Oversampling the Haar Wavelet Transform

Z.R. Struzik

Information Systems (INS)

INS-R0102 March 31, 2001

Report INS-R0102
ISSN 1386-3681

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Oversampling the Haar Wavelet Transform

Zbigniew R. Struzik

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
email: Zbigniew.Struzik@cwi.nl

ABSTRACT
The Haar wavelet representation and a number of related representations have been shown to be a simple and powerful technique for similarity matching of time series. In this report, we extend the standard formulation to the translation invariant oversampled system. This makes possible a particularly efficient incremental scheme for coefficient calculation. As an additional benefit, the oversampled scheme provides for easy incremental update of the decomposition on new input samples. The system is further extended over higher order scaling functions of smoother character and over wavelets with more vanishing moments.

2000 Mathematics Subject Classification: 28A80, 65U05, 68T10, 68P10
1999 ACM Computing Classification System: H1, I5, Jm, J2, E2
Keywords and Phrases: Haar wavelet, wavelet transform, multi-resolution, shift invariant, real-time, incremental, update
Note: full colour version of this paper can be downloaded from www.cwi.nl/~zbyszek

1. Introduction

The Haar wavelet transform has an established name as a simple and powerful technique for the multiresolution decomposition of time series. Unfortunately, the standard Haar wavelet transform has several limitations. The lack of translational shift invariance is very limiting when matching is to be performed over various intervals of the input data. In the standard formulation, for each shift in the input samples, the majority of the coefficients change and need to be recalculated. This is also the case when the input data is (continuously) updated with new samples.

Aiming at the possibility of such an incremental update (in particular in real-time), we present in this report an extended formulation of the Haar wavelet decomposition. It oversamples the decomposition by calculating coefficients on a shift invariant grid. While fixing the lack of shift invariance, this leads to severe redundancy of the oversampled representation, which may seem rather inefficient. However, using appropriate heuristics, we have been able to reduce computational costs in the oversampled scheme and provide the representation with the highly required features:

1. shift invariant coefficients;
2. incremental calculation of the coefficients;
3. incremental update on new sample(s);
4. the possibility to implement higher order systems.

An easy and very natural incremental update of the decomposition on new input samples comes almost automatically with the oversampled scheme.

Additionally, we have extend the Haar representation over higher order, smoother scaling kernels, as well as with wavelets of a higher degree of orthogonality to polynomials. These are obtained with

multiple/n-fold convolutions [1] of the usual components of the Haar wavelet (derivative operator and block scaling function). The oversampled representation can be used to implement the multifold convolutions in a straightforward manner.

2. The Haar Wavelet Transform

Conceptually, the wavelet transform [1, 2, 3] is an inner product of the time series with the scaled and translated wavelet $\psi(x)$, usually a n-th derivative of a smoothing kernel $\theta(x)$. The scaling and translation actions are controlled by two parameters; the scale parameter s 'adapts' the width of the wavelet to the resolution required and the location of the analysing wavelet is determined by the parameter b:

$$Wf(s,b) =< f, \psi_{(s,b)} >= \frac{1}{s} \int_{\Omega} dx\, f(x)\, \psi(\frac{x-b}{s}) \,. \tag{2.1}$$

For the continuous version (CWT), we have continuously running indices $s, b \in \mathbf{R}$ and $s > 0$, while for the discrete version (DWT, or just WT), the parameters are taken from a discrete, usually hierarchical (e.g. dyadic) grid of values s_i, b_j. Ω is the support of the $f(x)$ or the length of the time series.

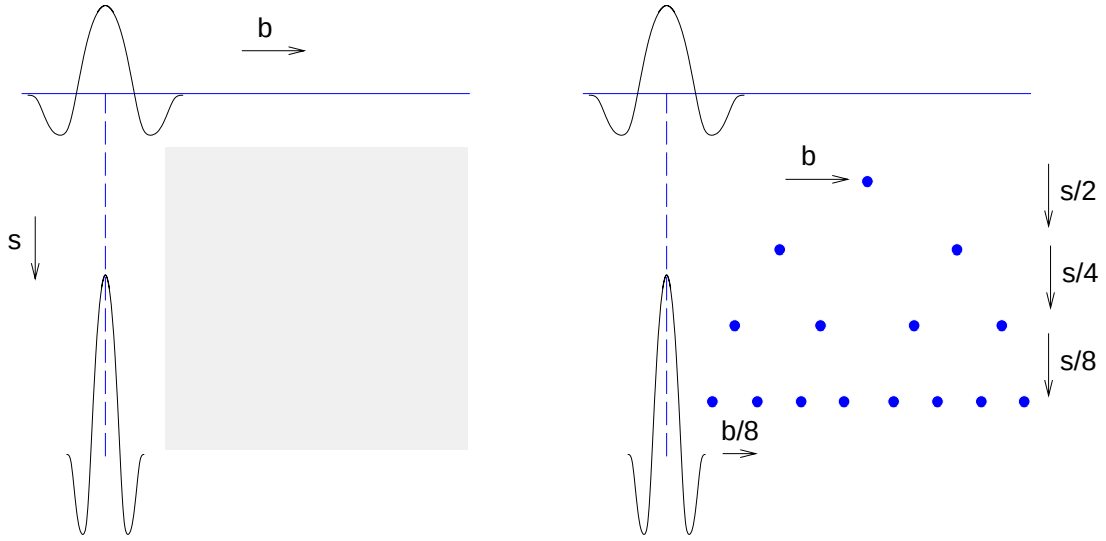


Figure 1: Continuous sampling of the parameter space (left) versus discrete (dyadic) sampling (right)

2.1 Discrete Haar Representation

The choice of the smoothing kernel $\theta(x)$ and the related wavelet $\psi(x)$ depends on the application and on the desired properties of the wavelet transform. A good default choice for the smoothing kernel is the Gaussian [2, 3, 4, 5, 6, 7]. The related wavelets, n-th derivatives of the Gaussian, have optimal localisation both in frequency and position. In this paper, for reasons related to inceremental construction of the wavelet coefficients (which will become apparent in the following), we will use a different smoothing function, namely a simple block function (and its successive convolutions with itself, see section 4):

[1] In this text for the reason of ease of use we will use the term 'convolution' instead of the 'inner product'. Strictly speaking, this is not correct since the two are not the same but differ in the sign of the argument of one of the operands. When using the term 'convolution' we therefore mean 'direct convolution' or 'inner product', as defined in 2.1.

$$\theta_{\square}(x) = \begin{cases} 1 & \text{for} \quad 0 \leq x < 1 \\ 0 & \text{otherwise} . \end{cases} \tag{2.2}$$

The wavelets obtained from this kernel are defined on finite support and go by the name of Alfred Haar, who in 1910 suggested this decomposition scheme [9]:

$$\psi_{\square}(x) = \begin{cases} 1 & \text{for} \quad 0 \leq x < \frac{1}{2} \\ -1 & \text{for} \quad \frac{1}{2} \leq x < 1 \\ 0 & \text{otherwise} . \end{cases} \tag{2.3}$$

For a particular choice of rescaling and position shift parameters (dyadic pyramidal scheme), the Haar system constitutes an orthonormal basis:

$$\psi_{m,n}(x) = 2^{-m}\psi(2^{-m}x - n), \qquad m = 1, 2, \ldots, L \qquad n = 0 \ldots, 2^{L-m} - 1 . \tag{2.4}$$

Assume an arbitrary time series $f = \{f_i\}, i = 0 \ldots 2^L - 1$, i.e. 2^L samples. Using the orthonormal basis just described, the function f can be represented with the linear combination of Haar wavelets:

$$f = f^{L+1} + \sum_{m=L}^{1} \sum_{l=0}^{2^{L-m}-1} c_{m,l}\, \psi_{\square m,l}, \tag{2.5}$$

where f^{L+1} is the most coarse approximation of the time series; $f^{L+1} =< f, \theta_{\square} >$ and each coefficient $c_{m,l}$ of the representation can be obtained as:

$$c_{m,l} =< f, \psi_{\square m,l} >=< f(x_i), \delta(i)\psi_{\square m,l}(x) > \ .$$

We used $\delta(i)$ to discretize the wavelet:

$$\delta(x) = \begin{cases} 1 & \text{for} \quad x = ind(f_i) \\ 0 & \text{otherwise} , \end{cases} \tag{2.6}$$

and $ind(f_i) = i$, is a function assigning a real value to the index of the discrete variable.

2.2 Multi-resolution Approximations

In particular, the approximations f^j of the time series f with the smoothing kernel $\theta_{\square j,k}$ form a 'ladder' of multi-resolution approximations:

$$f^{j-1} = f^j + \sum_{k=0}^{2^{L-j}-1} < f, \psi_{\square j,k} > \psi_{\square j,k}, \tag{2.7}$$

where $f^j =< f, \theta_{\square j,k} >$, and

$$\theta_{\square j,k} = 2^{-(j-1)}\theta_{\square}(2^{-(j-1)}x - k), \qquad j = 1, ..., L+1, \qquad k = 0, ..., 2^{L-(j-1)} - 1 .$$

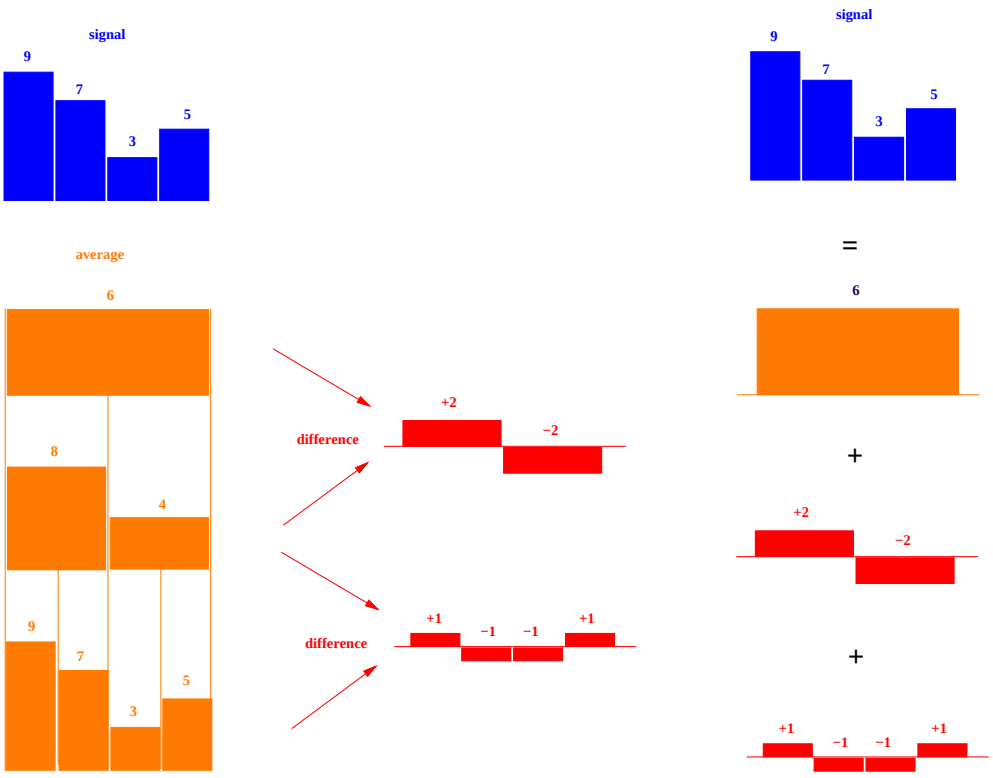


Figure 2: Decomposition of the example time series into Haar components. Right: reconstruction of the time series from the Haar components.

It is thus possible to 'move' from one approximation level $j-1$ to another level j by simply adding (subtracting if going from j to $j-1$) the detail contained in the corresponding wavelet coefficients $c_{j,k}, k = 0 \ldots 2^{L-j} - 1$.

In figure 2, we show an example decomposition and reconstruction with the Haar wavelet. The time series analysed is $f_{0..3} = \{9, 7, 3, 5\}$.

Note that the set of wavelet coefficients can be represented in a hierarchical (dyadic) tree structure, through which it is obtained. In particular, the reconstruction of each single point f_i of the time series is possible (without reconstructing all the $f_j \neq f_i$), by following a single path along the tree, converging to the point f_i in question. This path determines a unique 'binary address' of the point f_i.

The Haar wavelet is often used due its simplicity, compact support and ease of interpretation of the coefficients, see e.g Ref [8]. In the following subsection we will give a brief argument on the derivative of the analysed function.

2.3 Vanishing Moments and Stable Derivatives

Note that the Haar wavelet implements the operation of derivation at the particular scale at which it operates. From the definition of the Haar wavelet $\psi_{\square}$, (eq. 2.3, see also figure 3) we have:

$$\psi_{\square}(x) =< D(2\ x), \theta_{\square}(2\ x) > , \tag{2.8}$$

where D is the derivative operator

$$D(x) = \begin{cases} 1 & \text{for } x = 0 \\ -1 & \text{for } x = 1 \\ 0 & \text{otherwise .} \end{cases} \tag{2.9}$$

For the wavelet transform of f, we have the following:

$$\begin{aligned} < f(x), \psi_{l,n}(x) > &= \\ &= < f(x), < D_{l,n}(2\,x), \theta_{l,n}(2\,x) >> \\ &= < f(x), 2^{-1} < D_{l-1,n}(x), \theta_{l-1,n}(x) >> \\ &= << 2^{-1} D_{m,n}(x), f(x) >, \theta_{m,n}(x) > \\ &= 2^{-1} < Df_{m,n}(x), \theta_{m,n}(x) > , \end{aligned} \tag{2.10}$$

where Df is the derivative of the function f and θ is the smoothing kernel. (We skipped the $\square$ in the above derivation since it holds for any wavelet which satisfies 2.8.) The wavelet coefficients obtained with the Haar wavelet $\psi_{\square}$ are, therefore, proportional to the local averages of the derivative of the time series f at a given resolution. This property of our representation has already proved to be useful in time series mining [8]. Indeed, in the analysis of patterns in time series, the ability to represent local slope at various resolutions is an appealing feature.

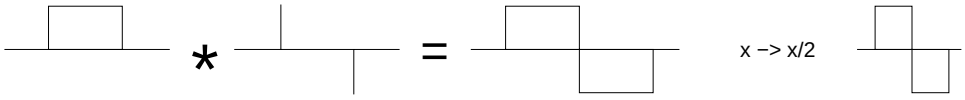


Figure 3: Convolution of the block function with the derivative operator gives the Haar wavelet after rescaling the time axis $x \to x/2$. $*$ stands for the inner product.

The ability of the wavelet to perform derivation (of n-th degree) is also referred to as the number of vanishing moments. Formally, the wavelet $\psi^{(n)}$ has n vanishing moments if

$$\int_{\Omega} x^m \, f(x) \, \psi(x) = 0 \, , \qquad \forall m, \; 0 \leq m < n \, , \tag{2.11}$$

that is, if it is orthogonal to polynomials of degree n and lower. This ability to filter out polynomials of degree n in input data is required in many applications [4, 10, 6, 7]. In particular, valuable information often resides in weak singular components which are often occluded by strong polynomial bias. For this reason, in section 4 we will extend the standard Haar wavelet (orthogonal to constants only) to wavelets with more vanishing moments.

3. Extending the Haar WT

So far we have discussed the classical Haar decomposition. We are now going to extend the decomposition in two directions. One is related to the grid of the decomposition. We will use oversampling, which will add translation invariance to the decomposition. (Recall that the Haar WT is defined on a dyadic grid and is not translation invariant.) The other extension concerns the Haar wavelet itself [11]. We will modify the definition to allow higher degree polynomials for the scaling function (the kernel). Recall that the Haar wavelet is derived from the block function (thus zero order polynomial) convolved with the derivative operator. Allowing multifold convolution of the block function

with itself will result in a smoother kernel. The number of times the derivative operator is convolved with the kernel is also subject to extension. This will allow us to increase the so-called number of vanishing moments of the wavelet (which indicates the orthogonality to polynomials). Note that the above extensions take place at the cost of giving up the orthogonality of the representation.

3.1 Oversampled, Scale-wise Incremental Decomposition

In order to provide shift invariance in our representation, we will oversample the Haar decomposition. In this work, the oversampling operation will only be performed on the position axis. The sampling grid for the scale axis will remain dyadic in scale. (Although not done here, the scale axis can also be oversampled.)

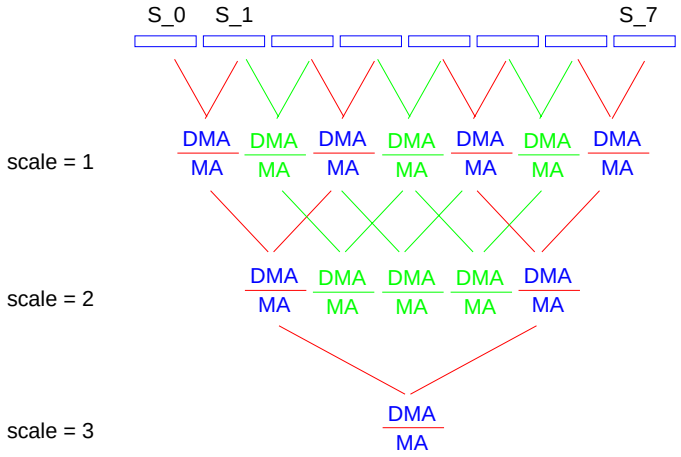


Figure 4: Oversampled Haar decomposition. Highlighted (in red) is the dyadic pyramid.

Oversampling on the position axis can be done to the highest resolution required, or any required intermediate, lower resolution. In this work, we will use the highest available resolution, that is that of the sampling rate of the input. This means that we let the position translation index run over the positions i of the time series f_i. Therefore, instead of Eq. 2.4, we have for the oversampled system:

$$\psi_{m_o,n_o}(x) = 2^{-m_o}\psi(2^{-m_o}x - n_o), \qquad \begin{cases} m_o &= 1, 2, \ldots, L \\ n_o &= BE, \ldots, 2^L - 1 - BE \\ BE &= 2^{m-1} - 1\ , \end{cases} \tag{3.1}$$

where BE denotes empty boundary coefficients and is introduced here in order to ensure alignment of the indices across scales. The bands which are empty are half the width of the wavelet (precisely $BE = 2^{s-1} - 1$ wide, where $s = 1, 2, 3 \ldots$ is the scale level). They are 'empty' because we can only position the wavelet so that it aligns with the time series at either end.

An oversampled grid obtained according to the above prescription is shown in figure 4. We localised the MA/DMA decomposition coefficients on the grid points. (The meaning of the MA/DMA and the interconnecting lines will be explained below.) Of course it is possible to select the dyadic grid from the oversampled representation for any arbitrary starting location. One such possible (centred) selection has been highlighted in figure 4.

Oversampling will also make possible the scale-wise incremental calculation of the Haar coefficients. Note that the basic scaling function is a block function, see Eq. 2.2. The convolution operation $< f, \theta >$ for this block function can be conveniently written as a sum of the input function f over the support of θ:

$$< f, \theta_{\square} >= \sum_{i \in \text{support}(\theta_{\square})} f_i \ . \tag{3.2}$$

Since this operation is equivalent to the moving average (MA) filtering (after proper normalisation), we denoted with MA each location m_o, n_o, where oversampled coefficients need to be calculated. Of course, replacing the multiplication operation with the summation can potentially have a substantial impact on the computational efficiency of the Haar WT algorithm.

The other heuristic for our incremental coefficient calculation is the observation that the MA coefficient at a particular scale can be obtained as a sum of MA coefficients at a lower scale: [2]

$$< f, \theta_{\square}(m,n) >=< f, \theta_{\square}(m+1,2n) > + < f, \theta_{\square}(m+1,2n+1) > \ . \tag{3.3}$$

In figure 4 we illustrate this principle, with line segments joining each MA coefficient at a higher scale, with two MA components at a lower scale. (At the input sample level, the input samples are taken instead of MA.)

Lastly, the actual Haar coefficients can be obtained directly from the two MA coefficients at a higher resolution (lower scale), compare Eq. 2.10:

$$< f, \psi_{\square}(m,n) >=< f, \theta_{\square}(m-1,2n) > - < f, \theta_{\square}(m-1,2n+1) > \ . \tag{3.4}$$

Since what is done here is the application of the convolution of the derivative operator and the moving average filter, we denoted it with DMA in figure 4 in all the locations where it is computed.

3.2 Padded Oversampled Decomposition

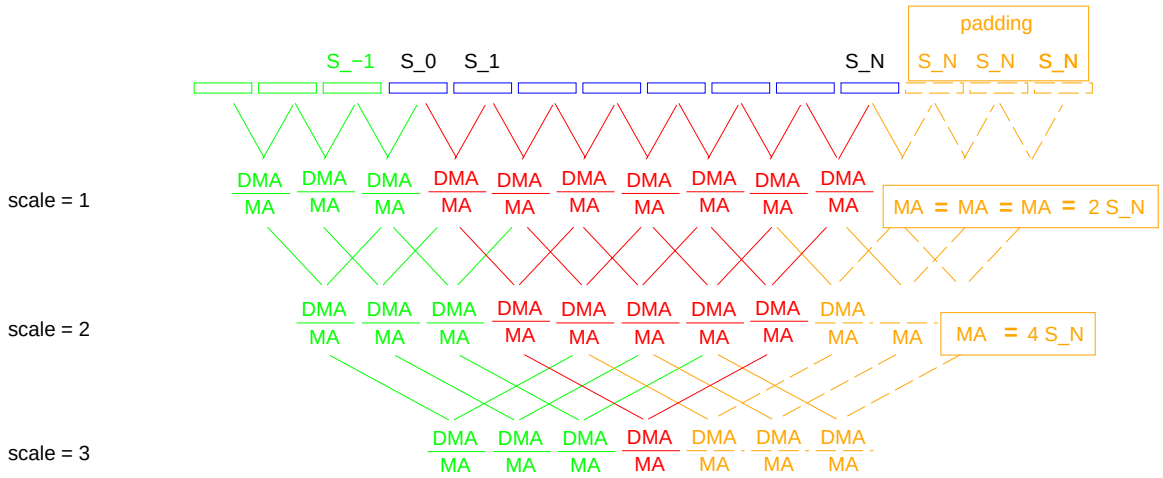


Figure 5: Padded Haar decomposition. The padding applied to the 'future' takes the last sample for the padded values. The padding for the 'past' uses the same principle or simply uses past values which are out of the current range of interest $(s_0, \ldots, s_N)$, $N = 2^L - 1$.

In some applications, we would like to have information about the approximation of the time series at all locations i corresponding to the input time series f_i. [3] This is still not possible in the oversampled

[2] Of course by the same argument, the coefficient at a particular scale can be obtained by summing up corresponding coefficients at any other arbitrary (lower) scale when dyadic scale increments are used.

[3] Actually, for the Haar WT, most natural is N locations if $2^L = N + 1$ is the number of input samples.

representation on both sides of the representation. Because we can only position the wavelet so that it aligns with the time series at either end, two void/'empty' bands are left at both ends. The width of these bands is proportional to the width of the wavelet (precisely $BE = 2^{s-1} - 1$ wide where $s = 1, 2, 3 \ldots$ is the scale level).

There are various solutions to this problem. One possibility requires the assumption of stationarity/periodicity and it provides 'padding' by joining both ends of the time series (wrapping around). The assumption of stationarity/periodicity is a relatively strong one and when it is evidently violated, for example by a strong linear bias, it often results in a large singularity, which can spoil all boundary coefficients, see figure 6.

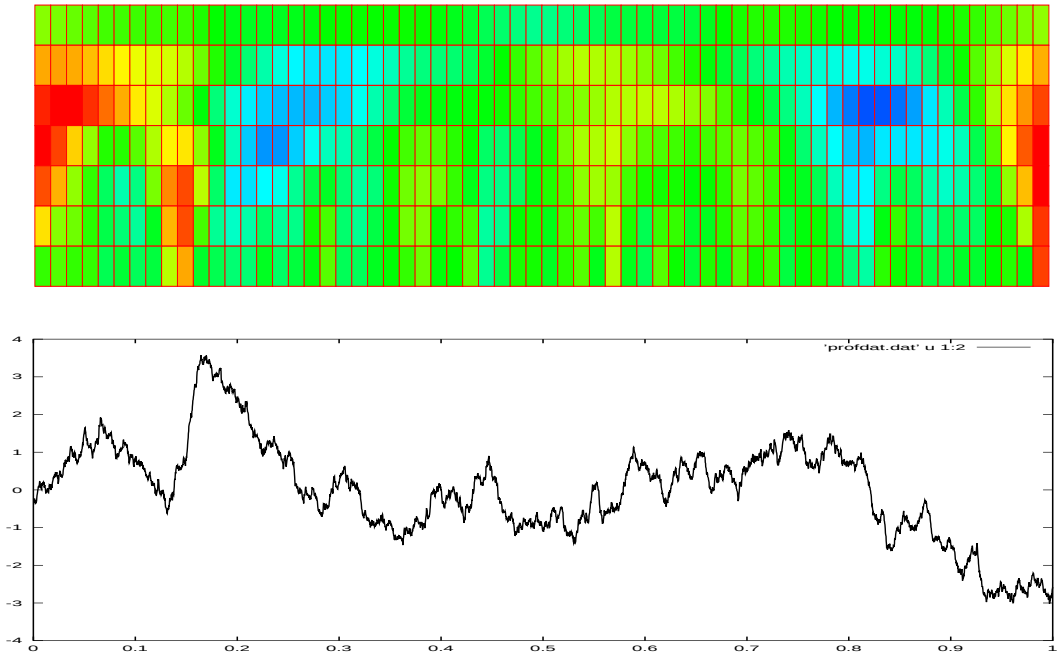


Figure 6: Example time series and (a part of) the oversampled Haar representation displayed in colour coding for coefficient amplitude values. (Rainbow colour coding from red for most positive to blue for most negative.) Periodical boundary condition is used.

Another approach, which we will favour here, is the 'last value' predictor, requiring a substantially milder assumption. As the name suggests, it extrapolates the time series with the last recorded value. Therefore padding is simply done with a required number of values equal to the last value of the time series. This can be done at both 'ends' of the time series. Of course, we will be padding with the first value for the 'past' and last value for the 'future'. In the case of (real time) processing with a continuous update, we will often have enough past samples to be able to use them for the past information and padding in the past will only be necessary on initialisation.

In figure 5, we show the example padding using the last known value. We highlight the coefficients which depend on the padded values. We also indicate the coefficients which need to be calculated from the padded values. Note that the calculation of these coefficients is particularly simple because all that needs to be done is some n-fold summation of the last time series value.

In the example shown in figure 5, we arranged the padding so that exactly N decomposition coefficients are calculated for each scale considered, where $N + 1 = 2^L$ is the length of the input time series. It is possible and it may be practical in some cases to calculate more boundary coefficients, for example to obtain $N + 2$ decomposition coefficients (or $N + 1$ if we permit asymmetry). Of course

such coefficients will show increasingly more distortion for larger padded areas used (proportionally to the ratio of true/padded input samples used).

3.3 Incremental Update

Oversampled representation makes possible a particularly simple update scheme. Update to the decomposition is beneficial when the representation is computed over partially overlapping (in time scale) windows of interest. In particular, when the representation has to be updated on each new sample or a packet of new samples (possibly in real time), an efficient update scheme is required.

Such an update scheme is almost automatically achieved with the oversampled representation. Only the coefficients which depend on the new sample need to be calculated. In an unpadded representation, there are only L such coefficients where L is the number of scales considered. In the case of padding, the padding coefficients will, of course, have to be recalculated (since each new sample is the last sample available and we pad using the last available sample).

An example scheme of one sample incremental update is shown in figure 7.

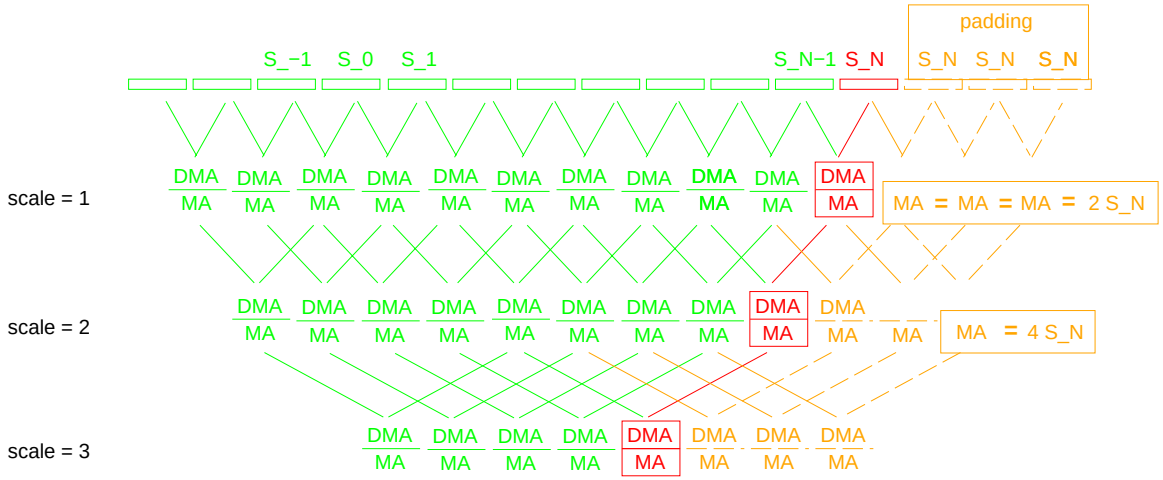


Figure 7: Update mechanism on one new sample with padding. Only the highlighted coefficients need to be calculated anew. This comprises new value coefficients (in red) and the new padding (in orange). Other coefficients (in green) need to be shifted by one index value.

4. Higher Order Extensions

So far we have been using the Haar wavelet, which can be represented through the convolution of the derivative operator with the block function 2.8. In some applications it will be beneficial to use a twofold extension to this wavelet. [4] The extensions can be applied to both components of the convolution.

4.1 Higher Order Smoothing Function

The block function in Eq. 2.8 serves as a smoothing kernel. It is possible to take a higher degree polynomial with different smoothing characteristics. The simplest way to achieve this is to permit multifold convolution of the block function with itself.

$$\theta_{\square}(2x) * \theta_{\square}(2x) = \theta_{\triangle}(x) \tag{4.1}$$

[4] In the following, we use the system presented by Roux in his thesis [11].

The effect of taking two block functions and convolving them with one another, Eq4.1, is shown in figure 8.

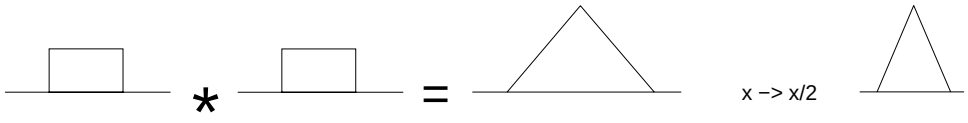


Figure 8: Convolution of the block function with itself gives a tent smoothing function after rescaling the time axis $x \to x/2$. $*$ stands for the inner product.

The convolution of three block functions delivers a smooth quadratic kernel, see Eq. 4.2 and figure 9,

$$\theta_{\square}(3x) * \theta_{\square}(3x) * \theta_{\square}(3x) = \theta_{\cap}(x) \tag{4.2}$$

and in the limit $n \to \infty$ the n-fold convolution of the block function with itself converges to the Gaussian kernel.

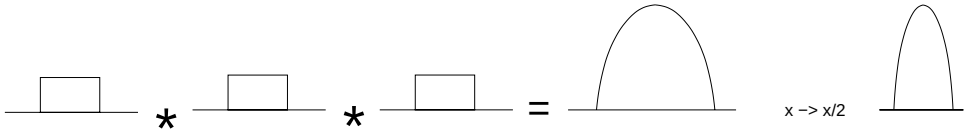


Figure 9: Convolution of three block functions gives a bump smoothing function after rescaling the time axis $x \to x/3$. $*$ stands for the inner product.

4.2 Higher Degree Derivation

In section 2.3, we have shown that the Haar wavelet can be decomposed into a convolution of the derivative operator and the block kernel. We can construct wavelets with other smoothing kernels, for example, as just discussed, with the multifold convolutions of the block kernel with itself. Such wavelets will still have one vanishing moment due to the (first/single) derivative operator, but they will have smoother characteristics, linear, quadratic or higher, according to the smoothing kernel used. (The wavelet with m vanishing moments is orthogonal to/ filters out polynomials of less than or equal to degree $m-1$.)

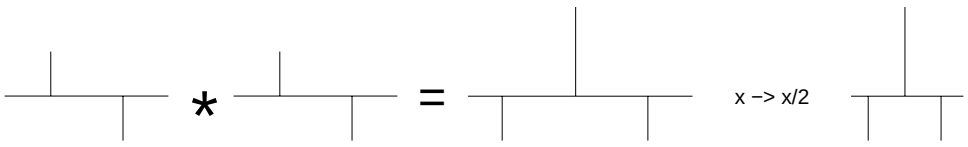


Figure 10: Convolution of the derivative operator with itself gives a second degree derivative operator (for comparison the time axis has been rescaled: $x \to x/2$). $*$ stands for the inner product.

We can take more convolutions of the derivative operator resulting in a higher degree of orthogonality to polynomials. In figure 10, we show a second derivative operator obtained from the convolution

of two (single) derivative operators. Convolved with a smoothing kernel, for example one of these introduced above (block function, tent or bump), it will give a wavelet with two vanishing moments. Such a wavelet can filter out constant bias and linear trends. This is a good enough generic choice and, therefore, wavelets with two vanishing moments are quite often used.

In figure 11, we present the types of wavelets which result from the convolution of the block function and derivative operator. The convolutions are taken up to the second degree in both components. This, of course, can be be extended to higher orders, should there be a need for a smoother wavelet type or a higher number of vanishing moments. Figure 11 is only intended to provide the intuition behind the construction scheme. In the Appendix, we give analytic formulas describing the wavelets shown in Figure 11, together with correct, detailed plots.

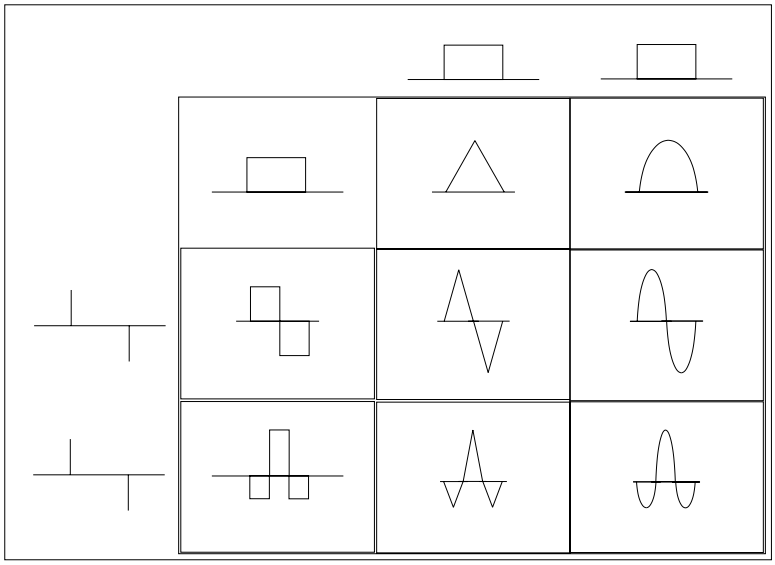


Figure 11: The array of inner (convolution) products of the derivative operator and the block function.

These wavelets can be used in the 'traditional' way, that is using the explicit convolution, Eq. 2.1 to obtain the corresponding wavelet transform coefficients. In the case of the block function it is, however, possible to replace the convolution with the summation over an appropriate support, as we have shown in section 3.1, Eq. 3.2. Due to the fact that the wavelets' kernels involved can be seen as convolutions of appropriate components, a similar approach can be used to replace the product with an appropriate sum. In the following, we will, therefore, describe an alternative incremental method of implementing the convolution implicitly in the oversampled representation.

5. Incremental Implementations of Higher Order Wavelets

In the previous section we have shown that higher order wavelets can be obtained through a convolution product of simple components like the block function or derivative operator. Such convolutions can easily be implemented on the oversampled Haar representation. Each oversampled level of the Haar representation contains shifted convolutions of the input function with the block kernel of an appropriate length, see Eq. 3.2 and Fig. 4.

5.1 Second Order Haar Type Wavelet Calculation from the Oversampled MA Representation

In order to perform a second convolution of the input with the block kernel, the results of the first convolution can be directly used. This is how the convolution of the input with the tent kernel is obtained:

$$\theta_{\triangle} * f = (\theta_{\square} * \theta_{\square}) * f = \theta_{\square} * (\theta_{\square} * f) \ .$$

Since, as noted before, the convolution with the block function simply amounts to a summation over an appropriate support, the second convolution step is obtained by summing up the oversampled block representation at the scale corresponding with the kernel used and over the support corresponding with the kernel width.

The tent representation coefficients at a particular scale can thus be obtained as the sum of MA coefficients (at a lower scale) already calculated:

$$< f, \theta_{\triangle}(m,n) >= \sum_{i=n}^{n+2^{m-1}} < f, \theta_{\square}(m-1,i) > \ . \tag{5.1}$$

Note, that it is not possible to derive the tent coefficients in an incremental way from the lower scale (tent) coefficients as was the case for the block coefficients, see Eq.3.3. This is because the tent function cannot be decomposed into the sum of two shifted tent functions at half the resolution. This, of course, also holds for higher degree kernels like the bump function. Therefore, all the higher degree kernels and the related wavelets will be obtained by a sum of lower degree kernel coefficients at the resolution considered.

The wavelets related with the tent function can now be obtained from the tent kernel representation. For the first degree of differentiation (one vanishing moment wavelet) we have:

$$\psi_{\triangle} * f = (D * \theta_{\square} * \theta_{\square}) * f = D * (\theta_{\square} * (\theta_{\square} * f)) \ . \tag{5.2}$$

Here, the convolution with the derivative operator is done as the last step on the already calculated tent kernel coefficients. Of course, since convolution is Abelian, the order of operations is arbitrary and we could have first convolved the block representation with the derivative operator D and later with the second block kernel.

The same approach can be used for a higher number of vanishing moments. In the case of two vanishing moments:

$$\psi^{(2)}_{\triangle} * f = (D * D * \theta_{\square} * \theta_{\square}) * f = D * (D * (\theta_{\square} * (\theta_{\square} * f))) \ . \tag{5.3}$$

Therefore, we can obtain the coefficients for the two vanishing moments (tent) wavelet from the coefficients obtained with the wavelet with one vanishing moment.

In order to give an intuitive feeling of the procedure, in figure 12 the two lowest resolution levels for the tent smoothing kernel are considered. They correspond with support lengths *two* and *four* samples for the block function smoothing kernel. In the incremental scheme, they give, respectively, *three* and *seven* sample tent linear smoothing functions in two convolutions $(f * \theta_{\square})$, $(f_{\square} * \theta_{\square})$, see 5.1. These two convolutions correspond with MA summation over two levels of dyadic hierarchy.

$$\begin{aligned} < f, \theta_{\triangle}(2,n) > &= \sum_{i=n}^{n+1} < f, \theta_{\square}(1,i) >= \\ &= < f, \theta_{\square}(1,n) > + < f, \theta_{\square}(1,n+1) >= \\ &= f_n + 2f_{n+1} + f_{n+2} \ . \end{aligned}$$

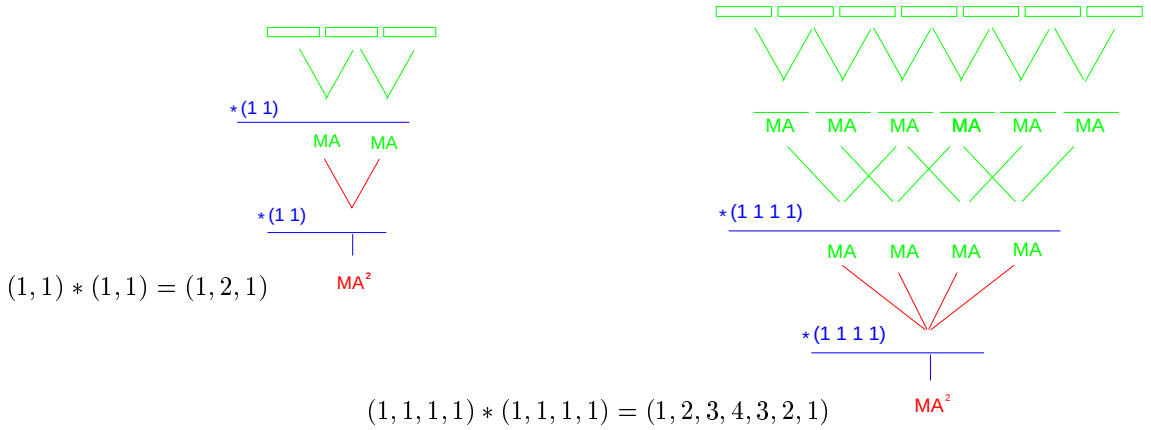


Figure 12: Second order Haar type wavelet calculation from the oversampled MA representation. The two lowest support lengths *two* and *four* samples are considered for the block function smoothing kernel. They give, respectively, *three* and *seven* sample tent linear smoothing functions in one-fold convolution (corresponding with MA summation over two levels of dyadic hierarchy).

$$\begin{aligned} < f, \theta_{\triangle}(3,n) > &= \sum_{i=n}^{n+3} < f, \theta_{\square}(2,i) >= \\ &= f_n + 2f_{n+1} + 3f_{n+2} + 4f_{n+3} + 3f_{n+4} + 2f_{n+5} + f_{n+6} \ . \end{aligned} \tag{5.4}$$

Note that with increasing scale m we obtain still better approximations of the tent function. For scale level m we have $2^{(m+1)} - 1$ samples of the tent smoothing function $\theta_{\triangle}$.

The one vanishing moment wavelet decomposition coefficients for this smoothing kernel can now be directly obtained from the two tent coefficients at a higher resolution; compare Eq. 5.2:

$$< f, \psi_{\triangle}(m,n) >=< f, \theta_{\triangle}(m-1,2n) > - < f, \theta_{\triangle}(m-1,2n+1) > \ . \tag{5.5}$$

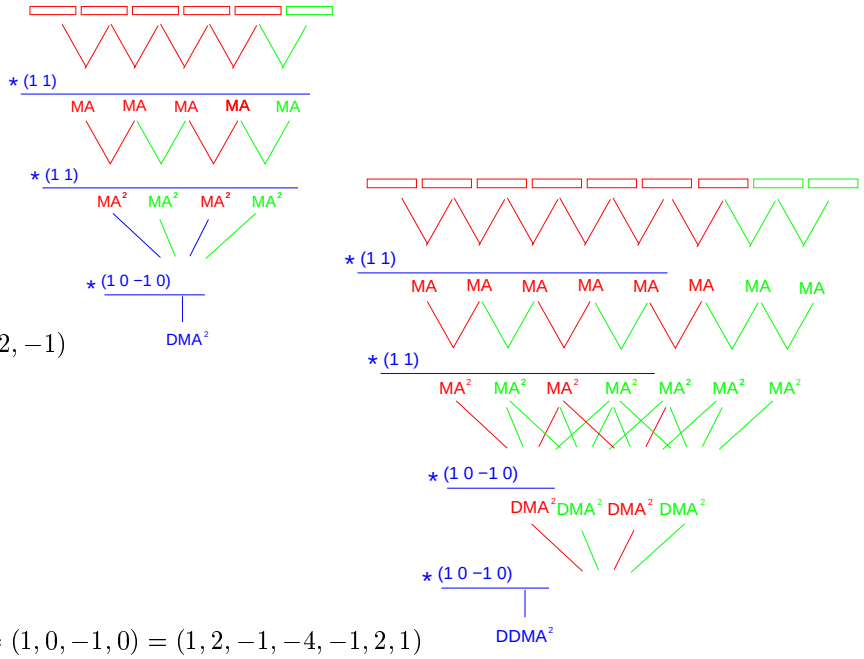
Of course, a higher number of vanishing moments can be obtained from the coefficients with a lower number of vanishing moments, see 5.3, in the same fashion as the higher order smoothing kernels are obtained from the block function. The two vanishing moments wavelet from the tent kernel will be obtained as

$$< f, \psi^{(2)}_{\triangle}(m,n) >=< f, \psi^{(1)}_{\triangle}(m-1,2n) > - < f, \psi^{(1)}_{\triangle}(m-1,2n+1) > \ . \tag{5.6}$$

The coefficients of the decomposition are therefore obtained in the incremental scheme without the need of actually providing the functional equation for the wavelet or precalculating the wavelet coefficients.

5.2 Third Order Haar Type Wavelet Calculation from the Oversampled MA Representation

The convolution of the input with the third order wavelet kernel, the bump function $\theta_{\cap}$, see Eq. 4.2, can be obtained in an incremental way, analogical to the second order calculation described above. From the already calculated tent coefficients, the bump coefficients are obtained by the convolution with the step function:



$$(1,1) * (1,1) * (1,0,-1,0) = (1,2,0,-2,-1)$$

$$(1,1) * (1,1) * (1,0,-1,0) * (1,0,-1,0) = (1,2,-1,-4,-1,2,1)$$

Figure 13: Second order, one and two vanishing moments Haar type wavelet calculation. The lowest support length of *two* samples is considered for the block function smoothing and the derivative operator. The product of two smoothing kernels with one (left figure) and two (right figure) derivative operators give a wavelet with respectively, one and two vanishing moments.

$$\theta_{\cap} * f = (\theta_{\triangle} * \theta_{\square}) * f = (\theta_{\triangle} * (\theta_{\square} * f)) \; . \tag{5.7}$$

Again, the convolution of the precalculated tent coefficients $(f * \theta_{\triangle})$ with a block function can be simply replaced by summing up the oversampled tent representation at the scale corresponding with the kernel used and over the support corresponding with the kernel width.

$$< f, \theta_{\cap}(m,n) >= \sum_{i=n}^{n+2^{m-1}} < f, \theta_{\triangle}(m-1,i) > \; .$$

In figure 14, the two lowest resolution levels for the bump kernel are considered. They correspond with support lengths *two* and *four* samples for the block function smoothing kernel. In the incremental scheme they give, respectively, *three* and *seven* sample bump linear smoothing functions, in three convolutions $(f * \theta_{\square})$, $(f_{\square} * \theta_{\square})$, $(f_{\triangle} * \theta_{\square})$, performed with MA summation over two levels of dyadic hierarchy:

$$
\begin{aligned}
< f, \theta_{\cap}(3,n) > &= \sum_{i=n}^{n+1} < f, \theta_{\triangle}(2,i) >= \\
&= < f, \theta_{\triangle}(2,n) > + < f, \theta_{\triangle}(2,n+1) >= \\
&= \sum_{i=n}^{n+1} < f, \theta_{\square}(1,i) > + \sum_{i=n+1}^{(n+1)+1} < f, \theta_{\square}(1,i) >= \\
&= f_n + 3f_{n+1} + 3f_{n+2} + f_{n+3} \ . \qquad (5.8)
\end{aligned}
$$

$$
\begin{aligned}
< f, \theta_{\cap}(4,n) > &= \sum_{i=n}^{n+3} < f, \theta_{\triangle}(3,i) >= \\
&= f_n + 3f_{n+1} + 6f_{n+2} + 10f_{n+3} + 12f_{n+4} + 12f_{n+5} + 10f_{n+6} + 6f_{n+7} + 3f_{n+8} + f_{n+9} \ . \qquad (5.9)
\end{aligned}
$$

Again, as was the case with the tent function, with increasing scale m we obtain still better approximations of the bump function. We need more samples to represent the more complex shape: for scale level m we have $2^{(m+1)} - 1 + 2^m - 1$ samples of the bump smoothing function $\theta_{\cap}$.

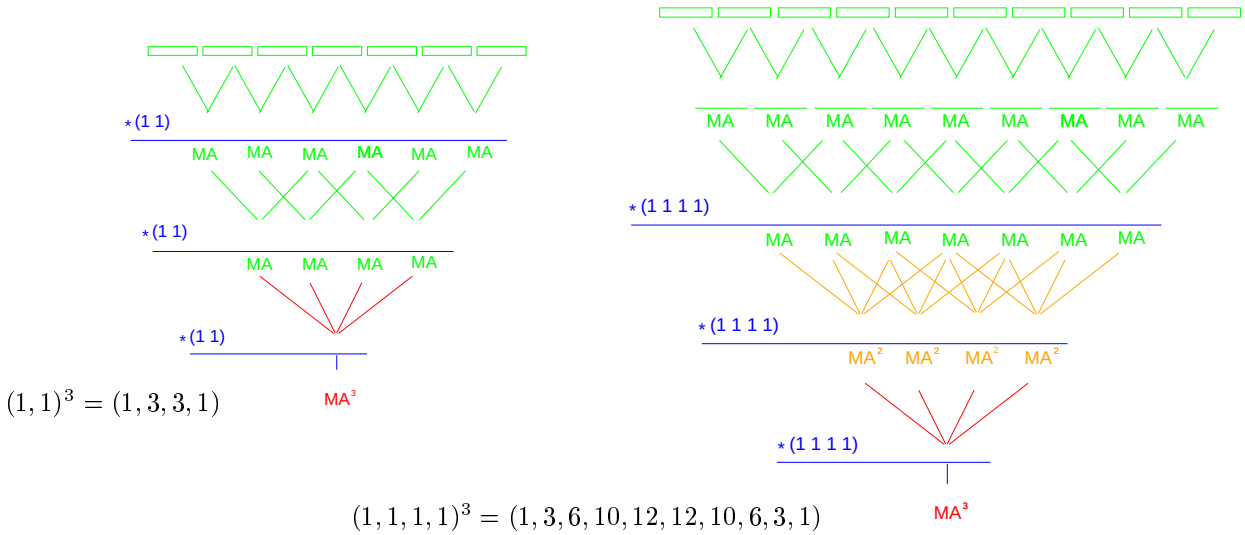


Figure 14: Third order Haar type wavelet calculation from the oversampled MA representation. The two lowest support lengths, *two* and *four* samples are considered for the block function smoothing kernel, giving respectively *four* and *ten* sample bump quadratic smoothing functions (three levels of dyadic hierarchy).

Similarly to the case of tent and block smoothing kernels, the one vanishing moment wavelet decomposition coefficients for the bump kernel can be directly obtained from the two bump coefficients at a higher resolution, compare Eq. 2.10:

$$
< f, \psi_{\cap}(m,n) >=< f, \theta_{\cap}(m-1,2n) > - < f, \theta_{\cap}(m-1,2n+1) > \ . \qquad (5.10)
$$

And the two vanishing moment wavelet decomposition coefficients for the bump kernel can be directly obtained from the two one vanishing moment bump coefficients at a higher resolution:

$$< f, \psi_{\cap}^{(2)}(m,n) >=< f, \psi_{\cap}(m-1,2n) > - < f, \psi_{\cap}(m-1,2n+1) > \ . \tag{5.11}$$

We refrained from plotting examples of this and higher orders wavelet constructions, as they become increasingly complicated and non-transparent.

6. CONCLUSIONS

In this report, we have extended the standard formulation of the Haar decomposition to the translation invariant oversampled system. A particularly efficient incremental scheme for coefficient calculation has been presented, which should provide a basis for fast implementations and improved storage.

As an additional benefit, the oversampled scheme provides for the easy incremental update of the decomposition on new input samples. Thus, efficient real time implementations of the algorithm seem possible.

The system has also been extended over higher order scaling functions of smoother character and over wavelets with more vanishing moments. These extensions will be practical in a variety of data analysis problems extending the scope of the data characteristics accessed with the Haar wavelet.

References

1. I. Daubechies, *Ten Lectures on Wavelets*. S.I.A.M., (1992).
2. M. Holschneider, *Wavelets - An Analysis Tool*. Oxford Science Publications, (1995).
3. S.G. Mallat and W.L. Hwang, Singularity Detection and Processing with Wavelets, *IEEE Trans. on Information Theory* **38**, pp. 617–643, (1992).
4. A. Arneodo, E. Bacry and J.F. Muzy, The Thermodynamics of Fractals Revisited with Wavelets, *Physica A*, **213**, 232–275, (1995).
   J.F. Muzy, E. Bacry and A. Arneodo, The Multifractal Formalism Revisited with Wavelets, *International Journal of Bifurcation and Chaos* **4**, pp. 245-302, (1994).
5. Z. R. Struzik, Local Effective Hölder Exponent Estimation on the Wavelet Transform Maxima Tree, in *Fractals: Theory and Applications in Engineering*, Eds: M. Dekking, J. Lévy Véhel, E. Lutton, C. Tricot, Springer Verlag, pp. 93–112, (1999).
6. Z.R. Struzik, The Wavelet Transform in the Solution to the Inverse Fractal Problem, *Fractals* **3**, No.2, pp. 329–350, (1995).
   Z.R. Struzik, From Coastline Length to Inverse Fractal Problem: The Concept of Fractal Metrology, *Thesis*, University of Amsterdam, (1996).
7. Z.R. Struzik, Determining Local Singularity Strengths and their Spectra with the Wavelet Transform, Fractals **8**, No 2, (2000).
8. Z.R. Struzik, A.P.J.M. Siebes, The Haar Wavelet Transform in the Time Series Similarity Paradigm, in *Principles of Data Mining and Knowledge Discovery*, Eds: J.M. Zytkow, J. Rauch, Springer-Verlag, pp 12-22, (1999)
9. A. Haar, Zur Theorie der orthogonalen Funktionnensysteme, *Math. Ann.* **69**, 331-71.
10. P.Ch. Ivanov, M.G. Rosenblum, L.A. Nunes Amaral, Z.R. Struzik, S. Havlin, A.L. Goldberger and H.E. Stanley, Multifractality in Human Heartbeat Dynamics, *Nature* **399**, pp. 461–465, (1999).
11. S. Roux, Analyse en ondelettes de l'auto-similarité de singnaux en turbulence pleinement développée, Thése de Doctorat, L'Université de la Mediterranée Aix-Marseille II. (1996).

Appendix

In the appendix, we give explicit analytic formulae for the kernels which are used in the paper. These kernels should be seen as limit cases. The incremental procedure described in the paper builds up approximations of these kernels gaining in quality (number of samples) with increasing scale level (decreasing resolution). The plots and formulae given are direct results of the convolutions of unit volume block function. Proper normalisation (on both x and y axis) is required to give L1 or L2 normalised wavelets.

$$\theta_{\triangle}(x) = \begin{cases} x+1 & \text{for} \quad -1 \leq x < 0 \\ -(x-1) & \text{for} \quad 0 \leq x < 1 \end{cases} \tag{6.1}$$

$$\psi_{\triangle}(x)^{(1)} = \begin{cases} x+1 & \text{for} \quad -1 \leq x < 0 \\ -2\,(x-\frac{1}{2}) & \text{for} \quad 1 \leq x < 2 \\ -(x-2) & \text{for} \quad 1 \leq x < 2 \end{cases} \tag{6.2}$$

$$\psi_{\triangle}(x)^{(2)} = \begin{cases} x+1 & \text{for} \quad -1 \leq x < 0 \\ -3\,(x-\frac{1}{3}) & \text{for} \quad 1 \leq x < 2 \\ 3\,(x-1\frac{2}{3}) & \text{for} \quad 2 \leq x < 3 \\ -(x-3) & \text{for} \quad 3 \leq x < 4 \end{cases} \tag{6.3}$$

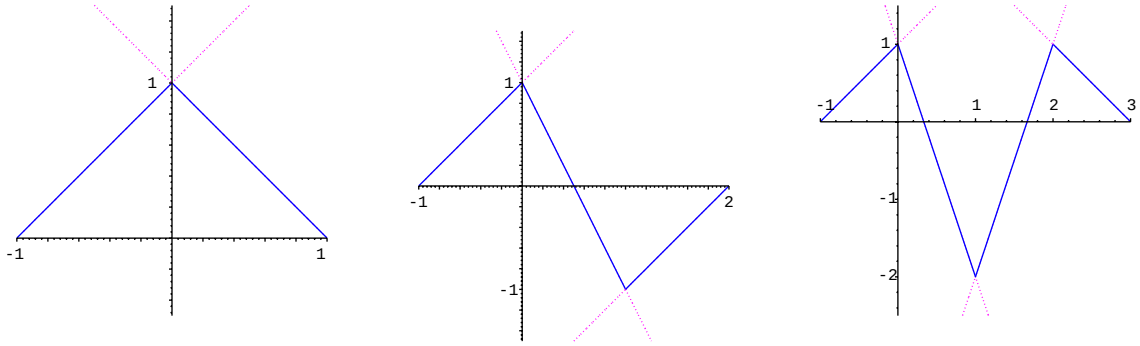


Figure 15: From left to right, convolution of two block kernels, convolution of two block kernels with one derivative operator and convolution of two block kernels with two derivative operators. After x, y normalisation they give the tent smoothing kernel and the tent wavelets with one and two vanishing moments.

$$\theta_{\cap}(x) = \begin{cases} \frac{1}{2}\,(x+1)^2 & \text{for} \quad -1 \leq x < 0 \\ 1-(\frac{1}{2}-x+x^2) & \text{for} \quad 0 \leq x < 1 \\ \frac{1}{2}\,(x-2)^2 & \text{for} \quad 1 \leq x < 2 \end{cases} \tag{6.4}$$

$$\psi_{\cap}(x)^{(1)} = \begin{cases} \frac{1}{2}\,(x+1)^2 & \text{for} \quad -1 \leq x < 0 \\ \frac{1}{4}-(\frac{1}{2}-x)^2+\frac{1}{2}\,(1-x^2) & \text{for} \quad 0 \leq x < 1 \\ -(\frac{1}{4}-(\frac{1}{2}-(x-1\frac{1}{3}))^2+\frac{1}{2}\,(1-(x-1\frac{1}{3})^2)) & \text{for} \quad 1 \leq x < 2 \\ -\frac{1}{2}\,(x-3)^2 & \text{for} \quad 2 \leq x < 3 \end{cases} \tag{6.5}$$

$$\psi_{\cap}(x)^{(2)} = \begin{cases} \frac{1}{2}\,(x+1)^2 & \text{for} \quad -1 \le x < 0 \\ \frac{1}{2}\,(\frac{1}{3} - 3\,(\frac{1}{3} - x)^2 + (1 - x^2)) & \text{for} \quad 0 \le x < 1 \\ -\frac{1}{2} - \frac{1}{2}\,((\frac{1}{3} - 3\,(\frac{1}{3} - (x-1))^2) + 2\,(\frac{2}{3} - \frac{3}{2}\,(\frac{2}{3} - (x-1))^2)) & \text{for} \quad 1 \le x < 2 \\ \frac{1}{2}\,(\frac{1}{3} - 3\,(\frac{1}{3} - (x - 2\frac{1}{2}))^2 + (1 - (x - 2\frac{1}{2})^2)) & \text{for} \quad 2 \le x < 3 \\ \frac{1}{2}\,(x-4)^2 & \text{for} \quad 3 \le x < 4 \end{cases} \tag{6.6}$$

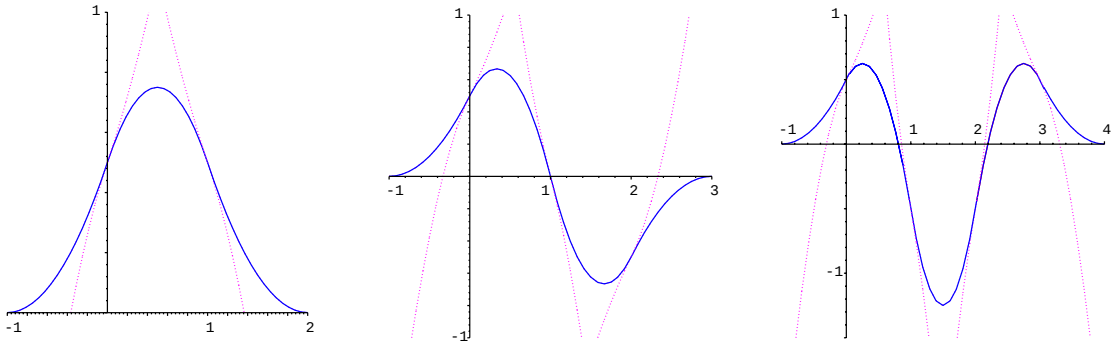


Figure 16: From left to right, convolution of three block kernels, convolution of three block kernels with one derivative operator and convolution of three block kernels with two derivative operatos. After x, y normalisation they give the bump smoothing kernel and the bump wavelets with one and two vanishing moments.