5.3.3. JPEG:

Bilder werden üblicherweise als Matrix gespeichert, deren Einträge pixelweise den Grauwert des entsprechend nummerierten Pixels enthalten, bzw. bei Farbbildern den Anteil einer der drei Farbkomponenten Rot, Grün oder Blau (= RGB).

RGB wird meist in der YUV-Form umgewandelt, bei der die Matrix Y die Helligkeit beschreibt, und U und V den Farbton (U,V lassen sich stärker komprimieren!).

JPEG zerlegt das Bild in 8 x 8 – Pixel große Blöcke, die zunächst getrennt bearbeitet werden. Auf jede dieser Teilmatrizen wird eine zweidimensionale Cosinus-Transformation angewandt: DCT

Warum DCT, warum nicht DFT?

Aus Anwendung der DCT erhält man für die Teilmatrizen wieder die Frequenzanteil-koeffizienten a_k und b_k in Teilmatrizen.

Die Teilmatrix mit den Frequenzanteilen wird nun quantisiert,d.h. die darin stehenden reellen Zahlen werden bestimmten Zahlenintervallen zugeordnet und Intervallweise jeweils durch eine Zahl angenähert, die für ein ganzes Intervall gilt. Maschinenzahlen!

Dadurch werden auch automatisch alle kleinen Komponenten, die in dem ersten Intervall mit den kleinsten Werten liegen, durch Null ersetzt.
Dies erzeugt einen Qualitätsverlust!

Die entstandene Gesamtmatrix aus den diskreten Zahlenwerten wird codiert – ***Huffman-Codierung***

Nachteile:

Fourier-Methoden haben Schwierigkeiten mit Kanten in Bildern

Kanten ←→ Unstetigkeiten

Durch stetige Funktionen cos(kx), sin(kx) sind Kanten schlecht darstellbar! Differenzierbarkeit!

Kosten der DCT: O(n log(n)) bei n Pixel. Zu teuer.

Daher DCT auf 8x8-Blöcke.

JPEG2000:

Anstatt Cosinus-Transformation auf 8 x 8 wendet man eine Wavelet-Transformation auf größere Teilmatrizen an.

Wavelet-Ansatzfunktionen haben keine Problem mit Kanten, vgl. B-Splines. Genauere Untersuchung folgt.

Kosten für Wavelet-Transformation O(n)

Beispiel: MATLAB Gibbs-Phänomen (recht.m)

Fourierkoeffizienten für Rechtecksignal;
Überschwingen an Kante unvermeidbar bei Darstellung durch endliches trigonometrisches Polynom!

5.3.4. Lineare Filter:

Betrachte Vektor v, dessen Komponenten wieder diskrete Werte einer Funktion oder eines Bildes darstellen (Sampling).

Zunächst 1-dimensional.
Wir *filtern* diesen Vektor, indem wir jede Komponente ersetzen durch eine gewichtete Kombination der Nachbarkomponenten.

So entspricht die *Maske*:

$\frac{1}{4}[1 \quad 2 \quad 1]$ der Operation $v_j = \frac{v_{j-1} + 2v_j + v_{j+1}}{4}$,

bzw. der Matrixmultiplikation

$$v \rightarrow \frac{1}{4}\begin{pmatrix} 2 & 1 & & & \\ 1 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2 & 1 \\ & & & 1 & 2 \end{pmatrix} \cdot v$$

Vorsicht am Rand! Fehlende Werte haben Einfluß auf Messwerte!
Man benötigt Annahmen für v_0 und v_{n+1}. Was ist sinnvoll?

Im zwei-dim. entspricht dies z.B. der Maske

$$\frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \rightarrow \quad v_{i,j} = \frac{v_{i,j-1} + v_{i-1,j} + 4v_{i,j} + v_{i,j+1} + v_{i+1,j}}{8},$$

Genauso Gaussfilter:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Dies entspricht glättenden Filtern, die zu einem weicheren Bild führen:
Mittelwertfilter, Glätter, Weichzeichner, oder Tiefpassfilter zur Abschwächung von Rauschen, bzw. hochfrequenten Anteilen.
Hochfrequente Störungen, die einzelne Komponente verändern, werden durch die Mittelwertbildung verringert!

Entsprechend kann man Hochpassfilter definieren, die die Unterschiede hervorheben, das Bild härter machen und niederfrequente (glatte) Anteile abschwächen (Differenzfilter, Scharfzeichner).

z.B. Laplacefilter:
(Sobelfilter)

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

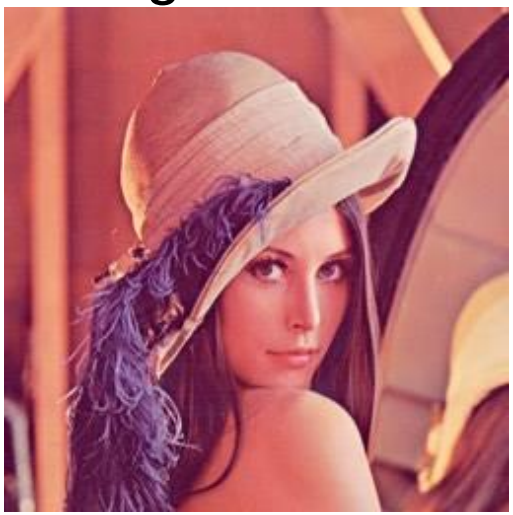
Bisher entsprechen die Methoden einer *Filterung im Ortsraum*.
Im Gegensatz dazu kann man auch *Filter im Phasenraum*
(Frequenz~) zum Entfernen von Rauschen (Noise) verwenden:

v → DFT(v) → Filter → IDFT(v)

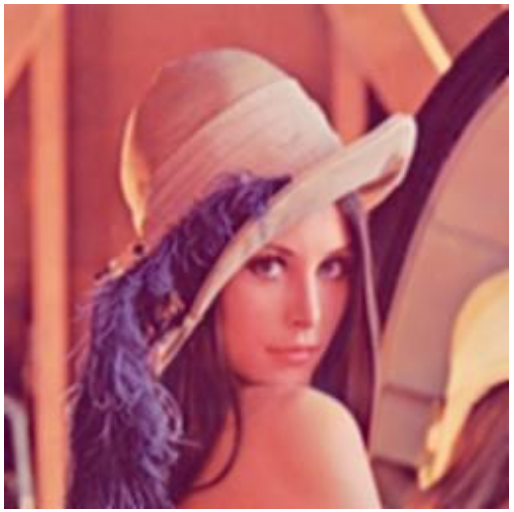
Also Transformation in (Fourier-)Koeffizientenraum,
dann Filtern der Koeffizienten = Gewichten/Löschen,
dann Rücktransformation (vgl. MP3)
Hochpassfilter, Tiefpass, Filterband,…

Filtern eines Bildes:

Original



Mittelwertfilter



Reduktion

Differenzfilter

Reduktion

5.3.5. Wavelets als rekusive Filter:

Betrachte **Kombination** von Tiefpassfilter und Hochpassfilter im 1-Dimensionalen zu Masken

$$\begin{bmatrix}1 & 1\end{bmatrix}/2 \qquad \text{und} \qquad \begin{bmatrix}1 & -1\end{bmatrix}/2$$

Ersetze den ursprünglichen Vektor verlustfrei durch tief-, bzw. hochpass-gefilterte Vektoren halber Länge (down sampling):

$$\begin{pmatrix} v_t \\ \hline v_h \end{pmatrix} = \frac{1}{2} \left(\begin{array}{cccccc} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & & \ddots & \ddots \\ \hline 1 & -1 & & & & \\ & & 1 & -1 & & \\ & & & & \ddots & \ddots \end{array} \right) \cdot v$$

oder

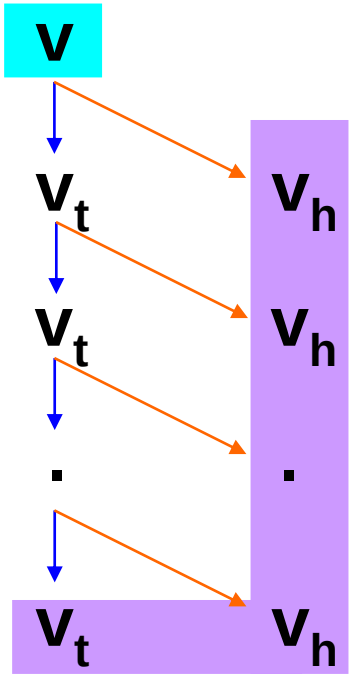
$$\frac{1}{2}\begin{pmatrix} \boxed{\begin{matrix}1 & 1\\ 1 & -1\end{matrix}} & & & & \\ & \boxed{\begin{matrix}1 & 1\\ 1 & -1\end{matrix}} & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & \\ & & & & \boxed{\begin{matrix}1 & 1\\ 1 & -1\end{matrix}} \end{pmatrix} \cdot v$$

(nur anders sortiert)

Der Differenzanteil v_h ist in der Regel klein und wird nicht weiterbearbeitet!

Der Mittelwertanteil (tiefpassgefiltert) v_t wird nach demselben Schema weiteraufgespalten wiederum in Tief/Hochpassanteil durch dieselben Masken.

Insgesamt:



Level 0 , Länge n

Level 1 , n/2

Level 2 , n/4

.

Level p , 1

Ersetze nun den Ausgangsvektor $v \in \mathbf{R}^n$ durch den letzten Mittelwertanteil auf Level p und sämtliche Differenzanteile v_h , das sind auch genau n Zahlen.
(v_h und v_t entsprechen quasi den Fourierkoeffizienten)

Vorteile:

Gesamtkosten der Transformation O(n)! Warum?
Differenz-Anteile meist klein → gut komprimierbar

Differenzanteil auf Level k enthält Information über das Verhalten von v auf diesem Level, bzw. in dieser Auflösung
→

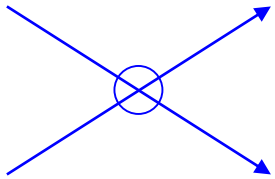
Multiskalenanalyse, Zerlegung des Vektors in verschiedene Frequenzbereiche = Skalen
Entspricht in etwa der Fourier-Analyse.

Filter müssen sparse sein damit billig,
leicht umkehrbar (invertierbar) damit Original rekonstruierbar!
(vgl. DFT und IDFT)
Aufspalten in Frequenzanteil (hoch – tief).

Algorithmische Ähnlichkeit zur Fourier-Transformation am Beispiel des Haar-Filters mit den Masken

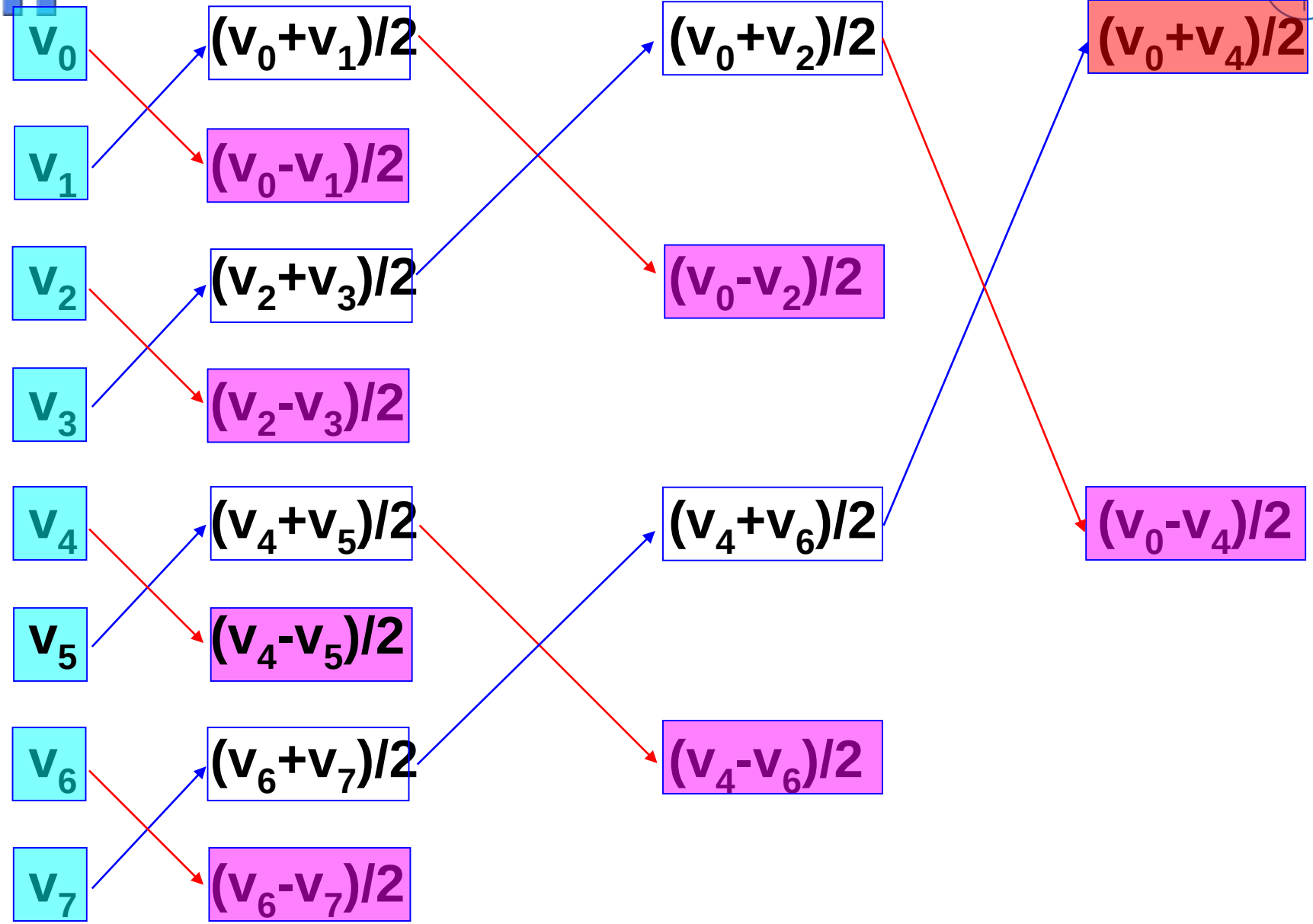
$$\frac{1}{2}[1 \quad 1] \qquad \text{und} \qquad \frac{1}{2}[1 \quad -1] \;:$$

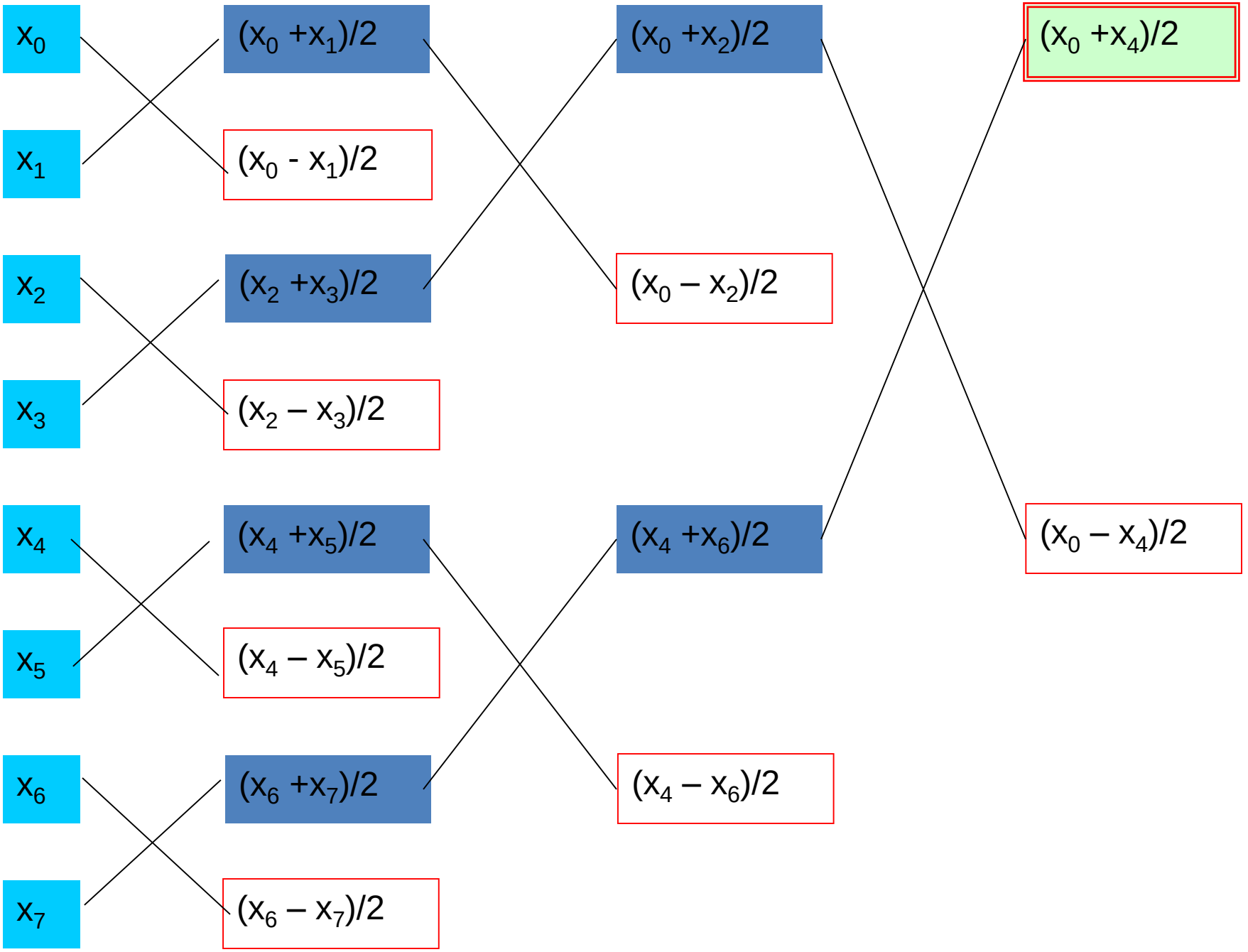
Butterfly



u → **(u+v)/2**

v → **(u-v)/2**


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
$(v_0+v_1)/2$
$(v_0-v_1)/2$
$(v_2+v_3)/2$
$(v_2-v_3)/2$
$(v_4+v_5)/2$
$(v_4-v_5)/2$
$(v_6+v_7)/2$
$(v_6-v_7)/2$
$(v_0+v_2)/2$
$(v_0-v_2)/2$
$(v_4+v_6)/2$
$(v_4-v_6)/2$
$(v_0+v_4)/2$
$(v_0-v_4)/2$


x_0
x_1
x_2
x_3
x_4
x_5
x_6
x_7
$(x_0 + x_1)/2$
$(x_0 - x_1)/2$
$(x_2 + x_3)/2$
$(x_2 - x_3)/2$
$(x_4 + x_5)/2$
$(x_4 - x_5)/2$
$(x_6 + x_7)/2$
$(x_6 - x_7)/2$
$(x_0 + x_2)/2$
$(x_0 - x_2)/2$
$(x_4 + x_6)/2$
$(x_4 - x_6)/2$
$(x_0 + x_4)/2$
$(x_0 - x_4)/2$

v wird dabei transformiert in
alle Differenzanteile und
den letzten Mittelwert.

Unterschied zur DFT:

keine Sortierphase,
einfacherer Butterfly,
nur die Mittelwertanteile werden weiter bearbeitet

→ *O(n)*

Bessere Filterkombination, z.B. Daubechy-Wavelets

$$\left[1+\sqrt{3} \quad 3+\sqrt{3} \quad 3-\sqrt{3} \quad 1-\sqrt{3}\right] / 4\sqrt{2}$$ Mittelwert-Filter

$$\left[1-\sqrt{3} \quad -3+\sqrt{3} \quad 3+\sqrt{3} \quad -1-\sqrt{3}\right] / 4\sqrt{2}$$ Diffferenz-filter

Filterkoeffizienten beschreiben rekursiv eine Funktion.

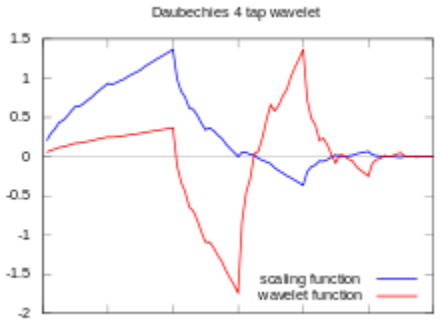
Sie beschreiben den Zusammenhang der Waveletfunktion von einer Skalierung zu einer anderen:

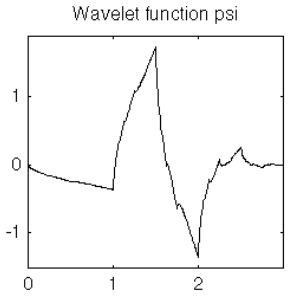
$$\Phi(x) = \sum_k a_k \Phi(2x - k)$$

$$\Phi(2^j x - k), \qquad W(2^j x - k)$$

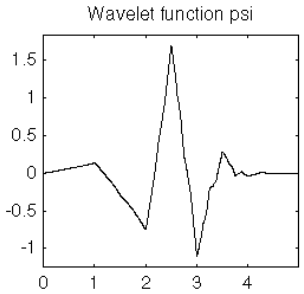
Funktion W(x) erzeugt Basis, vergleiche cos(x).

Daubechies Wavelets (orthogonal)

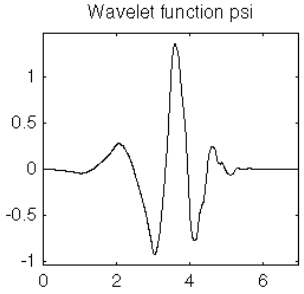




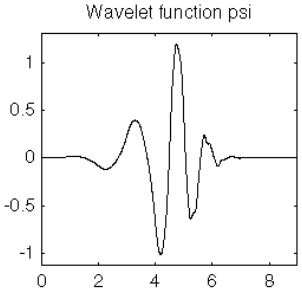
db2



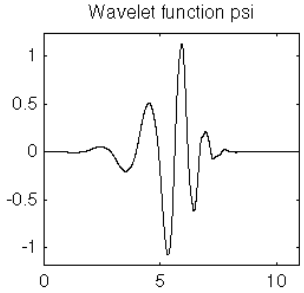
db3



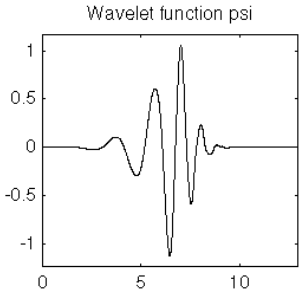
db4



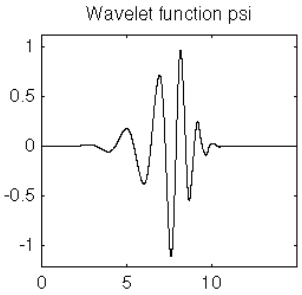
db5



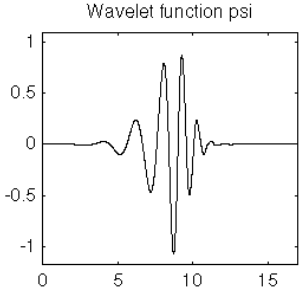
db6



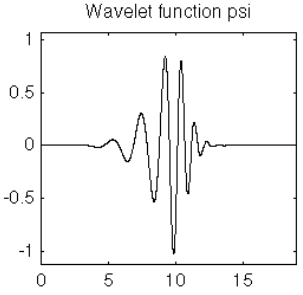
db7



db8

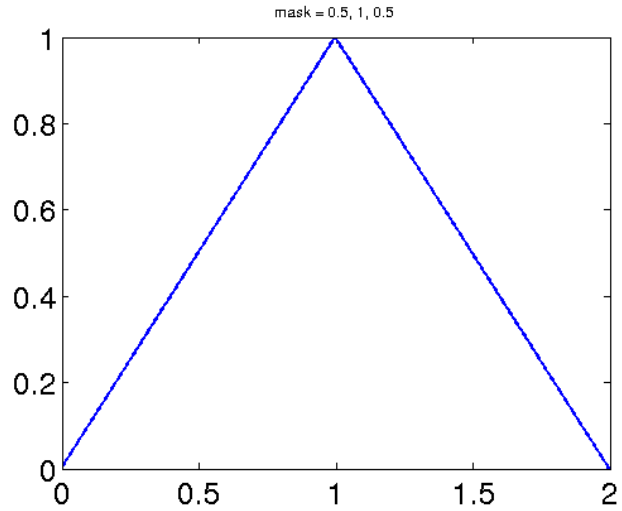


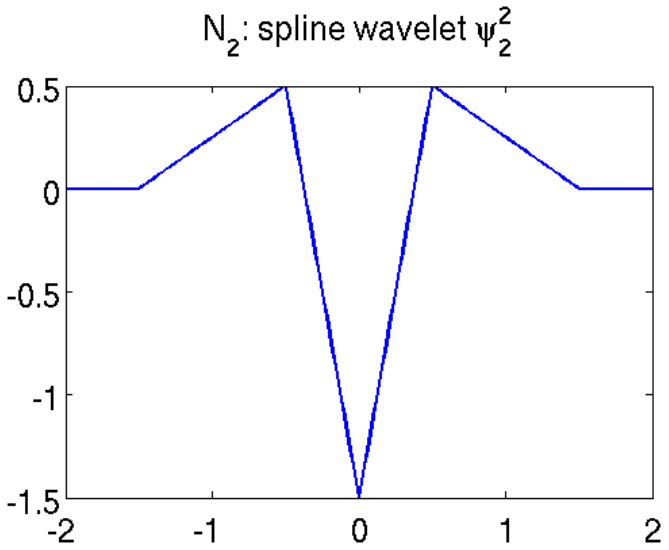
db9

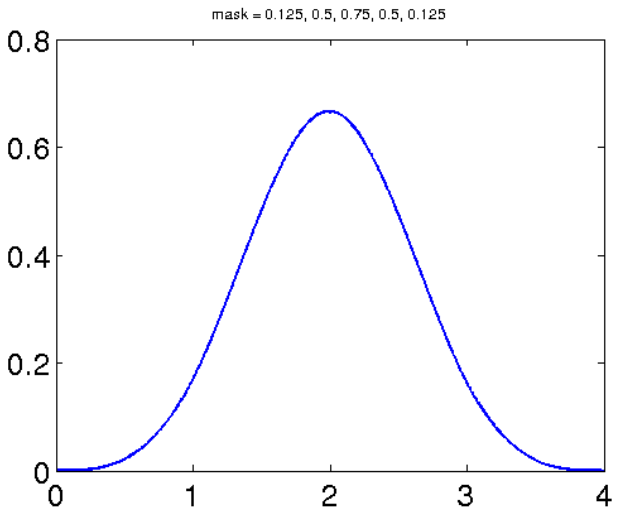


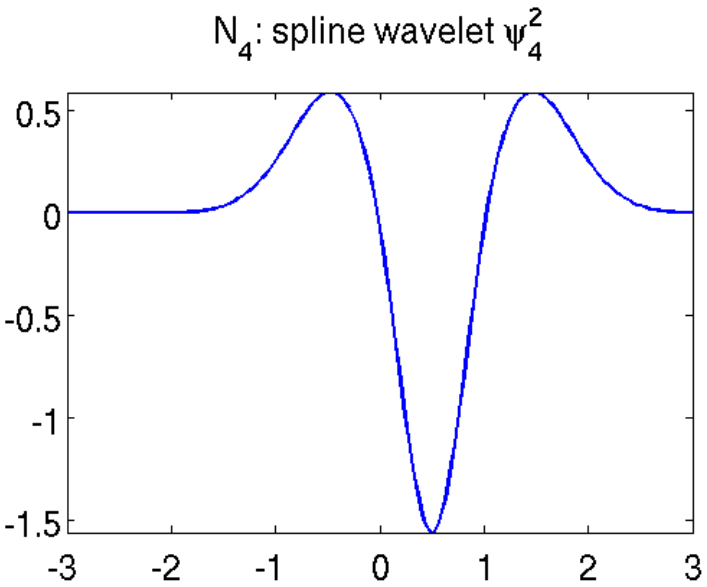
db10

Spline Wavelets

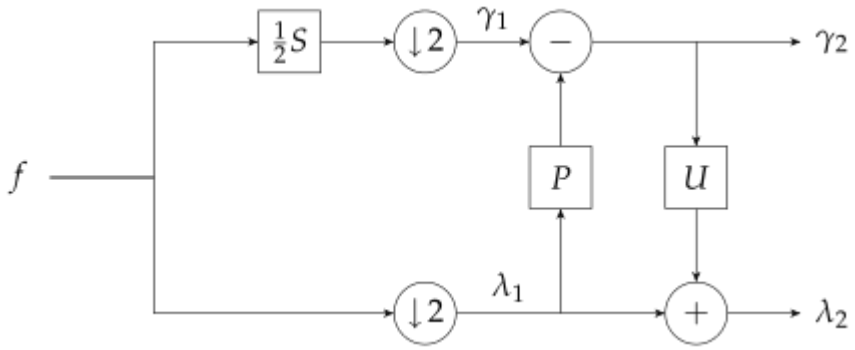




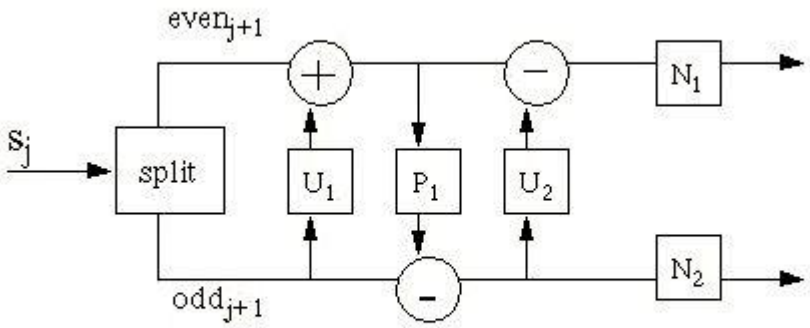




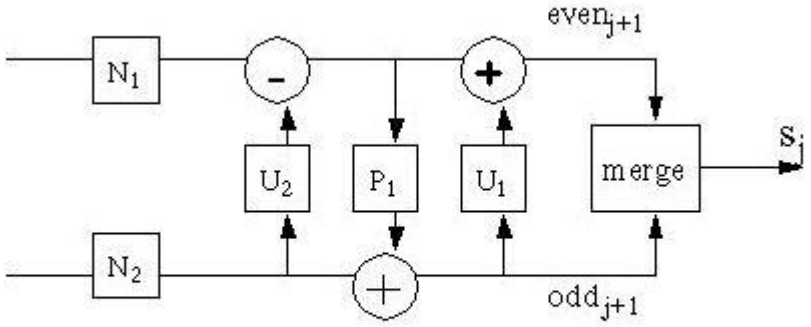
Lifting Scheme (ohne Funktion)



Splitten in Gerade/Ungerade wie Reißverschluss.
Dann zwei oder mehr Filteroperationen wie Predictor/Update.
Vorteil: Leicht umkehrbar.



Daubechies D4 forward wavelet transform

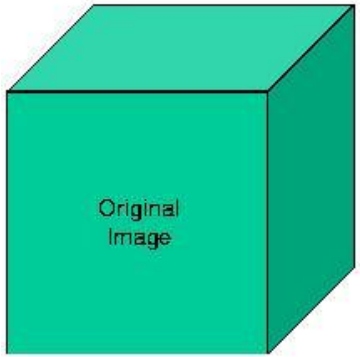


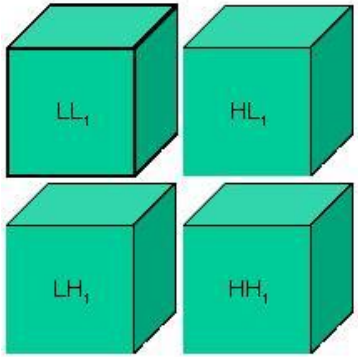
Daubechies D4 inverse wavelet transform

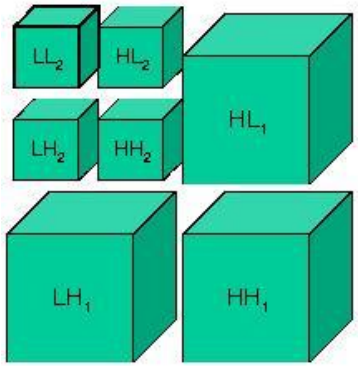
Wavelet als Verschmelzung der
klassischen Filter und
Fourieranalyse
mit der Idee
lokaler Basisfunktionen wie bei den B-Splines.

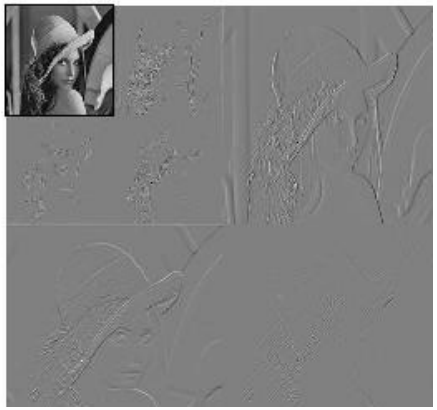
Haar-Filter ←→ Haar-Wavelet

- Orthogonale Wavelet,
- biorthogonale Wavelets,
- Frames,
- Lifting Scheme, …


Original
Image


LL1
HL1
LH1
HH1


LL2
HL2
LH2
HH2
HL1
LH1
HH1



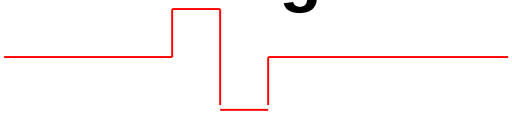
Fourier-Analyse
FFT

cos(kx), sin(kx)
Skalierung k
Alle gleichberechtigt
globaler Träger

keine Variations-möglichkeit
Kantenproblem
Volle Rekursion
Kosten O(n log(n))
Frequenz aus Fourierkoeffizienten

Multiskalen-Analyse
Wavelet

$W(2^j x-m)$, $\Phi(2^j x-m)$
Shift m, Skalierung 2^j
Mittelwert/Differenz
lokaler Träger



Wahl von Approximations-güte und Diff'barkeit
Lokal, adaptiv
Rekursion nur im Mittelwert
Kosten O(n)
Frequenz aus Waveletkoeffizienten