

Die Rechenregel $f(x+x_0) \;\circ\!\!-\!\!\bullet\; e^{j\omega x_0}F(\omega)$ liefert dann

$$\tfrac{1}{3}\varphi_0\big(\tfrac{1}{3}(x+1)\big) = \varphi_1(x) \;\circ\!\!-\!\!\bullet\; \Phi_1(\omega) \tag{D.6}$$

Die Funktion $\varphi_1(x)$ ist auch Lösung der Zwei-Skalen-Relation (2.20) für die durch (D.2) gegebenen Filterkonstanten $h_1(k)$. Der Kaskade-Algorithmus (2.23) konvergiert jedoch nicht für diese Filterkonstanten.

D.2 Berechnung der Filterkonstanten für das FBI-Filter

Ausgangspunkt ist (2.121) für $q = 4$; man erhält (2.131), hier geschrieben als

$$P(z) = 2^{-8}(1+z)^4(1+z^{-1})^4 \cdot Q(z)$$

mit

$$Q(z) = 2^{-3} \cdot \Big(208 - 131(z+z^{-1}) + 40(z^2+z^{-2}) - 5(z^3+z^{-3})\Big)$$

Für eine Faktorisierung $P(z) = H(z)\cdot\tilde{H}(z^{-1})$, bei der nicht ganz $Q(z)$ dem Faktor $\tilde{H}(z^{-1})$ „zugeschlagen“ wird (wie im ersten Beispiel CDF(3,5) in Abschnitt 2.8.4), ist es notwendig, die Nullstellen von $Q(z)$ zu bestimmen. Um Software zur Nullstellenbestimmung von Polynomen nutzen zu können, ist es sinnvoll, die negativen Potenzen auszuklammern und zu schreiben

$$Q(z) = 2^{-3} \cdot z^{-3}\Big(208z^3 - 131(z^4+z^2) + 40(z^5+z) - 5(z^6+1)\Big)$$

Entsprechende Software liefert z_1, z_1^*, $1/z_1$, $1/z_1^*$, z_2 und $1/z_2$, mit der numerischen Näherung

$$z_1 = 2,0311355 + j\cdot 1,7389508; \qquad z_2 = 3,0406605$$

als Nullstellen von $Q(z)$. Wir können daher $Q(z)$ nach dem Fundamentalsatz der Algebra schreiben als

$$Q(z) = 2^{-3}\cdot z^{-3}\cdot(-5)(z-z_1)(z-z_1^{-1})(z-z_1^*)(z-\tfrac{1}{z_1^*})(z-z_2)(z-z_2^{-1})$$

Die Umformungen

$$z - z_1^{-1} = z(1 - z_1^{-1}z^{-1}) = z \cdot z_1^{-1}(z_1 - z^{-1})$$

sowie dieselbe Umformung für $z - z_2^{-1}$ und

$$z - \tfrac{1}{z_1^*} = \frac{z}{z_1^*}(z_1^* - z^{-1})$$

liefern das Ergebnis

$$\begin{aligned} Q(z) &= -5 \cdot 2^{-3} z^{-3}(z - z_1)z \cdot z_1^{-1}(z_1 - z^{-1})(z - z_1^*)\frac{z}{z_1^*}(z_1^* - z^{-1})(z - z_2)z \cdot z_2^{-1}(z_2 - z^{-1}) \\ &= 5 \cdot 2^{-3} z_1^{-1}\frac{1}{z_1^*}z_2^{-1}(z_1 - z)(z_1 - z^{-1})(z_1^* - z)(z_1^* - z^{-1})(z_2 - z)(z_2 - z^{-1}) \end{aligned}$$

Mit der Konstanten $c_P := \frac{5}{2^{11} \cdot z_1 z_1^* z_2}$ erhalten wir damit das Ergebnis

$$P(z) = c_P(1 + z)^4(1 + z^{-1})^4 \cdot (z_1 - z)(z_1 - z^{-1})(z_1^* - z)(z_1^* - z^{-1})(z_2 - z^{-1}) \qquad \text{(D.7)}$$

Da $H(z)$ und $\tilde{H}(z)$ reell sein sollen und wir symmetrische Filter haben wollen, also $H(z) = H(z^{-1})$ oder $H(z^{-1}) = zH(z)$ und entsprechend für $\tilde{H}$, müssen wir für eine sinnvolle Faktorisierung $P(z) = H(z) \cdot \tilde{H}(z^{-1})$ diese Faktoren symmetrisch auf $H(z)$ und $\tilde{H}(z^{-1})$ verteilen, denn es muss gelten

$$H(z_0) = 0 \qquad \Longrightarrow \qquad H(z_0^*) = 0 \quad \text{und} \quad H(z_0^{-1}) = 0$$

und analog für $\tilde{H}$. Eine naheliegende Faktorisierung erhält man, wenn man die Nullstellen in $z = -1$ völlig symmetrisch auf $H(z)$ und $\tilde{H}(z^{-1})$ aufteilt, das Paar der zusätzlichen reellen Nullstellen $H(z)$ „zuschlägt“ und das „Quartett“ der komplexen Nullstellen zu $\tilde{H}(z)$ nimmt. Dies führt auf (2.132) und (2.133), hier nochmal aufgeführt:

$$\begin{aligned} \tilde{H}(z) &= \tilde{c}(1 + z)^2(1 + z^{-1})^2(z_1 - z)(z_1 - z^{-1})(z_1^* - z)(z_1^* - z^{-1}) \\ H(z) &= c(1 + z)^2(1 + z^{-1})^2(z_2 - z)(z_2 - z^{-1}) \end{aligned}$$

Mit dem Kaskade-Algorithmus kann man die zugehörigen Skalierungsfunktionen berechnen und feststellen, dass eine Vertauschung von $H(z)$ und $\tilde{H}(z)$ nicht sinnvoll ist, denn die hier vorgenommene Wahl führt zu einer regulärereren Synthesefunktion $\varphi(z)$ und einer weniger regulären Analysefunktion $\tilde{\varphi}(z)$, was als sinnvoller als die umgekehrte Wahl angesehen wird.

Literaturverzeichnis

[1] Werner Bäni. *Wavelets. Eine Einführung für Ingenieure*. Oldenbourg, München, 2002.

[2] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[3] Kathrin Berkner and Raymond O. Wells, Jr. A correlation-dependent model for denoising via nonorthogonal wavelet transforms. Technical Report CML report 98-07, Computational Mathematics Laboratory, Rice University Houston, TX 77005-1892, 1998. Available at
http://cml.rice.edu/98.html.

[4] Christopher M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. Technical report, Los Alamos National Laboratory, 1996.

[5] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms. A Primer*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[6] Bernard Maria Cena. *Reconstruction for Visualisation of Discrete Data Fields Using Wavelet Signal Processing*. PhD thesis, Department of Computer Science, University of Western Australia, 2000.

[7] Grace S. Chang, Zoran Cvetkovic, and Martin Vetterli. Resolution enhancement of images using wavelet transform extrema extrapolation. *Proc. ICASSP*, 4(May):2379–2382, 1995.

[8] R. R. Coifman and L. Donoho. Translation-invariant de-noising. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 125–150. Springer-Verlag, New York, 1995.

[9] Z. Cvetković and M. Vetterli. Discrete-time wavelet extrema representation: Design and consistent reconstruction. *IEEE Trans. Signal Process.*, 43(3):681–693, 1995.

[10] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, 1992.

[11] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[12] G. Davis and A. Nosratinia. Wavelet-based image coding: An overview. *Applied and Computational Control, Signals, and Circuits*, 1(1):to appear, 1998.
Available at
http://research.microsoft.com/~geoffd/papers/accsc.ps.gz.

[13] Carl de Boor and Amos Ron. Computational aspects of polynomial interpolation in several variables. *Math.Comp.*, 58:705–727, 1992.

[14] David L. Donoho. Smooth wavelet decompositions with blocky coefficient kernels. In Larry L. Schumaker and Glenn Webb, editors, *Recent Advances in Wavelet Analysis*, pages 259–308. Academic Press, Boston, 1994.

[15] David L. Donoho. De-noising by soft-thresholding. *IEEE Trans. Inform. Theory*, 41:613–627, 1995.

[16] G. Fernández, S. Periaswamy, and Wim Sweldens. LIFTPACK: A software package for wavelet transforms using lifting. In M. Unser, A. Aldroubi, and A. F. Laine, editors, *Wavelet Applications in Signal and Image Processing IV*, pages 396–408. Proc. SPIE 2825, 1996. Available at
http://cm.bell-labs.com/who/wim/papers/papers.html.

[17] Darrel R. Hankerson, Greg A. Harris, and Peter D. Johnson, Jr. *Introduction to Information Theory and Data Compression.* The CRC Press Series in Discrete Mathematics and Its Applications. CRC Press, Boca Raton etc., 1998.

[18] Barbara Burke Hubbard. *The World According to Wavelets. The Story of a Mathematical Technique in the Making.* A K Peters, Wellesley, Massachusetts, USA, 1996. deutsche Übersetzung in der FH-Bibliothek vorhanden, Signatur: 33TIE1201.

[19] Arne Jensen and Anders la Cour-Harbo. *Ripples in Mathematics. The Discrete Wavelet Transform.* Springer, Berlin, 2001.

[20] J. Kovačević and W. Sweldens. Wavelet families of increasing order in arbitrary dimensions. *IEEE Trans. Image Proc.*, 9(3):480–496, March 2000.

[21] F. Luisier and T. Blu. Image denoising by pointwise thresholding of the undecimated wavelet coefficients: A global SURE optimum. In *Proceedings of the Thirty-Second IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'07)*, pages I–593–I–596, Honolulu HI, USA, April 15-20, 2007. http://bigwww.epfl.ch/publications/luisier0702.html.

[22] F. Luisier, T. Blu, and M. Unser. A new SURE approach to image denoising: Interscale orthonormal wavelet thresholding. *IEEE Transactions on Image Processing*, 16(3):593–606, March 2007.

[23] S. Mallat. Zero-crossings of a wavelet transform. *IEEE Trans. Inform. Theory*, 37(4):1019–1033, 1991.

[24] S. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Trans. Inform. Theory*, 38(2):617–643, 1992.

[25] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Trans. Patt. Anal. Mach. Intell.*, 14(7):710–732, 1992.

[26] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, 1998.

[27] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[28] N. Saito. *Local Feature Extraction and Its Application Using a Library of Bases*. PhD thesis, Yale University, Yale, 1994.

[29] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Signal Process.*, 41(12):3445–3462, 1993.

[30] Alexander Stoffel. Remarks on the unsubsampled wavelet transform and the lifting scheme. *Signal Processing*, 69(2):177–182, 1998.

[31] G. Strang and N. Truong. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, 1996.

[32] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[33] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200, 1996.

[34] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics*, pages 15–87. ACM SIGGRAPH Course notes, 1996.

[35] M.J. Tsai, J. Villasenor, and F. Chen. Stack-run image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:519–521, 1996. Available at http://www.icsl.ucla.edu/~ipl/papers/stack_run.html.

[36] G. Uytterhoeven, D. Roose, and A. Bultheel. Wavelet transforms using the lifting scheme. Technical Report ITA-Wavelets Report WP 1.1 (Revised Version), April 28, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, 1997. Available at
http://www.cs.kuleuven.ac.be/~geert/Publications.html.

[37] Geert Uytterhoeven. *Wavelets: Software and Applications*. PhD thesis, Faculty of Applied Sciences of the Katholieke Universiteit Leuven, Department of Computer Science, Belgium, 1999.

[38] James S. Walker. *A Primer on Wavelets and their Scientific Applications*. The CRC Press Series in Discrete Mathematics and Its Applications. Chapmann & Hall/CRC, Boca Raton etc., 1999.

[39] Mladen Victor Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A. K. Peters, Wellesley, Massachusetts, 1994.

[40] L.L. Winger and A.N. Venetsanopoulos. Stack-run coding with space-frequency quantization. *ICIP*, 1:620–623, 1997.
Available at
http://www.dsp.toronto.edu/~lowell/icip97.ps.Z.

[41] I. H. Witten, R. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Comm. ACM*, 30(6):520–540, 1987.