Radboud University Nijmegen

Compression of images, Fourier and wavelets

Supervised by Theo Schouten

Maurice Samulski (m.samulski@iphicles.com)

July 2005

Contents

1 Introduction

Digital images have become an important source of information in the modern world of communication systems. In their raw form, these images require an enormous amount of memory. With the introduction of multimedia computing, the demand for processing, storing and transmitting images has increased exponentially. In the last two decades a considerable amount of research has been devoted to tackle the problem of image compression. There are two different compression categories: lossless and lossy. Lossless compression preserves the information in the image, which means that an image could be compressed and decompressed without loosing any information. In lossy compression information is lost but this is tolerated as it allows for a much higher compression ratio. Images need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable. Some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space. Lossy compression is useful in areas such as video conferencing, biomedical images, ECG and EEG diagrams, FBI fingerprint compression and a lot of other applications where compression plays a great role.

It's well known that about 200 years ago Joseph Fourier discovered that arbitrary functions could be represented by superposing sines and cosines. This sum or integral of sine functions multiplied by some coefficients ("amplitudes") is called the Fourier expansion. The physical interpretation of Fourier's result is that we can obtain every possible signal by placing a diversity of tuning forks together and then simultaneous hitting them with several strengths. Through fourier transformation we can analyse the measurement data of the frequency spectrum. By their definition, these functions are non-local and stretch out to infinity. They therefore do a very poor job in approximating sharp spikes. On the other hand, such an analysis is very suitable for analyzing stationary signals or composities of them, such as the sound being produced by one or several tuning forks. A special form of the Fourier transform, the Discrete Cosine Transform, is being used in the JPEG algorithm which will be explained in chapter 2.

However, a large class of signals from the physical world is not stationary. For the analysis of this type of signals both information from the time field and frequency field are important. In the past decades several solutions have been developed which are more or less able to represent a signal in the time and frequency domain at the same time. The idea behind these solutions is to cut the signal into several parts and then analyze the parts separately. It is clear that analyzing a signal in this way will give more information about when and where these frequencies arise, but it leads to a fundamental problem as well: how to cut the signal? In wavelet analysis the use of a fully scalable window solves this signal-cutting problem. The window is shifted along the signal and for every position the spectrum is calculated. Then this process is repeated many times with a shorter or longer window for every new cycle. In the end the result will be a collection of time-frequency representations of the signal, all with different resolutions which is in the literature often called multiresolution analysis. This concept will be explained in chapter 3.

In chapter 4 we experimentally compare the JPEG and JPEG2000 algorithms by compressing images which are known for their difficulty. The subjective quality results, as well as performance results will be presented.

2 Fourier Transforms

2.1 Fourier Series

The mathematical background on this transform was developed by the French mathematician Joseph Fourier, about 200 years ago. The basic idea behind this transform is that any signal can be seen as an infinite sum of a series of sinusoidal signals, where each sine can have a different amplitude and phase. In practical terms, it means that you can feed in an arbitrary signal and get out data that represents the individual frequencies and their amplitudes that make up the original data. This way of expressing a function is called the trigonometric expansion of a function which is a sum of the form

$$f(x) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(kx) + b_k \sin(kx)]$$

where the sum could be finite or infinite and k is an integer. Let's have a look at the following example. We let t be the independent variable representing time instead of x. A sine wave, such as $sin(kt)$, has a period of $\frac{2\pi}{k}$ and a frequency of k (i.e., vibrates k times in the $0 \leq t \leq 2\pi$). A signal such as

$$3\sin(t) - 100\sin(4t) - 20\sin(200t)$$

contains frequency components that vibrate at 1, 4 and 200 times per 2π-interval length. The component vibrating at a frequency of 4 dominates over the other frequency components because it has the largest coefficient. To see the relevance of this expression we take a look at a common task in signal analysis, data compression. The goal is to send a signal in a way that requires minimal data transmission. A possible approach is to express the signal f in terms of a trigonometric expansion, and then send only those coefficients that are larger in value than some specified tolerance value. The coefficients that are small and do not contribute substantially to the signal f can be thrown away. Another common task in signal analysis is the elimination of high-frequency noise, which can be done by setting the coefficients of the high frequency components (a_k and b_k for large k) to zero[3][13].

2.2 (Continuous) Fourier Transform

In this section we take a look at Fourier transform. The Fourier transform can be thought of as a continuous form of the Fourier series. A Fourier series decomposes a signal on $[-\pi, \pi]$ into components that vibrate at integer frequencies. Yet the Fourier transform decomposes a signal defined on an infinite time interval into complex exponential functions of different frequencies.

The way it does this, is defined by the following two equations:

$$FT(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} ft(t) \cdot e^{-2\pi i f t}\, dt \tag{1}$$

$$ft(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} FT(f) \cdot e^{2\pi i f t}\, df \tag{2}$$

where $i = \sqrt{-1}$, t stands for time, f stands for frequency and ft refers to the signal being processed. Note that ft refers to the signal in the time domain and FT refers to the signal in frequency domain. Equation (1) is the Fourier transform of ft(x) and equation (2) is the inverse Fourier transform FT(t) which is f(x). Important to notice is that the integration in equation (1) is over time. The information provided by the integral corresponds to all time instances, since the integration is from $-\infty$ to ∞ over time. However, the left hand side of equation (1) is a function of frequency. Therefore the integral is calculated for every value of f.[8]

That's why Fourier transform is not suitable for non-stationary signals, i.e. if the signal has time varying frequency. Only if the signal has the frequency f at all times, then the result obtained by Fourier transform makes sense[3].

The exponential term $e^{-2\pi i f t}$ in the two equations can also be written as:

$$\cos(2\pi f t) + i \cdot \sin(2\pi f t) \tag{3}$$

As you can see, equation (3) has a real part and imaginary part. The real part consists of cosine of frequency f and the imaginary part consists of sine of frequency f. Looking back at equation (1), we are basically multiplying the original signal $ft(t)$ with a complex expression which has sines and cosines of frequency f. After this we will integrate this product. If the result of this integration (which is nothing but some sort of infinite summation of the area under the sines) is a large value, then the signal $ft(t)$ has a dominant frequency component f. If the integration result in a small value, than this means that the signal does not have a major frequency component of f in it. If the integration result is zero, then the signal doesn't contain the frequency f at all.

To see how this Fourier transformation works, we will begin with a small example and compute the Fourier transform of a rectangular wave (figure 1(a)) defined as:

$$ft(t) = \begin{cases} 1 & \texttt{if} \quad -\pi \leq t \leq \pi \\ 0 & \texttt{otherwise} \end{cases}$$

For readability we let $\lambda = 2\pi f$. Now we have

$$ft(t)\, e^{-i\lambda t} = ft(t)\, (\cos\, \lambda t - i\, \sin\, \lambda t)$$

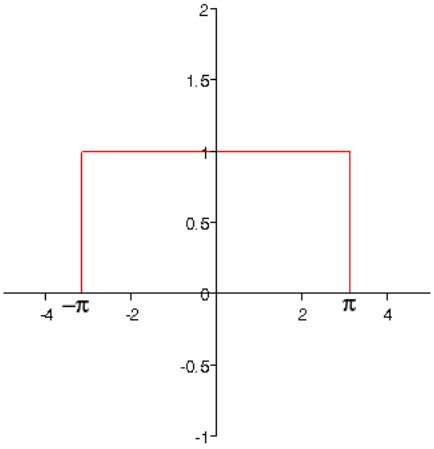
$ft(t)$ is an even function which means that it's symmetric with respect to the y-axis and that for each t, $ft(t) = ft(-t)$. Since $ft(t)$ is an even function, $ft(t) \sin(\lambda t)$ is an odd function, which means

that it's symmetric with respect to the origin and for each t, $f(-t) = -f(t)$. The integral of that odd function is zero over the real line.

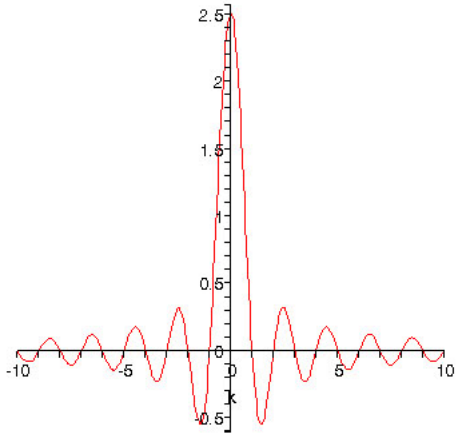
Therefore we could reduce the Fourier transform of $ft(t)$ to

$$\begin{aligned} FT(f) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} ft(t) \cdot \cos(\lambda t)\, dt \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} \cos(\lambda t)\, dt \\ &= \frac{\sqrt{2}}{2\sqrt{\pi}} \frac{2\sin(\lambda\pi)}{\lambda} \\ &= \frac{\sqrt{2}\sin(\lambda\pi)}{\sqrt{\pi}\lambda} \end{aligned}$$

As already mentioned, the Fourier transform measures the frequency component of $f(t)$ that vibrates with frequency f. In this example, $f(t)$ is a piecewise constant function. Since a constant function vibrates with zero frequency, we should expect that the largest values of $FT(f)$ occur when f is near zero. The graph of FT(f) in figure 1(b) clearly illustrates this.



(a) Rectangular wave



(b) Fourier transform of the rectangular wave

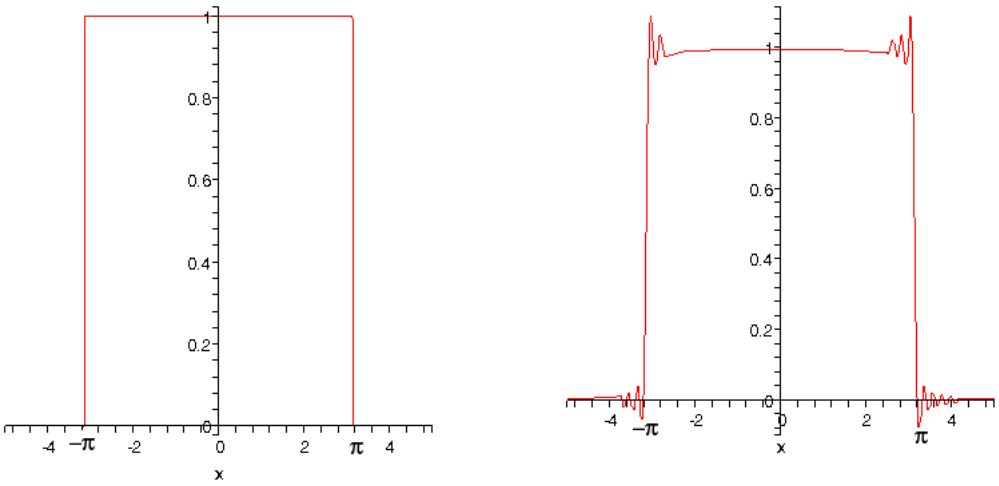
Figure 1: Example fourier transformation of square wave

The inverse of $FT(\lambda)$ can be calculated as follows:

$$\begin{aligned} ft(t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} FT(\lambda) \cdot \cos(\lambda t)\, d\lambda \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{\sqrt{2}\sin(\lambda\pi)}{\sqrt{\pi}\lambda} \cdot \cos(-\lambda t)\, d\lambda \\ &= \end{aligned}$$

$$\begin{cases} 1 & \texttt{if} \quad -\pi \leq t \leq \pi \\ 0 & \texttt{otherwise} \end{cases}$$

If we plot the inverse transform of FT(f) with a mathematical program such as Maple and use the integral from $-\infty$ to ∞, we get the result as shown in figure 2(a). However, if we want to do a faster calculation and let the integral go from -30 to 30 over the frequency we get the figure 2(b). The more harmonics, the more the inverse function looks like the original function.

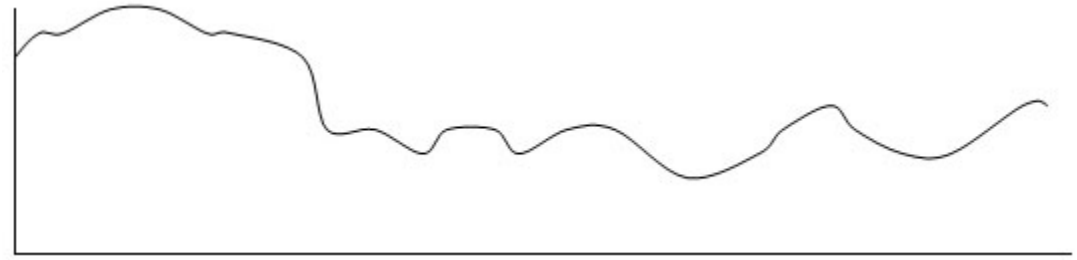


(a) Inverse transform of FT(f) with integral $\int_{-\infty}^{\infty}$ (b) Inverse transform of FT(f) with integral $\int_{-30}^{30}$

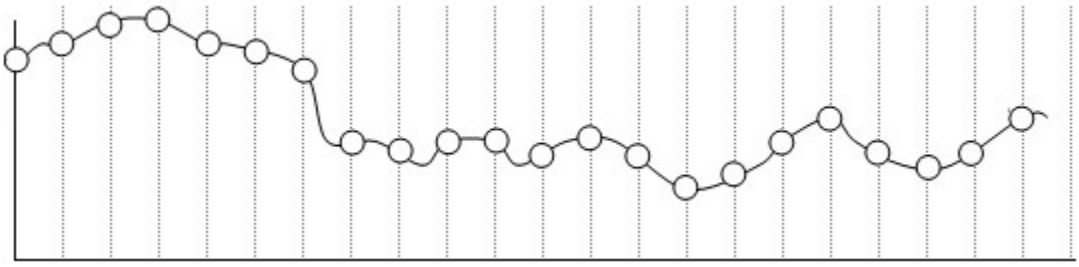
Figure 2: Inverse Fourier transforms of FT(f)

2.3 Discrete Fourier Transform

If digital computers are being used to perform the analysis, you need functions that are defined over discrete instead of continuous domains, which means that the time and the frequency variables are finite. It's suitable for many applications, for instance the signal coming from a compact disc player is a discrete data set.



(a) Continuous signal



(b) Discrete signal

Figure 3: A continuous signal sampled and converted to discrete form

The discrete Fourier transform of a discrete function of one variable $f(x)$ where $x = 0, 1, \ldots, n-1$ can be expressed as:

$$F(u) = \sum_{x=0}^{n-1} f(x) \cdot e^{-2\pi i \frac{ux}{n}} \qquad u = 0, \ldots, n-1 \tag{4}$$

where

$$\sum_{x=0}^{n-1} f(x) \triangleq f(0) + f(1) + \cdots + f(n-1)$$

$n \triangleq$ `number of time samples (integer)`

$f(x) \triangleq$ `input signal amplitude (real or complex) at x-th sampling interval`

To obtain the original function back from F(u), we can use the following inverse DFT:

$$f(x) = \frac{1}{n}\sum_{u=0}^{n-1} F(u) \cdot e^{2\pi i \frac{ux}{n}} \qquad x = 0, \ldots, n-1 \tag{5}$$

By applying the Fourier Transform algorithm on these n samples, we have made an implicit assumption. We have assumed that the signal is periodic over n samples[3][13].

As an example, we want to perform a 8 point DFT on a discretized version of a continuous one-dimensional input signal having frequency components 1000 Hz and 2000 Hz:

$x(t) = sin(2\pi\, 1000\, t) + \frac{1}{2} sin(2\pi\, 2000\, t + \frac{3\pi}{4})$

The period of x(t) is then $\frac{1}{f} = \frac{1}{1000}$ and because we want 8 samples per period, the sample time T_s will be $\frac{1}{8000}$ s or sample rate is 8000 samples/s.

Filling in the formula $t = n \cdot T_s$, you will get:

$x(n) = \sin(2\pi\, \frac{n}{8}) + \frac{1}{2}\sin(2\pi\, \frac{2n}{8} + \frac{3\pi}{4})$ where n = 0,1,...,7

Therefore

$$\begin{aligned}
X(0) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 0 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 0 \cdot \frac{n}{8})] = 0 + 0i \\
X(1) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 1 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 1 \cdot \frac{n}{8})] = 0 - 4i \\
X(2) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 2 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 2 \cdot \frac{n}{8})] = 1.414 + 1.414i \\
X(3) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 3 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 3 \cdot \frac{n}{8})] = 0 + 0i \\
X(4) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 4 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 4 \cdot \frac{n}{8})] = 0 + 0i \\
X(5) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 5 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 5 \cdot \frac{n}{8})] = 0 + 0i \\
X(6) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 6 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 6 \cdot \frac{n}{8})] = 1.4141.414i \\
X(7) &= \sum_{n=0}^{7} x(n)[\cos(2\pi\, 7 \cdot \frac{n}{8}) \cdot j\ \sin(2\pi\, 7 \cdot \frac{n}{8})] = 0 + 4i
\end{aligned}$$

In the figures below these solutions are graphically

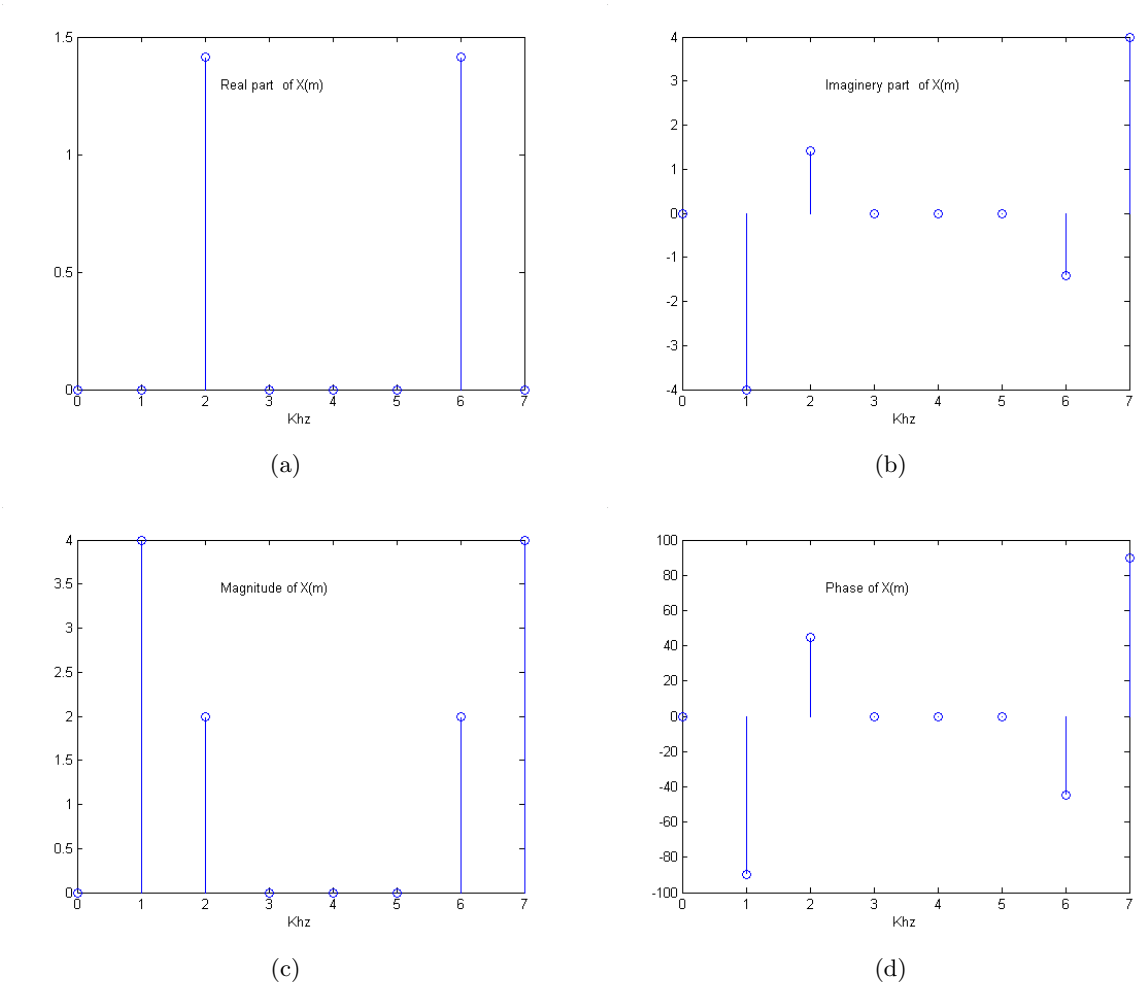


Figure 4: DFT graphs

X(n) is often represented by its magnitude and phase rather than it's real and imaginary parts, where

$$X_{magnitude}(n) = \sqrt{X_{real}(n)^2 + X_{imag}(n)^2}$$

$$X_{phase}(n) = \tan^{-1}[\frac{X_{imag}(n)}{X_{real}(n)}]$$

Basically, the magnitude tells how much of a certain frequency component is present. In our example, you see that there is a 1KHz signal of amplitude 4 and a 2KHz signal with amplitude 2. The phase tells where the frequency component is in the image. The 2KHz signal is shifted +45 degrees.

We also see that the magnitude of X(N-m) is also the magnitude of X(m) and the phase of X(N-m) is also the phase of X(m). Therefore we can conclude that if we perform an N point DFT of x(n), we will get N separate complex DFT outputs terms, but only the first N/2 terms are independent.

The originally discretized version can be retrieved by applying the inverse transform (5).

2.4 2-Dimensional Discrete Fourier Transform

In the sections above we discussed the one-dimensional Fourier transform. When working with digital images, we have to extend the discrete Fourier Transform to a two-dimensional form. Also we have to work with a finite number of discrete samples, not with continuous functions. These discrete samples are the pixels that compose an image. The extension to two-dimensional is straightforward. The discrete Fourier transform of a function f(x,y), which is the actual image, of size m x n can be expressed as:

$$F(u,v) = \frac{1}{mn} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} f(x,y) \cdot e^{-2\pi i(\frac{ux}{m}+\frac{vy}{n})} \qquad \texttt{for}\, u = 0,\ldots,m-1\, \texttt{and for}\, v = 0,\ldots,n-1 \quad (6)$$

and the inverse by

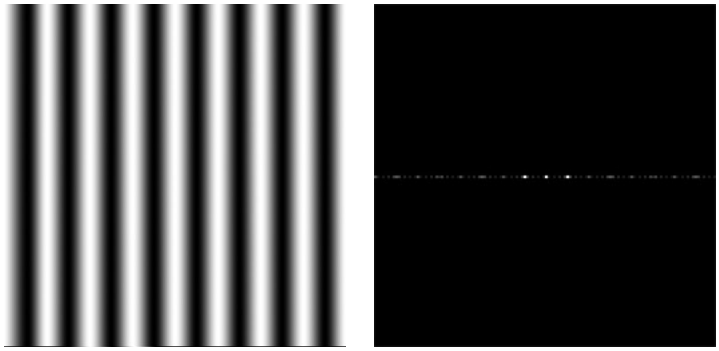
$$f(x,y) = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} F(u,v) \cdot e^{2\pi i(\frac{ux}{m}+\frac{vy}{n})} \qquad \texttt{for}\, x = 0,\ldots,m-1\, \texttt{and for}\, y = 0,\ldots,n-1 \quad (7)$$

The variables u and v used in equation 6 are the frequency variables, x and y are the spatial or image variables. Note that f(x,y) is the image and is real, but F(u,v) is the FT and is, in general, complex. In the literature, F(u,v) is often represented by its magnitude and phase rather than it's real and imaginary parts, where

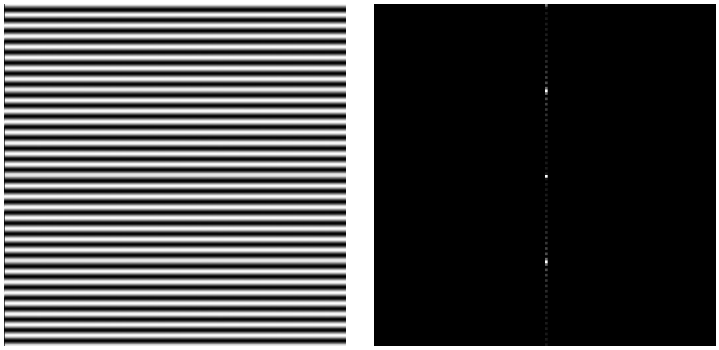
$$magnitude(F(u,v)) = \sqrt{R^2(u,v) + I^2(u,v)}$$
$$phase(F(u,v)) = \tan^{-1}[\frac{I(u,v)}{R(u,v)}]$$

Basically, the magnitude tells how much of a certain frequency component is present and the phase tells where the frequency component is in the image. To illustrate this consider the following.

First we will take a look at the "basis" functions for the discrete Fourier transform. The Fourier transform tries to represent all images as a summation of sine-like images. Therefore images that are pure sines have particularly simple DFTs which can be seen in figure 5. This figure shows two images with their discrete Fourier transforms. Notice that the DFT for each just has a single component, represented by two bright spots symmetrically placed about the center of the FT image. At the center of the image there's a dot which is the origin (0,0) of the frequency coordinate system and represents the average value of the image. Images usually have a large average value and lots of low frequency information. That's why many FT images usually have a bright blob of components near the center. The u-axis runs left to right through the center and represents the horizontal component of the frequency. The v-axis runs bottom to top through the center and represents the vertical component of the frequency. Notice that high frequencies in the vertical direction will cause bright dots away from the center in the vertical direction, and the high frequencies in the horizontal direction will cause bright dots away from the center in the horizontal direction.



(a) horizontal cosine of 8 cycles (b) DFT of figure (a)



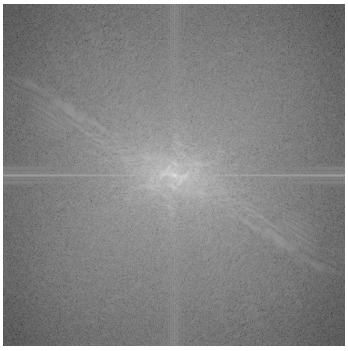
(c) vertical cosine of 32 cycles (d) DFT of figure (c)

Figure 5: Images that are pure cosines with particularly simple FTs

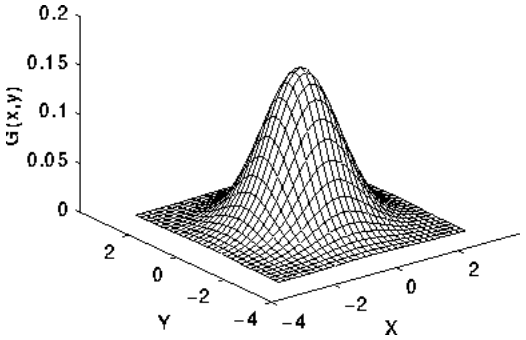
Many lossy image and sound compression methods employ this discrete Fourier transform: the signal is cut into short segments, each is transformed, and then the Fourier coefficients of high frequencies, which are assumed to be unnoticeable, are discarded[8][9].

Other image enhancements by manipulating the Fourier transform include Gaussian blurring, sharpening, low and high passes, contrast modifications, etc. In the images below you see an example of a Gaussian blur being applied. As you can see in figure 6(d) it cuts off the high frequencies by using the well known Gaussian curve, resulting in the loss of fine details in the resulting image.

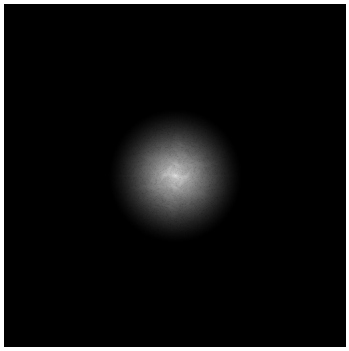
(a) Original image



(b) DFT of image (a)



(c) Gaussian blur curve



(d) Gaussian blur cutout on (c)

(e) Resulting image

Figure 6: Image with its FT and Gaussian blur applied on it

2.5 Fast Fourier Transform

There is no essential difference between FFT and DFT in the fact that they both carry out Fourier transform on discrete signals. The only difference is that the implementation of the algorithm is optimized to remove redundant calculations. These optimizations are only possible if the number of samples to be transformed is an exact power of two. The complexity of the DFT algorithm is $O(n^2)$ while FFT has a complexity of $O(n \cdot \log n)$ For a 1024 point transform (N=1024), this will give approximately a 100 fold speed improvement. This is why FFTs are important. Of course the actual speed improvement that is achieved in practice depends also on other factors associated with algorithm implementation and coding efficiency, but it's nevertheless a big speed improvement[3][13].

2.6 Discrete Cosine Transform

The Discrete Cosine Transform (DCT) is an algorithm that is widely used for image and video compression and is very similar to the discrete Fourier transform (DFT). It differs from DFT that it uses only real numbers and is the decomposition of a function into a series of even cosine components only. Despite this loss of information, and its assumption that the function to be analyzed is not only periodic but also even, the DCT is very common in image processing as it is computationally cheaper than DFT. It is also known to have better energy compaction properties than the DFT, which means that more information is encoded by the earlier DCT coefficients than by the DFT coefficients. This may initially appear paradoxical, that DCT ignores the odd sine components of a function and yet appears to encode more information. But it doesn't code more information than DFT, only it does so by using fewer and earlier coefficients. For this reason, the DCT is used in image and video compression algorithms as JPEG and MPEG[8].

The well known JPEG codec uses this DCT algorithm. It partitions the input image into non-overlapping 8x8 blocks (figure 7) and applies the DCT to each block which can cause the blocking artifact which is being discussed in the next chapter.

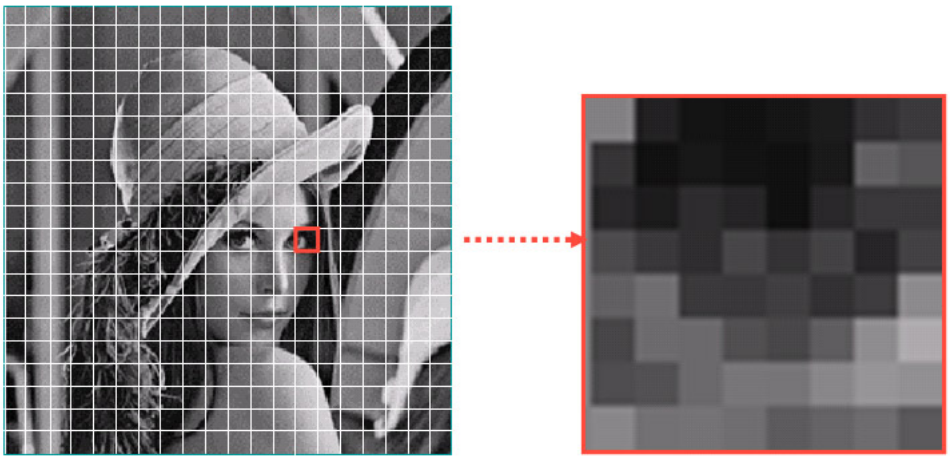


Figure 7: Dividing original image

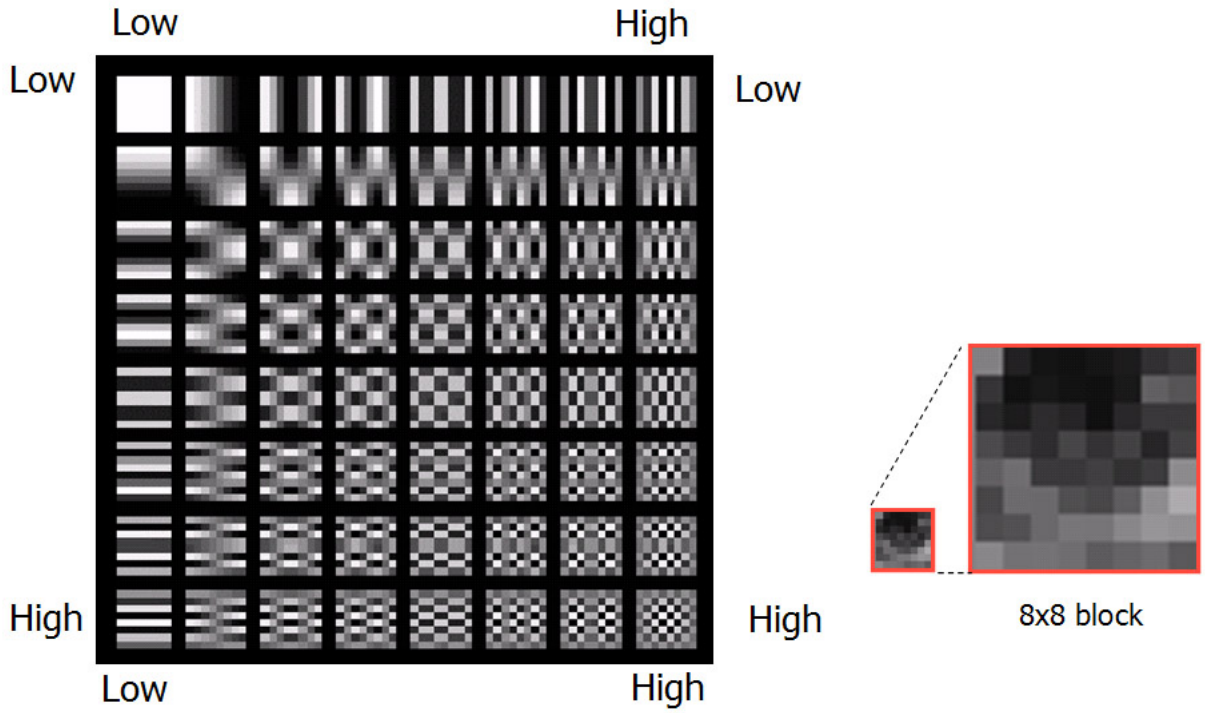


Figure 8: Comparing 8x8 block to the 64 basis functions

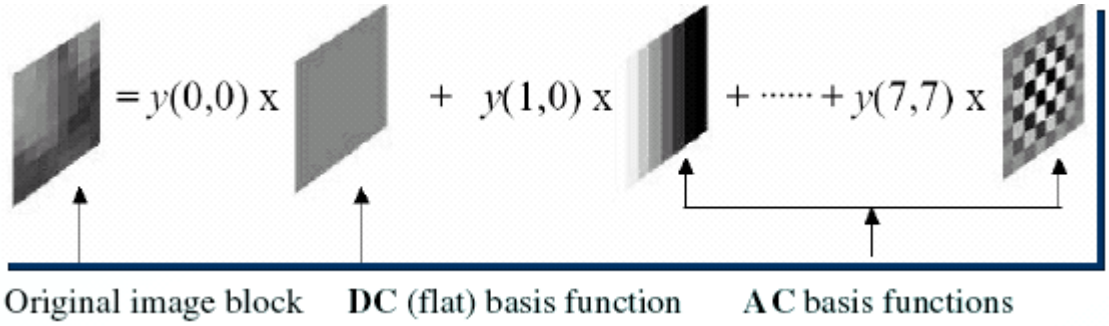


Figure 9: Image representation with DCT

DCT coefficients can be viewed as weighting functions that, when applied to the 64 cosine basis functions (see figure 8) of various spatial frequencies, will reconstruct the original block. De DC basis function will represent the average color and the AC basis functions will represent the differences. These coefficients are mapped to an array with length 64, using the zigzag method (figure 10). Because the human eye is most sensitive to low frequencies (upper left corner), less sensitive to high frequencies (lower right corner) in figure 8, we can truncate the coefficients which represent these high frequencies depending on the chosen quality. These high frequency coefficients are at the end of the array because of the zigzag method.

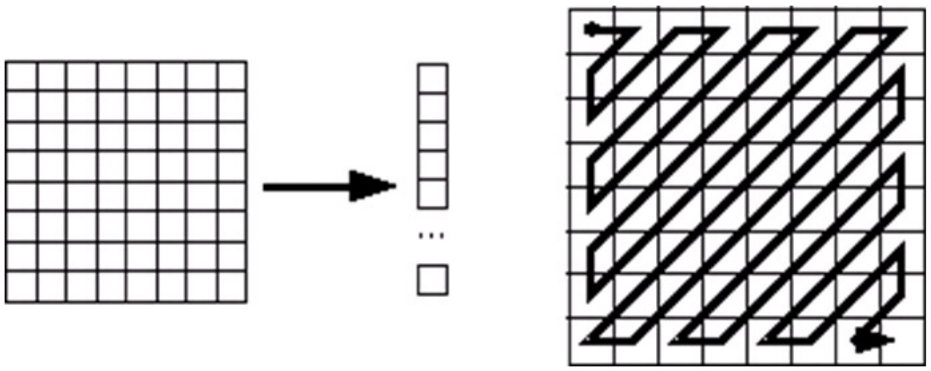


Figure 10: Conversion 8x8 matrix into an array of length 64

3 Wavelet Transforms

Wavelets are mathematical functions that satisfy certain properties and can be used to transform one function representation into basis functions like the Haar or Daubechies wavelets. The major advantage of using wavelets is that they can be used for analyzing functions at various scales; we could listen to an entire symphony or just the violin. Ideas such as multi-resolution, versions of an image at various resolutions, is very similar to how the human eye works. From the window of an airplane, for example, the forest appears to be a solid green plane. From the window of a car on the ground, this plane resolves into individual trees, and if you get out of the car and move closer, you begin to see branches and leaves. If you then pull out a magnifying glass, you might find a drop of dew on the leaf. As you zoom in at smaller and smaller scales, you can find details that you did not see before. Because the eye is much more sensitive to errors in low frequencies than in high frequencies, we can apply compression to the area's for which our eyes aren't sensitive. In practice, the wavelet transforms have great advantages over the traditional Fourier methods in analyzing signals which contain discontinuities and sharp spikes like most natural images.

3.1 The One-dimensional Haar Wavelet Transform

To get a sense for how these wavelets work and give an impression on what is meant by resolution and scale, we give a simple example that is based on the simplest wavelet basis: the Haar wavelet.

Suppose we have a one-dimensional data set containing eight pixels. For example, they could be the first row of an two-dimensional 8x8 pixel image:

$[\ 10 \quad 8 \quad 6 \quad 8 \quad 1 \quad 5 \quad 8 \quad 2\ ]$

We can represent this image in the Haar basis by computing a wavelet transform. To do this, we first average the pixels together, pairwise ($\frac{10+8}{2}$, $\frac{6+8}{2}$, etcetera), to get a new, lower resolution image with four pixel values

$[\ 9 \quad 7 \quad 3 \quad 5\ ]$

Clearly, some information has been lost in this averaging process. To recover the original eight pixel image from the four averaged values, we need to store detail coefficients, which capture the missing information. In our example the first detail coefficient is 1 because the average we calculated is 1 less than 9 and 1 more than 7. This single number allows us to recover the first two pixels of our original eight-pixel image. Our second detail coefficient is -1 because $7+(-1) = 6$ and $7-(-1) = 8$. Repeating this procedure over and over again will give the full decomposition, shown in table3.1.

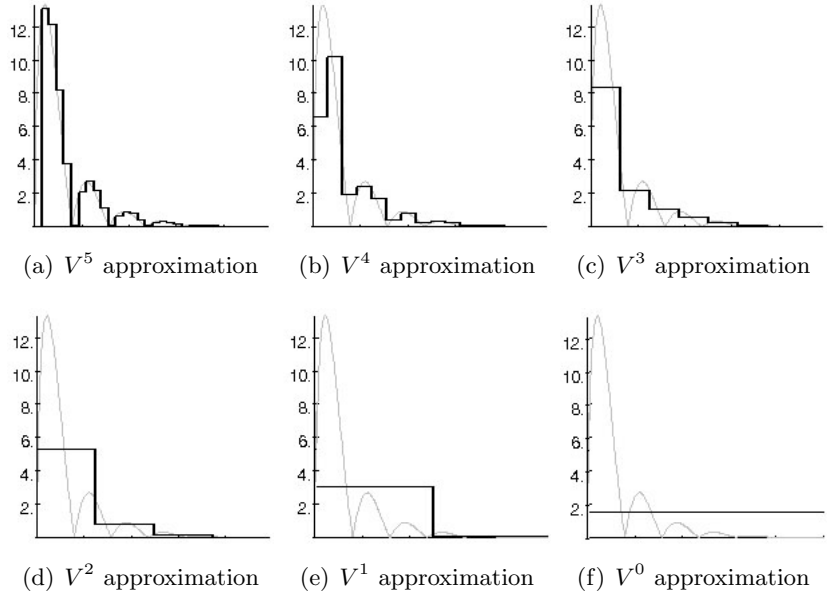
Resolution	Averages	Detail Coefficients
8	$[\ 10 \quad 8 \quad 6 \quad 8 \quad 1 \quad 5 \quad 6 \quad 4\ ]$	
4	$[\ 9 \quad 7 \quad 3 \quad 5\ ]$	$[\ 1 \quad -1 \quad -2 \quad 1\ ]$
2	$[\ 8 \quad 4\ ]$	$[\ 1 \quad -1\ ]$
1	$[\ 6\ ]$	$[\ 2\ ]$

Table 1: Decomposition of 8-pixel image

What we will do next is defining the *wavelet transform* of the original eight pixel image as follows: We begin with a single coefficient representing the overall average of the original image (in our example it is 6), followed by all the detail coefficients in order of increasing resolution. To clarify the notion of resolution, see figure 11. The wavelet transform of our original eight-pixel data set is then given by

$$[\ 6 \quad 2 \quad 1 \quad -1 \quad 1 \quad -1 \quad -2 \quad 1\ ]$$

Note that no information has been gained or lost by this process. With this transform we can reconstruct the image to any resolution by adding and subtracting the detail coefficients from the lower resolution versions. This may look wonderful and all, but what good is compression that takes eight values and compresses it to eight values? Simple, in images especially, the pixel values are similar to their neighbors. If they werent we would be looking at some random noise and not natural, real world images. When doing the averaging and subtracting with wavelets, the detail values are usually small numbers and sometimes even zero. These detail values can be compressed much better than the original pixel values. Furthermore, we can compress the image by removing the small coefficients from this transform, which only introduces some small errors in the reconstructed image. This is obviously a form of lossy image compression.



(a) V^5 approximation (b) V^4 approximation (c) V^3 approximation

(d) V^2 approximation (e) V^1 approximation (f) V^0 approximation

Figure 11: A sequence of decreasing resolution approximations to a function (grey line)

3.2 One-dimensional Haar Wavelet Basis Functions

There are two functions that play a primary role in wavelet analysis, the scaling function ϕ and the wavelet ψ. These two functions generate a family of functions that can be used to break up or reconstruct a signal. To emphasize the "marriage" involved in building this "family", ϕ is often called the "father wavelet" and ψ the "mother wavelet".

In the previous example we have shown how one-dimensional images can be treated as a sequence of coefficients. However, in most wavelet literature, there's another way presented to look at these images. They define images as piecewise-constant functions on the half-open interval $[0, 1)$ and use the concept of a *vector space* from linear algebra. For those who aren't familiar with linear algebra, we briefly explain some of the important concepts .

3.2.1 Basic linear algebra

- **Vector spaces**

  A vector space can be defined as a collection V of elements where

  1. For all $a, b \in \mathbb{R}$ and for all $u, v \in V \quad [au + bv \in V]$
  2. There exists an additive identity element $0 \in V$ such that
     - for all $u \in V, 0u = 0$, and
     - for all $u \in V, 0 + u = u$
  3. Other axioms for vector addition and multiplication are omitted here, but can be found in the regular linear algebra literature.

  The elements of a vector space V are called *vectors*, and the element 0 is called the *zero vector*. The vectors may be geometric vectors, or they may be functions as in the case of wavelets.

- **Inner products and orthogonality**

  For two vectors in $X = (x_1, x_2, x_3), Y = (y_1, y_2, y_3)$ in vector space $\mathbb{R}^3$, the standard inner product of X and Y is defined as

  $\langle X, Y \rangle = x_1y_1 + x_2y_2 + x_3y_3$

  Two vectors u, v in an inner product space are orthogonal if their inner product is zero. For example, the vectors $\begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$ and $\begin{pmatrix} 3 \\ -1 \\ 0 \end{pmatrix}$ are orthogonal to each other,
  since $1 \cdot 3 + 3 \cdot -1 + 2 \cdot 0 = 0$

A one-pixel image is just a function that is constant over the entire interval $[0, 1)$. We'll let V^0 be the vector space of all these functions. A two-pixel image has two constant pieces over the intervals $[0, \frac{1}{2})$ and $[\frac{1}{2}, 1)$. The space containing all these functions is called V^1. Generally speaking, V^j will include all piecewise-constant functions defined on the interval $[0, 1)$ with constant pieces over each of 2^j equal subintervals.

Every one-dimensional image with 2^j pixels is a vector in vector space V^j. Furthermore, every vector in V^j is also in V^{j+1}. That's because we can always describe a piecewise-constant function with two intervals as a piecewise-constant function with four intervals (by letting each interval in the first function correspond to a pair of intervals in the second function). So, the vector spaces V^j are nested as follows

$V^0 \subset V^1 \subset V^2 ... \subset V^{j-1} \subset V^j \subset V^{j+1} \subset ...$

So, V^j contains all relevant information up to a resolution scale of order 2^{-j}. As j gets larger, the resolution gets finer. The fact that $V^{j-1} \subset V^j$ means that no information is lost as the resolution gets finer. Now we can define a basis function for each vector space V^j. These functions are called *scaling functions* or father wavelets. The Haar scaling function is defined as

$$\phi_i^j(x) := \phi(2^j x - i) \qquad \texttt{where } i = 0, ..., 2^j - 1 \texttt{ and } \phi(x) = \begin{cases} 1 & \texttt{if } 0 \leq x < 1 \\ 0 & \texttt{otherwise} \end{cases}$$

Figure 12 shows the four (2^2) functions forming a basis for V^2 using the Haar scaling function.

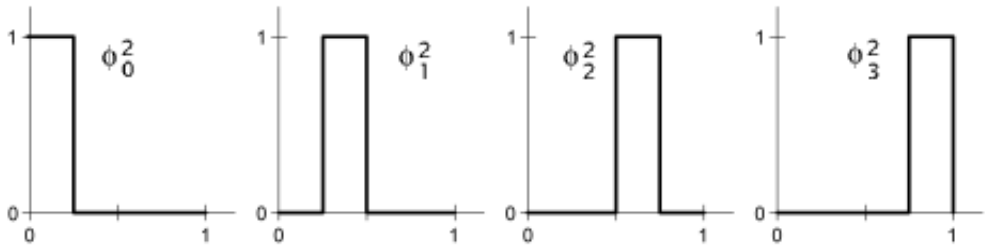


Figure 12: The four box functions forming a basis for the vector space V^2

As you can see, the function $\phi_i^j(x)$ is 1 at the i-th 'interval' and 0 otherwise. The graph of function $\phi_3^2(x)$ has been constructed as follows:

$$\begin{array}{c|c|c|c|c|c|c|c} \phi_3^2(0) & = & \phi_3^2(2^2 \cdot 0 - 3) & = & \phi(-3) & = & 0 \\ \phi_3^2(\frac{1}{4}) & = & \phi_3^2(2^2 \cdot \frac{1}{4} - 3) & = & \phi(-2) & = & 0 \\ \phi_3^2(\frac{1}{2}) & = & \phi_3^2(2^2 \cdot \frac{1}{2} - 3) & = & \phi(-1) & = & 0 \\ \phi_3^2(\frac{3}{4}) & = & \phi_3^2(2^2 \cdot \frac{3}{4} - 3) & = & \phi(0) & = & 1 \\ \phi_3^2(1) & = & \phi_3^2(2^2 \cdot 1 - 3) & = & \phi(1) & = & 0 \end{array}$$

Functions in V^{j+1} that are orthogonal to functions in V^j define a new vector space W^j. In other words W^j contains the detail in V^{j+1} that cannot be represented by V^j. Wavelets are those functions that span W^j. These basis functions have two properties:

1. the basis functions ψ_i^j of W^j, together with the basis functions ϕ_i^j of V^j form a basis for V^{j+1}

2. every basis function ψ_i^j of W^j is orthogonal to every basis function ϕ_i^j under the chosen inner product

The wavelets corresponding to Haar scaling functions (as in figure 12) are known as the *Haar wavelets* and are defined as

$$\psi_i^j(x) := \psi(2^j x - i) \qquad \texttt{where } i = 0, ..., 2^j - 1 \texttt{ and } \psi(x) = \begin{cases} 1 & \texttt{if } 0 \leq x < \frac{1}{2} \\ -1 & \texttt{if } \frac{1}{2} \leq x < 1 \\ 0 & \texttt{otherwise} \end{cases}$$

As an example, figure 13 shows the two Haar wavelets that span W^2 and correspond to the basis in V^2.

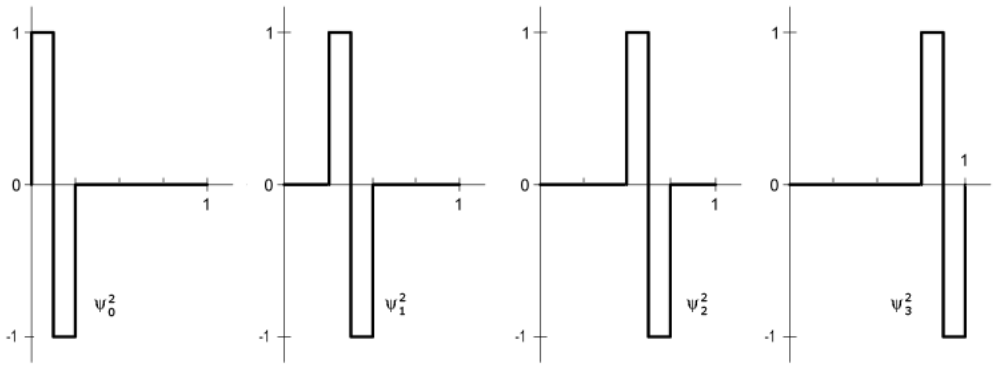


Figure 13: The Haar wavelets for W^2

3.3 Orthogonality and Orthonormality

It is important to note that the Haar basis has an important property, orthogonality, which many other wavelet basis don't have. Orthogonality in this respect is stronger than the requirement in the definition of wavelets that they need to be orthogonal to all scaling functions at the same resolution.

The basis functions in our example ϕ_0^0, ψ_0^0, ϕ_0^1, ..., are orthogonal to each other. One of the reasons an orthogonal basis is a nice property, is the easy with which we can obtain the inverse wavelet transform because of some mathematical advantages. Nonorthogonal bases are harder to work with in that respect.

Normalization is another property that can be desirable. A basis function $f(x)$ is normalized if the inner product of that orthogonal basis function with itself is 1, so $< f|f >= 1$. We can easily normalize the Haar basis by choosing a constant factor which will satisfy $< f|f >= 1$. If we take constant factor $2^{\frac{j}{2}}$, the unnormalized coefficients can be normalized by multiplying it with this factor.

We change the original definitions of Haar wavelets into:

$$\phi_i^j(x) := 2^{\frac{j}{2}}\,\phi(2^j x - i) \qquad \texttt{where } i = 0, ..., 2^j - 1$$
$$\psi_i^j(x) := 2^{\frac{j}{2}}\,\psi(2^j x - i) \qquad \texttt{where } i = 0, ..., 2^j - 1$$

With these modified definitions, the new normalized coefficients are obtained by multiplying each old coefficient with $2^{\frac{-j}{2}}$

In our example, the unnormalized coefficients

$$\begin{bmatrix} \psi_0^0 & \phi_0^0 & \phi_0^1 & \phi_1^1 & \phi_0^2 & \phi_1^2 & \phi_2^2 & \phi_3^2 \end{bmatrix}$$
$$\begin{bmatrix} 6 & 2 & 1 & -1 & 1 & -1 & -2 & 1 \end{bmatrix}$$

will become the normalized coefficients by multiplying with the constant factor $2^{\frac{-j}{2}}$

$$\begin{bmatrix} 6 & 2 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{2} & -\frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix}$$

Orthonormal means that the basis is both orthogonal and normalized[6].

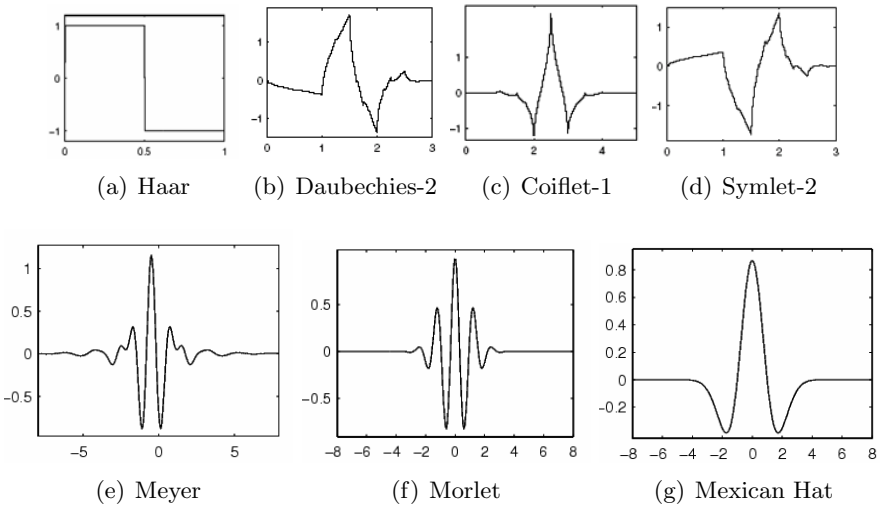
3.4 Two-dimensional Haar Transform

The one-dimensional Haar Transform can be easily extended to two-dimensional. In the two-dimensional case, we choose an input matrix instead of an input vector. To transform the input matrix, we first apply the one-dimensional Haar transform on each row. We take the resulting matrix, and then apply the one-dimensional Haar transform on each column. This will give us the final transformed matrix.

3.5 Daubechies and others

There are some limitations in using Haar's wavelets. Because the Haar's base functions are discontinuous functions, they are not really suitable for analyzing smooth functions. Images often contain smooth areas for which the Haar wavelet transform does not provide satisfactory results.

Therefore many other basis functions have been developed that can be used as the mother wavelet for Wavelet Transformation to improve the Haar wavelet transform. Because the mother wavelet produces all wavelet functions used in the transformation through translation and scaling, it will determine the characteristics of the resulting wavelet transform. Therefore, it's important that the details of the particular applications should be taken into account on which you choose the appropriate mother wavelet. For example, a certain mother wavelet might be appropriate for compressing ECG signals, but not for compressing fingerprints. In figure 14 you can find a collection of basis functions which you can use[11].



(a) Haar (b) Daubechies-2 (c) Coiflet-1 (d) Symlet-2

(e) Meyer (f) Morlet (g) Mexican Hat

Figure 14: Wavelet families

4 Comparison JPEG vs JPEG2000

The JPEG2000 standard has been approved in 2003 as an international standard for the compression of digital pictures. Unlike the conventional JPEG algorithm which uses the Fourier based DCT, this new standard is entirely wavelet-based; it incorporates the Daubechies (9,7)[6] wavelet for lossy compression and the reversible LeGall (5,3) wavelet for lossless compression.

To compare the JPEG2000 algorithm with the conventional JPEG algorithm we have selected a range of pictures which are known to be highly stressing for many image compression algorithms. Because the quality settings of the conventional JPEG algorithm differ in each implementation we choose the more general notation bits-per-pixel. For a full-colour (24 bit) image a compression ratio of 48:1 equals 24/48 or 0.5 bits-per-pixel.

Generally, there are two visible damages caused by image compression which are shortly described in the following sections.

4.1 Blocking artifacts

Blockiness is common to all Discrete Cosine Transform (DCT) based image compression techniques such as JPEG. The DCT is typically performed on 8x8 pixel squares in the image and the coefficients in each block are quantized separately. This leads to artificial horizontal and vertical borders between these blocks. If an image is compressed with very high compression ratios, it will look like some sort of mosaic.

4.2 Blur

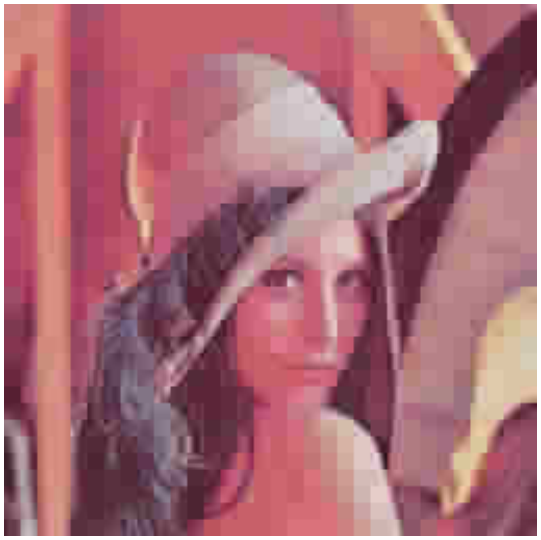
The loss of fine detail and the smearing of edges is called blur. This is caused by reduction of high frequencies in the encoding process. It is one of the main artifacts of wavelet based compression techniques such as JPEG2000. Also DCT-based compression cause some blur, but it's not the primary distortion.

4.3 Results: Image quality

Although the test results are subjective because they are based on our human visual experience, we come to the conclusion that JPEG2000 provides a slightly better image quality than JPEG when the compressed rates are 1-2 bits per pixel. If we compress the images a little more, at 0.5 bits per pixel, the JPEG2000 image is a distinct improvement, particular with respect to the block artifacts that appear in the JPEG compressed image. At 0.25 bits per pixel, the JPEG images begins to look like a mosaic while with JPEG2000 it gets a elegant blur across the image. JPEG2000 image files tend to be 20 to 60% smaller than their JPEG counterparts for the same subjective image quality depending on the type of pictures you are compressing [4]. However, the real quality advantage of JPEG2000 will only become clear by using very high compression ratios because it gets rid of JPEGs main problem: the infamous blocking artifacts.[5]

(a) Lena Original, 512x512x24b

(b) Monarch Original, 512x512x24b

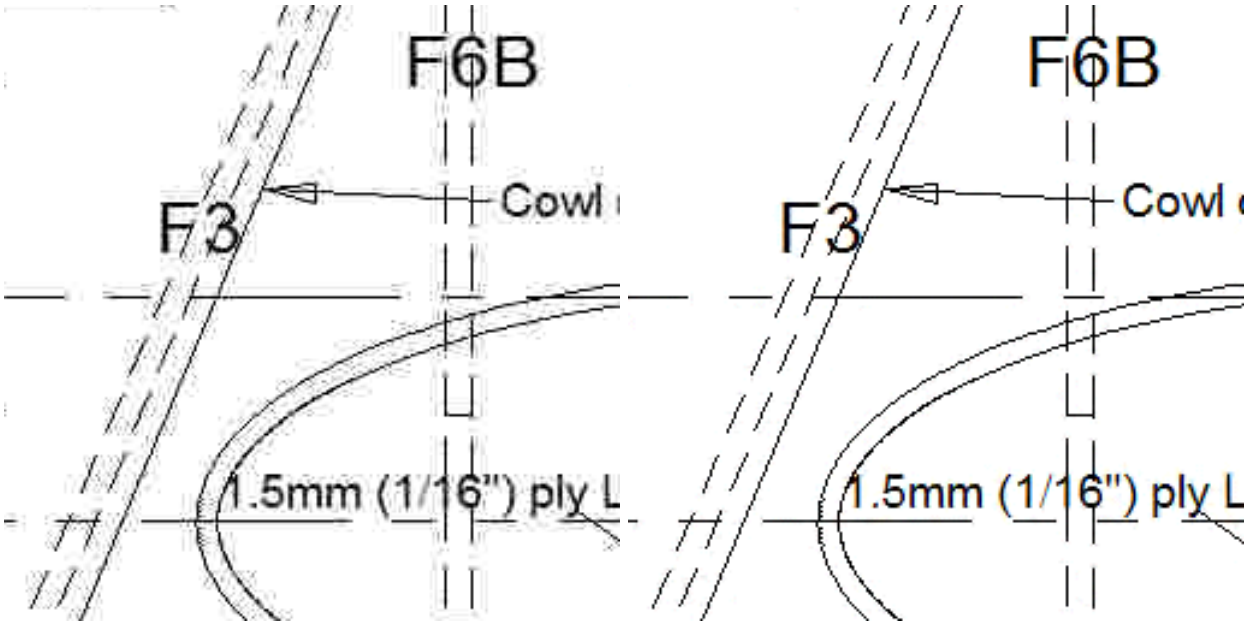


(c) Lena JPEG (0.2 bits/pixel)

(d) Monarch JPEG (0.1 bits/pixel)

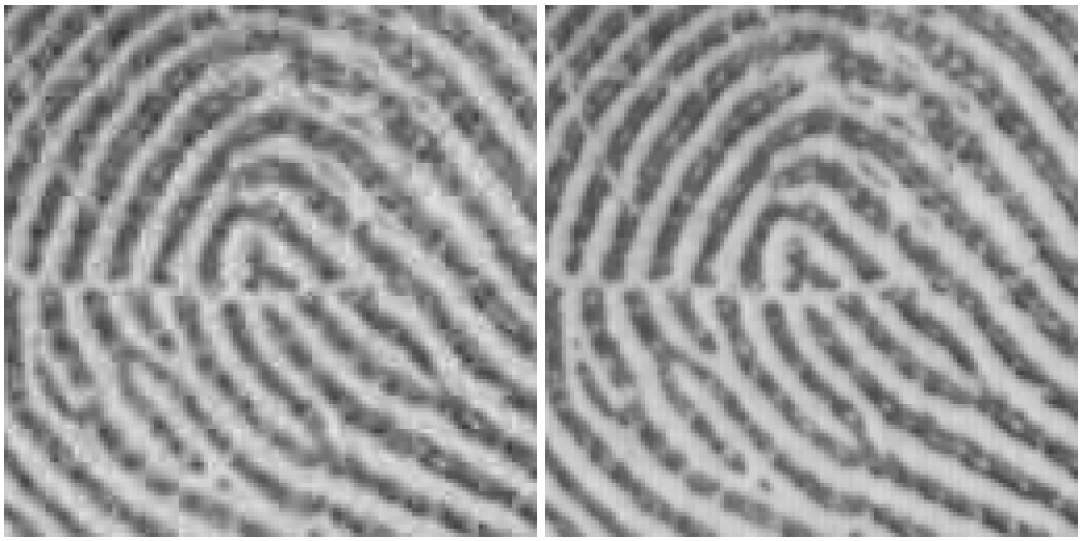
(e) Lena JPEG2000 (0.2 bits/pixel)

(f) Monarch JPEG2000 (0.1 bits/pixel)



(a) Building Plan JPEG (0.1 bits/pixel)

(b) Building Plan JPEG2000 (0.1 bits/pixel)



(c) Fingerprint JPEG (2 bits/pixel)

(d) Fingerprint JPEG2000 (2 bits/pixel)

Figure 16: Image quality comparison JPEG vs JPEG2000 part 2

If you look at figure 16(c) you see a 4x zoom of the center of an FBI fingerprint image. The graininess you see are the individual pixels in the 500 dpi scan. The whiter spots in the middle of the black edges are sweat pores, and they're tolerable points of identification in court as the little black points in the grooves between the ridges. Since these details are just a couple pixels wide, the compression algorithm needs to preserve these features correctly. This is a very tough problem since

most lossy algorithms like the conventional JPEG work by throwing out the highest frequencies (and thus the smallest details) in the image. At the JPEG version you see that the sweat pores are faded away or some other white points are introduced which renders the FBI fingerprint unusable in court.

4.4 Results: Performance

However, one has to pay a price for this quality improvement: a considerable increase in computational complexity and memory usage. In the table below we compare both algorithms by recording the time needed to compress the selected images known for their difficulty. We used the JasPer library[1] written in the C programming language for compressing images with the JPEG2000 algorithm. This library has been incorporated into numerous other software projects including ImageMagick, KDE and GhostScript and is freely available for academic uses and supports various image formats[1].

For compressing our test images with the conventional JPEG algorithm we used a widely used free library from the Independent JPEG Group[10].

Test image	Uncompressed size	Resolution	Color depth	Quality	JPEG2000 time	JPEG time
Construction plan	34 MB	5000x3477	16bit	0.75bpp	13.49 sec	1.98 sec
Lena	786 KB	512x512	24bit	0.75bpp	0.94 sec	0.37 sec
Tulips	1.2 MB	768x512	24bit	0.75bpp	0.78 sec	0.33 sec
Monarch	1.2 MB	768x512	24bit	0.75bpp	0.77 sec	0.36 sec

Table 2: Performance table JPEG vs JPEG2000

4.5 Test conclusions

These tests have lead to some interesting conclusions:

- The quality advantages are really visible when compressing with very high compression ratios in order of 0.1 or 0.2 bits/pixel. The JPEG2000 version of the monarch picture for example is visually much better than the JPEG version. Also the artifacts clearly visible in the JPEG version of the building plan are significantly reduced, which proves that JPEG2000 works better with sharp spikes in images.
- For most images we tested, the JPEG codec runs almost three times faster. This is probably due to the increase in computational complexity of the new JPEG2000 algorithm. Another reason is that the JPEG codec is been optimized throughout the years in its existence.
- The time needed to compress the construction plan image which has a very big resolution takes a lot of time with JPEG2000. Performance will decrease enormously when compressing big pictures in comparison with the conventional JPEG codec and uses considerable more system memory.

5 Conclusions

In the first chapters we gave an extensive description of the Fourier transform and wavelet transforms, the underlying concepts of JPEG and JPEG2000 and presented their advantages and disadvantages.

With wavelets it is possible to construct your own basis function, depending upon the application. We don't have this freedom in DCT, where one has only cosine functions as the basis set. Selecting a particular wavelet that is designed for a certain application will give the best result. But at the same time that is were a problem lies. Selecting a basis function for a particular application is difficult. Most of the time, the selection is done after experimenting with different sets of wavelets like the general purpose Daubechies wavelets.

During our experiments we saw that one of the main problems with DCT is that the given image is subdivided into 8x8 blocks. Therefore the correlation between these blocks is lost and blocking artifacts will occur. This is noticeable and annoying, particularly at low bit rates. In wavelets, no such blocking is done because the transformation is calculated over the entire image.

Generally we can conclude that compressing with JPEG2000 will result in better compression compared to the traditional JPEG algorithm with the same subjective image quality. For this improvement we have to pay a price: JPEG2000 uses far more memory and is slower, especially with high resolution images.

Although wavelets are being used in well known video codecs like DiVX and MPEG4, the JPEG2000 standard is not yet widely used. One of the reasons could be the lack of performance of current JPEG2000 algorithms. In the future these algorithms may be improved by using extended instruction sets which are available on most modern CPU's. Also other performance optimizations could be applied in the future, as was done with standard JPEG in the last years. An other reason that it's not used that much could be the ever increasing amount of disk and network capacities which eliminates the need for extremely high compression.

However, the Microsoft corporation recently announced that they will include JPEG2000 support in their new Internet Explorer and operating system Longhorn which could result in a breakthrough of JPEG2000 in the consumer market.

References

[1] M. D. Adams, "The Jasper Project Homepage", *Digital Signal Processing Group at University of Victoria*, 2001. Available from `http://www.ece.uvic.ca/ mdadams/jasper/`.

[2] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, "Image coding using the wavelet transform", *IEEE Computational Science and Engineering*, On Image Processing, vol. 1, pp. 205-220, April 1992.

[3] Albert Boggess and Francis J. Narcowich, "A First Course in Wavelets with Fourier Analysis", *Prentice Hall*, First Edition, 2001, pp. 37-57, pp. 91-104, pp. 131-143.

[4] , "The JPEG2000 still image coding system: An overview", *IEEE Trans. Consumer Electronics*, vol. 46, no. 4, pp. 1103-1127, 2000.

[5] Diego Santa Cruz, Touradj Ebrahimi, Joel Askelof, Mathias Larsson and Charilaos Christopoulos, "An analytical study of JPEG 2000 functionalities", *Swiss Federal Institute of Technology and Ericsson Research*, ISO/IEC JTC1/SC29/WG1. Available from `http://www.jpeg.org/public/wg1n1816.pdf`.

[6] Ingrid Daubechies, "Ten Lectures on Wavelets", *Rutgers University and AT&T Bell Labatories*, Philadelphia, 1992.

[7] R. DeVore, B. Jawerth and B. Lucier, "Image Compression Through Wavelet Transform Coding", *IEEE Trans. Information Theory*, Vol. 38, No. 2, Mar. 1992, pp. 719-746.

[8] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", *Prentice Hall*, Second Edition, 2002, pp. 147-215, pp. 349-404.

[9] Amara Graps, "An introduction to wavelets", *IEEE Computational Science and Engineering*, vol. 2, no. 2, Summer 1995, pp. 5061. Available from `http://www.amara.com/ftpstuff/IEEEwavelet.pdf`.

[10] Thomas G. Lane, "IJG JPEG Library", *Independent JPEG Group*, 1998. Available from `http://www.ijg.org/`.

[11] M. Misiti, Y. Misiti, G. Oppenhein and J.M. Poggi, "Wavelet Toolbox User's Guide", *The Mathworks, Inc.*, Chapter 1 Wavelets: a new tool for signal analysis, version 2.2 edition, July 2002. Available from `http://www.dii.unisi.it/ menegaz/DoctoralSchool2004/wavelet_tutorial_matlab.pdf`.

[12] James Z. Wang, "Wavelets and Imaging Informatics: A Review of the Literature", *School of Information Sciences and Technology, Pennsylvania State University*, Journal of Biomedical Informatics, vol. 34, pp. 129141, April 2001.

[13] Eric W. Weisstein, "Fourier Transform", *From MathWorld–A Wolfram Web Resource*. Available from `http://mathworld.wolfram.com/FourierTransform.html`.