CS – 4802
Final Project

Wavelet Transform using Haar Wavelets

Submitted by: Jiri Sumbera
Submitted to: Dr. Taylor
Submitted on: 02-12-01

Wavelet Transform using Haar Wavelets

Introduction

Image transforms are very important in digital processing they allow to accomplish less with more. For example the Fourier Transform may be used to effectively compute convolutions of images[1] or the Discrete Cosine Transform may be used to significantly decrease space occupied by images without noticeable quality loss.
Wavelet Transform (WT) is a relatively new concept as a whole, even it though it incorporates some of the transforms, which have been know for long time. For fuller discussion of the topic, the reader is strongly advised to read the Appendix and the sources contained in the bibliography.

Problem Statement
Within application developed for CS-4802 labs:

- add the ability to perform both **forward and inverse WT using the Haar functions** as the set of basis functions
- add the ability to: - threshold frequencies in the wavelet domain
  - extract image using only first n x n coefficients

Secondary Goals:
- investigate on the usability of WT as an effective compression technique.

Implementation

1) Within the previously developed source code new 'JWavelet' class responsible for handling of wavelets was created. The source code for the 1-D WT implemented in the functions FastWT and Haar_2 comes directly from [1] p.267-269.

2) Secondary functions necessary for thresholding and frequency removal were implemented.

3) Due to the fact that the wavelet coefficients are real numbers (i.e. both positive and negative) two methods of displaying (not storing!) the image were created:
   a. **true mode** – negative numbers in the interval (-absmax[2], 0) are represented by graylevels in the range 0-127, positive numbers (0, absmax) range 129-255 and zero values are 128.
   b. **absolute value mode** – absolute values of coefficients are taken and the interval (0, absmax) is represented by graylevels 255-0.

   Both methods also use logarithmic conversion to spread out the values evenly.

[1] as in the frequency domain convolution is simply multiplication
[2] absmax = max (|max|,|min|)

Results

Wavelet transform

The following pictures show a 256x256 original and its true mode and absolute value mode wavelet domain representations as well as a schematic picture showing how the frequency information is stored[3]. It is not surprising to see the original image emerge in each of the sub-rectangles as the position of the coefficients is directly related to the translation parameter of the WT[4].

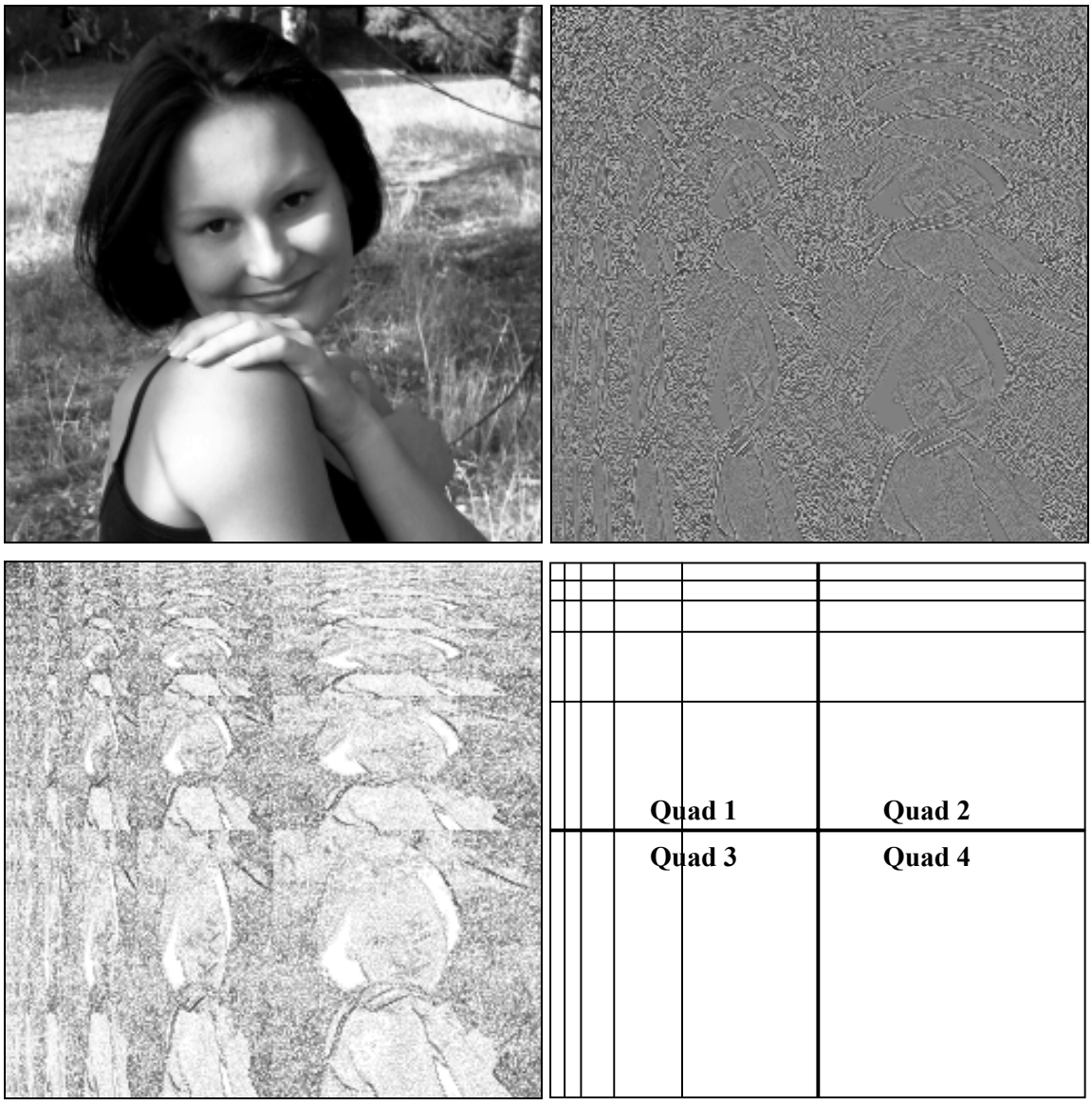


Figure 1 – a) original picture, b) true mode wavelet representation, c) absolute mode wavelet representation, d) schematic diagram of frequency content storage within the transform

[3] (the relation between j and frequency is $f \approx 2^j$, subscripts refer to horizontal and vertical information)
[4] Again, for more details why this is so, please refer to the Appendix.

The absolute mode image shows more clearly where the important coefficients (i.e. non-zero) really are, whereas the true mode image shows that positive and negative coefficients usually occur together (i.e. when a sharp edge is present in the image).

Effect of thresholding

The original wavelet domain representation contains many regions with zero values (show as black in Fig. 1-c). In fact, the number of zeros[5] is 3992, which represents about 6.1% of all the points. The frequency coefficients range from –40 to 111. The following pictures show the effect of thresholding on the reconstructed image.

Thresholding in the interval (-1,1) (i.e. all coefficients between –1 an 1 were set to zero) increased the abundance of zeros to 45.1%. Notice, the image bears almost no signs of quality loss.

Figure 2 – a) thresholded in (-1,1) b) thresholded in (-2,2) c) thresholded in (-4,4) d) thresholded in (-8,8)

[5] using the **double** storage type computing precision

Threshold (-2,2) increased the abundance of zeros to 64.1% and first quality depreciation can be observed. Threshold (-4,4) increased the percentage of zeros to 82.8%. It is worthwhile noticing that quality loss is most obvious in regions with gradual changes in intensity. On the other hand, edges and regions of high contrast are still quite clearly distinguishable. Threshold (-8,8) deteriorates the quality of the image considerably, however it should be noted that only 5.0% of the points were left non-zero.

Since the maximum magnitudes of negative and positive coefficients are not the same (the lowest coefficient being –40, highest 111), the next thresholding effect to be investigated is the importance of each. The levels were chosen to split up the highest thresholding region of the previous set into two, i.e. (-8, 0) and (0, 8) as well as to exclude the extreme values, i.e. (-40, -20) and (56, 111).

Figure 3 – a) thresholded in (-8, 0), b) thresholded in (0, 8), c) thresholded in (-40, -20), d) thresholded in (55, 111)

Due to the fact that none of the first two images appears significantly better than the other and also that the percentage of zeros induced by the two operations are 50.2% and 50.9%, it can be concluded that coefficient values close to zero have equal contribution independent of their sign. The third image (zero % = 6.2%; zero count = 4089) reveals that these coefficients carry important information, which should not be discarded. The fourth image reveals even more: due to the fact that zero count changed by one, it was easy to deduce it was the value of the first pixel (which corresponds to the overall graylevel[6]).

Effect of Extraction using only NxN coefficients[7]

The idea behind this approach is to be able to extract at least some amount of information about the whole image from a low volume of data[8] - also known as progressive extraction.

Figure 4 – images extracted using the first a) 32x32, b) 64x64, c) 128x128, d) 96x96 coefficients

[6] Please refer to the appendix, to find out why the first pixel carries the graylevel information.
[7] In fact, the extraction was simulated by setting the other coefficients to zero.
[8] This is useful during image download as an overall, continuously improving image may be reconstructed.

The images on Fig 4a)b)c) show how the effect of the NxN extraction is equivalent to image pixelizing or downsampling[9]. In addition, the last image shows NxN extraction, where N is not a power of 2. Clearly 96x96 pixels contain 225% more information than a 64x64 image, however this extra data is unevenly distributed as the precision doubles in the first quadrant only, the second and third quadrants are only partially affected[10].

Conclusion

In the above pages it was shown that reasonable level of thresholding may greatly reduce the number of data points required for holding the data at a relatively low quality loss. Such a thresholding is appropriate only in the case of coefficients whose absolute values are low. In addition, the various thresholds shown that the at first unexplainable asymmetry in the extreme values of negative and positive coefficients could be explained by asserting the spectrum was symmetrical, the only asymmetrical value being the overall graylevel coefficient stored in pixel [1,1][11].

Finally it was also shown that wavelet transform is capable of supporting progressive extraction[12], which makes it ideal for bandwidth-limited applications, e.g. web-based graphics.

Further possible directions of improvement (which could not be pursued due to the limited scope of this paper) include:

- investigating the effect of rounding-off of the wavelet coefficients to the nearest integer, or fraction possibly with adaptive rounding-off.
- investigating the whole nature of the wavelet coefficients spectrum with the aim of reduction in the storage space required to store the coefficients
- implementing other wavelet basis, such as the Daubechies basis

Bibliography

[1] *J.R.Parker*, **Algorithms for Image Processing and Computer Vision**, John Viley 1996, p. 250-254

[2] *RobiPolikar,* **The Wavelet Tutorial**, http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html

[9] which is exactly what it is – frequency f in a large image corresponds to frequency 2f in a downsampled image

[10] In quadrant 2, resolution improves in the vertical direction, while in quadrant 3 in the horizontal one

[11] In other words the original wavelet spectra could be greatly enhanced if the first pixel was excluded from the calculations of the extreme values – see the appendix for more.

[12] provided the data is accessed in a manner as to follow two sides of an increasing square as opposed to classical access by consecutive lines. This approach would also lead to great savings for images which are not powers of two (as the consecutive length of zeros would be much longer, i.e. ideal for RLC – see appendix for more

Appendix I - Other image-related concepts

The images below show the improvements in wavelet domain representation obtained by ignoring the first wavelet coefficient during the scaling.

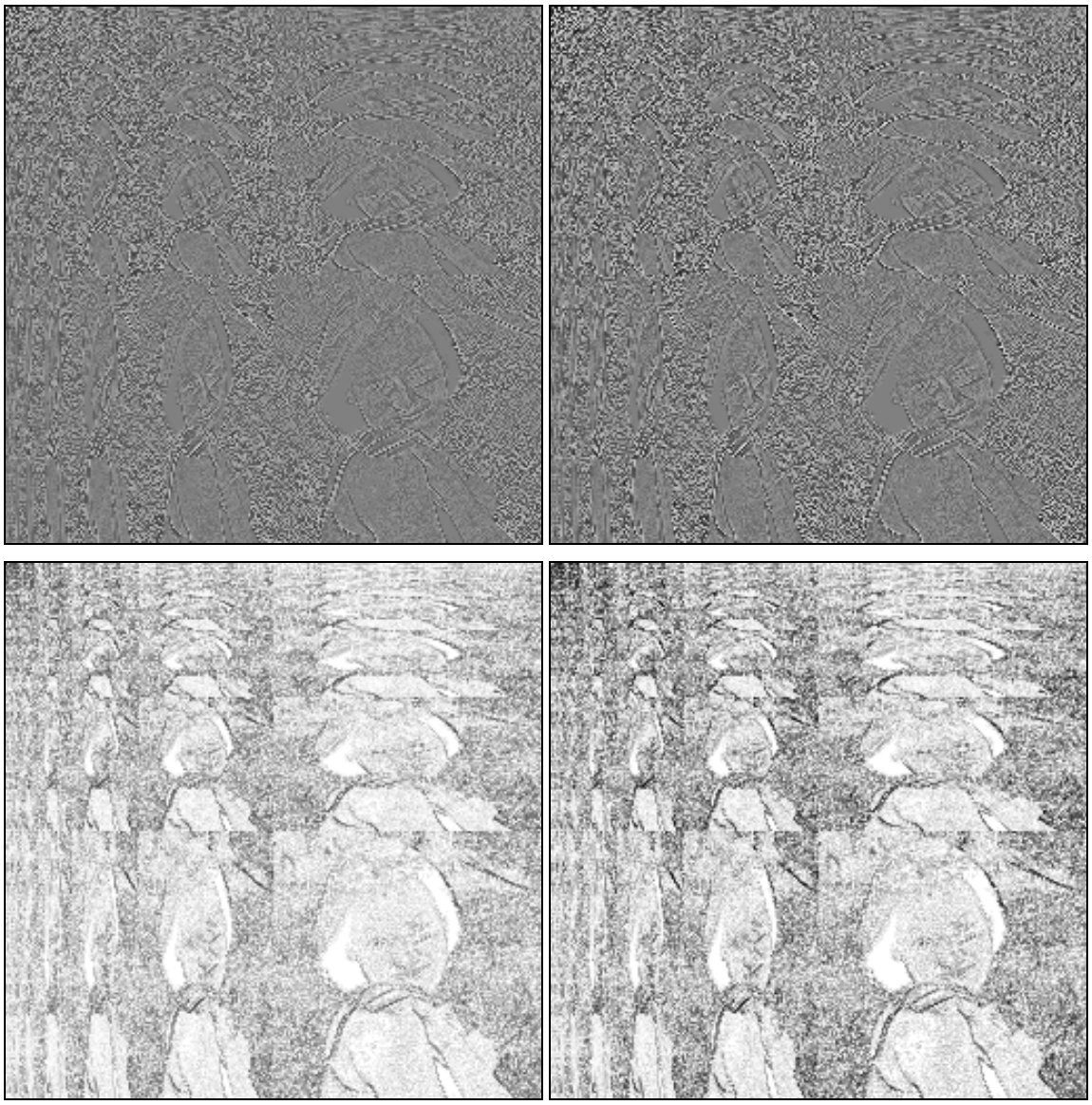


Figure5 – true mode wavelet representation – a)without ignoring the first pixel, b) ignoring it absolute mode wavelet representation – c)without ignoring the first pixel, d) ignoring it

The images below show the effect on the wavelet coefficients by using image dimensions, which are not powers of two. Since the images are padded with zeros, corresponding multiple stripes in the wavelet domain are created

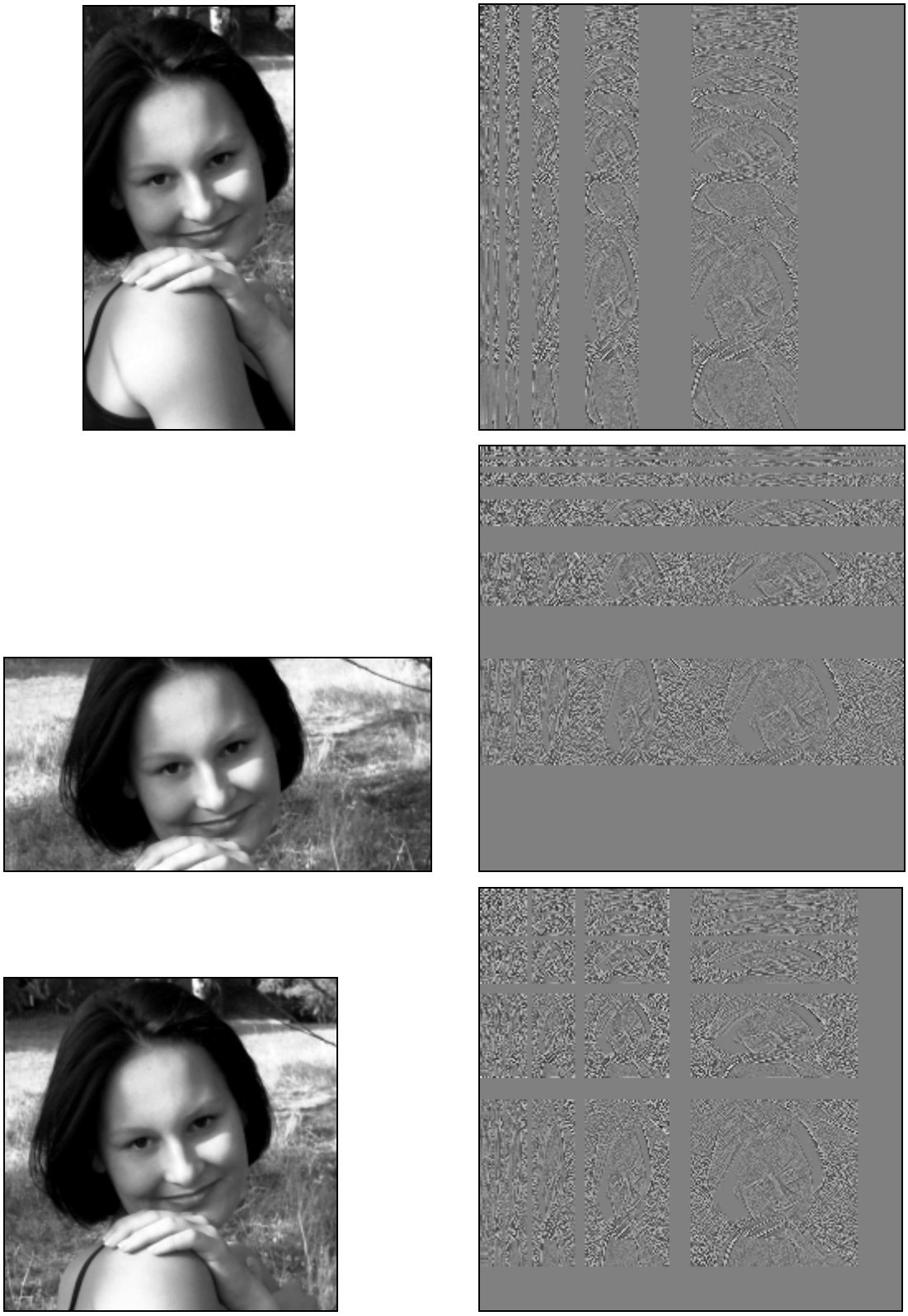


Figure 6 – original images and their WT for sizes 128x256, 256x128 and 202x202

Appendix II - Mathematical Introduction to Wavelet Transform

Before touching on the mathematical theory behind WT, credits should be given again to the following two sources.

[1] *J.R.Parker*, **Algorithms for Image Processing and Computer Vision**, John Viley 1996, p. 250-254
[2] *RobiPolikar,* **The Wavelet Tutorial**,
http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html

Source [2] was used as the main source of pictures and general introduction into wavelet theory. Source [1] provided mathematical and practical information on WT using the Haar basis.

Vector and its coefficients

Any vector (function) can be written as a linear combination of basis vector (functions)

$$f(x) = \sum_{i=1}^{N} c_i g_i(x)$$

where N is the dimension of the vector space (e.g. 3 for our Euclidian world or *inf* for the space of square-integrable functions of one variable on an given interval (a, b) = $L^2(a,b)$).
The constants c_i can be calculated by the scalar product of the function *f(t)* with the corresponding basis function $g_i(x)$.

$$scalar\ product\ (f, g_i) = \langle f, g_i \rangle = \int f(x) g_i^*(x) dx$$

In other words, every function of the $L^2(a,b)$ space the can be written in the form.

$$f(x) = \sum_{i=1}^{N} \langle f, g_i \rangle g_i(x)$$

For example if trigonometric functions are used as the basis functions (i.e. $g_n(x) = 1, \cos x, \sin x, \cos 2x, \sin 2x, \ldots$), the well-known Fourier series is obtained.

Fourier Transform and its Limitations.

The one-dimensional Fourier Transform (FT) and its inverse are specified by the following equations

$$X(f) = \int_{-\infty}^{\infty} x(t) \bullet e^{-2j\pi ft} dt \ldots\ldots (1)$$

$$x(t) = \int_{-\infty}^{\infty} X(f) \bullet e^{2j\pi ft} df \ldots\ldots (2)$$

Robi Polikar, Ames IA. 1994

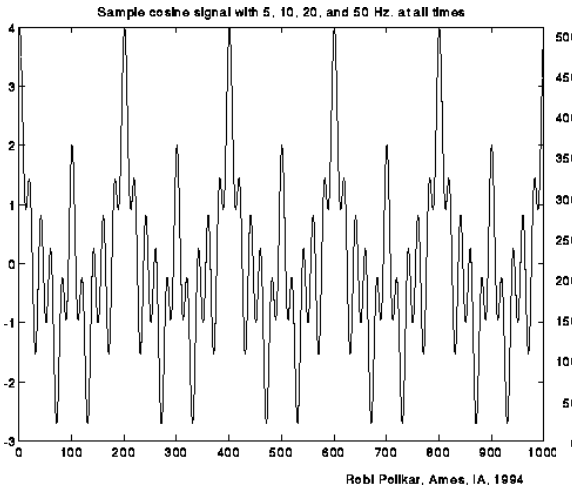
In the view of the above definitions, FT can be though of as just a generalized version of the Fourier series – the summation is replaced by the integral and a 'countable' set of basis functions is replaced by a uncountable one, this time consisting of cosines for the real part and sines for the complex part of the analyzed signal[13].
Due to the fact that each of the basis functions spans over the total length of the analyzed signal, this creates what is known as time-frequency resolution problem, which is best illustrated with an example.

[13] which can now be a complex number, that is another generalization of FT

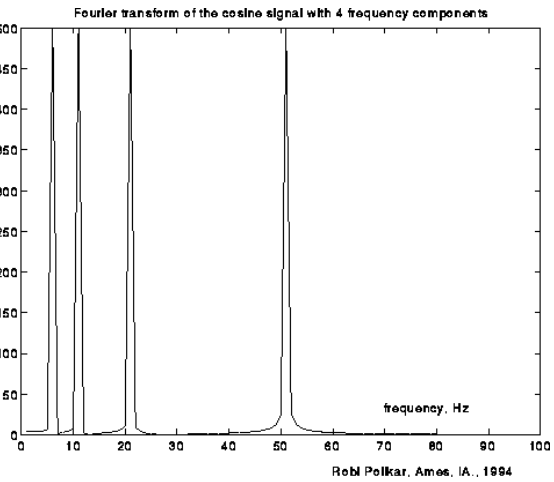
Suppose the Fourier transform of a simple signal containing a superposition of four different frequencies (namely 5, 10, 20 and 50Hz) is taken

$$x(t)=cos(2*pi*5*t)+cos(2*pi*10*t)+cos(2*pi*20*t)+cos(2*pi*50*t)$$

time-domain signal

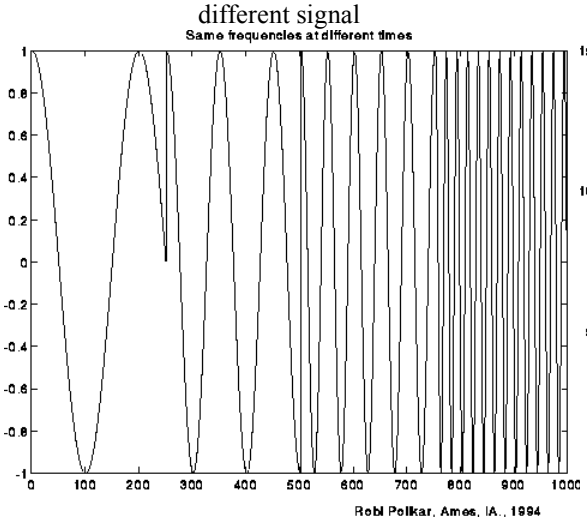


frequency-domain signal

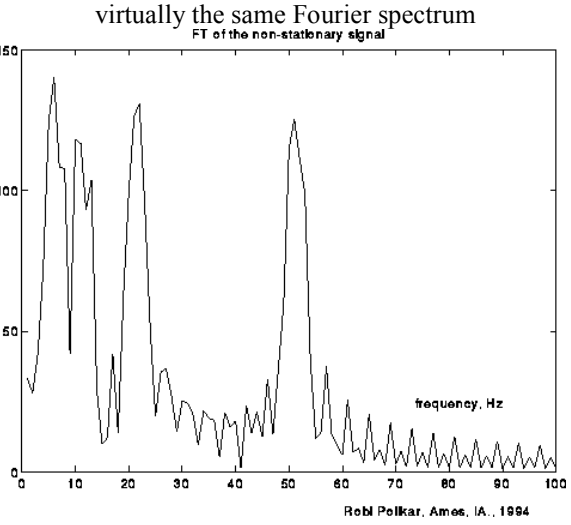


The Fourier spectrum clearly reveals the presence of these four frequencies. Now consider a signal contains the same frequencies as the previous one, only this time they are not superimposed, but appear sequentially as the picture below shows.

different signal



virtually the same Fourier spectrum



It should be noted how similar the frequency-domain representations of both signals are. (The small peaks are purely a consequence of discontinuities present in the second signal ("ringing") and could be completely eliminated if a the signal was chosen more carefully.) The important conclusion is that FT does give information about what frequencies are present in the signal, however it lacks the ability to correlate the frequencies with the time of their presence. This fact does not even come into consideration when only stationary signals are processed, however as soon as non-stationary signals are to be analyzed, it becomes a serious handicap

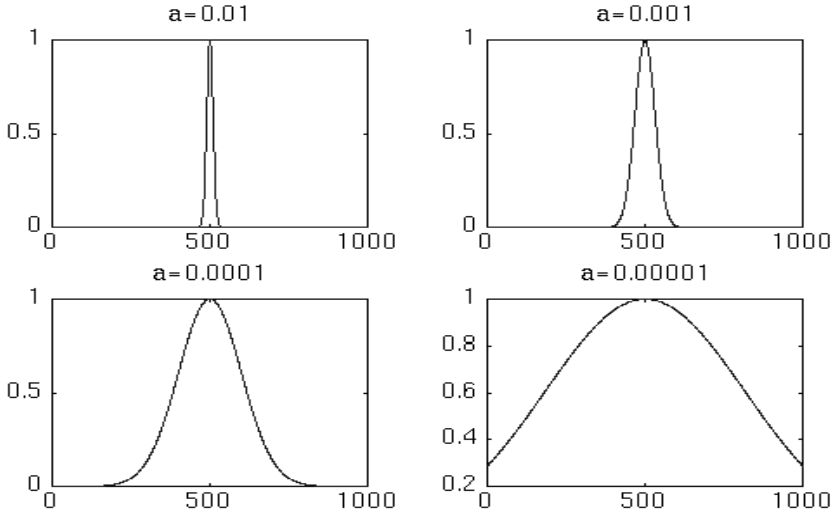
Short Term Fourier Transform

One approach of getting past the limitations is provided by the 'windowed' version of FT, known as Short Term Fourier Transform (STFT). The basic idea is select a short time frame within the transformed function and then perform standard FT. This is accomplished by multiplying the function by a fast decreasing window function (i.e. a function which is significantly non-zero on a finite interval). The window function has to be given a position along the signal axis and consequently additional parameter to specify the transform arises. In other words, STFT is a transform from one-dimensional space into two-dimensional one.

$$STFT^{w}{}_{f(t)}(f,t) = \int \left[f(t) w_t^*(t-t') \right] e^{-i2\pi ft}\, dt$$

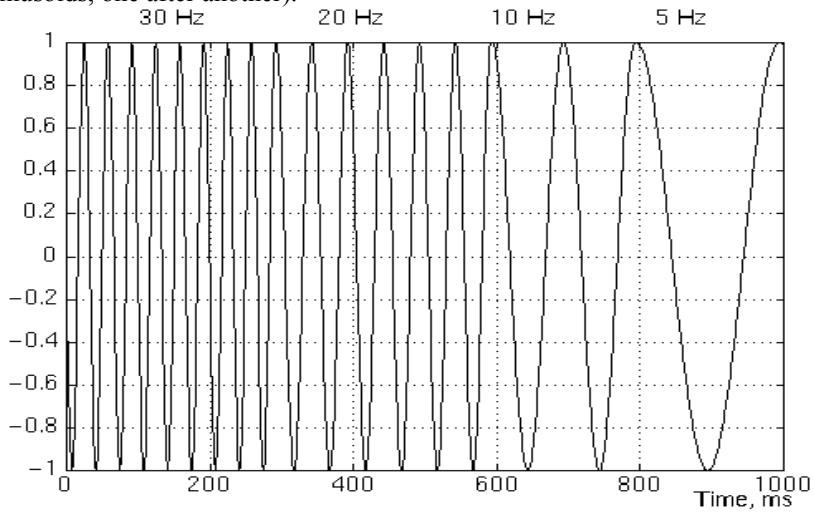
The picture below shows how a window function (in this case a Gaussian function**)** can be modulated using an extra parameter to provide different window widths.

$$w(t) = e^{-\frac{at^2}{2}}$$

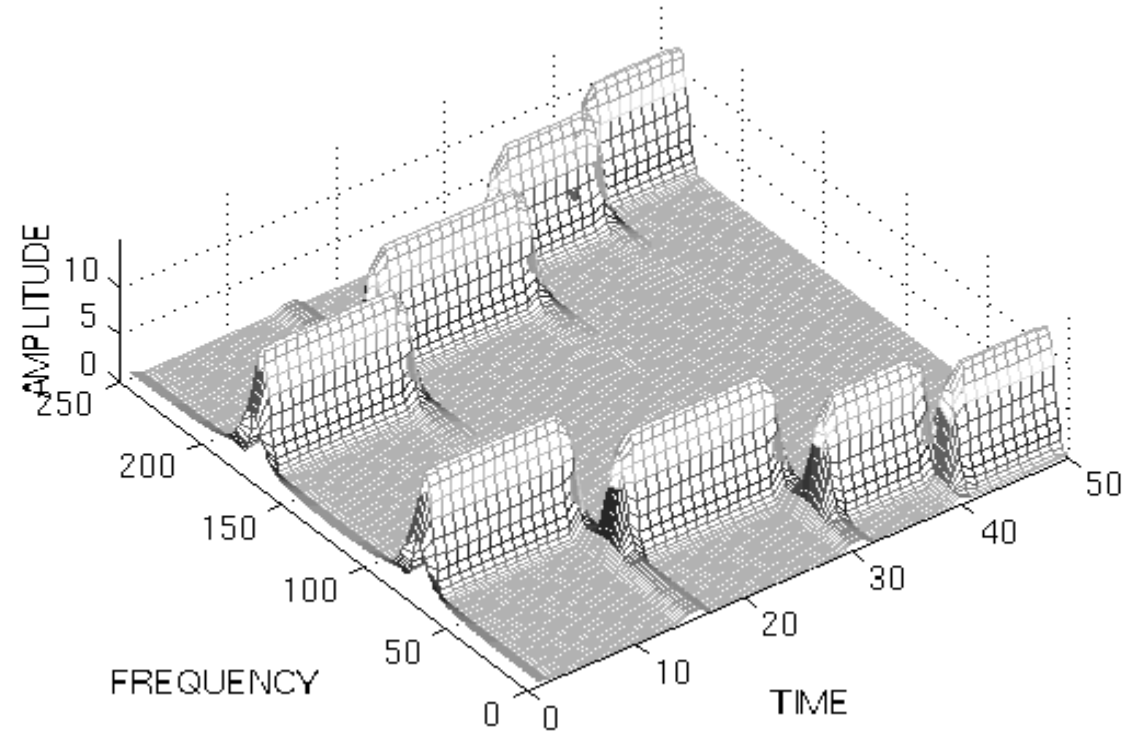


The size of the window, determines the resolution of the transform. If a narrow window is used (which is obtained by using a relatively large value of parameter *a*) good time resolution is obtained at the expense of poor frequency resolution. On the contrary, wide windows give good frequency resolution and poor time resolution (small value of parameter *a*).

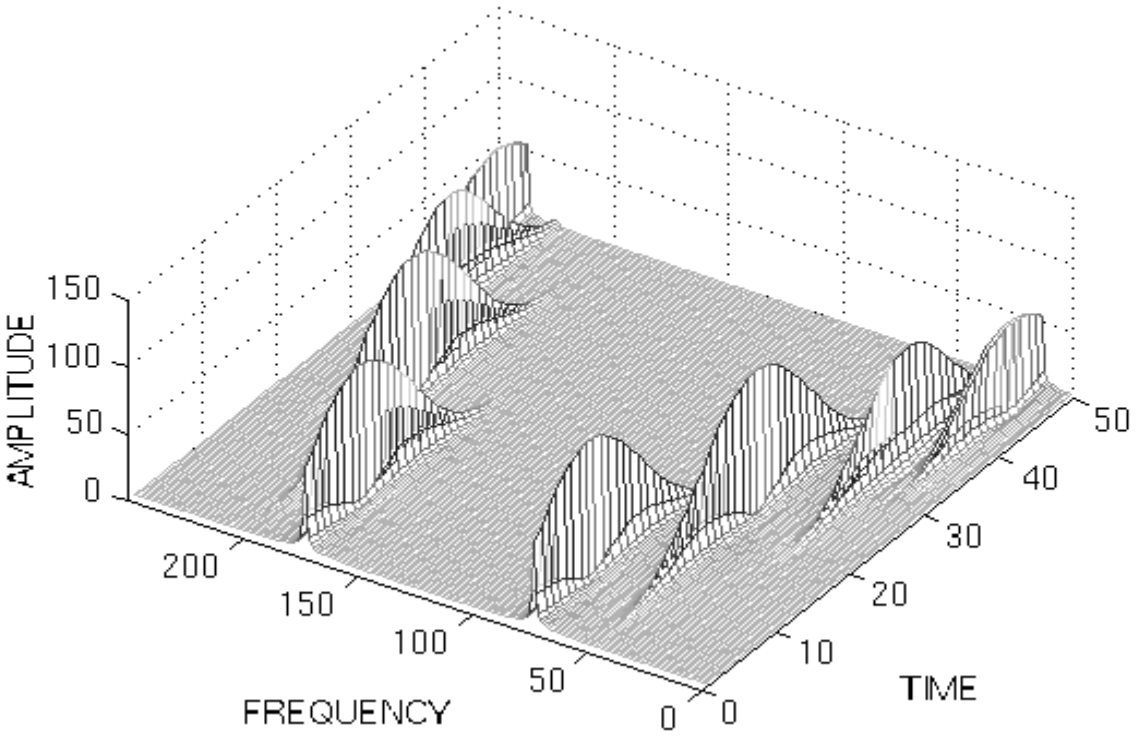
The pictures bellow illustrate what is meant by poor and good resolutions. Assume the following signal (i.e. four different sinusoids, one after another).



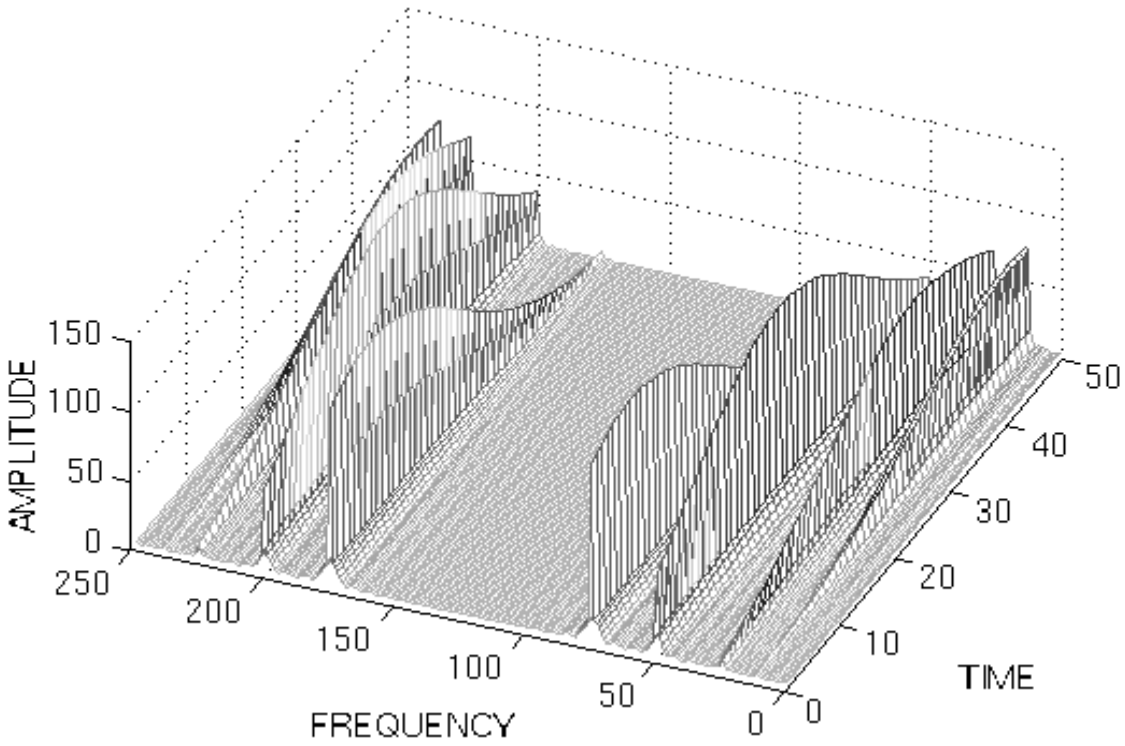
If a narrow window is used, the time resolution is rather good, but the frequency information is blurred.



If a wider window is used, the frequency resolution improves, but at the expense of time resolution.



Even wider windows almost completely destroy the time information and the obtained image resembles the original Fourier spectrum (in fact $\lim_{a\to\infty} STFT = FT$)



Continuous Wavelet Transform

The above problem can be partly resolved (although not completely – see Heisenberg's Principle below) by using, what is known as Multiresolution Analysis (MRA) technique. Similarly to STFT, the Continuous Wavelet Transform (CWT)is also a transformation from one-dimensional space into two-dimensional space using a gliding window function.

$$W_{f(x)}(s,t) = \int f(x) w^*_{s,t}(x) dx$$

(compare with $STFT^{w}{}_{f(x)}(f,t) = \int [f(x) w^*_t(x - 't)] e^{-i2\pi xt} dx$)

The main two differences between CWT and STFT are:
1) The Fourier transforms of the windowed signals are **not taken**.
2) The **width of the window is changed as the transform is computed** for every single spectral component.
3) Instead of frequency, the scale parameter *s* is used. Scale can be thought of as being the reciprocal of frequency. In addition to that, it is also used to scale the defining function of the whole group of windows– the **mother wavelet *w(x)*** – as the equation below shows.

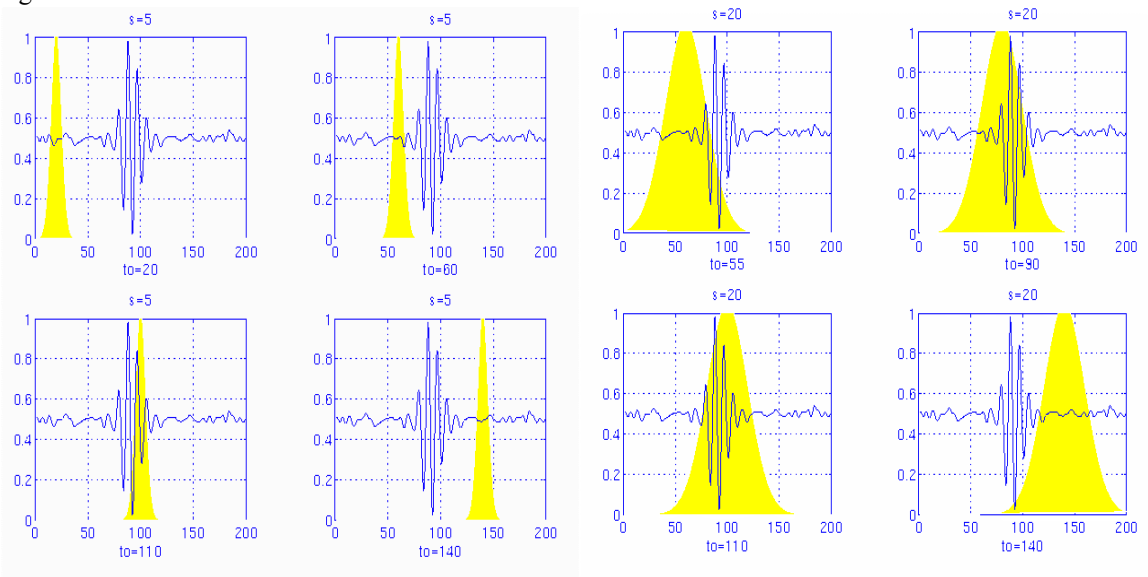
$$w_{s,t}(x) = \frac{1}{\sqrt{s}} w\left(\frac{x-t}{s}\right)$$

The wavelet window has to fulfill two criteria:

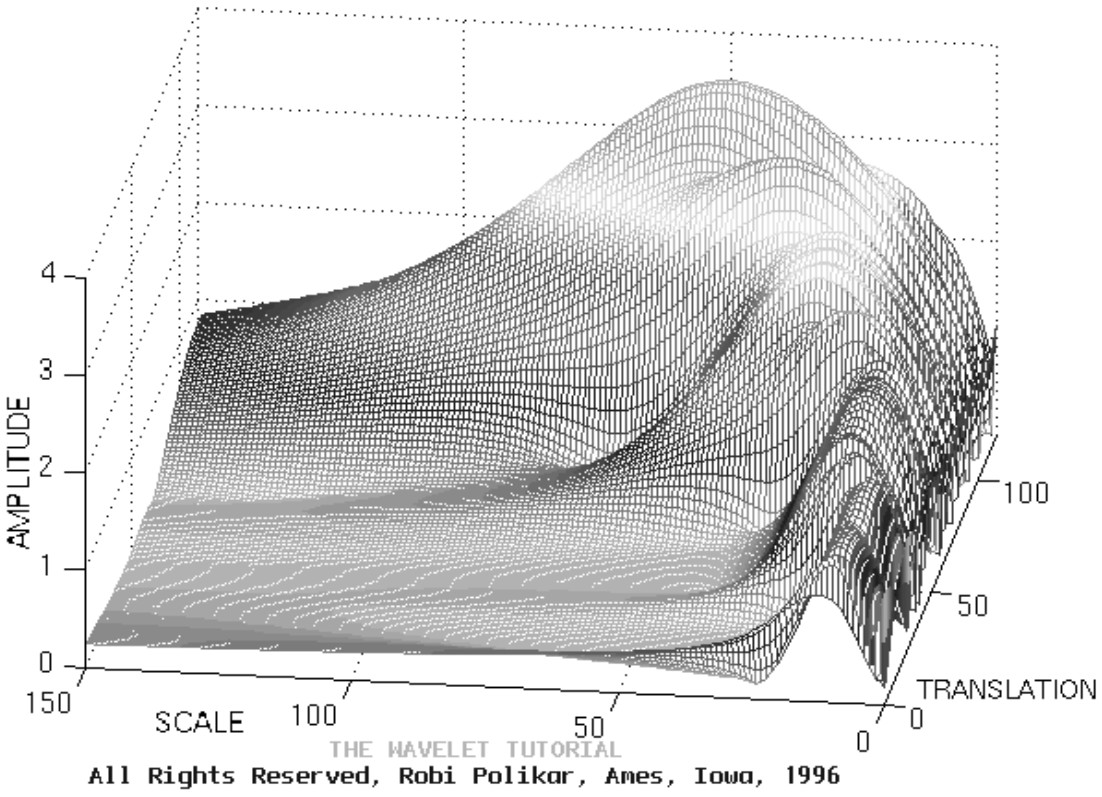
1) The function has to be of local character (i.e. finite-supported or fast decreasing function as STFT window function)

2) It has to be oscillatory (i.e. $\int w(x)dx = 0$)

The following pictures simulate how the wavelet function is translated and scaled along the non-stationary signal.

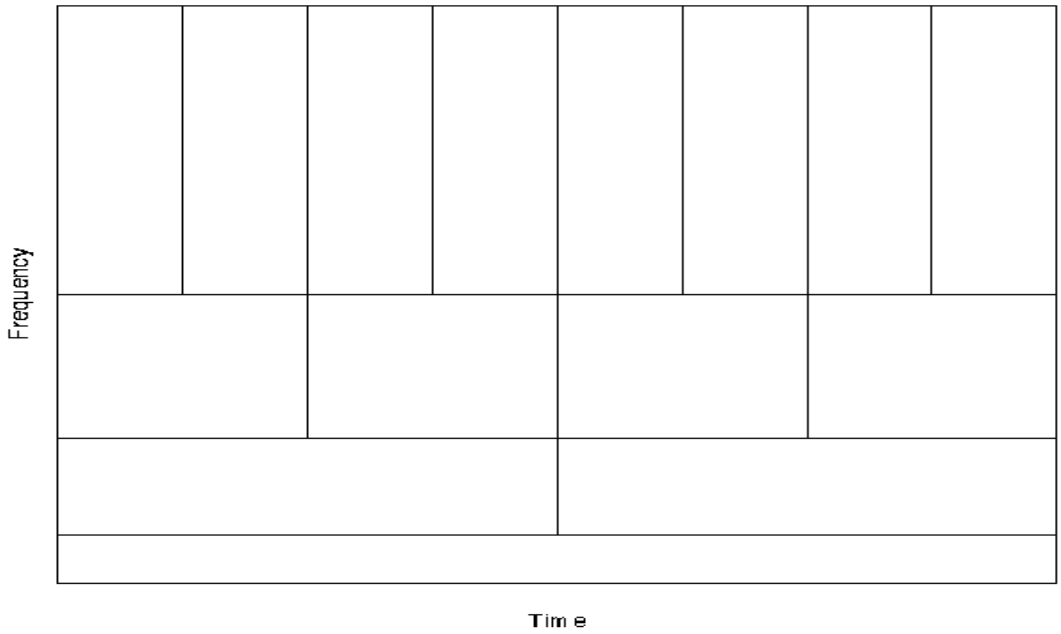


The following picture then shows the CWT of the same signal as investigated previously using STFT. It should be noted that translation is strictly correlated with time and scale corresponds to the reciprocal of frequency. The picture illustrates the fact that CWT has a good time and poor frequency resolution at high frequencies (narrow scales at low scale values imply poor frequency resolution), and good frequency and poor time resolution at low frequencies (wide scale corresponds to ambiguous scale or better frequency resolution).



The picture below illustrates **Heisenberg's uncertainty principle** for time and frequency, which gives an absolute lower bound on the uncertainty of their product.

$$(\Delta f)(\Delta t) \geq \frac{\pi}{4}$$



The areas of all rectangles are the same even though they have different widths and heights. Similarly good time resolutions (small width in the above picture) will inherently possess poor frequency resolutions (large value of height) and vice versa good frequency resolution is bound to be poorly located in time. It may be interesting to note that STFT would give rectangles with constant shape. This flexibility of CWT is its primary advantage over STFT – WT provides for multiresolution analysis within the space of a single transform.

Discretization of Continuous Wavelet series

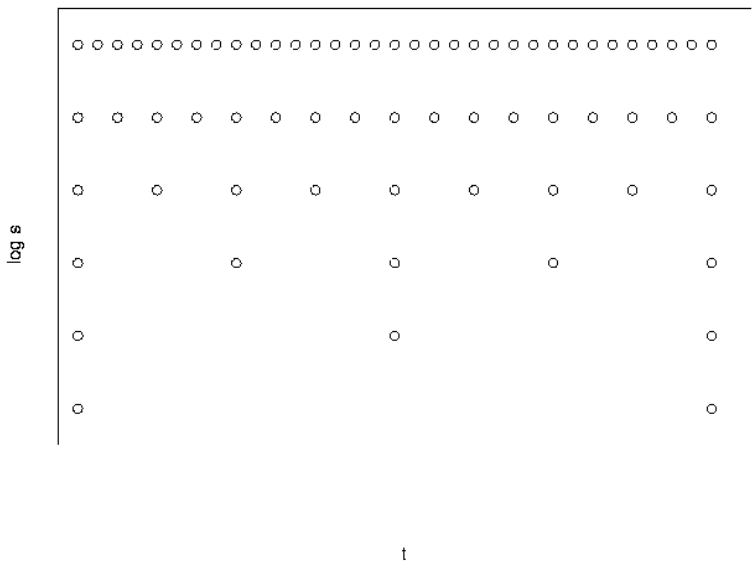
It is apparent that for most functions it is not practical to compute STFT and WT transforms using analytical equations. Instead the discretized version of these transformed is usually performed. Most of these calculations are performed using computers, which already work with discrete, sampled data. Normally, the sampling is done with a uniform sampling rate, however for the case of WT the sampling rate can be varied without loss of precision.

Nyquist's rule

If the s-t plane is sampled with rate N_1 at scale s_1, the sampling rate at a different scale s_2 is proportional to the ratio of scales.

$$N_2 = \frac{s_1}{s_2} N_1 = \frac{f_2}{f_1} N_1$$

The above equation implies what is intuitively understood – lower frequencies (higher scales) need to be sampled at lower rates than higher frequencies (lower scales). This significantly reduces computing time without any loss of precision.



Similar to the relationship between continuous Fourier transform, Fourier series and the discrete Fourier transform, there is a continuous wavelet transform, a semi-discrete wavelet transform (also known as wavelet series) and a discrete wavelet transform.

If we assume a set of functions of the form

$$w_{s,t}(x) = \frac{1}{\sqrt{s}} w\left(\frac{x-t}{s}\right)$$

and sample them according to the rule $s = s_0^j$ (and correspondingly $t = k\, s_0^j\, t_0$) the functions transform to

$$w_{j,k}(x) = \frac{1}{\sqrt{s_0^j}} w\left(\frac{x - k s_0^j t_0}{s_0^j}\right) = s_0^{\frac{-j}{2}} w\left(s_0^{-j} - k t_0\right)$$

If $\{w_{j,k}\}$ constitutes an orthonormal or biorthogonal basis, than the WT series may be defined as follows

$$W_{j,k}^{f(x)} = \int f(x) w_{j,k}^{*}(x) dx$$

and the function f(x) can be written in terms of a linear combination of the basis functions

$$f(x) = \sum_{j,k} c_{j,k} W_{j,k}^{f(x)}$$

In other words, the function is defined in terms of its coefficients using the $\{W^{f(x)}_{j,k}\}$ basis.

For the purposes of digital image processing, it is very convenient to scale these wavelets by powers of two. In other words $s_0 = 1/2$ and t = 1

$$w_{j,k}(x) = 2^{\frac{j}{2}} w\left(2^j - k\right)$$

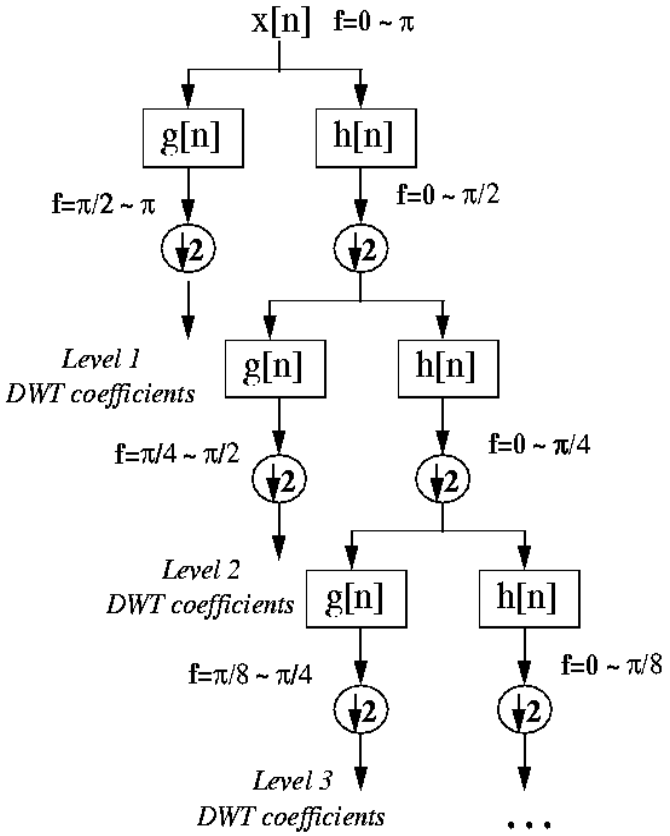
Discrete Wavelet Transform

The following algorithm is also known as **subband coding**.
Assume the highest frequency component that exists in a signal will be π radians. To provide sufficient information about the signal without its loss due to sampling, the signal is sampled at **Nyquist's rate** (which is twice the maximum frequency that exists in the signal). Next, the signal is passed through two filters – one half band lowpass and one half band high band filter. The output of the highpass filter is preserved. The remaining part of the procedure is best summarized by Robi Polikar:

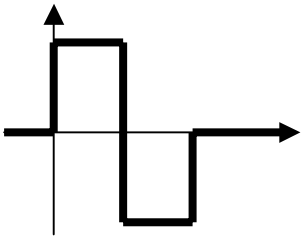
> *"After passing the signal through a half band lowpass filter, half of the samples can be eliminated according to the Nyquist's rule, since the signal now has a highest frequency of π/2 radians instead of π radians Simply discarding every other sample will* ***subsample*** *the signal by two, and the signal will then have half the number of points. The scale of the signal is now doubled. Note that the lowpass filtering removes the high frequency information, but leaves the scale unchanged. Only the subsampling process changes the scale. Resolution, on the other hand, is related to the amount of information in the signal, and therefore, it is affected by the filtering operations. Half band lowpass filtering removes half of the frequencies, which can be interpreted as losing half of the information. Therefore, the resolution is halved after the filtering operation. Note, however, the subsampling operation after filtering does not affect the resolution, since removing half of the spectral components from the signal makes half the number of samples redundant anyway. Half the samples can be discarded without any loss of information. In summary, the lowpass filtering halves the resolution, but leaves the scale unchanged. The signal is then subsampled by 2 since half of the number of samples is redundant. This doubles the scale."*

[1] -The Wavelet Tutorial by Robi Polikar, part 4

The above process can now be recursively applied on the subsampled output of the lowpass filter, always storing the part of the signal produced by the highpass filter. In other words the DWT can be though of as consisting of two processes. The filtering is associated with retrieving of the wavelet coefficients corresponding to the current scale (i.e. highpass filter) and the scaling corresponds to subsampling of the remaining data (i.e. lowpass filter subsampling).
The picture on the right shows this process graphically. g[n] represents the highpass filtering and retrieval of wavelet coefficients. h[n] corresponds to lowpass filtering and subsampling.



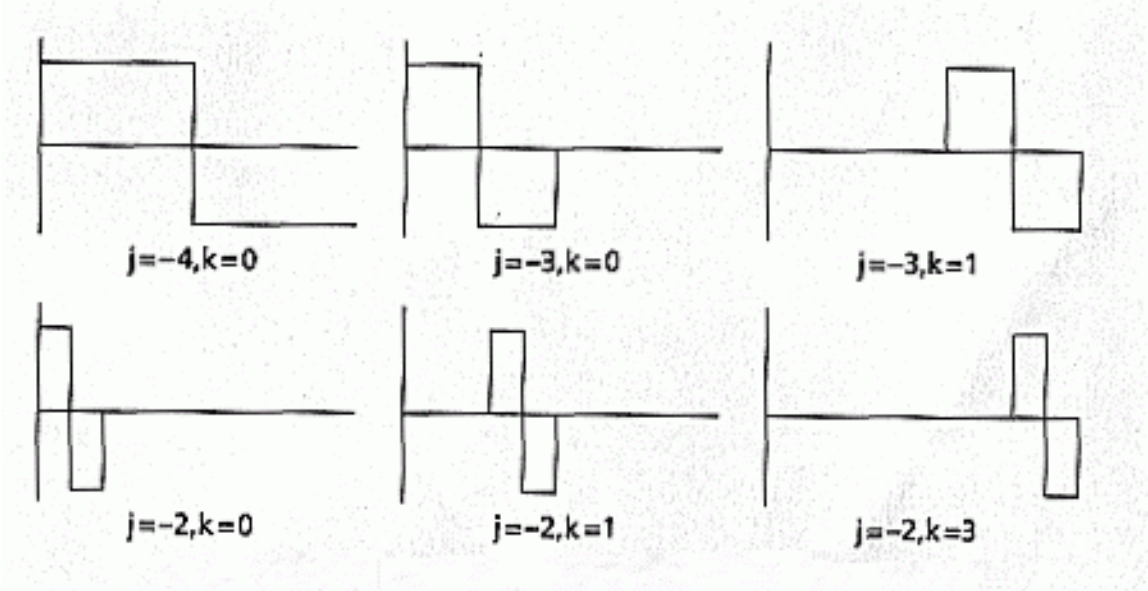
The Haar wavelet



The Haar wavelet is the simplest wavelet.

$$w_H(x) = \begin{cases} 1 & 0 < x < 1/2 \\ -1 & 1/2 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

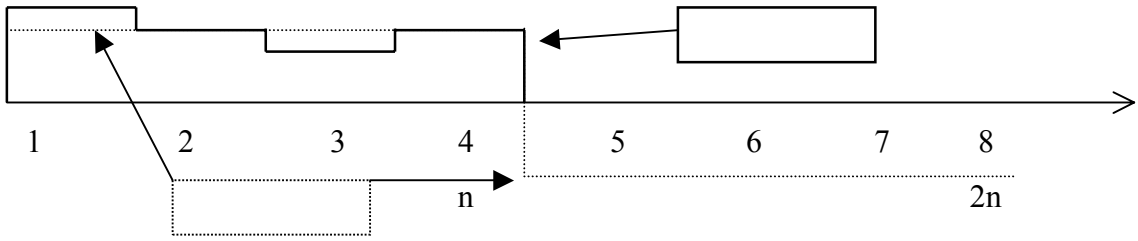
The mother wavelet obviously satisfies the two wavelet requirements, as it is both **local** and **oscillatory**. The picture below shows the shapes of Haar wavelets for various scales and translations.



Fast Wavelet Transform

The algorithm for the one-dimensional Fast Wavelet (Haar) Transform was implemented using the optimized code from [1] as the basis source code. In a nutshell it can be summarized as follows:

Assume the data is contained in vector with length, which is a power of 2, e.g. n = 256 = 2^8 (i.e. a row of an image). Consequently the wavelet corresponding to the highest possible frequencies that could be contained in the vector (i.e. the smallest scales as $s = 2^j$, $j = -1 \Rightarrow s = 2$) spans over two successive data points (pixels) and the lowest possible frequencies (i.e. the largest scales as $s = 2^j$, $j = -8 \Rightarrow s = 2^8$) span over the whole length of the vector. Interestingly enough, one more frequency – the sub-frequency j = -9 – can be contained in the image. This frequency corresponds to the mean value of all data (i.e. average level of gray) and spans over twice the vector length as the picture below shows (n=4)



Consequently the range of frequencies that are contained in the image can be specified in either scale (s spans from 2 to 512) or in terms of its exponent (j spans from –1 to –9)

The following is an implementation of subband coding for Haar wavelets:

1) The image is passed through two filters: a low-pass filter and a high pass filter. The low-pass filter lets only frequencies below a certain value through (i.e. $j < -1$) and the high-pass filter similarly allows only the highest frequencies in the image (i.e. $j = -1$) through. The wavelet coefficients for those frequencies (i.e. $c_{-1,k}$) are then calculated. (again it is easy to see that to completely cover a vector of length n with scale 2 wavelets, only n/2 wavelets and hence only $n/2 = 256/2$ coefficients are necessary)
2) Next, step 1 is repeated for those frequencies, which originally passed through the low-pass filter. In other words frequencies with j = -2 are extracted. Since the sampling rate required for storing these frequencies is now half the original only $n/4 = 256/4$ coefficients are required (i.e. it takes $n/4$ wavelets with scale 4 to cover the vector).
3) The filtering and sampling process is then repeated for all remaining frequencies ($j = -3..-8$) requiring progressively fewer and fewer coefficients to be kept ($n/8..n/256 = 256/8..256/256$). Finally the coefficient of the sub-frequency is computed.

Due to the fact that the total number of coefficients for scales j = -1..-8 is $\sum_{j=-1}^{-8} \frac{n}{2^{-j}} = \sum_{j=1}^{8} \frac{256}{2^{j}} = 2^{8} - 1$, all the necessary coefficients (including the sub-frequency j = -9) can be stored in $2^8 - 1 + 1 = 2^8 = n$ spaces. In other words, the transform occupies the same number of data slots as the original data and thus can be stored in place of it. The most efficient way of storing the coefficients is by increasing frequency (i.e. decreasing scale).