$$p_1(t) * p_2(t) = \int_{-\infty}^{\infty} p_1(\tau) p_2(t-\tau) d\tau \qquad (2.10)$$

This is actually the basis of systems theory where the output of a system is the convolution of a stimulus, say p_1, and a system's response, p_2, By inverting the time axis of the system response, to give $p_2(t-\tau)$ we obtain a memory function. The convolution process then sums the effect of a stimulus multiplied by the memory function: the current output of the system is the cumulative response to a stimulus. By taking the Fourier transform of Equation 2.10, the Fourier transform of the convolution of two signals is

$$\begin{aligned}\Im\left[p_1(t) * p_2(t)\right] &= \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} p_1(\tau) p_2(t-\tau) d\tau \right\} e^{-j\omega t} dt \\ &= \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} p_2(t-\tau) e^{-j\omega t} dt \right\} p_1(\tau) d\tau \end{aligned} \qquad (2.11)$$

Now since $\Im\left[p_2(t-\tau)\right] = e^{-j\omega\tau} Fp_2(\omega)$ (to be considered later in Section 2.6.1), then

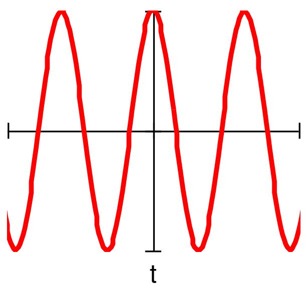
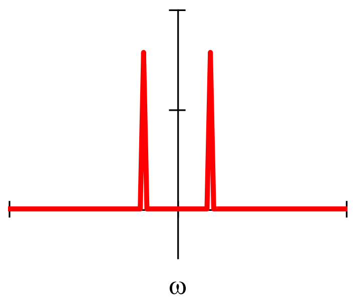
$$\begin{aligned}\Im\left[p_1(t) * p_2(t)\right] &= \int_{-\infty}^{\infty} Fp_2(\omega) p_1(\tau) e^{-j\omega\tau} d\tau \\ &= Fp_2(\omega) \int_{-\infty}^{\infty} p_1(\tau) e^{-j\omega\tau} d\tau \\ &= Fp_2(\omega) \times Fp_1(\omega) \end{aligned} \qquad (2.12)$$

As such, the frequency domain dual of convolution is multiplication; the convolution integral can be performed by *inverse Fourier transformation* of the product of the transforms of the two signals. A frequency domain representation essentially presents signals in a different way but it also provides a different way of processing signals. Later we shall use the duality of convolution to speed up the computation of vision algorithms considerably.

Further, *correlation* is defined to be

$$p_1(t) \otimes p_2(t) = \int_{-\infty}^{\infty} p_1(\tau) p_2(t+\tau) d\tau \qquad (2.13)$$

where $\otimes$ denotes correlation ($\odot$ is another symbol which is used sometimes, but there is not much consensus on this symbol – if comfort is needed: "in esoteric astrology $\odot$ represents the creative spark of divine consciousness" no less!). Correlation gives a measure of the match between the two signals $p_2(\omega)$ and $p_1(\omega)$. When $p_2(\omega) = p_1(\omega)$ we are correlating a signal with itself and the process is known as *autocorrelation*. We shall be using correlation later, to find things in images.

Time Domain Signals	Frequency Domain Spectra
	

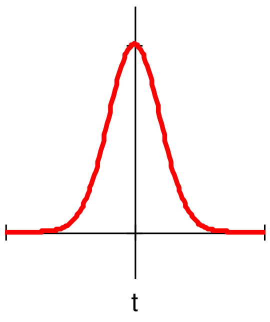
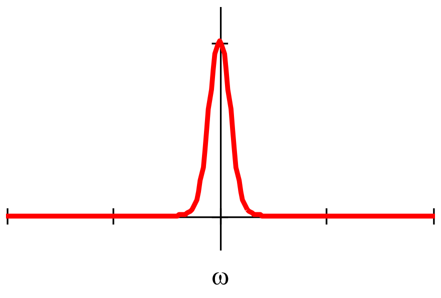
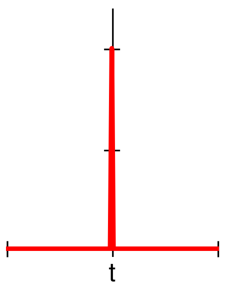
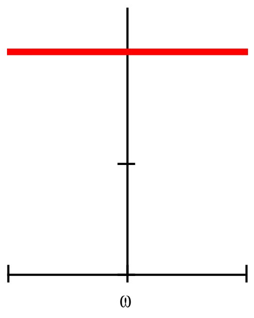
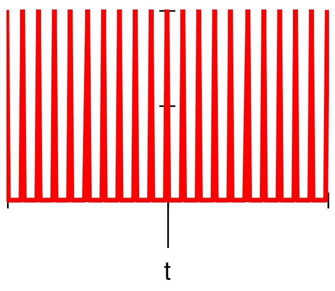
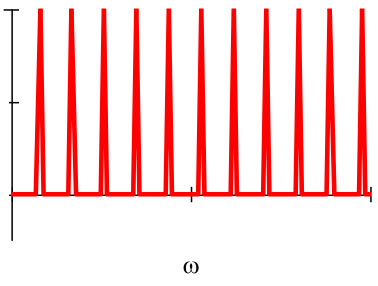
(a) cosine wave	**(b)** Fourier transform of cosine wave
 **(c)** Gaussian function	 **(d)** spectrum of Gaussian function
 **(e)** delta function	 **(f)** frequency content of delta function
 **(g)** sampling function in time domain	 **(h)** transform of sampling function

Figure 2.6 Fourier Transform Pairs

Before proceeding further, we also need to define the *delta function*, which can be considered to be a function occurring at a particular time interval:

$$delta(t-\tau) = \left| \begin{array}{ll} 1 & \text{if} \quad t=\tau \\ 0 & \text{otherwise} \end{array} \right. \tag{2.14}$$

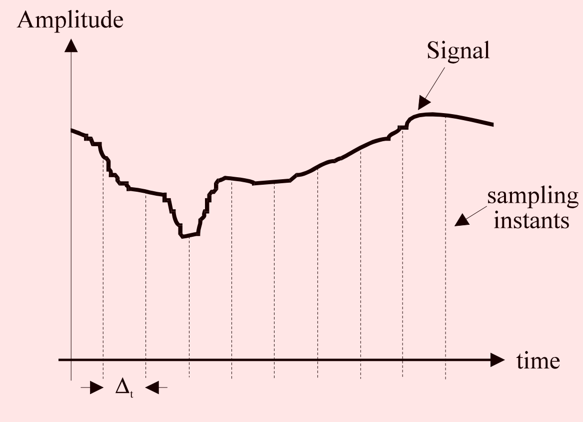
The relationship between a signal's time domain representation and its frequency domain version is also known as a *transform pair*: the transform of a pulse (in the time domain) is a sinc function in the frequency domain. Since the transform is symmetrical, the Fourier transform of a sinc function is a pulse.

There are other Fourier transform pairs, as illustrated in Figure 2.6. Firstly, Figures 2.6(a) and (b) show that the Fourier transform of a cosine function is two points in the frequency domain (at the same value for positive and negative frequency) – we expect this since there is only one frequency in the cosine function, the frequency shown by its transform. Figures 2.6(c) and (d) show that the transform of the *Gaussian function* is another Gaussian function, this illustrates linearity (for linear systems it's Gaussian in, Gaussian out which is another version of GIGO). Figure 2.6(e) is a single point (the delta function) which has a transform that is an infinite set of frequencies, Figure 2.6(f), an alternative interpretation is that a delta function contains an equal amount of all frequencies. This can be explained by using Equation 2.5 where if the pulse is of shorter duration (T tends to zero), the sinc function is wider; as the pulse becomes infinitely thin, the spectrum becomes infinitely flat.

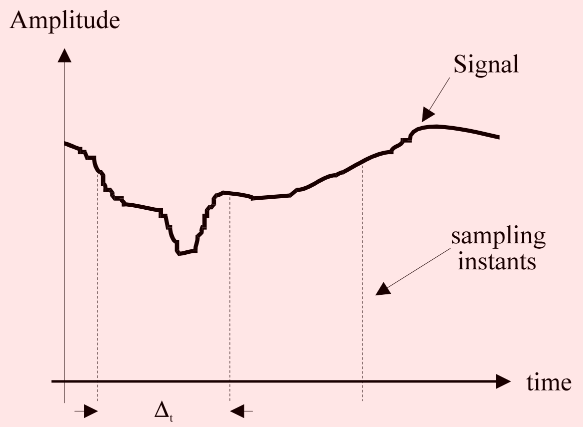
Finally, Figures 2.6(g) and (h) show that the transform of a set of uniformly-spaced delta functions is another set of uniformly-spaced delta functions, but with a different spacing. The spacing in the frequency domain is the reciprocal of the spacing in the time domain. By way of a (non-mathematical) explanation, let us consider that the Gaussian function in Figure 2.6(c) is actually made up by summing a set of closely spaced (and very thin) Gaussian functions. Then, since the spectrum for a delta function is infinite, as the Gaussian function is stretched in the time domain (eventually to be a set of pulses of uniform height) we obtain a set of pulses in the frequency domain, but spaced by the reciprocal of the time domain spacing. This transform pair is actually the basis of sampling theory (which we aim to use to find a criterion which guides us to an appropriate choice for the image size).

2.4 The Sampling Criterion

The *sampling criterion* specifies the condition for the correct choice of sampling frequency. *Sampling* concerns taking instantaneous values of a continuous signal, physically these are the outputs of an A/D converter sampling a camera signal. Clearly, the samples are the values of the signal at sampling instants. This is illustrated in Figure 2.7 where Figure 2.7(a) concerns taking samples at a high frequency (the spacing between samples is low), compared with the amount of change seen in the signal of which the samples are taken. Here, the samples are taken sufficiently fast to notice the slight dip in the sampled signal. Figure 2.7(b) concerns taking samples at a low frequency, compared with the rate of change of (the maximum frequency in) the sampled signal. Here, the slight dip in the sampled signal is not seen in the samples taken from it.



(a) sampling at high frequency



(b) sampling at low frequency

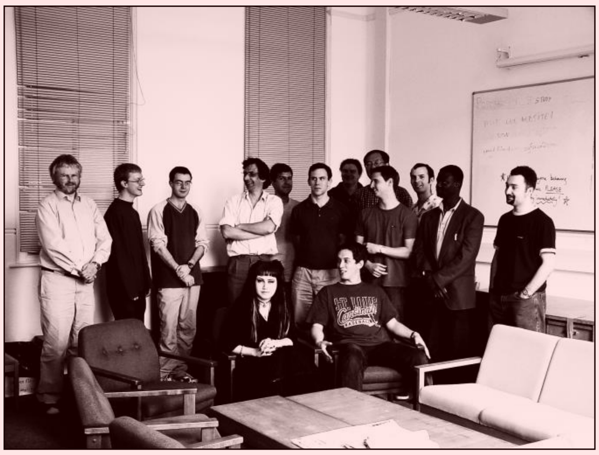
Figure 2.7 Sampling at Different Frequencies

We can understand the process better in the frequency domain. Let us consider a time-variant signal which has a range of frequencies between $-f_{max}$ and f_{max} as illustrated in Figure 2.9(b). This range of frequencies is shown by the Fourier transform where the signal's spectrum exists only between these frequencies. This function is sampled every Δ_t s: this is a sampling function of spikes occurring every Δ_t s. The Fourier transform of the sampling function is a series of spikes separated by $f_{sample} = 1/\Delta_t$ Hz. The Fourier pair of this transform was illustrated earlier, Figures 2.6(g) and (h).

The sampled signal is the result of multiplying the time-variant signal by the sequence of spikes, this gives samples that occur every Δ_t s, and the sampled signal is shown in Figure 2.9(a). These are the outputs of the A/D converter at sampling instants. The frequency domain analogue of this sampling process is to convolve the spectrum of the time-variant signal with the spectrum of the sampling function. Convolving the signals, the convolution process, implies that we take the spectrum of one, flip it along the horizontal axis and then slide it across the other. Taking the spectrum of the time-variant signal and sliding it over the spectrum of the spikes, results in a spectrum where the spectrum of the original signal is repeated every $1/\Delta_t$ Hz, f_{sample} in Figure 2.9(b-d). If the spacing between samples is Δ_t, the repetitions of the time-variant signal's spectrum are spaced at intervals of $1/\Delta_t$, as in Figure 2.9(b). If the sample spacing is large, then the time-variant signal's spectrum is replicated close together and the spectra collide, or interfere, as in Figure 2.9(d). The spectra just touch when the sampling frequency is twice the maximum frequency in the signal. If the frequency domain spacing, f_{sample}, is more than twice the maximum frequency, f_{max}, the spectra do not collide or interfere, as in Figure 2.9(c). If the sampling frequency exceeds twice the maximum frequency then the spectra cannot collide. This is the *Nyquist sampling criterion*:

In order to reconstruct a signal from its samples, the sampling frequency must be at least twice the highest frequency of the sampled signal.

If we do not obey Nyquist's sampling theorem the spectra collide. When we inspect the sampled signal, whose spectrum is within $-f_{max}$ to f_{max}, wherein the spectra collided, the corrupt spectrum implies that by virtue of sampling, we have ruined some of the information. If we were to attempt to reconstruct a signal by inverse Fourier transformation of the sampled signal's spectrum, processing Figure 2.9(d) would lead to the wrong signal whereas inverse Fourier transformation of the frequencies between $-f_{max}$ and f_{max} in Figures 2.9(b) and (c) would lead back to the original signal. This can be seen in computer images as illustrated in Figure 2.8 which show an image of a group of people (the computer vision research team at Southampton) displayed at different spatial resolutions (the contrast has been increased to the same level in each sub-image, so that the effect we want to demonstrate should definitely show up in the print copy). Essentially, the people become less distinct in the lower resolution image, Figure 2.8(b). Now, look closely at the window blinds behind the people. At higher resolution, in Figure 2.8(a), these appear as normal window blinds. In Figure 2.8(b), which is sampled at a much lower resolution, a new pattern appears: the pattern appears to be curved - and if you consider the blinds' relative size the shapes actually appear to be much larger than normal window blinds. So by reducing the resolution, we are seeing something different, an *alias* of the true information – something that is not actually there at all, but appears to be there by result of sampling. This is the result of sampling at too low a frequency: if we sample at high frequency, the interpolated result matches the original signal; if we sample at too low a frequency we can get the wrong signal. (For these reasons people on television tend to wear nonchequered clothes – or should not!). Note that this effect can be seen, in the way described, in the printed version of this book. This is because the printing technology is very high resolution. If you were to print this page offline (and sometimes even to view it), e.g. from a Google Books sample, the nature of the effect of aliasing depends on the resolution of the printed image, and so the aliasing effect might also be seen in the high resolution image as well as in the low resolution version – which rather spoils the point.


(**a**) high resolution

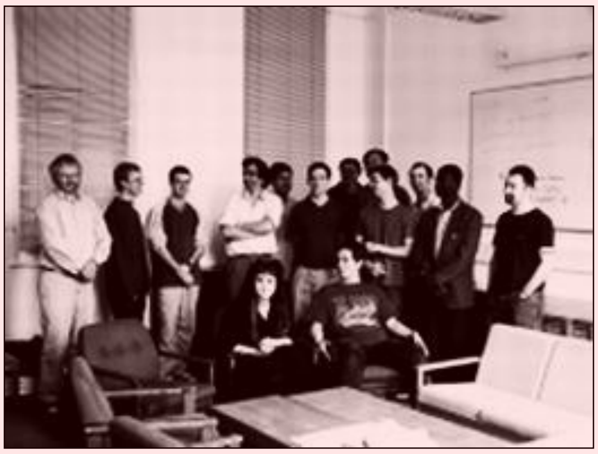
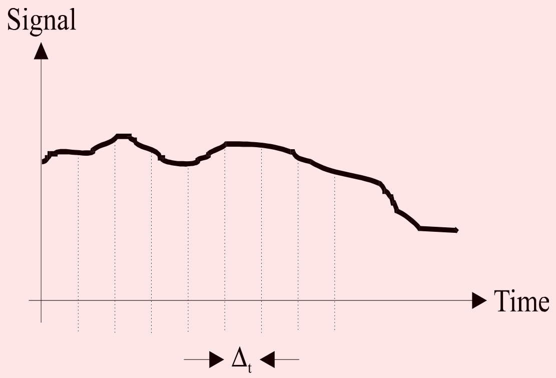

(**b**) low resolution (aliased)

Figure 2.8 Aliasing in Sampled Imagery

In art, Dali's picture "Gala Contemplating the Mediterranean Sea, which at 20 meters becomes the portrait of Abraham Lincoln" (Homage to Rothko) is a classic illustration of sampling. At high resolution you see a detailed surrealist image, with Mrs Dali as a central figure. Viewed from a distance - or for the shortsighted, without your spectacles on - the image becomes a (low resolution) picture of Abraham Lincoln. For a more modern view of sampling [Unser00] is well worth a look. The *compressive sensing* approach [Donoho06] takes advantage of the fact that many signals have components that are significant, or nearly zero, leading to cameras which acquire significantly fewer elements to represent an image. This provides an alternative basis for compressed image acquisition without loss of resolution.



(**a**) sampled signal

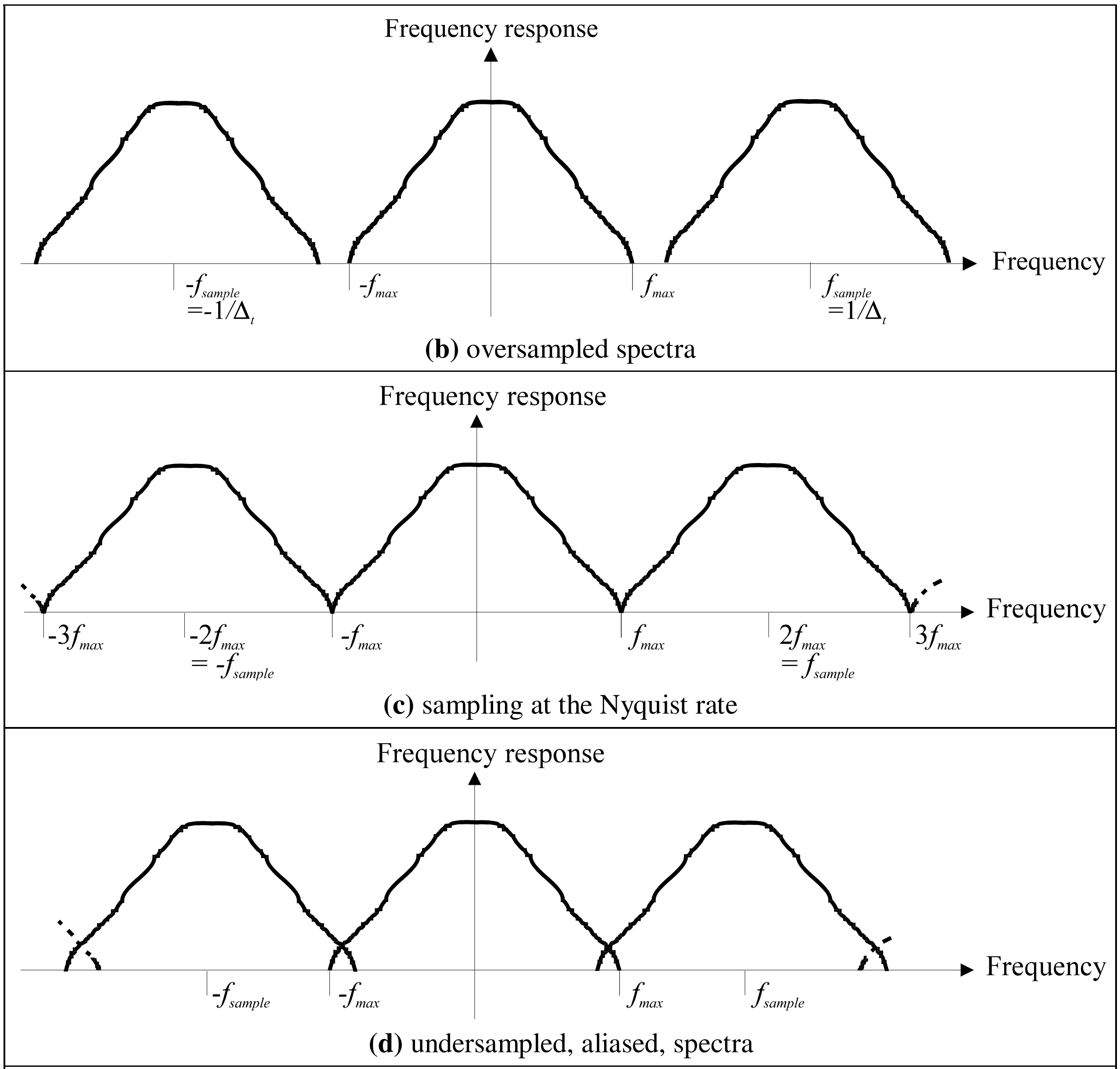
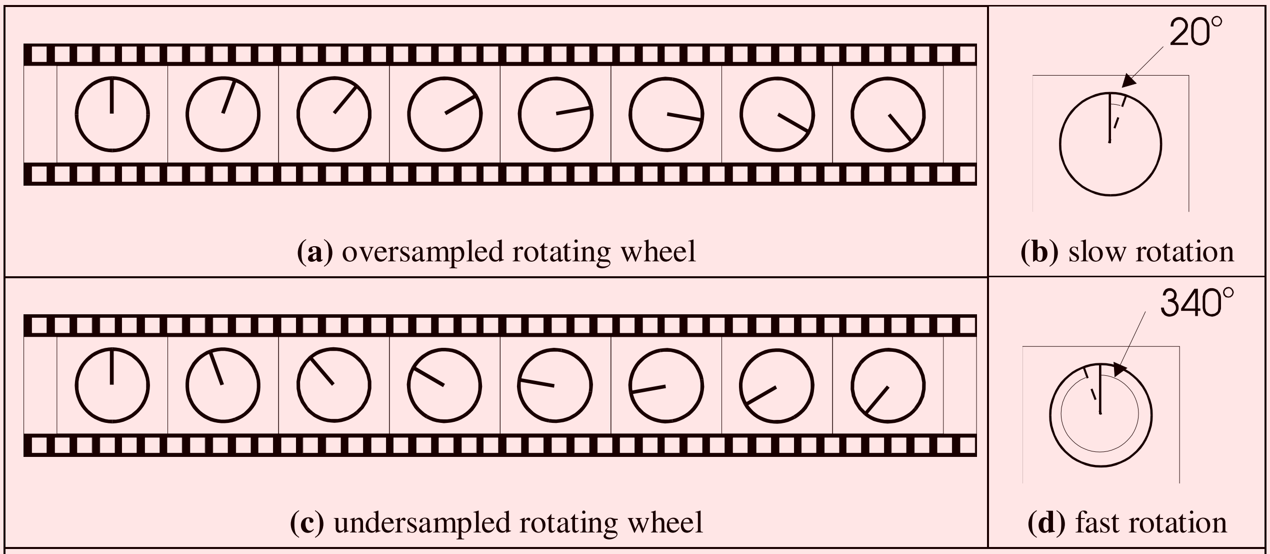


Figure 2.9 Sampled Spectra

Obtaining the wrong signal is called *aliasing*: our interpolated signal is an alias of its proper form. Clearly, we want to avoid aliasing, so according to the sampling theorem we must sample at twice the maximum frequency of the signal coming out of the camera. The maximum frequency is defined to be 5.5 MHz so we must sample the camera signal at 11 MHz. (For information, when using a computer to analyse speech we must sample the speech at a minimum frequency of 12 kHz since the maximum speech frequency is 6 kHz.) Given the timing of a video signal, sampling at 11 MHz implies a minimum image resolution of 576×576 pixels. This is unfortunate: 576 is not an integer power of two which has poor implications for storage and processing. Accordingly, since some image processing systems have a maximum resolution of 512×512, they must anticipate aliasing. This is mitigated somewhat by the observations that:

1. globally, the lower frequencies carry more information whereas locally the higher frequencies contain more information so the corruption of high frequency information is of less importance; and
2. there is limited depth of focus in imaging systems (reducing high frequency content).

But aliasing can, and does, occur and we must remember this when interpreting images. A different form of this argument applies to the images derived from digital cameras. The basic argument that the precision of the estimates of the high order frequency components is dictated by the relationship between the effective sampling frequency (the number of image points) and the imaged structure, naturally still applies.



(a) oversampled rotating wheel **(b)** slow rotation **(c)** undersampled rotating wheel **(d)** fast rotation

Figure 2.10 Correct and Incorrect Apparent Wheel Motion

The effects of sampling can often be seen in films, especially in the rotating wheels of cars, as illustrated in Figure 2.10. This shows a wheel with a single spoke, for simplicity. The film is a sequence of frames starting on the left. The sequence of frames plotted Figure 2.10(a) is for a wheel which rotates by 20^o between frames, as illustrated in Figure 2.10(b). If the wheel is rotating much faster, by 340^o between frames, as in Figure 2.10(c) and Figure 2.10(d), to a human viewer the wheel will appear to rotate in the opposite direction. If the wheel rotates by 360^o between frames, it will appear to be stationary. In order to perceive the wheel as rotating forwards, then the rotation between frames must be 180^o at most. This is consistent with sampling at more than twice the maximum frequency. Our eye can resolve this in films (when watching a film, we bet you haven't thrown a wobbly because the car's going forwards whereas the wheels say it's going the other way) since we know that the direction of the car must be consistent with the motion of its wheels, and we expect to see the wheels appear to go the wrong way, sometimes.

2.5 The Discrete Fourier Transform (DFT)

2.5.1 One Dimensional Transform

Given that image processing concerns sampled data, we require a version of the Fourier transform which handles this. This is known as the *discrete Fourier transform* (DFT). The DFT of a set of N points $\mathbf{p}_x$ (sampled at a frequency which at least equals the Nyquist sampling rate) into sampled frequencies $\mathbf{Fp}_u$ is:

$$\mathbf{Fp}_u = \frac{1}{N}\sum_{x=0}^{N-1}\mathbf{p}_x e^{-j\left(\frac{2\pi}{N}\right)xu} \tag{2.15}$$

where the scaling coefficient $1/N$ ensures the d.c. coefficient $\mathbf{Fp}_0$ is the average of all samples. Equation 2.15 is a discrete analogue of the continuous Fourier transform: the continuous signal is replaced by a set of samples, the continuous frequencies by sampled ones, and the integral is replaced by a summation. If the DFT is applied to samples of a pulse in a window from sample 0 to sample $N/2-1$ (when the pulse ceases), the equation becomes:

$$\mathbf{Fp}_u = \frac{1}{N}\sum_{x=0}^{\frac{N}{2}-1} A e^{-j\left(\frac{2\pi}{N}\right)xu} \tag{2.16}$$

And since the sum of a geometric progression can be evaluated according to:

$$\sum_{k=0}^{n} a_0 r^k = \frac{a_0\left(1-r^{n+1}\right)}{1-r} \tag{2.17}$$

the discrete Fourier transform of a sampled pulse is given by:

$$\mathbf{Fp}_u = \frac{A}{N}\left(\frac{1-e^{-j\left(\frac{2\pi}{N}\right)\left(\frac{N}{2}\right)u}}{1-e^{-j\left(\frac{2\pi}{N}\right)u}}\right) \tag{2.18}$$

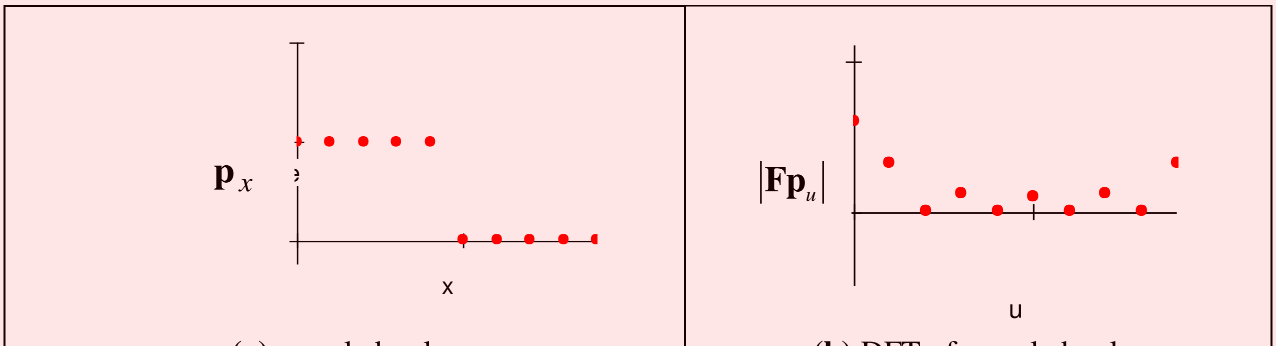
By rearrangement, we obtain:

$$\mathbf{Fp}_u = \frac{A}{N} e^{-j\left(\frac{\pi u}{2}\right)\left(1-\frac{2}{N}\right)} \frac{\sin\left(\pi u/2\right)}{\sin\left(\pi u/N\right)} \tag{2.19}$$

The modulus of the transform is:

$$\left|\mathbf{Fp}_u\right| = \frac{A}{N}\left|\frac{\sin\left(\pi u/2\right)}{\sin\left(\pi u/N\right)}\right| \tag{2.20}$$

since the magnitude of the exponential function is 1. The original pulse is plotted Figure 2.11(a) and the magnitude of the Fourier transform plotted against frequency is given in Figure 2.11(b)



(a) sampled pulse

(b) DFT of sampled pulse

Figure 2.11 Transform Pair for Sampled Pulse

This is clearly comparable with the result of the continuous Fourier transform of a pulse, Figure 2.3, since the transform involves a similar, sinusoidal, signal. The spectrum is equivalent to a set of sampled frequencies; we can build up the sampled pulse by adding up the frequencies according to the Fourier description. Consider a signal such as that shown in Figure 2.12(a). This has no explicit analytic definition, as such it does not have a closed Fourier transform; the Fourier transform is generated by direct application of Equation 2.15. The result is a set of samples of frequency, Figure 2.12(b).

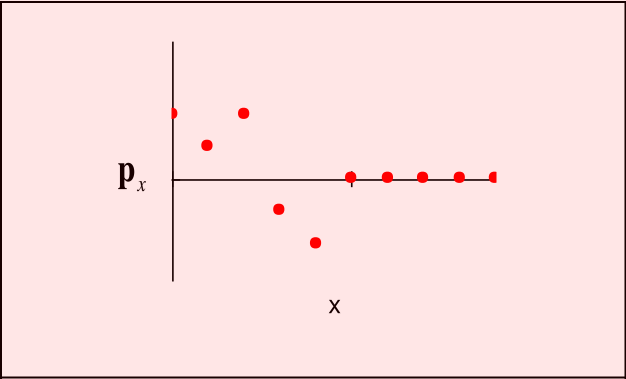
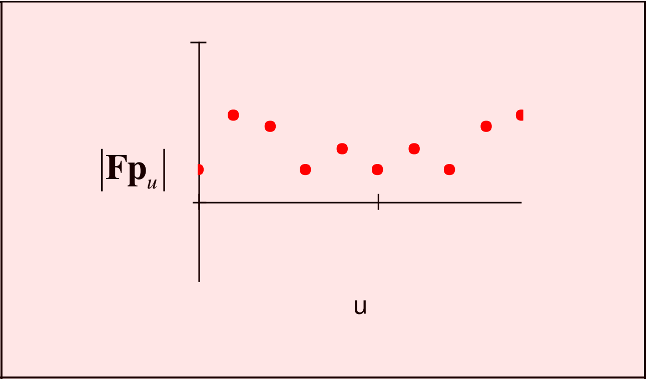
	
(a) sampled signal	**(b)** transform of sampled signal

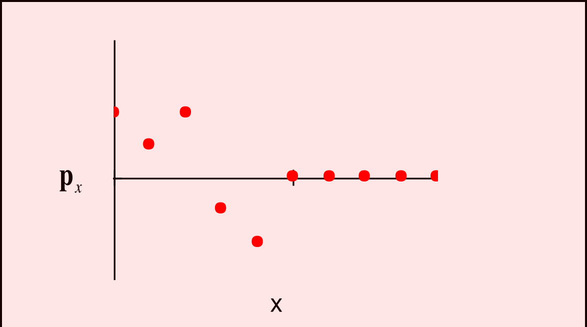
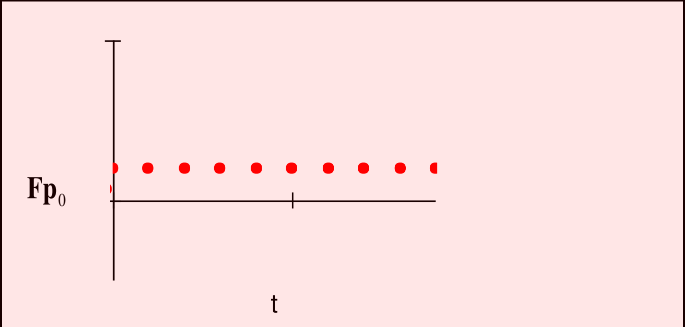
Figure 2.12 A Sampled Signal and its Discrete Transform

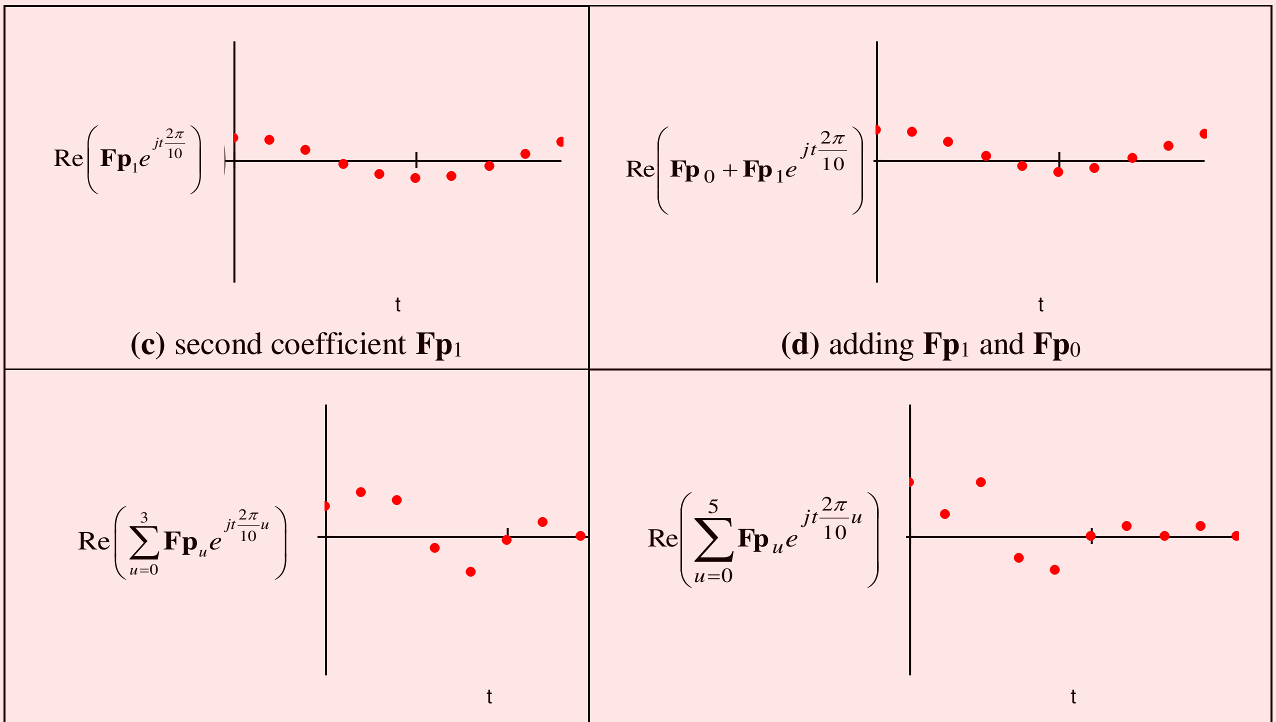
The Fourier transform in Figure 2.12(b) can be used to reconstruct the original signal in Figure 2.12(a), as illustrated in Figure 2.13. Essentially, the coefficients of the Fourier transform tell us how much there is of each of a set of sinewaves (at different frequencies), in the original signal. The lowest frequency component $\mathbf{Fp}_0$, for zero frequency, is called the *d.c. component* (it is constant and equivalent to a sinewave with no frequency) and it represents the average value of the samples. Adding the contribution of the first coefficient $\mathbf{Fp}_0$, Figure 2.13(b), to the contribution of the second coefficient $\mathbf{Fp}_1$, Figure 2.13(c), is shown in Figure 2.13(d). This shows how addition of the first two frequency components approaches the original sampled signal. The approximation improves when the contribution due to the fourth component, $\mathbf{Fp}_3$, is included, as shown in Figure 2.13(e). Finally, adding up all six frequency components gives a close approximation to the original signal, as shown in Figure 2.13(f).

This process is, of course, the *inverse DFT*. This can be used to reconstruct a sampled signal from its frequency components by:

$$\mathbf{p}_x = \sum_{u=0}^{N-1} \mathbf{Fp}_u e^{j\left(\frac{2\pi}{N}\right)ux} \tag{2.21}$$

Note that there are several assumptions made prior to application of the DFT. The first is that the sampling criterion has been satisfied. The second is that the sampled function replicates to infinity. When generating the transform of a pulse, Fourier theory assumes that the pulse repeats outside the window of interest. (There are window operators that are designed specifically to handle difficulty at the ends of the sampling window.) Finally, the maximum frequency corresponds to half the sampling period. This is consistent with the assumption that the sampling criterion has not been violated, otherwise the high frequency spectral estimates will be corrupted.

	
(a) original sampled signal	**(b)** first coefficient $\mathbf{Fp}_0$



(c) second coefficient $\mathbf{Fp}_1$

(d) adding $\mathbf{Fp}_1$ and $\mathbf{Fp}_0$

(e) adding $\mathbf{Fp}_0$, $\mathbf{Fp}_1$, $\mathbf{Fp}_2$ and $\mathbf{Fp}_3$

(f) adding all six frequency components

Figure 2.13 Signal Reconstruction from its Transform Components

.

2.5.2 Two Dimensional Transform

Equation 2.15 gives the DFT of a one-dimensional signal. We need to generate Fourier transforms of images so we need a *two-dimensional discrete Fourier transform*. This is a transform of pixels (sampled picture points) with a two dimensional spatial location indexed by coordinates x and y. This implies that we have two dimensions of frequency, u and v, which are the horizontal and vertical spatial frequencies, respectively. Given an image of a set of vertical lines, the Fourier transform will show only horizontal spatial frequency. The vertical spatial frequencies are zero since there is no vertical variation along the y axis. The two dimensional Fourier transform evaluates the frequency data, $\mathbf{FP}_{u,v}$, from the $N \times N$ pixels $\mathbf{P}_{x,y}$ as:

$$\mathbf{FP}_{u,v} = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathbf{P}_{x,y} e^{-j\left(\frac{2\pi}{N}\right)(ux+vy)} \tag{2.22}$$

Where the scaling coefficient $1/N^2$ makes the d.c. coefficient $\mathbf{FP}_{0,0}$ equal the average of all points in the image (in Matlab the scaling coefficient is 1.0). The Fourier transform of an image can actually be obtained *optically* by transmitting a laser through a photographic slide and forming an image using a lens. The Fourier transform of the image of the slide is formed in the front focal plane of the lens. This is still restricted to transmissive systems whereas reflective formation would widen its application potential considerably (since optical computation is just slightly faster than its digital counterpart). The magnitude of the 2D DFT of an image of vertical bars (Figure 2.14(a)) is shown in Figure 2.14(b). This shows that there are only horizontal spatial frequencies; the image is constant in the vertical axis and there are no vertical spatial frequencies.

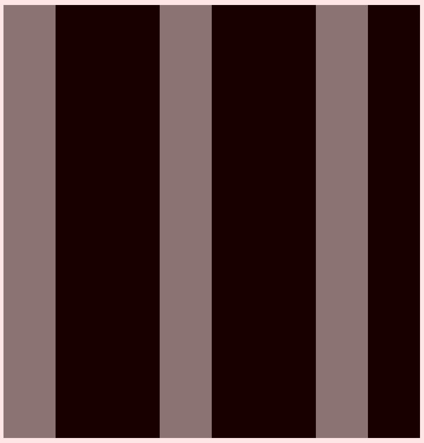
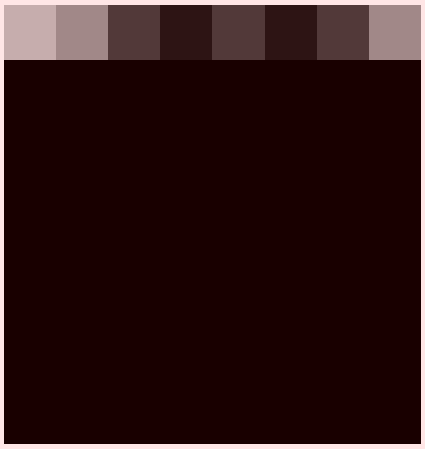
	
(a) image of vertical bars	**(b)** Fourier transform of bars

Figure 2.14 Applying the 2D Discrete Fourier Transform

The *two-dimensional (2D) inverse DFT* transforms from the frequency domain back to the image domain, to reconstruct the image. The 2D inverse DFT is given by:

$$\mathbf{P}_{x,y} = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\mathbf{FP}_{u,v}e^{j\left(\frac{2\pi}{N}\right)(ux+vy)} \tag{2.23}$$

The contribution of different frequencies is illustrated in Figure 2.15 where we have images showing in the first row (a)-(d) the position of the image transform components (presented as log[magnitude]), in the second row (f)-(i) the image constructed from that single component, and in the third row (j)-(m) the reconstruction (by the inverse FT) using frequencies up to and including that component. There is also the image of the magnitude of the Fourier transform, (e). We shall take the transform components from a circle centred at the middle of the transform image. In Figure 2.15 the first column is the transform components at radius 1 (which are low frequency components), the second column is the radius 4 components, the third column is at radius 9 and the fourth column is the radius 25 components (the higher frequency components). The last column has the complete Fourier transform image, (e), and the reconstruction of the image from the transform, (n). As we include more components we include more detail; the lower order components carry the bulk of the shape, not the detail. In the bottom row, the first components plus the d.c. component give a very coarse approximation Figure 2.15(j) when the components up to radius four are added we can see the shape of a face Figure 2.15(k); the components up to radius 9 order allow us to see the face features Figure 2.15(l), but they are not sharp; we can infer identity from the components up to radius 25 Figure 2.15(m), noting that there are still some image artefacts on the right hand side of the image; when all components are added Figure 2.15(n) we return to the original image. This also illustrates *coding*, as the image can be encoded by retaining fewer of the components of the image than are in the complete transform. Figure 2.15 (m) is a good example of where an image of acceptable quality can be reconstructed, even when about half of the components are discarded. There are considerably better coding approaches than this, though we shall not consider coding in this text, and compression ratios can be considerably higher and still achieve acceptable quality. Note that it is common to use logarithms of magnitude to display Fourier transforms (Section 3.3.1) as otherwise the magnitude of the d.c. component can make the transform difficult to see.

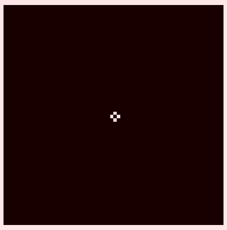
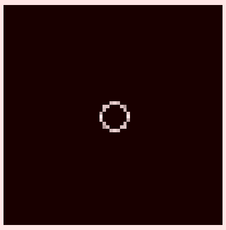
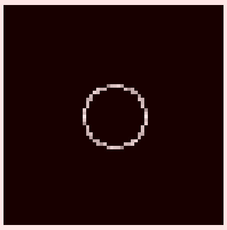
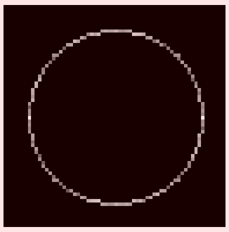
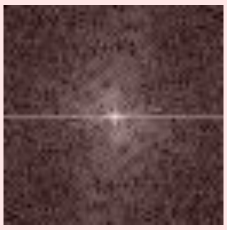
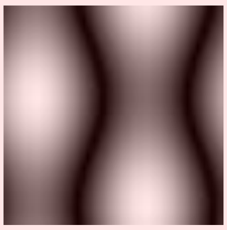
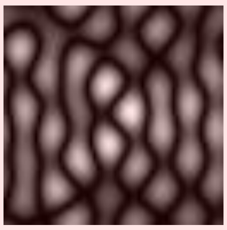
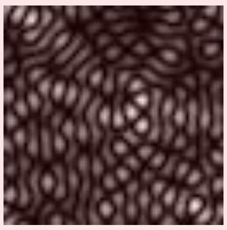
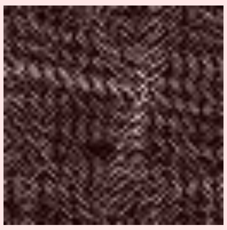
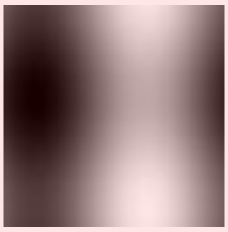
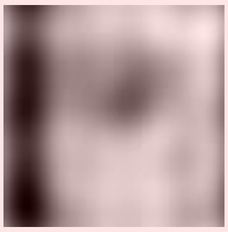
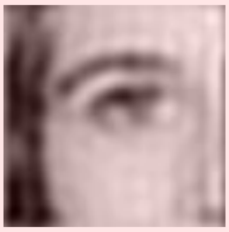
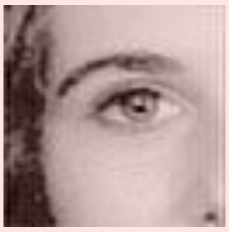
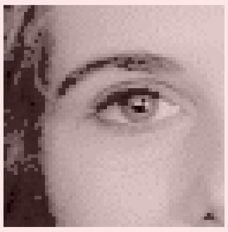
 **(a)** transform radius 1 components	 **(b)** transform radius 4 components	 **(c)** transform radius 9 components	 **(d)** transform radius 25 components	 **(e)** complete transform
 **(f)** image by radius 1 components	 **(g)** image by radius 4 components	 **(h)** image by radius 9 components	 **(i)** image by radius 25 components	
 **(j)** reconstruction up to 1st	 **(k)** reconstruction up to 4th	 **(l)** reconstruction up to 9th	 **(m)** reconstruction up to 25th	 **(n)** reconstruction with all

Figure 2.15 Image Reconstruction and Different Frequency Components

One of the important properties of the FT is *replication* which implies that the transform repeats in frequency up to infinity, as indicated in Figure 2.9 for 1D signals. To show this for 2D signals, we need to investigate the Fourier transform, originally given by $\mathbf{FP}_{u,v}$, at integer multiples of the number of sampled points $\mathbf{FP}_{u+mM,v+nN}$ (where m and n are integers). The Fourier transform $\mathbf{FP}_{u+mM,v+nN}$ is, by substitution in Equation 2.22:

$$\mathbf{FP}_{u+mN,v+nN} = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)((u+mN)x+(v+nN)y)} \tag{2.24}$$

so,

$$\mathbf{FP}_{u+mN,v+nN} = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)(ux+vy)} \times e^{-j2\pi(mx+ny)} \tag{2.25}$$

and since $e^{-j2\pi(mx+ny)} = 1$ (since the term in brackets is always an integer and then the exponent is always an integer multiple of 2π) then

$$\mathbf{FP}_{u+mN,v+nN} = \mathbf{FP}_{u,v} \tag{2.26}$$

which shows that the replication property does hold for the Fourier transform. However, Equation 2.22 and Equation 2.23 are very slow for large image sizes. They are usually implemented by using the *Fast Fourier Transform* (FFT) which is a splendid rearrangement of the Fourier transform's computation which improves speed dramatically. The FFT algorithm is beyond the scope of this text but is also a rewarding topic of study (particularly for computer scientists or software engineers). The FFT can only be applied to square images whose size is an integer power of 2 (without special arrangement). Calculation actually involves the *separability* property of the Fourier transform. Separability means that the Fourier transform is calculated in two stages: the rows are first transformed using a 1D FFT, then this data is transformed in columns, again using a 1D FFT. This process can be achieved since the sinusoidal *basis functions* are orthogonal. Analytically, this implies that the 2D DFT can be decomposed as in Equation 2.27

$$\mathbf{FP}_{u,v} = \frac{1}{MN}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}\mathbf{P}_{x,y}e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} = \frac{1}{MN}\sum_{x=0}^{N-1}\left\{\sum_{y=0}^{M-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)(vy)}\right\}e^{-j\left(\frac{2\pi}{M}\right)(ux)} \tag{2.27}$$

where M and N are the numbers of columns and rows, respectively. Equation 2.27 shows how separability is achieved, since the inner term expresses transformation along one axis (the y axis), and the outer term transforms this along the other (the x axis).

```
function [Fourier] = F_transform(image)

image=double(image);
[rows, cols] = size(image);
%we deploy equation 2.27, so that we can handle non square images
for u=1:cols % along the horizontal axis
    for v=1:rows % down the vertical axis
        sumx=0;
        for x=1:cols
            %first we transform the rows
            sumy=0;
            for y=1:rows %Eq 2.27 inner bracket
                sumy=sumy+image(y,x)*exp(-1j*2*pi*(v-1)*(y-1)/rows);
            end
            %then we do the columns Eq 2.27 outer
            sumx=sumx+sumy*exp(-1j*2*pi*(u-1)*(x-1)/cols);
        end %and finally normalise
        Fourier(v,u) = sumx/(rows*cols);
    end
end
```

Code 2.1 2D DFT, Exponential Form implementing Equation 2.27

Code 2.1 illustrates the implementation of Equation 2.27 in Matlab. The implementation simply evaluates the complex exponent in the definition. Since the computational cost of a 1D FFT of N points is O($N\log_2(N)$), the cost (by separability) for the 2D FFT is O($N^2\log_2(N)$) whereas the computational cost of the 2D DFT is O(N^3). This implies a considerable saving since it suggests that the FFT requires much less time, particularly for large image sizes (so for a 1024×1024 image, if the FFT takes seconds, the DFT will take minutes). The 2D FFT is available in Matlab using the `fft2` function which gives a result equivalent to Equations 2.22 or 2.27. You can note the difference in time between executing the code in Code 2.1 (or our Python version) and Matlab's own FFT operator (note there is some difference due to compiled vs interpreted code; do not run the basic version on a large image as it will take a very long time). The inverse 2D FFT, Equation 2.23, can be implemented using the Matlab `ifft2` function. (The difference between many Fourier transform implementations essentially concerns the chosen scaling factor, though the order of the frequency components differs from the basic equations in the Matlab functions.) The direct Matlab implementation of the 2D DFT in Equation 2.27 is given in Code 2.1. This is simply called using the command `b=F_Transform(a)` and the routine enforces the change from an integer format to double precision, as needed when using complex numbers in Matlab. It is easier to work in double precision throughout when developing code; integer formats can be used to speed real implementations (with caution: an early ARIANE space rocket blew up given erroneous conversion of a 32-bit integer to a 16-bit version).

In general, Equation 2.27 is difficult to compute since it requires evaluation of complex exponents. The result of a complex exponent is a complex number, thus the implementation requires representing functions, operations and variables for storage and processing using complex numbers. Although operations with complex numbers are well supported in mathematical packages, this equation obscures the true interpretation of the Fourier definition. Fortunately, by algebraic manipulation we can rewrite Equation 2.27 in a form that separates the imaginary and real parts. By recalling Euler's formula $e^{-jwt} = cos(wt) - jsin(wt)$, then Equation 2.27 becomes

$$\boldsymbol{FP}_{u,v} = \frac{1}{MN}\sum_{x=0}^{N-1}\sum_{y=\mathbf{0}}^{\boldsymbol{M-1}} \boldsymbol{P}_{x,y}\left(cos\left(\frac{2\pi}{M}vy\right) - jsin\left(\frac{2\pi}{M}vy\right)\right)\left(cos\left(\frac{2\pi}{N}ux\right) - jsin\left(\frac{2\pi}{N}ux\right)\right) \quad (2.28)$$

By grouping terms, we have that

$$\mathbf{FP}_{u,v} = \frac{1}{MN}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} \mathbf{P}_{x,y}\left(\left(\cos\left(\frac{2\pi}{N}ux\right)\cos\left(\frac{2\pi}{M}vy\right) - \sin\left(\frac{2\pi}{N}ux\right)\sin\left(\frac{2\pi}{M}vy\right)\right) - j\left(\cos\left(\frac{2\pi}{N}ux\right)\sin\left(\frac{2\pi}{M}vy\right) + \sin\left(\frac{2\pi}{N}ux\right)\cos\left(\frac{2\pi}{M}vy\right)\right)\right) \quad (2.29)$$

Equations 2.27 and 2.28 represent the same transform, but Equation 2.28 permits computation of the real and imaginary parts as trigonometric functions. More importantly, the trigonometric form shows how the values of u and v define the frequency used in the transform. This equation also shows the symmetry of the transform; since $\sin(-t) = -\sin(t)$ and $\cos(-t) = \cos(t)$, then the magnitude of the transform defined in Equation 2.7 is symmetrical at the origin.

```
for u in range(-maxFreqW, maxFreqW + 1):
    entryW = u + maxFreqW

    for v in range(-maxFreqH, maxFreqH + 1):
        entryH = v + maxFreqH
        coeff[entryH, entryW] = [0, 0]

        for x in range(0, width):
            sumY = [0, 0]

            for y in range(0, height):
                sumY[0] += inputImage[y,x] * cos(y * wh * v)
                sumY[1] += inputImage[y,x] * sin(y * wh * v)
            coeff[entryH, entryW][0] += sumY[0]*cos(x*ww*u)-sumY[1]*sin(x*ww*u)
            coeff[entryH, entryW][1] -= cos(x*ww*u)*sumY[1]+sin(x * ww*u)* sumY[0]
```

Code 2.2 2D DFT, Trigonometric Form

The implementation of the trigonometric form is shown in Code 2.2. To illustrate separability, the implementation computes the inner and outer summations in Equation 2.28, though the same result would be obtained if the real and imaginary parts were computed by using Equation 2.29. In any case, the implementation should perform the sum for the real and imaginary parts by evaluating sines and cosines at different frequencies. In the code, the array `sumY[0]` and `coeff[u,v][0]` store the real values and `sumY[1]` and `coeff[u,v][1]` the imaginary values. Finally the coefficients need to be scaled by $\frac{1}{MN}$ according to Equation 2.28.

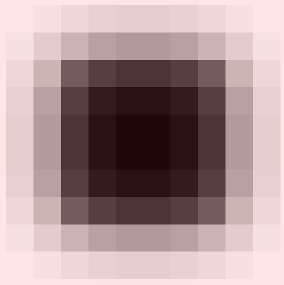
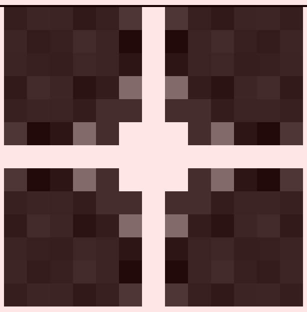
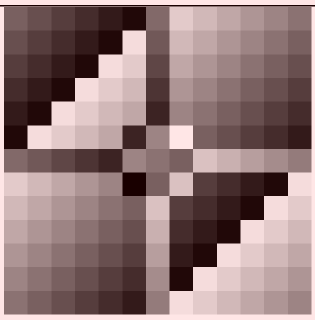
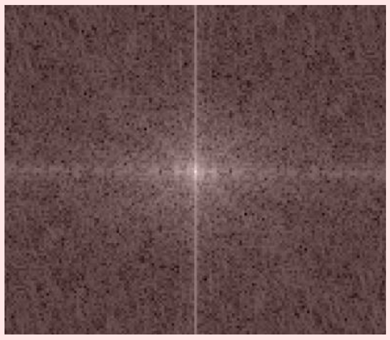
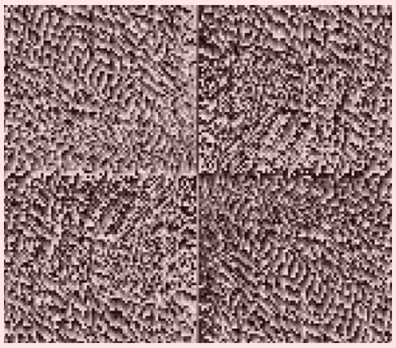
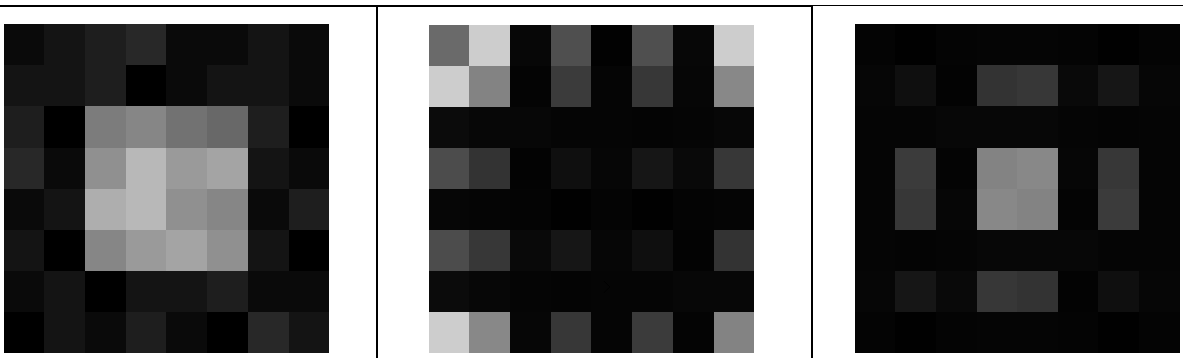
 **(a)** image of square	 **(b)** magnitude	 **(c)** phase
(d) image	 **(e)** magnitude	 **(f)** phase

Figure 2.16 2D DFT Examples

Figure 2.16 shows two examples of the DFT obtained with Code 2.2. The results of the Fourier expansion are not presented by showing the imaginary and real values of $\mathbf{FP}_{u,v}$, but the complex values are shown as magnitude and phase (Equations 2.7 and 2.8). Note the symmetries in the magnitude and phase that result from the symmetries of the cosine and sine functions.

Equation 2.29 reveals the nature of the Fourier transform as a description of an image using frequency; each value $\mathbf{FP}_{u,v}$ defines sine and cosine waves at a given frequency. The values of u and v produce waves with frequency that increases as these values increase. As such, the position of each component reflects its frequency: low frequency components are near the origin and high frequency components are further away. Notice also that frequencies can be positive and negative and the lowest frequency component, for zero frequency – the d.c. component – represents the average value of the samples. In Code 2.2, we used this interpretation where u and v represent frequencies. As such, the loops in the implementation iterate over a frequency range. These ranges can be arbitrarily selected to obtain only the fine detail by just computing high frequencies or to obtain coarse features by choosing low frequencies. As we discussed in Section 2.4, the maximum frequency we can use is given by the half of image size (or the result will contain aliasing). In the implementation, the result of the summation for each pair u, v is stored in a cell in the output array `coeff`. The index to the cell is stored in the variables `entryW` and `entryH`. These values are set such that the zero frequency is stored in the centre of the output array and the negative frequencies are to the left of and down from the centre position. This way to select the output cells is just a standard way to organise and visualise the frequencies. It is important to notice that the loops can be changed to iterate over the output array (since Equations 2.29 and 2.27 are the same transform), but iterating in frequency gives a clear meaning to the frequency content in the output.

Code 2.1 shows the origin of the transform (low frequency components) at the corners of the transform. The image of the square in 2.17(a) produces the result for 2.17(b). A spatial transform is easier to visualize if the d.c. (zero frequency) component is in the centre, with frequency increasing towards the edge of the image. This can be arranged either by rotating each of the four quadrants in the Fourier transform by 180°. An alternative is to *reorder* the original image to give a transform which has been shifted to the centre. Both operations result in the image in 2.17(c) wherein the transform is much more easily seen. Note that this is aimed to improve visualization and does not change any of the frequency domain information, only the standard way it is displayed.



(a) image of square **(b)** original DFT **(c)** rearranged DFT

Figure 2.17 Rearranging the 2D DFT for Display Purposes.

To rearrange the image so that the d.c. component is in the centre, the frequency components need to be reordered. This can be achieved simply by multiplying each image point $\mathbf{P}_{x,y}$ by $-1^{(x+y)}$. Since $\cos(-\pi) = -1$, then $-1 = e^{-j\pi}$ (the minus sign is introduced just to keep the analysis neat) so we obtain the transform of the multiplied image as:

$$\begin{aligned}\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)(ux+vy)} \times -1^{(x+y)} &= \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)(ux+vy)} \times e^{-j\pi(x+y)} \\ &= \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)\left(\left(u+\frac{N}{2}\right)x+\left(v+\frac{N}{2}\right)y\right)} \\ &= \mathbf{FP}_{u+\frac{N}{2},v+\frac{N}{2}}\end{aligned} \tag{2.30}$$

According to Equation 2.30, when pixel values are multiplied by $-1^{(x+y)}$, the Fourier transform becomes shifted along each axis by half the number of samples. According to the replication theorem, Equation 2.26, the transform replicates along the frequency axes. This implies that the centre of a transform image will now be the d.c. component. (Another way of interpreting this is that rather than look at the frequencies centred on where the image is, our viewpoint has been shifted so as to be centred on one of its corners - thus invoking the replication property.) This brings equivalence between the trigonometric form (for the magnitude see Figures 2.16(b) and (d)) and the exponential form (see Figure 2.17(b)). The operator `Rearrange`, in Code 2.3, is used prior to transform calculation and leads to the image of Figure 2.17(c), and all later transform images.

```
function rearranged = rearrange(image)
%get dimensions
[rows,cols]=size(image);

%rearrange image
for x = 1:cols %address all columns
  for y = 1:rows %address all rows
    rearranged(y,x)=image(y,x)*((-1)^(y+x)); %Eq. 2.30
  end
```

```
end
```

Code 2.3 Reordering for Transform Calculation

The full effect of the Fourier transform is shown by application to an image of much higher resolution. Figure 2.18(a) shows the image of a group of people and Figure 2.18(b) shows its transform. The transform reveals that much of the information is carried in the lower frequencies since this is where most of the spectral components concentrate. This is because the image has many regions where the brightness does not change a lot, such as in the foliage. The high frequency components reflect change in intensity. Accordingly, the higher frequency components arise from things that change fast and from the borders of objects.

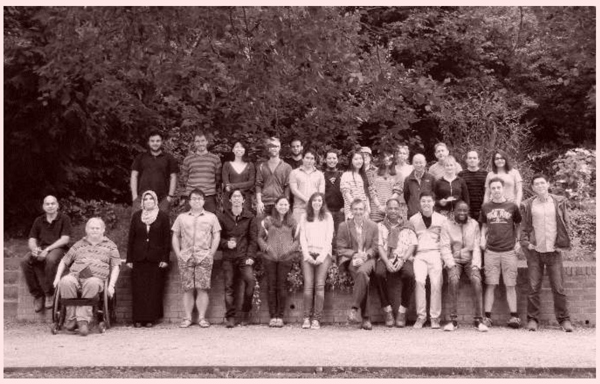
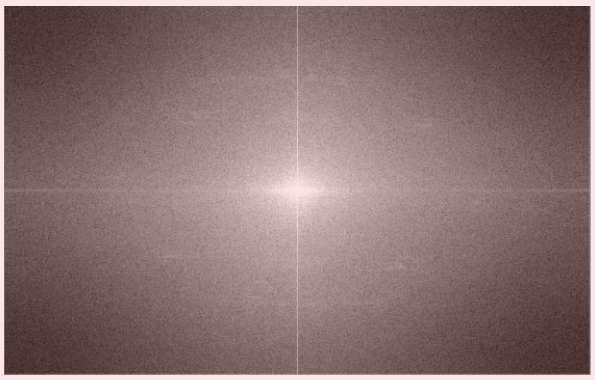
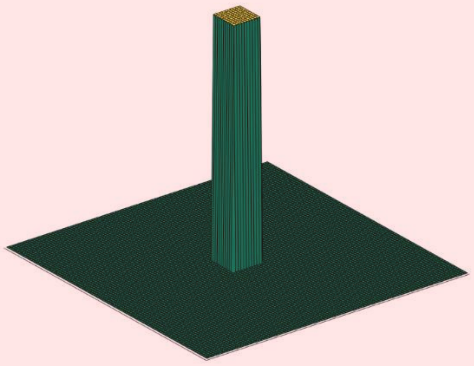
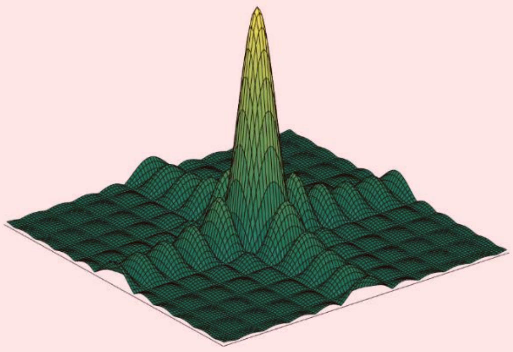
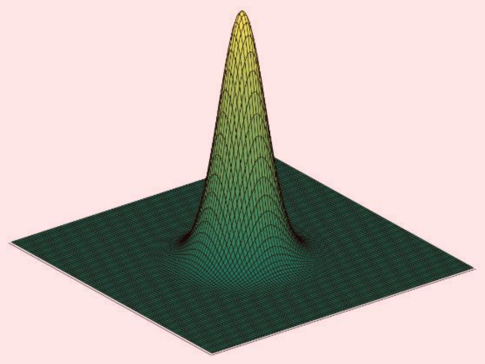
 **(a)** image of group	 **(b)** transform of image of group

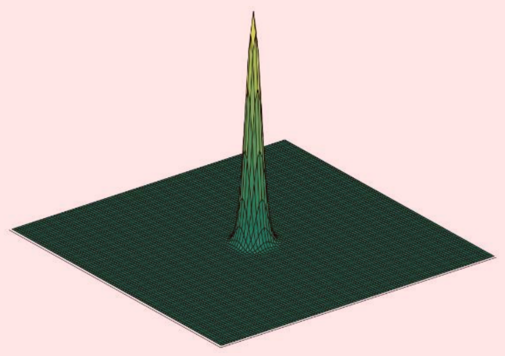
Figure 2.18 Applying the Fourier Transform to the Image of a Group of People

As with the 1D Fourier transform, there are 2D Fourier transform pairs, illustrated in Figure 2.19. The 2D Fourier transform of a two dimensional pulse, Figure 2.19(a), is a two dimensional sinc function, in Figure 2.19(b). The 2D Fourier transform of a Gaussian function, in Figure 2.19(c), is again a two dimensional Gaussian function in the frequency domain, in Figure 2.19(d).

Image Domain	**Transform Domain**
 **(a)** square	 **(b)** 2D sinc function



(c) Gaussian



(d) Fourier transform of Gaussian

Figure 2.19 2D Fourier Transform Pairs

2.6 Properties of the Fourier Transform

2.6.1 Shift Invariance

The decomposition into spatial frequency does not depend on the position of features within the image. If we shift all the features by a fixed amount, or acquire the image from a different position, the magnitude of its Fourier transform does not change. This property is known as *shift invariance*. By denoting the delayed version of $p(t)$ as $p(t-\tau)$, where τ is the delay, and the Fourier transform of the shifted version as $\Im\left[p(t-\tau)\right]$, we obtain the relationship between a time domain shift in the time and frequency domains as:

$$\Im\left[p(t-\tau)\right] = e^{-j\omega\tau}P(\omega) \tag{2.31}$$

Accordingly, the magnitude of the Fourier transform is:

$$\left|\Im\left[p(t-\tau)\right]\right| = \left|e^{-j\omega\tau}P(\omega)\right| = \left|e^{-j\omega\tau}\right|\left|P(\omega)\right| = \left|P(\omega)\right| \tag{2.32}$$

and since the magnitude of the exponential function is 1.0 then the magnitude of the Fourier transform of the shifted image equals that of the original (unshifted) version. We shall use this property later in Chapter 7 when we use Fourier theory to describe shapes. There, it will allow us to give the same description to different instances of the same shape, but a different description to a different shape. You do not get something for nothing: even though the magnitude of the Fourier transform remains constant, its phase does not. The phase of the shifted transform is:

$$\arg\left(\Im\left[p(t-\tau)\right]\right) = \arg\left(e^{-j\omega\tau}P(\omega)\right) \tag{2.33}$$

The implementation of a `shift` operator, Code 2.4, uses the modulus operation `%` to enforce the cyclic shift. The `shiftDistance` is a parameter that defines the length of the horizontal shift along the *x* axis.

```
for x,y in itertools.product(range(0, width), range(0, height)):
    xShift = (x - shiftDistance) % width
    shiftImage[y,x] = inputImage[y,xShift]
```

Code 2.4 Shifting an Image

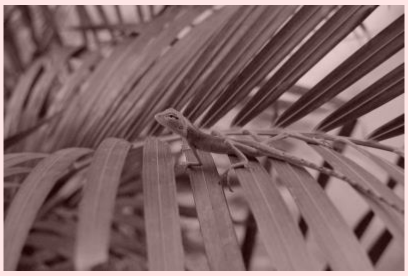
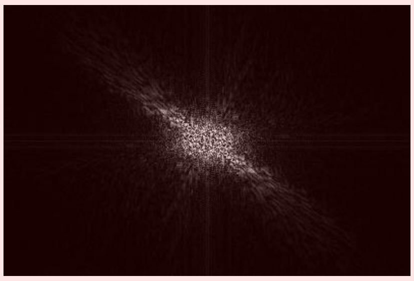
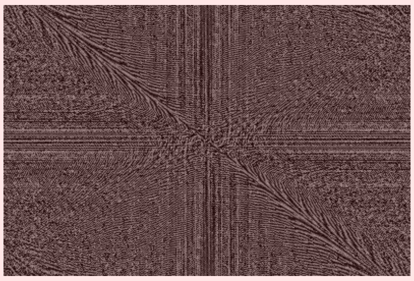
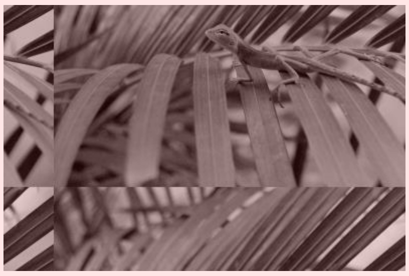
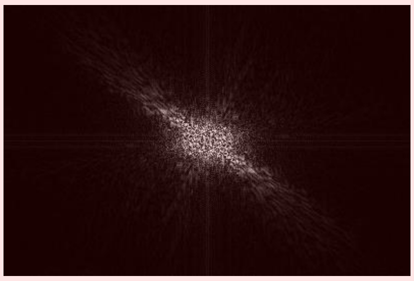
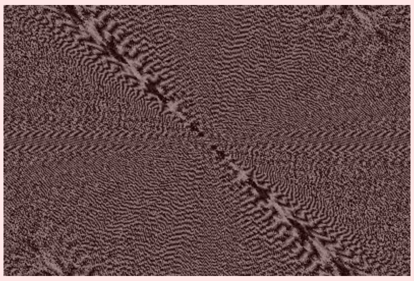
		
(a) original image	**(b)** magnitude of Fourier transform of original image	**(c)** phase of Fourier transform of original image
		
(d) shifted image	**(e)** magnitude of Fourier transform of shifted image	**(f)** phase of Fourier transform of shifted image

Figure 2.20 Illustrating Shift Invariance

This process is illustrated in Figure 2.20. An original image, Figure 2.20(a), is shifted along the *x* and the *y* axes, Figure 2.20(d). The shift is cyclical, so parts of the image wrap around; those parts at the top of the original image appear at the base of the shifted image. The Fourier transform of the original image and of the shifted image are identical: Figure 2.20(b) appears the same as Figure 2.20(e). The phase differs: the phase of the original image Figure 2.20(c) is clearly different from the phase of the shifted image, Figure 2.20(f).

The differing phase implies that, in application, the magnitude of the Fourier transform of a face, say, will be the same irrespective of the position of the face in the image (i.e. the camera or the subject can move up and down), assuming that the face is much larger than its image version. This implies that if the magnitude of the Fourier transform is used to analyse an image of a human face or one of cloth, to describe it by its spatial frequency, we do not need to control the position of the camera, or the object, precisely.

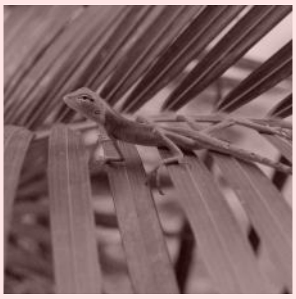
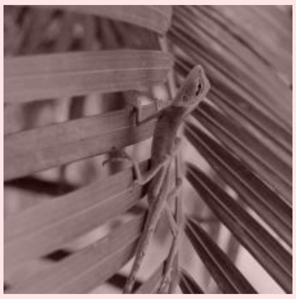
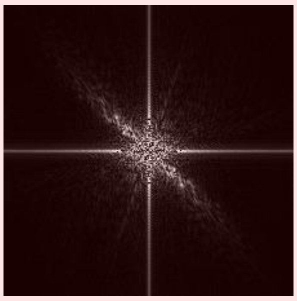
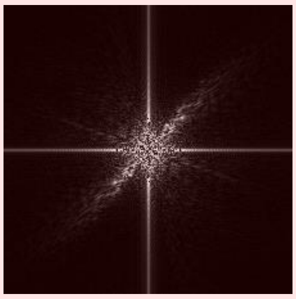
			
(a) original image	**(b)** rotated image	**(c)** transform of original image	**(d)** transform of rotated image

Figure 2.21 Illustrating Rotation

2.6.2 Rotation

The Fourier transform of an image *rotates* when the source image rotates. This is to be expected since the decomposition into spatial frequency reflects the orientation of features within the image. As such, orientation dependency is built into the Fourier transform process.

This implies that if the frequency domain properties are to be used in image analysis, via the Fourier transform, the orientation of the original image needs to be known, or fixed. It is often possible to fix orientation, or to estimate its value when a feature's orientation cannot be fixed. Alternatively, there are techniques to impose invariance to rotation, say by translation to a polar representation, though this can prove to be complex.

The effect of rotation is illustrated in Figure 2.21. An image, Figure 2.21(a), is rotated by 90º to give the image in Figure 2.21(b). Comparison of the transform of the original image, Figure 2.21(c), with the transform of the rotated image, Figure 2.21(d) shows that the transform has been rotated by 90º, by the same amount as the image. In fact, close inspection of Figures 2.21(c) and (d) shows that the diagonal axis is consistent with the normal to the axis of the leaves (where the change mainly occurs), and this is the axis that rotates.

2.6.3 Frequency Scaling

By definition, time is the reciprocal of frequency. So if an image is compressed, equivalent to reducing time, its frequency components will spread, corresponding to increasing frequency. Mathematically the relationship is that the Fourier transform of a function of time multiplied by a scalar λ, $p(\lambda t)$, gives a frequency domain function $P(\omega/\lambda)$, so:

$$\Im\left[p(\lambda t)\right] = \frac{1}{\lambda} P\left(\frac{\omega}{\lambda}\right) \tag{2.34}$$

This is illustrated in Figure 2.22 where the image of spots (a 3D calibration target), Figure 2.22(a), is reduced in scale, Figure 2.22(b), thereby increasing the spatial frequency. The DFT of the original image is shown in Figure 2.22(c) which reveals that the large spatial frequencies in the original image are arranged in a star-like pattern. As a consequence of scaling the original image, the spectrum will spread from the origin consistent with an increase in spatial frequency, as shown in Figure 2.22(d). This retains the star-like pattern, but with points at a greater distance from the origin.

The implications of this property are that if we reduce the scale of an image, say by imaging at a greater distance, we will alter the frequency components. The relationship is linear: the amount of reduction, say the proximity of the camera to the target, is directly proportional to the scaling in the frequency domain.

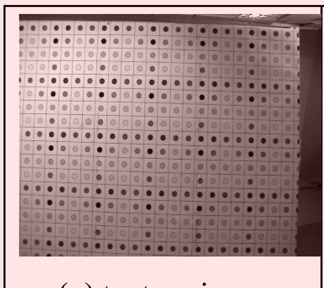
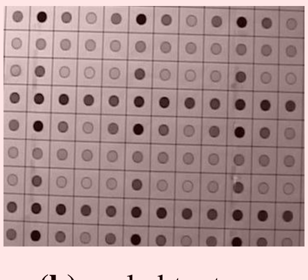
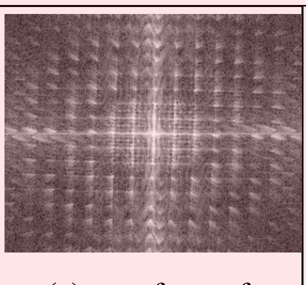
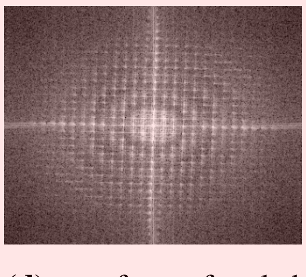
			
(a) texture image	**(b)** scaled texture image	**(c)** transform of original texture	**(d)** transform of scaled texture

Figure 2.22 Illustrating Frequency Scaling

2.6.4 Superposition (Linearity)

The *principle of superposition* is very important in systems analysis. Essentially, it states that a system is linear if its response to two combined signals equals the sum of the responses to the individual signals. Given an output O which is a function of two inputs I_1 and I_2, the response to signal I_1 is $O(I_1)$, that to signal I_2 is $O(I_2)$, and the response to I_1 and I_2, when applied together, is $O(I_1 + I_2)$, the superposition principle states:

$$O(I_1 + I_2) = O(I_1) + O(I_2) \tag{2.35}$$

Any system which satisfies the principle of superposition is termed *linear*. The Fourier transform is a linear operation since, for two signals p_1 and p_2:

$$\Im[p_1 + p_2] = \Im[p_1] + \Im[p_2] \tag{2.36}$$

In application this suggests that we can separate images by looking at their frequency domain components. This is illustrated for one-dimensional signals in Figure 2.23. One signal is shown in Figure 2.23(a) and a second is shown in Figure 2. 23(c). The Fourier transforms of these signals are shown in Figures 2.23(b) and (d). The addition of these signals is shown in Figure 2.23(e) and its transform in Figure 2.23(f). The Fourier transform of the added signals differs little from the addition of their transforms, Figure 2.23(g). This is confirmed by subtraction of the two, Figure 2.23(h) (some slight differences can be seen, but these are due to numerical error).

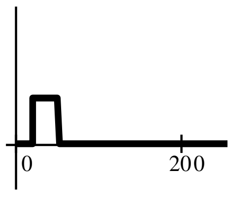
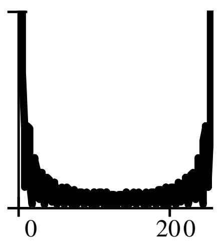
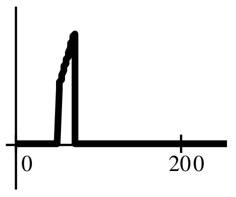
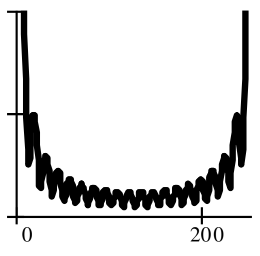
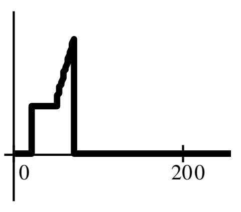
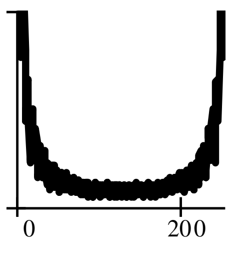
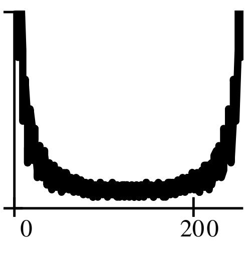
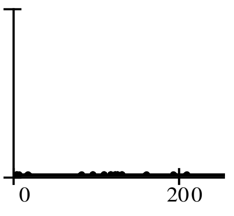
 **(a)** signal 1	 **(b)** $\Im$ (signal 1)	 **(c)** signal 2	 **(d)** $\Im$ (signal 2)
 **(e)** signal 1 + signal 2	 **(f)** $\Im$ (signal 1 + signal 2)	 **(g)** $\Im$ (signal 1) + $\Im$ (signal 2)	 **(h)** difference: (f)-(g)

Figure 2.23 Illustrating Superposition

By way of example, given the image of a fingerprint in blood on cloth it is very difficult to separate the fingerprint from the cloth by analysing the combined image. However, by translation to the frequency domain, the Fourier transform of the combined image shows strong components due to