the texture (this is the spatial frequency of the cloth's pattern) and weaker, more scattered, components due to the fingerprint. If we suppress the frequency components due to the cloth's texture, and invoke the inverse Fourier transform, then the cloth will be removed from the original image. The fingerprint can now be seen in the resulting image.

2.6.5 The Importance of Phase

You might reasonably ask: "what is the importance of phase?". It is actually a pretty reasonable question. *Phase* is about how signals are arranged (or add up), whereas magnitude is about how big the signals are. As such, one might intuitively think that the magnitude is more important than the phase. It is more complex than this: phase can actually be viewed to be more important (though you need both magnitude and phase to reconstruct a signal). To illustrate this we shall take two images, one of the eye and the other of an ear (OK, ears are rather ugly but they are unique to their owner) as shown in Figure 2.24. Here we have reconstructed images by taking the magnitude of one image and the phase (the argument) of another. From Eq 2.6 we have

$$\mathbf{Fp}(\omega)=\mathrm{Re}(\mathbf{Fp}(\omega))+j\,\mathrm{Im}(\mathbf{Fp}(\omega))=\mathit{magnitude}\times\cos(\mathit{phase})+j\times\mathit{magnitude}\times\sin(\mathit{phase}) \qquad (2.37)$$

where magnitude and phase are calculated according to Equations 2.7 and 2.8, respectively. The rearranged Fourier transform is then

$$\mathbf{Fp}(\omega)=|\mathbf{Fp}(\text{image 1})|\times\cos\left(\mathit{arg}\left(\mathbf{Fp}(\text{image 2})\right)\right)+j\times|\mathbf{Fp}(\text{image 1})|\times\sin\left(\mathit{arg}\left(\mathbf{Fp}(\text{image 2})\right)\right)$$

When image 1 is the eye and image 2 is the ear then the reconstructed image (via the inverse FT) looks most like the image of the ear, Figure 2.24(c); when it is the other way found, the image is much closer to the eye, Figure 2.24(d). So it is the phase that is controlling the reconstruction in this case, not the magnitude (the bit represented by the magnitude hardly shows in the reconstructions here). Clearly the phase is very important in the representation of a signal.

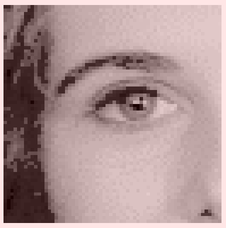
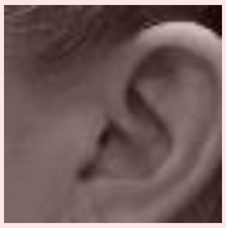
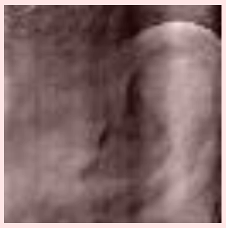
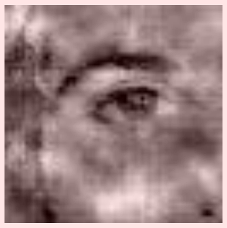
			
(a) eye image	**(b)** ear image	**(c)** reconstruction from magnitude(eye) and phase(ear)	**(d)** reconstruction from magnitude(ear) and phase(eye)

Figure 2.24 Illustrating the Importance of Phase

2.7 Transforms other than Fourier

2.7.1 Discrete Cosine Transform

The *Discrete Cosine Transform* (DCT) [Ahmed74] is a real transform that has great advantages in energy compaction. Its definition for spectral components $\mathbf{DP}_{u,v}$ is:

$$\mathbf{DP}_{u,v} = \begin{vmatrix} \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y} & \text{if} \quad u=0 \quad \text{and} \quad v=0 \\ \frac{2}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}\times\cos\left(\frac{(2x+1)u\pi}{2N}\right)\times\cos\left(\frac{(2y+1)v\pi}{2N}\right) & \text{otherwise} \end{vmatrix} \tag{2.38}$$

There are many variants of the definition of the DCT and we are concerned only with principles here. The inverse DCT is defined by

$$\mathbf{P}_{x,y} = \frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\mathbf{DP}_{u,v}\times\cos\left(\frac{(2x+1)u\pi}{2N}\right)\times\cos\left(\frac{(2y+1)v\pi}{2N}\right) \tag{2.39}$$

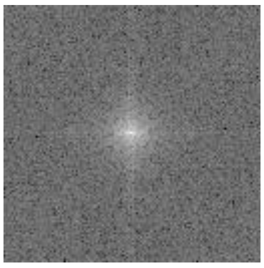
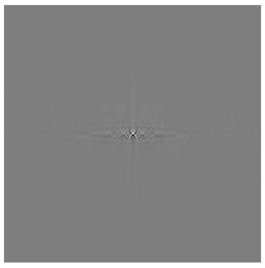
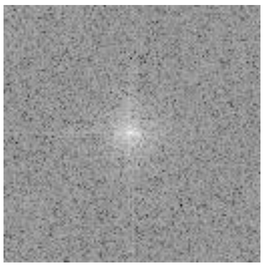
		
(a) Fourier transform magnitude	**(b)** discrete cosine transform	**(c)** Hartley transform

Figure 2.25 Comparing Transforms

A fast version of the DCT is available, like the FFT, and calculation can be based on the FFT. Both implementations offer about the same speed. The Fourier transform is not actually optimal for *image coding* since the Discrete Cosine transform can give a higher compression rate, for the same image quality. This is because the cosine basis functions can afford for high energy compaction. This can be seen by comparison of Figure 2.25(b) with Figure 2.25(a), which reveals that the DCT components are much more concentrated around the origin, than those for the Fourier Transform. This is the compaction property associated with the DCT. The DCT has actually been considered as optimal for image coding, and this is why it is found in the JPEG and MPEG standards for coded image transmission.

The DCT is actually shift variant, due to its cosine basis functions. In other respects, its properties are very similar to the DFT, with one important exception: it has not yet proved possible to implement convolution with the DCT. It is actually possible to calculate the DCT via the FFT. This has been performed in Figure 2.25(b).

The Fourier transform essentially decomposes, or decimates, a signal into sine and cosine components, so the natural partner to the DCT is the *Discrete Sine Transform* (DST). However, the DST transform has odd basis functions (sine) rather than the even ones in the DCT. This lends the DST transform some less desirable properties, and it finds much less application than the DCT.

2.7.2 Discrete Hartley Transform

The *Hartley transform* [Hartley42] is a form of the Fourier transform, but without complex arithmetic, with result for the face image shown in Figure 2.25(c). Oddly, though it sounds like a very rational development, the Hartley transform was first invented in 1942, but not rediscovered and then formulated in discrete form until 1983 [Bracewell83]. One advantage of the Hartley transform is that the forward and inverse transform is the same operation; a disadvantage is that phase is built into the order of frequency components since it is not readily available as the argument of a complex number. The definition of the Discrete Hartley Transform (DHT) replaces the exponent basis by the cas function defined as

$$\text{cas}(t) = \cos(t) + \sin(t) \tag{2.40}$$

Thus the transform components $\mathbf{HP}_{u,v}$ are:

$$\mathbf{HP}_{u,v} = \frac{1}{\sqrt{MN}}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}\mathbf{P}_{x,y}\left(\cos\left(\frac{2\pi}{M}vy\right)+\sin\left(\frac{2\pi}{M}vy\right)\right)\left(\cos\left(\frac{2\pi}{N}ux\right)+\sin\left(\frac{2\pi}{N}ux\right)\right) \tag{2.41}$$

For a square image, this can be written as

$$\mathbf{HP}_{u,v} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}\left(\sin\left(\frac{2\pi}{N}(ux+vy)\right)+\cos\left(\frac{2\pi}{N}(ux-vy)\right)\right) \tag{2.42}$$

The inverse Hartley transform is the same process, but applied to the transformed image.

$$\mathbf{P}_{x,y} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{HP}_{x,y}\left(\sin\left(\frac{2\pi}{N}(ux+vy)\right)+\cos\left(\frac{2\pi}{N}(ux-vy)\right)\right) \tag{2.43}$$

The implementation is then the same for both the forward and the inverse transforms. Again, a fast implementation is available - the *Fast Hartley Transform* [Bracewell84] (though some suggest that it should be called the Bracewell transform, eponymously). It is actually possible to calculate the DFT of a function, *F*(*u*), from its Hartley transform, *H*(*u*). The analysis here is based on 1-dimensional data, but only for simplicity since the argument extends readily to two dimensions. By splitting the Hartley transform into its odd and even parts, *O*(*u*) and *E*(*u*), respectively we obtain:

$$H(u) = O(u) + E(u) \tag{2.44}$$

where:

$$E(u) = \frac{H(u) + H(N-u)}{2} \tag{2.45}$$

and

$$O(u) = \frac{H(u) - H(N-u)}{2} \tag{2.46}$$

The DFT can then be calculated from the DHT simply by

$$F(u) = E(u) - j \times O(u) \tag{2.47}$$

Conversely, the Hartley transform can be calculated from the Fourier transform by:

$$H(u) = \text{Re}[F(u)] - \text{Im}[F(u)] \tag{2.48}$$

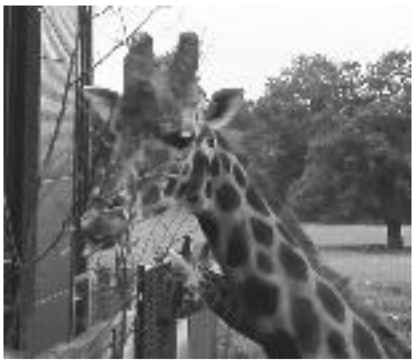
where Re[ ] and Im[ ] denote the real and the imaginary parts, respectively. This emphasises the natural relationship between the Fourier and the Hartley transform. The image of Figure 2.25(c) has been calculated via the 2D FFT using Equation 2.48. Note that the transform in Figure 2.25(c) is the complete transform whereas the Fourier transform in Figure 2.25(a) shows magnitude only. Naturally, as with the DCT, the properties of the Hartley transform mirror those of the Fourier transform. Unfortunately, the Hartley transform does not have shift invariance but there are ways to handle this. Also, convolution requires manipulation of the odd and even parts.

Code 2.5 illustrates the computation of the Hartley transform in Equation 2.28. Similar to Code 2.2 the values are defined for a range of frequencies and the result is processed such that the zero frequency is at the centre of the output array. Notice that this equation is also separable, so it is possible to follow a similar implementation of the Fourier transform by maintaining the first cosine and sine sum in one direction and then performing the multiplication with the terms in the other direction. In this implementation, the components are computed by combining the product of both directions.

```
for u in range(-maxFreqW, maxFreqW + 1):
    entryW = u + maxFreqW
    for v in range(-maxFreqH, maxFreqH + 1):
        entryH = v + maxFreqH
        for x,y in itertools.product(range(0, width), range(0, height)):
            coeff[entryH, entryW] += inputImage[y,x] *                        \
                           (cos(x*ww*u) + sin(x*ww*u)) * (cos(y*wh*v) + sin(y*wh*v))
```

Code 2.5 Hartley Transform

Figure 2.26 shows an example of the transform obtained using Code 2.5. Figure 2.26 (a) shows the input image and Figure 2.26 (b) shows the result of the transform. The resulting components have positive and negative values, so to show the results in an image it is necessary to perform some normalisation. The image shown in Figure 2.26 (b) shows the log of the absolute value, so it is comparable to the Fourier magnitude. Notice that the transform is symmetrical and it is rather similar to the results in Figure 2.16(e). This is because both, the Fourier and Hartley transforms decompose the image into frequency components.


(a) image

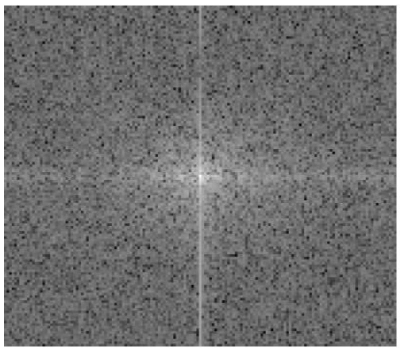

(b) transform modulus

Figure 2.26 Applying the Hartley Transform

2.7.3 Introductory Wavelets

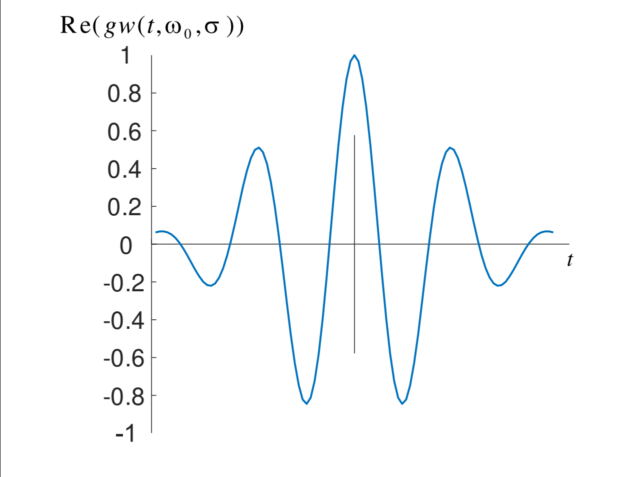
2.7.3.1 Gabor Wavelet

Wavelets are more recent approach to signal processing than the Fourier transform, being introduced only in the nineties [Daubechies90]. Their main advantage is that they allow multi-resolution analysis (analysis at different scales, or resolution). Furthermore, wavelets allow decimation in space and frequency, simultaneously. Earlier transforms actually allow decimation in frequency, in the forward transform, and in time (or position) in the inverse. In this way, the Fourier transform gives a measure of the frequency content of the whole image: the contribution of the image to a particular frequency component. Simultaneous decimation allows us to describe an image in terms of frequency which occurs at a position, as opposed to an ability to measure frequency content across the whole image. Clearly this gives us a greater descriptional power, which can be used to good effect.

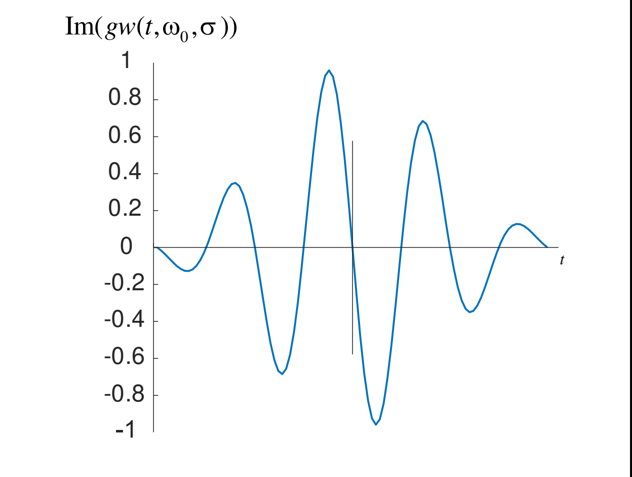
First though, we need a basis function, so that we can decompose a signal. The basis functions in the Fourier transform are sinusoidal waveforms at different frequencies. The function of the Fourier transform is to convolve these sinusoids with a signal to determine how much of each is present. The *Gabor wavelet* is well suited to introductory purposes, since it is essentially a sinewave modulated by a Gaussian envelope. The Gabor wavelet *gw* is given by

$$gw(t,\omega_0,\sigma) = e^{-j\omega_0 t} e^{-\left(\frac{t-t_0}{\sigma}\right)^2} \tag{2.49}$$

where $\omega_0 = 2\pi f_0$ is the modulating frequency, t_0 dictates position and σ controls the width of the Gaussian envelope which embraces the oscillating signal. An example Gabor wavelet is shown in Figure 2.27 which shows the real and the imaginary parts (the modulus is the Gaussian envelope). Increasing the value of ω_0 increases the frequency content within the envelope whereas increasing the value of σ spreads the envelope without affecting the frequency. So why does this allow simultaneous analysis of time and frequency? Given that this function is the one convolved with the test data, then we can compare it with the Fourier transform. In fact, if we remove the term on the right hand side of Equation 2.49, we return to the sinusoidal basis function of the Fourier transform, the exponential in Equation 2.1. Accordingly, we can return to the Fourier transform by setting σ to be very large. Alternatively, setting f_0 to zero removes frequency information. Since we operate in between these extremes, we obtain position and frequency information simultaneously.



(a) real part



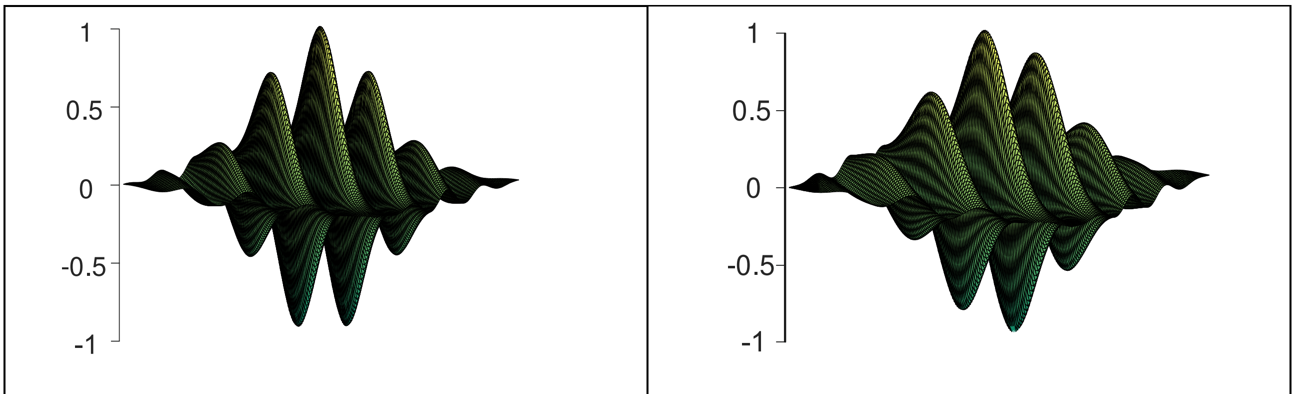
(b) imaginary part

Figure 2.27 An Example Gabor Wavelet

Actually, an infinite class of wavelets exists which can be used as an expansion basis in signal decimation. One approach [Daugman88] has generalised the Gabor function to a 2D form aimed to be optimal in terms of spatial and spectral resolution. These 2D Gabor wavelets are given by

$$gw2\mathrm{D}(x,y,\omega_0,\sigma) = \frac{1}{\sigma\sqrt{\pi}} e^{-\left(\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}\right)} e^{-j\omega_0\left((x-x_0)\cos(\theta)+(y-y_0)\sin(\theta)\right)} \tag{2.50}$$

where x_0 and y_0 control position, $\omega_0 = 2\pi f_0$ controls the frequency of modulation along either axis, and θ controls the direction (orientation) of the wavelet (as implicit in a two dimensional system). Naturally, the shape of the area imposed by the 2D Gaussian function could be elliptical if different variances were allowed along the *x* and *y* axes (the frequency can also be modulated differently along each axis). Figure 2.28, of an example 2D Gabor wavelet, shows that the real and imaginary parts are even and odd functions, respectively; again, different values for f_0 and σ control the frequency and envelope's spread respectively, the extra parameter θ controls rotation.

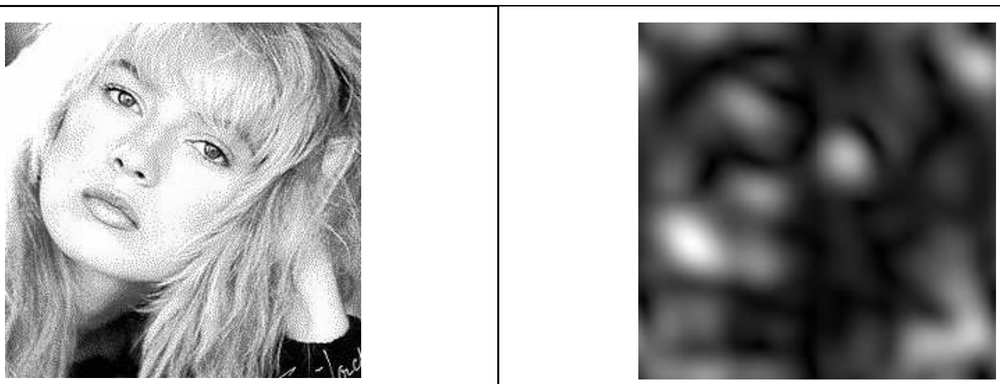


(a) real part

(b) imaginary part

Figure 2.28 Example 2-Dimensional Gabor Wavelet

The function of the wavelet transform is to determine where and how each wavelet specified by the range of values for each of the free parameters occurs in the image. Clearly, there is a wide choice that depends on application. An example transform is given in Figure 2.29. Here, the Gabor wavelet parameters have been chosen in such a way as to select face features: the eyes, nose and mouth have come out very well. (Not that it's anything to do with Gabor, but the lady in the image had an interesting career: she started as lady of the night, went on to be a pop star and then wrote a book about it. In her career she had a lot of positions, especially at the start.) These features are where there is local frequency content with orientation according to the head's inclination. Naturally, these are not the only features with these properties, the cuff of the sleeve is highlighted too! But this does show the Gabor wavelet's ability to select and analyse localised variation in image intensity.



(a) original image

(b) after Gabor wavelet transform

Figure 2.29 An Example Gabor Wavelet Transform

However, the conditions under which a set of continuous Gabor wavelets will provide a complete representation of any image (i.e. that any image can be reconstructed) were developed later. However, the theory is naturally very powerful, since it accommodates frequency and position simultaneously, and further it facilitates multi-resolution analysis - the analysis is then sensitive to scale which is advantageous since objects which are far from the camera appear smaller than those which are close. We shall find wavelets again, when processing images to find low-level features. Amongst applications of Gabor wavelets, we can find measurement of iris texture to give a very powerful security system [Daugman93] and face feature extraction for automatic face recognition [Lades93]. Wavelets continue to develop [Daubechies90] and have found applications in image texture analysis [Laine93], in coding [daSilva96] and in image restoration [Banham96]. Unfortunately, the discrete wavelet transform is not shift invariant, though there are approaches aimed to remedy this (see for example [Donoho95]). As such, we shall not study it further and just note that there is an important class of transforms that combine spatial and spectral sensitivity, and it is likely that this importance will continue to grow.

2.7.3.2 Haar Wavelet

Though Fourier laid the basis for frequency decomposition, the original wavelet approach is now attributed to Alfred Haar's work in 1909. This uses a binary approach, rather than a continuous signal, and has led to fast methods for finding features in images [Oren97] (especially the object detection part of the Viola-Jones face detection approach [Viola01]). Essentially, the binary functions can be considered to form averages over sets of points, thereby giving means for compression and for feature detection. If we are to form a new vector (at level h+1) by taking averages of pairs of elements (and retaining the integer representation) of the N points in the previous vector (at level h of the $\log_2(N)$ levels) as,

$$\mathbf{p}_i^{h+1} = \frac{\mathbf{p}_{2\times i}^h + \mathbf{p}_{2\times i+1}^h}{2} \quad i \in 0 \ldots \frac{N}{2} - 1; h \in 1, \ldots \log_2(N) \tag{2.51}$$

By way of example, consider a vector of points at level 0 as

$$\mathbf{p}^0 = \begin{bmatrix} 1 & 3 & 21 & 19 & 17 & 19 & 1 & -1 \end{bmatrix} \tag{2.52}$$

then the first element in the new vector becomes $(1+3)/2 = 2$ and the next element is $(21+19)/2 = 20$ and so on, so the next level is

$$\mathbf{p}^1 = \begin{bmatrix} 2 & 20 & 18 & 0 \end{bmatrix} \tag{2.53}$$

And is naturally half the number of points. If we also generate some detail, which is how we return to the original points, then we have a vector

$$\mathbf{d}^1 = \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix} \tag{2.54}$$

and when each element of the detail $\mathbf{d}^1$ is successively added and subtracted from the elements of $\mathbf{p}^1$ as $\begin{bmatrix} \mathbf{p}_0^1 + \mathbf{d}_0^1 & \mathbf{p}_0^1 - \mathbf{d}_0^1 & \mathbf{p}_1^1 + \mathbf{d}_1^1 & \mathbf{p}_1^1 - \mathbf{d}_1^1 & \mathbf{p}_2^1 + \mathbf{d}_2^1 & \mathbf{p}_2^1 - \mathbf{d}_2^1 & \mathbf{p}_3^1 + \mathbf{d}_3^1 & \mathbf{p}_3^1 - \mathbf{d}_3^1 \end{bmatrix}$ by which we obtain

$$\begin{bmatrix} 2+(-1) & 2-(-1) & 20+1 & 20-1 & 18+(-1) & 18-(-1) & 0+1 & 0-1 \end{bmatrix}$$

which returns us to the original vector $\mathbf{p}^0$ (Eqn. 2.52). If we continue to similarly form a series of decompositions (averages of adjacent points), together with the detail at each point, we generate

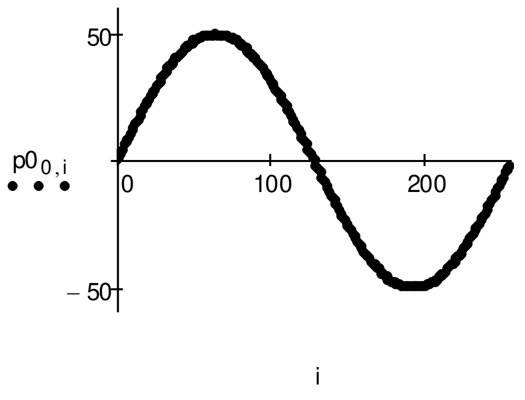
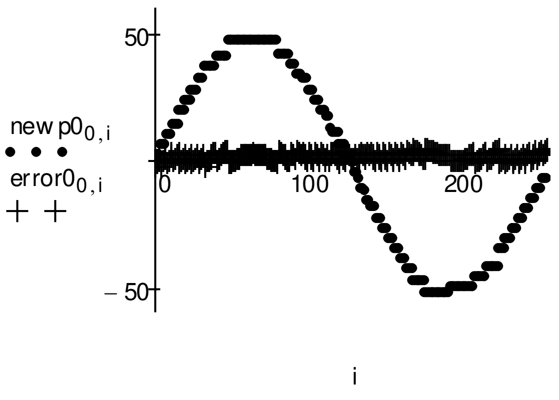
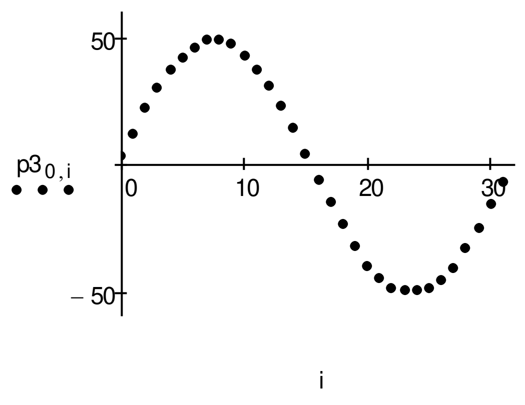
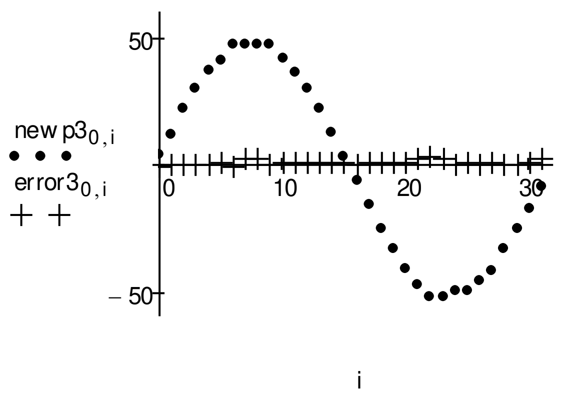
$$\mathbf{p}^2 = \begin{bmatrix} 11 & 9 \end{bmatrix}; \ \mathbf{d}^2 = \begin{bmatrix} -9 & 9 \end{bmatrix} \tag{2.55}$$

$$\mathbf{p}^3 = \begin{bmatrix} 10 \end{bmatrix}; \ \mathbf{d}^3 = \begin{bmatrix} 1 \end{bmatrix} \tag{2.56}$$

We can then store the image as a code

$$\begin{bmatrix} \mathbf{p}^3 & \mathbf{d}^3 & \mathbf{d}^2 & \mathbf{d}^1 \end{bmatrix} = \begin{bmatrix} 10 & 1 & -9 & 9 & -1 & 1 & -1 & 1 \end{bmatrix} \tag{2.57}$$

The process is illustrated in Fig. 2.30 for a sinewave. Fig. 2.30(a) shows the original sinewave, (b) shows the decomposition to level 3, and it is a close but discrete representation whereas (c) shows the decomposition to level 6, which is very coarse. The original signal can be reconstructed from the final code, and this is without error. If the signal is reconstructed by filtering the detail to reduce the amount of stored data, the reconstruction of the original signal (d) at level 0 is quite close to the original signal, and the reconstruction at the other levels is similarly close as expected. The reconstruction error is also shown in (d) to (f). Components of the detail (of magnitude less than one) were removed, achieving a compression ratio of approximately 50%. Naturally, a Fourier transform would encode the signal better, as the Fourier transform is best suited to representing a sinewave. Like Fourier this discrete approach can encode the signal, we can also reconstruct the original signal (reverse the process), and shows how the signal can be represented at different scales, since there are less points in the higher levels.

Binary decomposition	Reconstruction after filtering/ compression
 **(a)** original signal, $\mathbf{p}^0$	 **(d)** level 0, $\mathbf{p}^0$
 **(b)** level 3, $\mathbf{p}^3$	 **(e)** level 3, $\mathbf{p}^3$

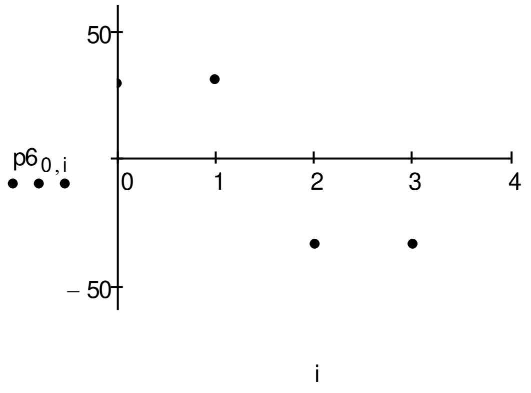
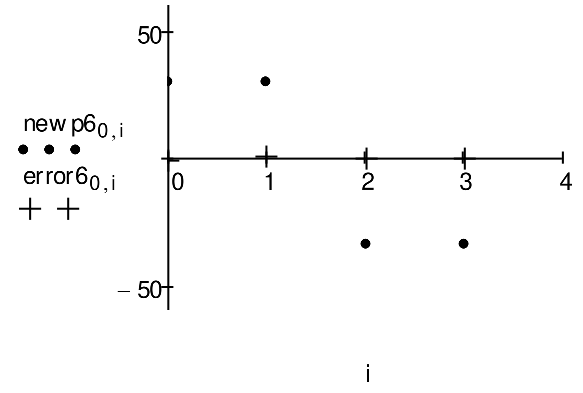
 **(c)** level 6, $\mathbf{p}^6$	 **(f)** level 6, $\mathbf{p}^6$

Figure 2.30 Binary Signal Decomposition and Reconstruction

By Eqn. 2.57 this gives a set of numbers of the same size as the original data, and is an alternative representation from which we can reconstruct the original data. There are two important differences:

i) we have an idea of scale by virtue of the successive averaging ($\mathbf{p}^1$ is similar in structure to $\mathbf{p}^0$, but at a different scale) ;

and ii) we can compress (or code) the image by removing the small numbers in the new representation (by setting them to zero, noting that there are efficient ways of encoding structures containing large numbers of zeros).

A process of successive averaging and differencing can be expressed as a function of the form in Fig. 2.31. This is a mother wavelet which can be applied at different scales, but retains the same shape at those scales. So we now have a binary decomposition rather than the sinewaves of the Fourier transform.

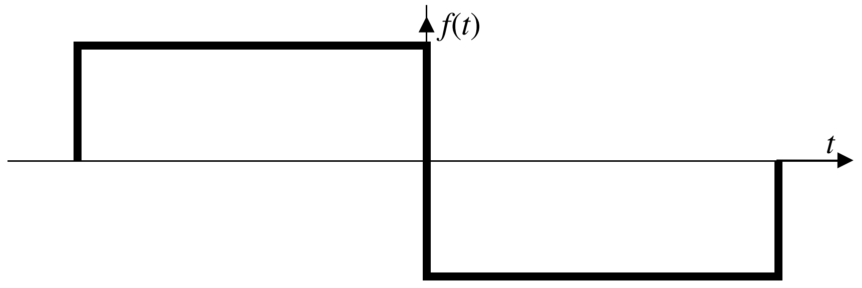
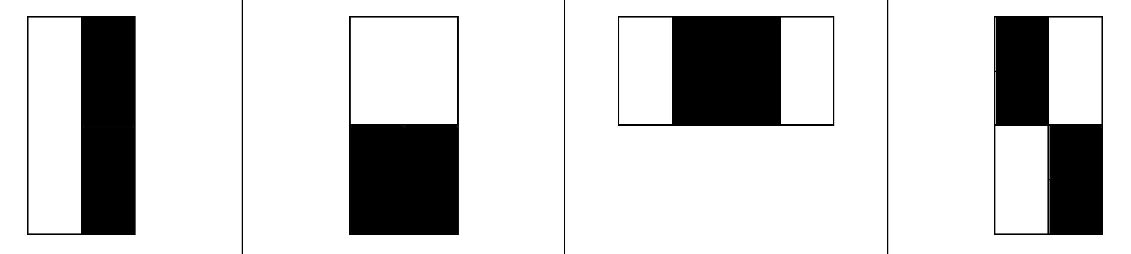


Figure 2.31 Example Haar Wavelet Function

To detect objects, these wavelets need to be arranged in two dimensions. These can be arranged to provide for object detection, by selecting the two-dimensional arrangement of points. By defining a relationship which is a summation of the points in an image prior to a given point

$$\mathbf{sP}_{x,y} = \sum_{x'<x,y'<y} \mathbf{P}_{x',y'} \tag{2.58}$$

Then we can achieve wavelet type features which are derived by using these summations. Four of these wavelets are shown in Fig. 2.32. These are placed at selected positions in the image to which they are applied. There are white and black areas: the sum of the pixels under the white area(s) is subtracted from of the sum of the pixels under the dark area(s), in a way similar to the earlier averaging operation in Eqn. 2.52. The first template, Fig. 2.32(a), will detect shapes which are brighter on one side than the other; the second (b) will detect shapes which are brighter in a vertical sense; the third will detect a dark object which has brighter areas on either side. There is a family of these arrangements, and that can apply at selected levels of scale. By collecting the analysis, we can determine objects whatever their position, size (objects further away will appear smaller) or rotation. We will dwell on these topics later and how we find and classify shapes. The point here is that we can achieve some form of binary decomposition in two dimensions, as opposed to the sine/ cosine decomposition of the Gabor wavelet whilst retaining selectivity to scale and position (similar to the Gabor wavelet). This is also simpler, so the binary functions can be processed more quickly.



(a) horizontal **(b)** vertical **(c)** bar **(d)** diagonal

Figure 2.32 Example Haar Wavelet Image Functions

2.7.4 Other Transforms

Decomposing a signal into sinusoidal components was actually one of the first approaches to transform calculus, and this is why the Fourier transform is so important. The sinusoidal functions are actually called *basis functions*, the implicit assumption is that the basis functions map well to the signal components. As such, the Haar wavelets are binary basis functions. There is (theoretically) an infinite range of basis functions. Discrete signals can map better into collections of binary components rather than sinusoidal ones. These collections (or sequences) of binary data are called sequency components and form the basis of the *Walsh transform* [Walsh23], which is a global transform when compared with the Haar functions (like Fourier compared with Gabor). This has found wide application in the interpretation of digital signals, though it is less widely used in image processing (one disadvantage is the lack of shift invariance). The *Karhunen-Loéve* transform [Karhunen47] [Loéve48] (also called the *Hotelling* transform from which it was derived, or more popularly *Principal Component Analysis*) is a way of analysing (statistical) data to reduce it to those data which are informative, discarding those which are not.

2.8 Applications using Frequency Domain Properties

Filtering is a major use of Fourier transforms, particularly because we can understand an image, and how to process it, much better in the frequency domain. An analogy is the use of a graphic equaliser to control the way music sounds. In images, if we want to remove high-frequency information (like the hiss on sound) then we can filter, or remove, it by inspecting the Fourier transform. If we retain low-frequency components, we implement a *low-pass filter*. The low-pass filter describes the area in which we retain spectral components, the size of the area dictates the range of frequencies retained, and is known as the filter's bandwidth. If we retain components within a circular region centred on the d.c. component, and inverse Fourier transform the filtered transform then the resulting image will be blurred. Higher spatial frequencies exist at the sharp