4.7 THE WAVELET TRANSFORM

Notation:

$\underline{\underline{\boldsymbol{Q}}}$	**Delay**	$\underline{\underline{\boldsymbol{Q}}}\,\underline{\boldsymbol{h}}(k)=\underline{\boldsymbol{h}}(k-1)$
$\underline{\underline{\boldsymbol{Q}}}^{-1}$	**Advance**	$\underline{\underline{\boldsymbol{Q}}}^{-1}\underline{\boldsymbol{h}}(k)=\underline{\boldsymbol{h}}(k+1)$
$(\downarrow 2)$	**Downsampling operator**	$(\downarrow 2)\underline{\boldsymbol{x}}=\underline{\boldsymbol{x}}_{even}=(x_0,x_2,x_4\ \mathrm{K})$ $(\downarrow 2)\underline{\underline{\boldsymbol{Q}}}^{-1}\underline{\boldsymbol{x}}=\underline{\boldsymbol{x}}_{odd}=(x_1,x_3,x_5\ \mathrm{K})$
$(\uparrow 2)$	**Upsampling operator**	$(\uparrow 2)\underline{\boldsymbol{x}}=(x_0,0,x_1,0,x_2,0\ \mathrm{K})$
$(\uparrow 2)(\downarrow 2)\underline{\boldsymbol{x}}=(x_0,0,x_2,0,x_4,0\ \mathrm{K})$ has z-transform		$\frac{1}{2}\left[X(z)+X(-z)\right]=X_{even}(z^2)$

Definitions and Relations:

$\underline{\boldsymbol{L}}=(\downarrow 2)\underline{\boldsymbol{C}}=(\downarrow 2)\sqrt{2}\underline{\boldsymbol{H}}_0$	**Lowpass analysis channel**	Filter and downsample
$\underline{\boldsymbol{B}}=(\downarrow 2)\underline{\boldsymbol{D}}=(\downarrow 2)\sqrt{2}\underline{\boldsymbol{H}}_1$	**Highpass analysis channel**	Filter has $\sum\underline{\boldsymbol{h}}_1(k)=0$
$\sqrt{2}\underline{\boldsymbol{F}}_0(\uparrow 2)$	**Lowpass synthesis channel**	Upsample and filter with $F_0(z)=H_1(-z)$
$\sqrt{2}\underline{\boldsymbol{F}}_1(\uparrow 2)$	**Highpass synthesis channel**	Upsample and filter with $F_1(z)=-H_0(-z)$
$\sum\underline{\boldsymbol{h}}_0(n)\underline{\boldsymbol{h}}_0(n+2k)=\delta(k)$ $H_0(z)H_0(z^{-1})+H_0(-z)H_0(-z^{-1})=2$		**Orthogonal lowpass**
$F_W(a,\tau)=\frac{1}{\sqrt{\lvert a\rvert}}\int_{-\infty}^{\infty}f(t)w\left(\frac{t-\tau}{a}\right)dt$		**Continuous time wavelet equation**
$\phi(t)$	from the **dilation equation**	$\phi(t)=\sum 2\underline{\boldsymbol{h}}(k)\phi(2t-k)$
$w(t)$	from the **wavelet equation**	$w(t)=\sum 2\underline{\boldsymbol{h}}_1(k)\phi(2t-k)$
$w_{jk}(t)=2^{j/2}w(2^jt-k)$	**Normalised wavelet on** $[k\Delta t,(k+N)\Delta t]$	

4.7.1 Introduction

The wavelet transform is very similar to short time Fourier transform (STFT). Both operates in time and frequency. Both create a two-dimensional matrix or space from a one-dimensional vector or function. More precisely, wavelet transform operates in time and *scale*. To look ahead, the most vulgar difference between the two is that while the STFTs frequency distribution is linear (all the absolute bandwidths are the same), wavelet transform has a *logarithmic frequency distribution*: all the **relative** bandwidths are the same, we speak about "*constant-Q*" analysis. Since the nature is logarithmic in many cases (just think of hearing, vision, etc.), presumably the wavelet transform describes it in a more effective way.

The most important field for wavelets is signal compression or coding for images and speech (the so-called subband coding). The HDTV standard is based on Fourier transforms. Wavelets were just too new for competing in that video race, but they have great chances for audio.

Other important applications are high-fidelity audio compression, ECG compression (**compression**); noise-removal, discontinuity detection, multiscale feature detection (**identification**); transmultiplexers, discrete multitone modulation transceivers (**telecommunication**) and adaptive filtering/regulation (**process control**).

Due to its localising abilities, wavelet transform also has the same application field as of STFT. Scalogram can be used instead of spectrogram in parameter-estimation and pattern-recognition problems of **non-stationary signals**. Wavelets provide an alternative to the classical STFT, Gabor Transform or the time-frequency analysis based on the Wigner-Ville distribution. For coding speech, the time variation is irregular and requires non-uniform intervals.

The idea of looking at a signal at various scales and analysing it with various resolutions has in fact emerged independently in many different fields of science. First results with wavelets were achieved by Alfred Haar in 1910. It goes to say that Haar did not call his function a *"wavelet"* in that time. In the mid-eighties, researchers of the "French-school", lead by a geophysicist, a theoretical physicist and a mathematician (namely, Morlet, Grossman and Meyer) built strong mathematical foundation around the subject and named their work "Ondelettes" (Wavelets). We also have to mention work of Daubechies and Mallat.

The wavelets are easier to understand or imagine through its correspondence with *musical notation*. In music, each note specifies a frequency and a position (and duration) in time. Its vertical placements gives frequency, its horizontal placement indicates time. A musical score is almost a wavelet decomposition — except that it has fractional jumps in frequency.

By wavelets, the scale corresponds to the pitch, and the time axis indicates where the sound starts and how long its duration. The value of the wavelet transform shows the amplitude of the signal of the given frequency at the given moment; it is the "loudness". You can see the beginning of the

Toccata and Fugue in D-minor from J. S. Bach and its wavelet transform. Try to see the similarity. It is not very difficult, is it? The blur comes from the long fade time (echo). The sound of the organ is almost a pure sine, thus there are no considerable harmonics.

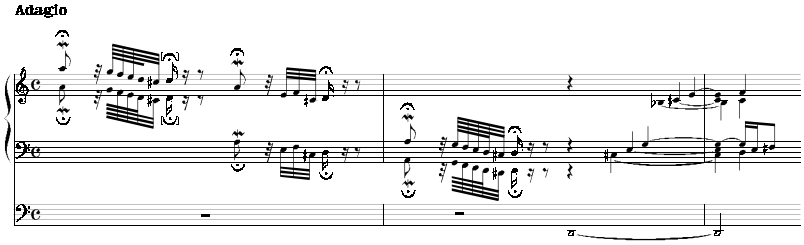


Figure 1.a. Score from the Toccata and Fugue in D minor of Johann Sebastian Bach. BWV 565

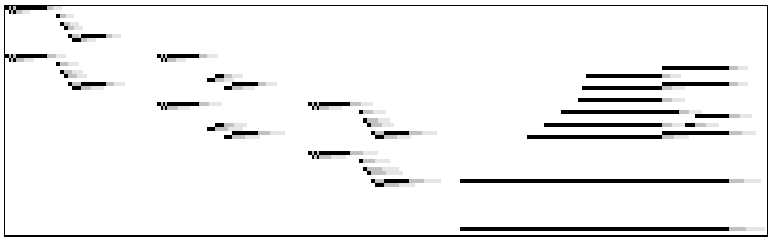


Figure 1.b. The corresponding wavelet transform of the score.
Vertical axis: scale (frequency), in $2^{1/12}$ steps. Horizontal axis: time, in 64ths.

If we made an STFT of the same score and same number of bands, the frequency resolution would be too coarse for the bass and unnecessarily fussy for high frequency components. In numbers: there would be 3-4 redundant empty bands between neighbouring halfnotes in the highest region, while there would be 4 halfnotes in the lowest band of STFT making them indistinguishable.

4.7.2 Wavelet Theory

The short-time Fourier transform aims localising the basis function in the signal. The windowing functions $g(t)$ achieve that by having good compact support — they can be Gabor windows with the Gaussian factor e^{-t^2}, or a simple rectangular window, or they can have finite length. The shifted window $g(t-\tau)$ is localised around time τ. Then the windowed transform:

$$F(\omega,\tau)=\frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)g(t-\tau)e^{-i\omega t}dt \quad . \tag{1}$$

Reconstruction from STFT:

$$f(t)=\frac{1}{\|g\|}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\omega,\tau)g(t-\tau)e^{i\omega t}d\omega d\tau \tag{2}$$

Instead of having a modulation and a windowing function, for wavelets, there is only one function, $w(t)$. The position variable τ still comes from the translation. But *instead of the frequency variable we have a scale variable.* Instead of modulating the wavelets we rescale them:

$$w_{a,\tau}(t)=\frac{1}{\sqrt{|a|}}w\left(\frac{t-\tau}{a}\right). \tag{3}$$

from what

$$F_W(a,\tau)=\frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty} f(t)w\left(\frac{t-\tau}{a}\right)dt \tag{4}$$

This is the equation for the ***Continuous Wavelet Transform*** **(CWT)**, or the **Integral Wavelet Transform**. The continuous wavelet transform is redundant, just as the windowed Fourier transform (STFT) was. If we select a good wavelet, it will be enough to know the CWT at a discrete set of scales and positions.

Important to note that the natural scale is *logarithmic.* Thus the differential is not da but da/a. The frequency window da is proportional to a (remember, this is why it is often called *constant-Q*). Convolution with $\frac{1}{a}w\left(\frac{1}{a}\right)$ gives a unitary operator, when we integrate with respect to logarithmic measure da/a. Now the **reconstruction** is:

$$f(t)=\frac{1}{C}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_W(a,\tau)\frac{1}{\sqrt{|a|}}w\left(\frac{t-\tau}{a}\right)\frac{da\,d\tau}{a^2} \tag{5}$$

The constant for normalising is $C = 2\pi \int |\hat{w}|^2 d\omega / |\omega|$. Effectively C is finite, when the transform of the wavelet is zero at $\omega = 0$. This means that *the integral of the wavelet is zero.* E.g. any smooth decaying function $w(t)$ is a mother wavelet for the CWT provided it has no DC component. In practice the compact support of $w(t)$ is needed in order to have good time localisation. $w(t)$ also can be real or complex, with the resulting wavelet transform being real or complex. In the latter case the complex conjugate of the basis function is used in the equations.

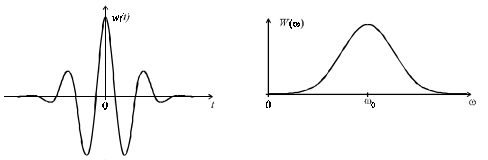
The signal $f(t)$ is supposed to be in $\mathbf{L}^2(\mathbf{R})$, i.e. it must be square integrable in order to being wavelet transformable. While the DC and pure sinusoid signals have no such property, all finite value functions with compact support are suitable. If we sample the signal for finite duration, this requirement is automatically fulfilled.

Some often used wavelet basis function with their Fourier transform:

Modulated Gauss (Morlet)

$$w(t) = e^{i\omega_0 t} e^{-t^2/2}$$

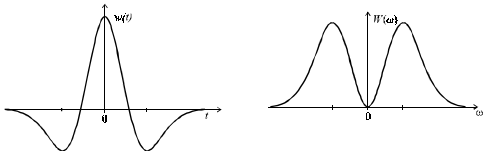
$$W(\omega) = \sqrt{2\pi}\, e^{-\frac{(\omega-\omega_0)^2}{2}}$$



Gauss second derivative

$$w(t) = \left(1 - t^2\right) e^{-t^2/2}$$

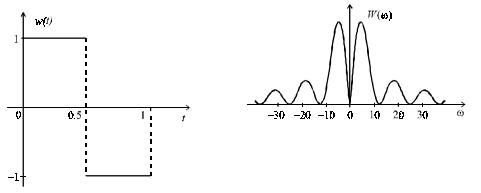
$$W(\omega) = \sqrt{2\pi}\, \omega^2 e^{-\omega^2/2}$$



Haar

$$w(t) = \begin{cases} 1 & \text{for } 0 \le t < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \le t < 1 \\ 0 & \text{otherwise} \end{cases}$$

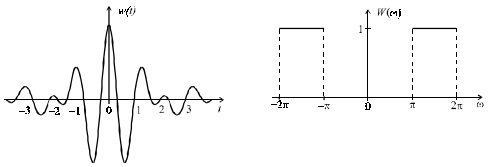
$$W(\omega) = i\, e^{-i\omega/2} \frac{\sin^2(\omega/4)}{\omega/4}$$



Shannon

$$w(t) = \frac{\sin(\pi t/2)}{\pi t/2} \cos\left(\frac{3\pi t}{2}\right)$$

$$W(\omega) = \begin{cases} 1 & \text{for } \pi \le |\omega| < 2\pi \\ 0 & \text{otherwise} \end{cases}$$



The wavelets of **Daubechies** (maxflat orthogonal with *2p* filter coefficients, used in identification) and **Mallat** (two-dimensional, used for image compression) are also very popular nowadays.

The orthonormal property of Haar and Shannon wavelets are easy to prove. These two function groups have opposing localisation ability. Haar wavelet is accurate in time but its spectrum differs from zero for $\omega \to \infty$, it has no compact support. Shannon wavelet has no compact support in time, therefore it can not localise precisely events in time. However, it acts good in frequency because it realises an ideal bandpass filter.

There are many other orthonormal wavelets between these two extreme, what have acceptable accuracy both in time and in frequency.

To see the power of wavelet transform, let us see some example! The scalogram shows sudden changes in the signal with characteristic pattern. These jumps create high-frequency components. Thus for small a (what means high frequency) the scalogram would have large values. For example, having a Dirac-delta at $t=t_0$:

$$CWT(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} w\left(\frac{t-\tau}{a}\right) \delta(t-t_0)\,dt = \frac{1}{\sqrt{a}} w\left(\frac{t_0-\tau}{a}\right) \tag{6}$$

If $a \to 0$, $|\,\mathrm{CWT}(a,\tau)| \to \delta(t_0-\tau)$, e.g. for small a's the place of the impulse is visible immediately. The use a complex wavelet basis brings further advantages detecting jumps. With modulated Gauss wavelet in this case

$$CWT(a,\tau) = \frac{1}{\sqrt{a}} w\left(\frac{t_0-\tau}{a}\right) = \frac{1}{\sqrt{a}} e^{-j\omega_0\left(\frac{t_0-\tau}{a}\right)} e^{-\left(\frac{t_0-\tau}{a}\right)^2}. \tag{7}$$

Its phase is

$$\theta = -\omega_0\left(\frac{t_0-\tau}{a}\right). \tag{8}$$

If we create its plot, one can see lines belonging to constant phase converging towards the impulse on the a-τ-plane. With other jumps this phenomena is not so much conspicuous, however, it is still enough to localise the disturbance. Of course, it also depends on the wavelet basis to be used.

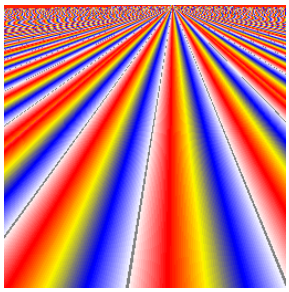


Figure 2. Phase plot of a Dirac-delta's WT at t_0=0.6.
[Horizontal axis: time 0 to 1. Verical axis: scale]

4.7.3 Implementations of the Wavelet Transform

There are four ways to view the integral operation (or the inner product) in the wavelet transform:

1. It computes the inner product or the cross-correlation of input signal $f(t)$ with scaled wavelet $w(t/a)/\sqrt{a}$, at time shift τ/a. It therefore computes the "similarity" between them, or the component of the signal that is "common" to the wavelet.
2. It is the output of a bandpass filter of impulse response $w(-t/a)/\sqrt{a}$, of input $f(t)$, at the instant τ/a.
3. It also computes the inner product or the cross-correlation of a scaled signal $f(at)$ with the wavelet $\sqrt{a}\,w(t)$, at time shift τ/a.
4. The same way, it is also the output of a bandpass filter of impulse response $\sqrt{a}\,w(-t)$, of input $f(at)$, at the instant τ/a.

From these points it is clear that the following two methods for computing the wavelet transform are equivalent. The first realisation (Figure 3.a) is simpler, assumed we can scale the signal $f(t)$. In discrete time and decimation by 2 it is easily done — this is the base of discrete FWT.

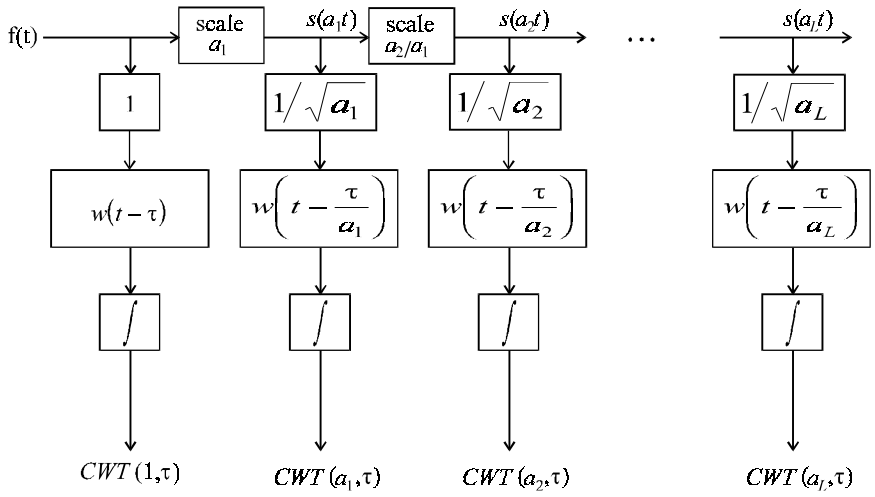


Figure 3.a. Computing FWT by scaling the signal

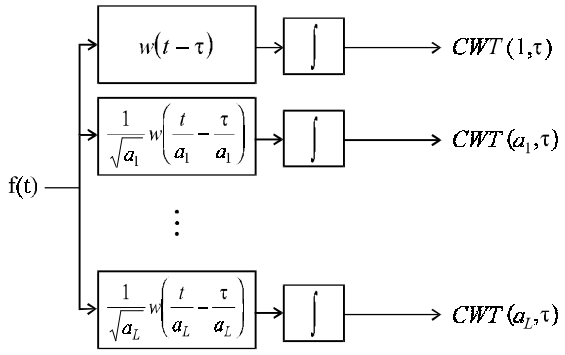


Figure 3.b. Filter bank implementation of the wavelet transform

Similarly to Fourier transform, wavelet transform also has different discrete versions:

Discrete parameter wavelet transform

$$DPWT(j,k) = a_0^{-j/2} \int_{-\infty}^{\infty} f(t) w\left(a_0^{-j} t - k\tau_0\right) dt \tag{9}$$

Discrete time wavelet transform

$$DTWT(j,k) = a_0^{-j/2} \sum_r f(r) w\left(a_0^{-j} r - k\tau_0\right) \tag{10}$$

Discrete wavelet transform

$$DWT(j,k) = 2^{-j/2} \sum_r f(r) w\left(2^{-j} r - k\right) \tag{11}$$

If $w(k)$ are samples of a continuous function, DWT also computes the DTWT. Generally, $w(k)$ does not need to have a continuous version. We will get the same (11) equation later from the discrete time multiresolution, using another approach.

4.7.4 The Discrete Wavelet Transform

Multilevel Transforms by Recursion

A key idea for wavelets is the concept of "*scale*". Sums and differences of neighbours are at the finest scale. We move to a larger scale by taking sums and differences again. This is recursion — the same transform at a new scale. It leads to a **multiresolution** of the original signal. The key point is the multilevel construction, which is clear in a flow-graph. Suppose, we have a four-element $\underline{x}$ input to transform, from what we calculate sums and differences to obtain $\underline{y}$. In the next level we make the same to get $\underline{z}$.

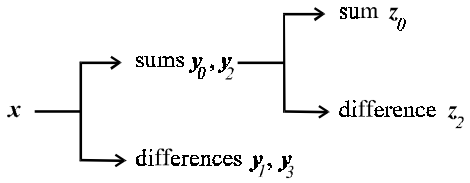


Figure 4. Multilevel by recursion

This two-step wavelet transform is executed by a product of two matrices:

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

which is invertible, so is the transform invertible. To draw the flow-graph of the inverse, just reverse the arrows. The wavelet transform becomes unitary, when we divide all entries by $\sqrt{2}$.

Perhaps one more transform could be mentioned. It is the Walsh-Hadamard transform, which iterates *also on the differences* y_1 *and* y_3. Instead of the "logarithmic" or "dyadic" tree for wavelets, we have a complete binary tree for Walsh-Hadamard, e.g. we calculate all possible independent sums and differences:

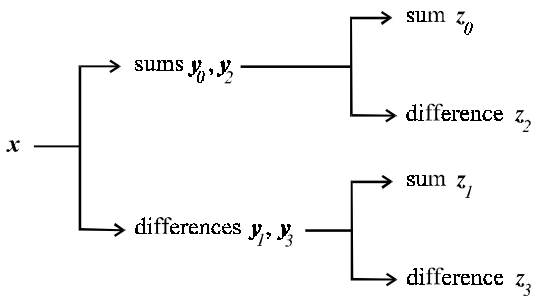


Figure 5. Complete binary tree of Walsh-Hadamard Transform

This also counts as a **wavelet packet** (those include every binary tree). You would not want to miss the "Hadamard matrix" for this transform, which is exceptional. All entries are 1 and –1 and all rows are orthogonal:

$$\begin{bmatrix} 1 & & 1 & \\ & 1 & & 1 \\ 1 & & -1 & \\ & 1 & & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \end{bmatrix}=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix},$$ which is symmetric and

when divided twice by $\sqrt{2}$, this matrix is also orthonormal.

Do not forget the disadvantage compared to wavelets! All sixteen entries are non-zero. The complete tree takes more computation than the wavelet tree. For signals of length N (a power of 2), the Hadamard matrix has N^2 nonzeros and the wavelet matrix has only $N + N\log_2 N$. Those are the costs *without recursion*, jumping from the original $\underline{x}$'s to the final $\underline{z}$'s.

When the computations are recursive, as they always should be, we have a product of very sparse matrices. The flow-graph gives $2N\log_2 N$ for the complete Walsh-Hadamard transform (*exactly matching the FFT*). This is good but wavelets are slightly better. The transform computes the sum and difference of $N-1$ pairs. This needs only $2N-2$ calls to memory.

Wavelets in Discrete Time

Wavelets are localised waves. Instead of oscillating forever, they drop to zero. They come from the **iteration** of filters (with rescaling). The link between discrete-time filters and continuous-time wavelets is in the limit of a logarithmic filter-tree. In each level, we separate the signal with a low- and a highpass filter after downsampling, thus we get the *logarithmic frequency distribution*.

Downsampling (decimation) means that we drop every second sample. It transposes the signal to twice as high frequency range. Then if we use the same filter coefficients, **relatively** it will have the same effect as if we

halved filter cut-off frequencies at each level — this corresponds to the new scale parameter:

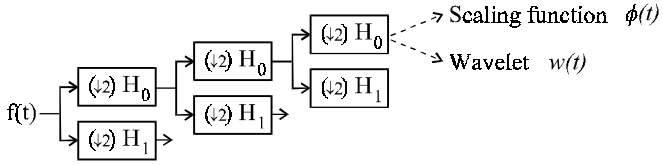


Figure 6. Scaling function and wavelets from iteration of the lowpass filter

The parallels between multiresolution in continuous time and a filter tree in discrete time are almost perfect. Multiresolution uses longer and longer wavelets, as we climb the pyramid. The filter bank separates lower and lower frequencies, as we iterate. We are speaking of the analysis half, where inputs are separated by *scale*.

Continuous time	**Discrete time**
multiresolution	filter bank tree
rescaling $t \to 2t$	downsampling $\omega \to \frac{\omega}{2}$
averaging with $\phi(t)$	lowpass filter
averaging with $w(t)$	highpass filter
orthogonal bases	orthogonal matrices
wavelet coefficients	analysis bank output
sum of wavelet series	synthesis bank output
fast wavelet transform	product of filter matrices

Scaling functions and wavelets have remarkable properties. They inherit orthogonality, or biorthogonality, from the filter bank. Because of the repeated rescaling that products them, wavelets decompose a signal into details at all scales. The wavelet $w(t)$ and its shifts $w(t-k)$ are at unit scale. The wavelets $w(2^j t)$ and $w(2^j t - k)$ are at scale 2^{-j}. The biorthogonal $\tilde{\phi}(t)$ and $\tilde{w}(t)$ come from iterating the synthesis bank.

Wavelets produce a natural multiresolution of every image, including the all-important edges. Where the low frequency part of the Fourier transform is often a blur, the output from the lowpass channel is a useful compression.

Wavelets are basis functions $w_{jk}(t)$ in continuous time. The special feature of the wavelet basis is that all functions $w_{jk}(t)$ (the so-called "baby-wavelets") are constructed from a single mother wavelet $w(t)$. This wavelet is a small wave (a pulse). Normally it starts at time $t=0$ and ends at time $t=N$.

A typical wavelet w_{jk} is compressed j times and shifted k times. Its formula is

$$w_{jk}(t) = w\left(2^j t - k\right), \tag{12}$$

or in normalised form:

$$w_{jk}(t) = 2^{j/2} w\left(2^j t - k\right). \tag{13}$$

For continuous parameters $w_{a\tau}(t) = \frac{1}{\sqrt{a}} w\left(\frac{t-\tau}{a}\right)$. (It lead to the continuous wavelet transform, CWT).

As we describe the connection between filter banks and wavelets, you will see that it is the "*highpass filter*" that leads to $w(t)$. The "*lowpass filter*" leads to a scaling function $\phi(t)$. In most cases the lowpass filter comes first — **the scaling function is obtained before the wavelet**. In fact the scaling function (in continuous time) comes from infinite repetition **L L ... L** of the lowpass filter, with rescaling each iteration. The wavelet follows from $\phi(t)$ by just *one* application of the highpass filter.

Multiresolution

At a given resolution of a signal or an image, the scaling functions $\phi\left(2^j t - k\right)$ are a basis for the set of signals. The new details at level j are represented by the wavelets $w\left(2^j t - k\right)$. Then the smooth signals plus the details, the ϕ's and w's, combine into a **multiresolution** of the signal at the finer level $j + 1$. Averages come from the scaling functions, details come from the wavelets:

signal at level j (local averages) + details at level j (local differences) → signal at level $j+1$

The signal is divided into different ***scales*** of resolution, rather than different frequencies. Multiresolution divides the frequencies into octave bands, from ω to 2ω, instead of uniform bands from ω to $\omega + \Delta\omega$. This matching of long time with low frequency and short time with high frequency occurs in a natural way for wavelets. It is one of the attracts of a wavelet decomposition.

Scaling Function and Wavelets

We will now produce a continuous-time scaling function $\phi(t)$ and a wavelet $w(t)$ from the filter coefficients. The dilation equation for the scaling function $\phi(t)$ is

$$\phi(t) = \sqrt{2} \sum_{k=0}^{N} \underline{c}(k) \phi(2t - k). \tag{14}$$

In terms of the original lowpass coefficients $\underline{\boldsymbol{h}}(k)$, the extra factor is 2:

$$\phi(t)=2\sum_{k=0}^{N}\underline{\boldsymbol{h}}(k)\phi(2t-k). \tag{15}$$

This involves $\phi(t)$ in continuous time, and a set of coefficients $\underline{\boldsymbol{c}}(k)$ or $\underline{\boldsymbol{h}}(k)$ from discrete time. The presence of both t and $2t$ is the key. Without the 2 we would have an ordinary constant-coefficient equation (look for exponential solutions $e^{\lambda t}$). With two scales, there are major changes:

1. There may or may not be a solution $\phi(t)$.
2. The solution is zero outside the interval $0\le t<N$.
3. The solution seldom has an elementary formula.
4. The solution is not likely to be a smooth function.

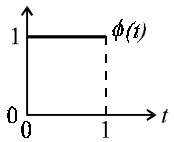
Formally, we can find an expression for the Fourier transform of $\phi(t)$. It is an infinite product. The inverse Fourier transform yields $\phi(t)$ as "distribution" — not necessarily continuous, possibly involving delta functions.

Taking Haar's filter with $2\underline{\boldsymbol{h}}(0)=1$ and $2\underline{\boldsymbol{h}}(1)=1$, the dilation equation is

$$\phi(t)=\phi(2t)+\phi(2t-1). \tag{16}$$

The graph of $\phi(t)$ *is compressed by* 2, to give the graph of $\phi(2t)$. It is *shifted right by* $\frac{1}{2}$, it becomes the graph of $\phi(2t-1)$. The dilation equation is fulfilled when $\phi(t)$ is the *box function*:

$$\phi(t)=\begin{cases}1 & \text{for } 0\le t<1\\ 0 & \text{otherwise.}\end{cases}$$



The graphs of $\phi(2t)$ and $\phi(2t-1)$ are half-size boxes. Their sum is the full-size box $\phi(t)$.

The dilation equation is linear, so any multiple of the box function is also a solution. It is convenient to normalise so that *the integral of* $\phi(t)$ *from* $-\infty$ *to* ∞ *equals one*. Integrating both sides of the dilation equation, we get:

If $\int_{-\infty}^{\infty}\phi(t)\,dt=1$ *then* $2\underline{\boldsymbol{h}}(0)+2\underline{\boldsymbol{h}}(1)+\cdots+2\underline{\boldsymbol{h}}(N)=2$, i.e. the sum of the coefficients is one. This is our lowpass filter convention.

Note: the difference between the normalising constants 1, $\sqrt{2}$ and 2 is quite clear. The *filter bank* has a factor $\sqrt{2}$ to account for the downsampling step.

There are half as many components and half as many terms in *energy*. To compensate, we need this factor. For the *dilation equation*, we preserve *area*. Rescaling the time to half requires renormalising the coefficients by 2.

Important: There are filters in which $H(0)$ is not exactly 1. If it is near enough, this still deserves the name "lowpass filter". (Just think of Tchebishev's approximation with even order.) *Such filters do not lead to wavelets!* There is a **must** on $H(0)=1$ and $H(\pi)=0$ for wavelets. Its reason will be proven at the Continuous Wavelet Transform.

The Wavelet Equation

The equation for the wavelet involves the highpass coefficients $\underline{\boldsymbol{d}}(k)$. It is a direct equation that gives $w(t)$ immediately and explicitly from $\phi(t)$:

$$w(t) = \sqrt{2}\sum \underline{\boldsymbol{d}}(k)\phi(2t-k). \tag{17}$$

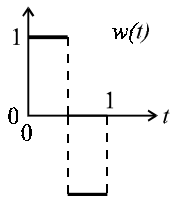
In terms of the original coefficients $\underline{\boldsymbol{h}}_1(k)$, the factor $\sqrt{2}$ becomes 2:

$$\textbf{Wavelet equation} \qquad w(t) = 2\sum \underline{\boldsymbol{h}}(k)\phi(2t-k). \tag{18}$$

In our example, $\phi(t)$ is a box function and its dilations $\phi(2t-k)$ are half-boxes. Then the wavelet is a difference of half-boxes:

$$w(t) = \phi(2t) - \phi(2t-1), \text{ explicitly}$$

$$w(t) = \begin{cases} 1 & \text{for } 0 \le t < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \le t < 1 \\ 0 & \text{otherwise.} \end{cases}$$



This is the ***Haar wavelet***. A wavelet is a small wave. Haar wavelet is zero outside a bounded interval (this is true for almost all other wavelets): ***compact support*** *corresponds to FIR.*

Important: For other wavelets, testing orthogonality is not desirable and usually not possible. We will not have elementary formulas for $\phi(t)$ and $w(t)$. Instead, we will know the coefficients $\underline{\boldsymbol{c}}(k)$ in the dilation equation and $\underline{\boldsymbol{d}}(k)$ in the wavelet equation. ***All information about $\phi(t)$ and $w(t)$ — their support interval, their orthogonality, their smoothness, and their vanishing moments — will be determined by and from the $\underline{c}$'s and $\underline{d}$'s.***

Wavelets from Filters

The previous sections reached the dilation equation and the wavelet equation. Those equations are the crucial connections between wavelets and filters. Historically, their development was separate. Now you have to see them together. The lowpass filter $\underline{c}(0), \ldots, \underline{c}(N)$ determines the scaling function $\phi(t)$. Then the highpass coefficients produce the wavelets.

Working with $\phi(t)$ and $w(t)$, we really have three basic jobs:

1. Connect the *properties* of $\phi(t)$ and $w(t)$ to properties of the $\underline{c}$'s and $\underline{d}$'s.
2. Compute the *coefficients* in $f_i(t) = \sum_k a_{jk}\phi_{jk}(t)$ and $f(t) = \sum_j \sum_k b_{jk} w_{jk}(t)$.
3. *Construct* $\phi(t)$ by actually solving the dilation equation.

Theorem: *If the coefficients $\underline{c}(k)$ and $\underline{d}(k)$ come from an orthogonal filter bank, so they have double-shift orthogonality, then*

a. The scaling functions $\phi(t-n)$ are orthonormal to each other:

$$\int_{-\infty}^{\infty} \phi(t-n)\phi(t-m)\,dt = \delta(m-n) \tag{19}$$

b. The scaling functions are orthogonal to the wavelets:

$$\int_{-\infty}^{\infty} \phi(t-m)w(t-n)\,dt = 0 \tag{20}$$

c. The wavelets $w_{jk}(t) = 2^{j/2}w(2^j t - k)$ at all scales are orthonormal:

$$\int_{-\infty}^{\infty} w_{jk}(t)w_{JK}(t)\,dt = \delta(j-J)\delta(k-K). \tag{21}$$

To find the recursion for coefficients, shift equations by k and set $n = 1 - 2k$:

$$\textbf{\textit{Dilation equation:}}\quad \phi(t-k) = \sum \sqrt{2}\underline{c}(n)\phi(2t-2k-n) = \sum \underline{c}(1-2k)\phi_{11}(t) \tag{22}$$

$$\textbf{\textit{Wavelet equation:}}\quad w(t-k) = \sum \sqrt{2}\underline{d}(n)\phi(2t-2k-n) = \sum \underline{d}(1-2k)\phi_{11}(t) \tag{23}$$

Multiply by $f_1(t)$ and integrate with respect to t. Since the basis functions are orthonormal, the integral gives the coefficients of $f_1(t)$ in each basis:

$$a_{0k} = \sum \underline{c}(1-2k)a_{11} \text{ and } b_{0k} = \sum \underline{d}(1-2k)a_{11}. \tag{24}$$

This is the key to recursion. It is the action of a filter bank, which inputs $a_{1\mathbb{1}}$ and outputs a_{0k} and b_{0k}. But we have to watch indices, because an ordinary convolution would be $\sum \underline{c}(k-\mathbb{1})a_{1\mathbb{1}}$ and downsampling would give $\sum \underline{c}(2k-\mathbb{1})a_{1\mathbb{1}}$. There is a time reversal between this filter $\underline{\underline{C}}$ and the transpose filter $\underline{\underline{C}}^T$ that appears in the recursion: $\underline{c}^T(n)=\underline{c}(-n)$ and $\underline{d}^T(n)=\underline{d}(-n)$.

Going between levels of a multiresolution is *subband filtering* with $\underline{\underline{C}}^T$ and $\underline{\underline{D}}^T$:

Theorem:

$$a_{jk}=\sum_{\mathbb{1}}\underline{c}(\mathbb{1}-2k)a_{j+1,\mathbb{1}} \text{ and } b_{jk}=\sum_{\mathbb{1}}\underline{d}(\mathbb{1}-2k)a_{j+1,\mathbb{1}}. \tag{25}$$

In vector notation this is $\underline{a}_j=(\downarrow 2)\underline{\underline{C}}^T\underline{a}_{j+1}$ *and* $\underline{b}_j=(\downarrow 2)\underline{\underline{D}}^T\underline{a}_{j+1}$. *The pyramid is*

$$a_{j+1}\xrightarrow{C^T}a_j\xrightarrow{C^T}a_{j-1}\cdots a_1\xrightarrow{C^T}a_0,\quad a_{j+1}\xrightarrow{D^T}b_j,\ a_j\xrightarrow{D^T}b_{j-1},\ a_1\xrightarrow{D^T}b_0$$

Theorem: $a_{j+1,\mathbb{1}}$ *comes from* a_{jk} *and* b_{jk} *by a* ***synthesis*** *filter bank:*

$$a_{j+1,\mathbb{1}}=\sum \boldsymbol{c}(2k-\mathbb{1})a_{jk}+\sum \boldsymbol{d}(2k-\mathbb{1})b_{jk}. \tag{26}$$

The inverse pyramid is the ***fast inverse wavelet transform****:*

$$a_0\xrightarrow{C}a_1\xrightarrow{C}a_2\cdots a_j\xrightarrow{C}a_{j+1},\quad b_0\xrightarrow{D}a_1,\ b_1\xrightarrow{D}a_2,\ b_j\xrightarrow{D}a_{j+1}$$

A possible solution for the dilation equation comes from piece-wise constant approximation, the so-called ***cascade algorithm***. The pieces become shorter (their length is 2^{-i}) with the levels. If $\phi^{(i)}(2t-n)$ converges suitably to a limit $\phi(t)$, then this limit function solves the dilation equation. In this case we can make the following statements (they are not quite two-way implications!):

Convergence of $\phi^{(i)}(t)$ to $\phi(t)$ needs $H=0$ at $z=-1$

Orthogonality of $\phi(t-k)$ needs $H(z)H\left(z^{-1}\right)$ to be halfband.

4.7.5 Wavelet Transform by Multiresolution

The wavelet transform operates in continuous time (on functions) and in discrete time (on vectors). The input is $f(t)$ or $\underline{\boldsymbol{x}}(n)$. The output is the set of coefficients b_{jk}, which express the input in the wavelet basis. For functions and infinite signals, this basis is necessarily infinite. In continuous time the coefficients involve integrals of $f(t)$ times $w_{jk}(t)$. For finite length vectors with L components, there will be L basis vectors and L coefficients. Thus the discrete wavelet transform is expressed by an L by L matrix. To calculate coefficients b_{jk}, we are solving a linear system. The inverse transform involves the inverse matrix.

Then we show how the b_{jk} can be found recursively. Levels j and $j-1$ are connected. This reorganises the matrix multiplication. The discrete wavelet transform (DWT) becomes the *fast* wavelet transform (FWT). The central idea in this fast recursion is *multiresolution*.

First come the two directions, synthesis and analysis, for an orthonormal basis:

Synthesis of a function: $$f(t) = \sum_{j,k} b_{jk} w_{jk}(t) \tag{27}$$

Analysis of a function: $$b_{JK} = \int_{-\infty}^{\infty} f(t) w_{JK}(t)\, dt\,. \tag{28}$$

In the matrix case, the wavelets are ordinary vectors. They go into the columns of the ***wavelet matrix S***. Each wavelet vector has a position in time given by k and a position in scale given by j.

Synthesis in discrete time: $$\underline{\boldsymbol{x}} = \underline{\underline{\boldsymbol{S}}}\,\underline{\boldsymbol{b}} \tag{29}$$

Analysis in discrete time: $$\underline{\boldsymbol{b}} = \underline{\underline{\boldsymbol{A}}}\,\underline{\boldsymbol{x}} \tag{30}$$

In general the two matrices are related by $\underline{\underline{\boldsymbol{A}}} = \underline{\underline{\boldsymbol{S}}}^{-1}$. For Haar and all orthonormal wavelets $\underline{\underline{\boldsymbol{A}}} = \underline{\underline{\boldsymbol{S}}}^{T}$, the columns of $\underline{\underline{\boldsymbol{S}}}$ are the same as the rows of $\underline{\underline{\boldsymbol{A}}}$.

Tree-Structured Filter Bank

We move from transform towards *fast* transforms. The wavelet basis and the Fourier basis have special properties, beyond orthogonality. The scales j and $j-1$ are closely related, just as frequencies ω and $\omega/2$ are closely related. By taking the advantage of these relations, the multiplications by $\underline{\underline{\boldsymbol{S}}}$ and $\underline{\underline{\boldsymbol{S}}}^{T}$ can be reorganised. We will explain the special properties, and then derive fast transforms (starting with Haar wavelets).

The recursive nature of wavelets is clearest when we construct a tree of filter banks (Figure 7). The highpass filter $\underline{\underline{\boldsymbol{D}}}$ computes differences of the input. The downsampling step symbolised by ($\downarrow$2) keeps the even-numbered

differences $\left(\boldsymbol{x}(2k)-\boldsymbol{x}(2k-1)\right)/\sqrt{2}$. These are final outputs because they are not transformed again. They are at the end of their branch, in this "logarithmic tree". These outputs b_{jk} are at the fine-mesh level. The factor $r=1/\sqrt{2}$ is included to produce unit vectors in the matrix $\underline{\underline{\boldsymbol{S}}}$, below.

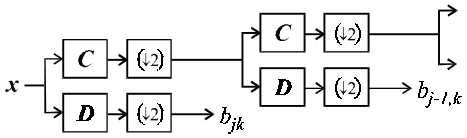


Figure 7. The logarithmic tree that leads to wavelets

The lowpass filter $\underline{\underline{\boldsymbol{C}}}$ computes averages. Again the (↓2) step keeps the even samples. These averages are not final outputs, because they will be filtered again by $\underline{\underline{\boldsymbol{D}}}$ and $\underline{\underline{\boldsymbol{C}}}$. The averages and differences of all levels follow the fundamental recursion:

Averages (lowpass filter) $$a_{j-1,k}=\tfrac{1}{\sqrt{2}}\left(a_{j,2k}+a_{j,2k+1}\right) \tag{31}$$

Differences (highpass filter) $$b_{j-1,k}=\tfrac{1}{\sqrt{2}}\left(a_{j,2k}-a_{j,2k+1}\right) \tag{32}$$

These filters are anti-causal. ***There is a time-reversal here.*** That requirement is built in to convolutions and inner products.

You must see the finite case in terms of matrices. With length $L=4$, there are two fine differences, one coarse difference, and the overall average. The columns of $\underline{\underline{\boldsymbol{S}}}$ are the basis vectors for the Haar wavelets:

$$\begin{bmatrix}1&1&1&\\1&1&-1&\\1&-1&&1\\1&-1&&-1\end{bmatrix} \text{ is scaled by } r=\frac{1}{\sqrt{2}} \text{ to } \underline{\underline{\boldsymbol{S}}}=\begin{bmatrix}r^2&r^2&r&0\\r^2&r^2&-r&0\\r^2&-r^2&0&r\\r^2&-r^2&0&-r\end{bmatrix}. \tag{33}$$

The scaling gives unit vectors in the columns and rows, because $2r^2=1$ and $4r^4=1$. The inverse matrix appears in analysis, as we create the tree of averages and differences:

$$\underline{\underline{\boldsymbol{A}}}=\underline{\underline{\boldsymbol{S}}}^{-1}=\underline{\underline{\boldsymbol{S}}}^{T}=\begin{bmatrix}r^2&r^2&r^2&r^2\\r^2&r^2&-r^2&-r^2\\r&-r&0&0\\0&0&r&-r\end{bmatrix}. \tag{34}$$

The first row gives the overall average a_{00}. The second gives b_{00}. The third and fourth rows give the finer differences, coming earlier in the tree of filters. The transform $\underline{\boldsymbol{b}}=\underline{\underline{\boldsymbol{A}}}\,\underline{\boldsymbol{x}}$ contains these four numbers.

Fast Wavelet Transform (FWT)

The tree of filters is the fast wavelet transform! The FWT expresses the analysis matrix $\underline{\underline{A}}$ as a product of simple average-difference matrices, coming from the downsampled filters in the tree:

$$\underline{\underline{A}} = \begin{bmatrix} r & r & & \\ r & -r & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} r & r & & \\ & & r & r \\ r & -r & & \\ & & r & -r \end{bmatrix} \tag{35}$$

The matrix on the right is first in the tree. The coefficients r, r are in the lowpass filter, and downsampling leaves matrix $\underline{\underline{L}}$ in the top of two rows. The rows of $\underline{\underline{L}}$ have a double shift. The coefficients r, $-r$ come from highpass (or bandpass) filter, downsampled to leave $\underline{\underline{B}}$. ***There is again a time-reversal from –r, r.*** The pattern you see in the 4 by 4 Haar matrix is the product in (35):

$$\textit{Analysis tree:} \qquad \underline{\underline{A}} = \begin{bmatrix} \underline{\underline{L}} & \\ \underline{\underline{B}} & \\ \hline & \underline{\underline{I}} \end{bmatrix} \begin{bmatrix} \underline{\underline{L}} \\ \underline{\underline{B}} \end{bmatrix}. \tag{36}$$

This pattern applies to transforms of all lengths $L = 2^j$. ***It will apply to all wavelets!*** The fast transform expresses $\underline{\underline{A}}$ (and later $\underline{\underline{S}}$) using matrices with many zeros. It is the matrix form of the *pyramid algorithm*.

For length $L = 2^j$, there will be J levels in the tree and J matrices in $\underline{\underline{A}}$. The matrix on the right, from the start of the tree, has two nonzeros in each row. Filters with T coefficients will produce T nonzeros in each row. The finest factor has TL non-zero entries.

The next factor has $TL/2$ coefficients in the top half, processing a shorter input. (The identity matrix in the lower right costs nothing. It leaves the differences alone.) The third stage of the tree has $TL/4$ coefficients. The total for the FWT (= factored form of $\underline{\underline{W}}^{-1}$) comes from J factors:

$$TL\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{J-1}}\right) = TL\sum_{i=0}^{J-1}\frac{1}{2^i} < 2TL \tag{37}$$

The same count applies to synthesis step $\underline{x} = \underline{\underline{S}}\underline{b}$. The factors of $\underline{\underline{S}}$ give the inverse of equation (35). The synthesis tree has the inverse matrices in opposite order:

$$\underline{\underline{S}} = \begin{bmatrix} r & & r & \\ r & & -r & \\ & r & & r \\ & r & & -r \end{bmatrix} \begin{bmatrix} r & r & & \\ r & -r & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \tag{38}$$

The matrix on the right inverts the last analysis filter (still with time-reversal). It is the first step in the synthesis tree. The product of J matrices is the whole synthesis bank, which reconstructs $\underline{x}$ in Figure 8.

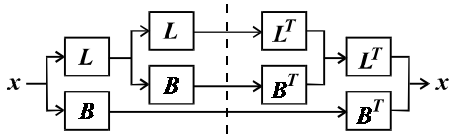


Figure 8. Analysis bank followed by synthesis bank: $\underline{\underline{S}}\,\underline{\underline{A}}\underline{x} = \underline{x}$.
The matrices are factored by the tree, producing the FWT.

Wavelet Packets and the Best Basis

Previously only the lowpass filter was iterated. It was assumed that lower frequencies contained more important information than higher frequencies. For many signals this is not true. A *wavelet packet basis allows any dyadic tree structure* (Figure 9). At each point in the tree we have the option to send the signal through the lowpass-highpass filter bank, ***or not***.

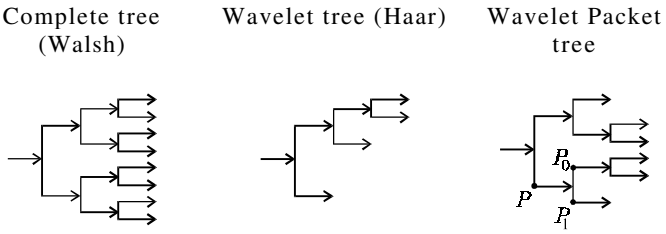


Figure 9. Each wavelet packet tree yields a basis, including Walsh and Haar

One possibility is the *logarithmic tree*, with lowpass iteration only. Another possibility is the *complete tree*, analogous to the Short Time Fourier Transform (STFT). Haar iterates only the lowpass filter, Walsh iterated both filters. Wavelet packets make up the entire family of bases. Each one is associated with a particular *quadtree*, because it comes from splitting into two (or nor splitting) at each step. The decision to split or to merge should be aimed at achieving minimum distortion D — subject to cost and capacity constraints on the rate R.

The wavelet packet bases are given by simple recursions (not by simple formulas). They inherit the properties of the filter bank — orthogonality or biorthogonality.

We emphasise the recursive form of every wavelet packet decomposition. The coefficient sequence at a point P is treated in exactly the same way as the original sequence $\boldsymbol{x}(n)$ at the original root of the tree. The packet splits or not. If it splits then the two new branches end in two new decision points.

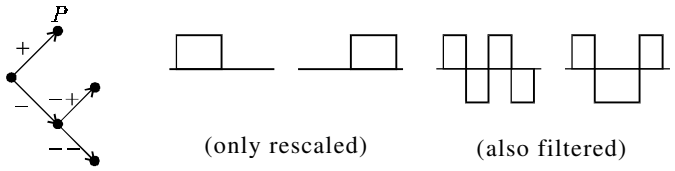


Figure 10. Creating wavelet packets

There is a corresponding Fourier packet. Its decision is whether to break the interval in two. Instead of a fixed length 8 for all DFT blocks, and 8×8 for an image, the splitting decision is made locally — depending on the signal. The "butterfly" in the FFT is like the lowpass-highpass bank for wavelets. But the FFT admits all frequencies 0, 1, 2, 3, ... where the wavelets are dyadic and octave-based (1, 2, 4, 8, ...).

Multidimensional Filters and Wavelets

Images are two-dimensional. Processing images is an extremely important application of subband filtering. We certainly need two-dimensional filters! Their construction can be extremely easy or extremely hard — depending on the type of filter:

1. *Separable*: Products of one-dimensional filters (easy).
2. *Non-separable*: Genuinely two-dimensional filters (hard).

Each filter bank is associated with a subsampling matrix $\underline{\underline{\boldsymbol{M}}}$. In d dimensions this matrix is $d \times d$. In one dimension it contains a usual number M. In two dimensions we consider the two leading possibilities:

$$\text{(separable) } \underline{\underline{\boldsymbol{M}}}_s = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad \text{(non-separable quincunx) } \underline{\underline{\boldsymbol{M}}}_q = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix},$$

These matrices have eigenvalues $|\lambda| > 1$, as required. Their determinates are M_s=4 and M_q=2. The light-face symbol M still represents the number of channels. M is also the number of scaling functions plus wavelets.

Figure 11 shows the lattice of integers and the quincunx sublattice of samples:

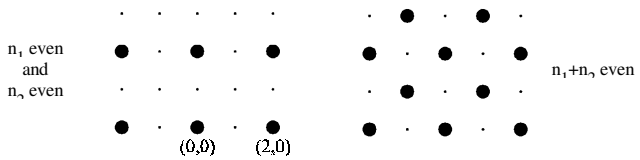


Figure 11. The sampling lattices for separable and quincunx

Notice how M_s=4 separable sublattices cover the whole lattice. Similarly M_q=2 quincunx sublattices (*staggered grids*) cover all mesh points. The separable lattices are strongly oriented in the horizontal and vertical directions! The quincunx lattice has an extra symmetry at 45° and –45° . It is closer to isotrophic, meaning independent of direction. A diagonal edge in an image will be captured far better by the quincunx. But those filters are harder to design, if we want PR and especially if we want orthogonality.

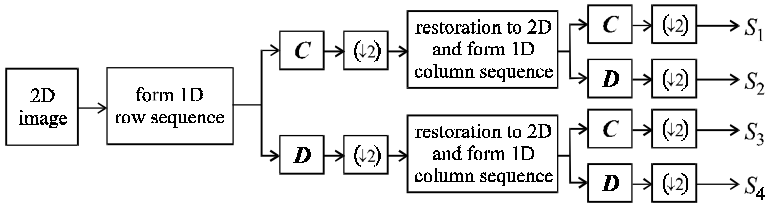


Figure 12. A possible realisation of two-dimension MRA

Wavelets and Image Compression

Just for comparison: at compression rate of 15:1 and 20:1 the blocking from the DCT is sever with the widely-spread JPEG compression. Lines that are separated in the true image are found to merge during compression. It does not happen for wavelets. Experiments indicate that the improvement over the JPEG algorithm (the DCT in 8 × 8 blocks) increases with the compression ratio. At high compression, the blocking effects in the JPEG output overwhelm the signal.

The difficulty is to measure the distortion that our eyes actually perceive. It is not the L^2 norm! Energy is a convenient measure of error, but it does not agree with human perception.

Subband coding uses wavelets (this means tree-structured filter banks) avoids blocking at medium bit-rate, because its basis functions have variable length. Long basis functions represent flat background (low frequency). Short basis functions represent regions with texture. At low bit-rate, wavelet coding suffers from *ringing* when high frequencies (textures) are deleted. The ringing artifacts are significant around edges with high intensity. They have the shape of the basis functions that are emphasised in synthesis.

4.7.6 Parallel Implementations

Since DWT requires intensive computations, several architectural solutions using special purpose parallel processors have been proposed in order to meet the real time requirement in many applications. The solutions include parallel filter architecture — as it was shown in Figure 3.b. —, SIMD linear array architecture, SIMD multigrid architecture [Vish], [ChVO], 2-D block based architecture [Kang], and the AWARE's wavelet transform processor (WTP) [Awa].

The first three architectures are special purpose parallel processors that implement the high-level abstraction of the pyramid algorithm by brute force parallelism. The two-dimensional (2-D) block-based architecture is a VLSI implementation that uses four multiply and accumulate (MAC) units to execute the forward and inverse transforms. It requires a small on-chip memory and implements 2-D wavelet transform directly without data transposition. However, this feature can be a drawback in certain applications. In addition, the block-based architecture may introduce block boundary effects degrading the visual quality.

The Aware's WTP is capable of computing forward and inverse wavelet transforms for 1-D input data using a maximum of six filter coefficients. It can be cascaded to execute transforms using higher order filters. The WTP has been clocked of 30 MHz and offers 16 bit precision. The DWT computation is executed in a synchronous pipeline fashion and is under complete user control. However, the Aware's WTP is a complex design requiring extensive user control. Programming such a device is therefore tedious, difficult, and time consuming.

Another approach is to use an efficient systolic array architecture [Grze] for computing DWT. The VLSI architecture [GMPY] computes both high- and low-pass frequency coefficients in the same clock cycle and thus has an efficient hardware utilisation. The design is simple, modular, and cascadable for computation of 1-D or 2-D data streams of fairly arbitrary size. You can read more about systolic array processors in the subchapter 2.5.1 of this book.

The Systolic Array Architecture

A digit serial wavelet architecture using the pyramid algorithm was outlined in [PaNi]. The systolic array (DWT-SA) architecture in an improvement over the above architecture. Here, only one set of multipliers and adders has been employed in contrast to two parallel computational hardware employed in [PaNi]. The multiplier and adder set performs all necessary computations to generate all high- and low-pass coefficients. Thw DWT-SA architecture does not use any external or internal memory modules to store the intermediate results and therefore avoids the delays caused by memory access and memory refresh timing. In addition, since a set of registers controlled by a

global clock is employed, the control circuitry does not need to take to intermediate products in and out of the memory. This results in a simple and efficient systolic implementation for 1-D DWT computation.

In order to compute separable 2-D DWT, two modules (one for row transform and another for columns) of this architecture can be used along with a transposer. With DWT-SA, it is possible to calculate 2-D DWT for a 512*512 monochrome image in real time (13 ms with 20 MHz clock).

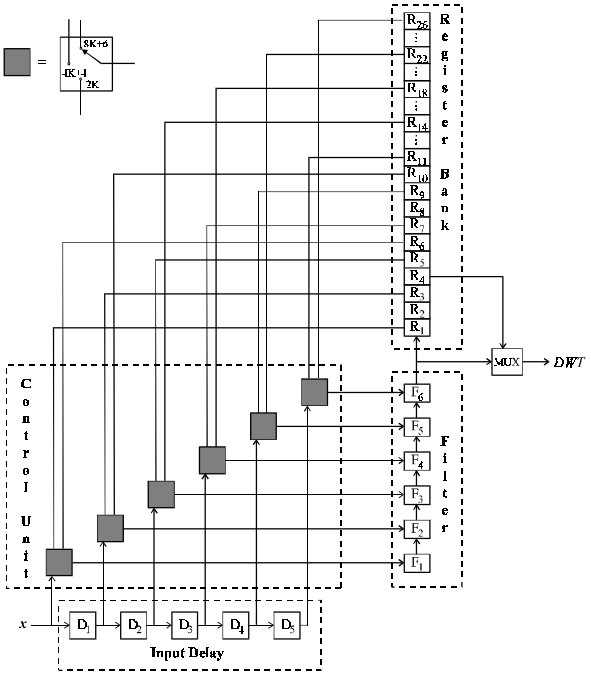


Figure 13.a. The DTW-SA architecture

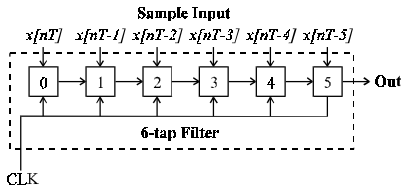


Figure 13.b. Systolic operation of the six tap filter

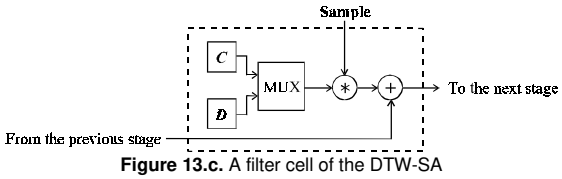


Figure 13.c. A filter cell of the DTW-SA

4.7.7 References and Further Readings

[Awa] *Aware Wavelet Transform Processor (WTP) Preliminary*, Aware Inc., Cambridge, MA.

[Chan] Y.T. Chan: *Wavelet Basics*. Kluwer Acad., Boston (1995)

[Chui] C.K. Chui: *An Introduction to Wavelets.* Academic Press (1992)

[ChVO] C. Chakrabarti, M. Vishwanath, R. M. Owens: *Architectures for Wavelet Transforms*. Proc IEEE VLSI Signal Processing Workshop, 1993, pp. 507-515.

[Dau] Ingrid Daubechies: *Ten Lectures on Wavelets*. SIAM (1992)

[GMPY] Aleksander Grzeszczak, Mrinal K. Mandal, Sethuraman Panchanathan, Tet Yeap: *VLSI Implementation of Discrete Wavelet Transform*. IEEE Trans. on VLSI Systems, vol. 4., No. 4. (Dec. 1996) pp. 421-433.

[Grze] A. Grzeszczak: *VLSI Architecture for Discrete Wavelet Transform*. M.A.Sc. thesis, Dep. Elec. Eng., Univ. Ottawa, Ont., Canada (1995)

[Kang] Y. Kang: *Low-power Design of Wavelet Processors*. Proc SPIE, vol. 2308, 1993, pp. 1800-1806.

[Mat] The MATLAB Wavelet Toolbox

[PaNi] K. K. Parhi, T. Nishitani: *VLSI Architectures for Discrete Wavelet Transform*. IEEE Trans. on VLSI Systems, pp. 191-202., June 1993

[RiVe] Oliver Rioul, Martin Vetterli: *Wavelets and Signal Processing*. IEEE SP Magazine, Oct. 1991, pp. 14-37.

[StNg] Gilbert Strang, Truong Nguyen: *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley (1996) [With bibliography.]
(http://saigon.ece.wisc.edu/~waveweb/Tutorials/book.html)

[Vish] M. Vishwanath: *Discrete Wavelet Transform in VLSI*. Proc. IEEE Int. Conf. Appl. Specific Array Processors, 1992, pp. 218-229.

http://www.wavelet.org

Wavelet digest: wavelet@math.scarolina.edu Subj: subscribe
http://www.scarolina.edu/~wavelet
ftp://maxwell.math.scarolina.edu/pub/wavelet

[StNg] (on which this chapter is mostly based), [Dau] and [Chan] can be used as basics literature to learn more about wavelets. The others deal with narrow fields of applying wavelet transforms thus they are for expert waveleteers.