2 Images, Sampling and Frequency Domain Processing

2.1 Overview

In this chapter, we shall look at the basic theory which underlies image formation and processing. We shall start by investigating what makes up a picture and look at the consequences of having a different number of points in the image. We shall also look at images in a different representation, known as the frequency domain. In this, as the name implies, we consider an image as a collection of frequency components. We can actually operate on images in the frequency domain and we shall also consider different transformation processes. These allow us different insights into images and image processing which will be used in later chapters not only as a means to develop techniques, but also to give faster (computer) processing.

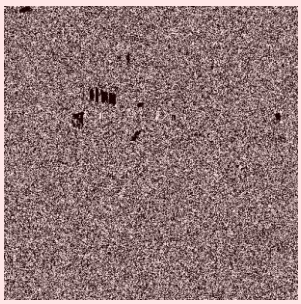
Main topic	Sub topics	Main points
Images	Effects of differing numbers of points and of number range for those points.	*Grayscale*, *colour*, *resolution*, dynamic range, storage.
Fourier transform theory	What is meant by the frequency domain, how it applies to discrete (sampled) images, how it allows us to interpret images and the sampling resolution (number of points).	*Continuous Fourier transform* and properties, sampling criterion, *discrete Fourier transform* and properties, image transformation, transform duals. *Inverse Fourier transform*. Importance of *magnitude* and *phase*.
Consequences of transform approach	Basic properties of Fourier transforms, other transforms, frequency domain operations.	*Translation* (shift), *rotation* and *scaling*. *Principle of Superposition* and linearity. *Walsh*, *Hartley*, *Discrete Cosine* and *Wavelet transforms*. *Filtering* and other operations.

Table 2.1 Overview of Chapter 2

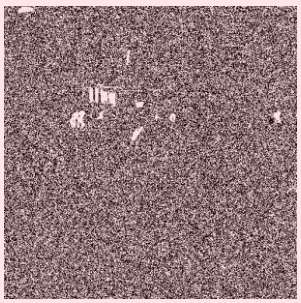
2.2 Image Formation

A computer image is a matrix (a two dimensional array) of *pixels***.** The value of each pixel is proportional to the brightness of the corresponding point in the scene; its value is, of course, usually derived from the output of an A/D converter. We can define a square image as $N \times N$ m-bit pixels where N is the number of points and m controls the number of brightness values. Using m bits gives a range of 2^m values, ranging from 0 to 2^m-1. If m is 8 this gives brightness levels ranging between 0 and 255, which are usually displayed as black and white, respectively, with shades of grey in-between, as they are for the *grayscale image* of a scene in Figure 2.1(a). Smaller values of m give fewer available levels reducing the available contrast in an image. We are concerned with images here, not their formation; imaging geometry (pinhole cameras et al) is to be found in Chapter 10.

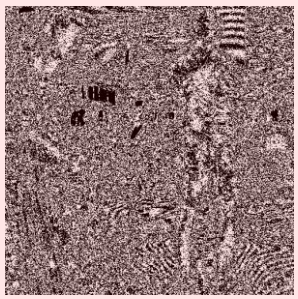
(a) original image



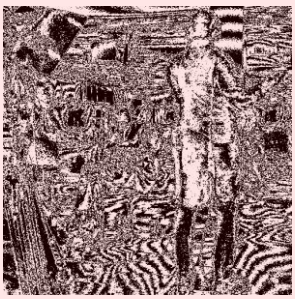
(b) bit 0 (LSB)



(c) bit 1

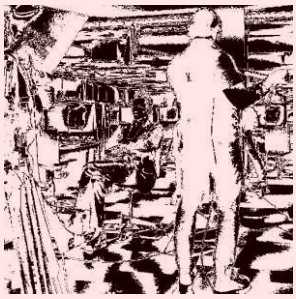


(d) bit 2

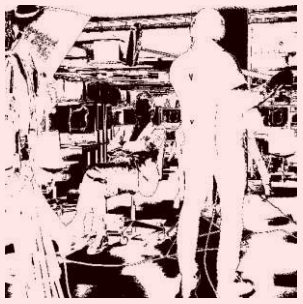


(e) bit 3

(f) bit 4



(g) bit 5



(h) bit 6

(i) bit 7 (MSB)

Figure 2.1 Decomposing an Image Into its Bits

The ideal value of *m* is actually related to the *signal to noise ratio* (*dynamic range*) of the camera. This is stated as approximately 45 dB for an analogue camera and since there are 6 dB per bit, then 8 bits will cover the available range. Choosing 8-bit pixels has further advantages in that it is very convenient to store pixel values as bytes, and 8-bit A/D converters are cheaper than those with a higher resolution. For these reasons images are nearly always stored as 8-bit bytes, though some applications use a different range. These are the 8-bit numbers encountered in Section 1.5.2, stored as unsigned 8-bit bytes (`uint8`). The relative influence of the eight bits is shown in the image of the subjects in Figure 2.1. Here, the least significant bit, bit 0 (Figure 2.1(b)), carries the least information (it changes most rapidly) and is largely noise. As the order of the bits increases, they change less rapidly and carry more information. The most information is carried by the most significant bit, bit 7 (Figure 2.1(i)). Clearly, the fact that there are people in the original image can be recognised much better from the high order bits, much more reliably than it can from the other bits (also notice the odd effects which would appear to come from lighting in the middle order bits). The variation in lighting is hardly perceived by human vision.

Colour images (also mentioned in Section 1.5.2) follow a similar storage strategy to specify pixels' intensities. However, instead of using just one image plane, colour images are represented by three intensity components. These components generally correspond to red, green, and blue (the RGB model) although there are other colour schemes. For example, the CMYK colour model is defined by the components cyan, magenta, yellow, and black. In any colour mode, the pixel's colour can be specified in two main ways. First, you can associate an integer value, with each pixel, that can be used as an index to a table that stores the intensity of each colour component. The index is used to recover the actual colour from the table when the pixel is going to be displayed, or processed. In this scheme, the table is known as the image's palette and the display is said to be performed by colour mapping. The main reason for using this colour representation is to reduce memory requirements. That is, we only store a single image plane (i.e., the indices) and the palette. This is less than storing the red, green and blue components separately and so makes the hardware cheaper and it can have other advantages, for example when the image is transmitted. The main disadvantage is that the quality of the image is reduced since only a reduced collection of colours is actually used. An alternative to represent colour is to use several image planes to store the colour components of each pixel. This scheme is known as true colour and it represents an image more accurately, essentially by considering more colours. The most common format uses eight bits for each of the three RGB components. These images are known as 24-bit true colour and they can contain 16777216 different colours simultaneously. In spite of requiring significantly more memory, the image quality and the continuing reduction in cost of computer memory make this format a good alternative, even for storing the image frames from a video sequence. Of course, a good compression algorithm is always helpful in these cases, particularly, if images need to be transmitted on a network. Here we will consider the processing of grey level images only since they contain enough information to perform feature extraction and image analysis; greater depth on colour analysis/parameterisation is to be found in Chapter 11. Should the image be originally in colour, we will consider processing its luminance only, often computed in a standard way. In any case, the amount of memory used is always related to the image size.

(a) 64×64 **(b)** 128×128 **(c)** 256×256

Figure 2.2 Effects of Differing Image Resolution

Choosing an appropriate value for the image size, N, is far more complicated. We want N to be sufficiently large to resolve the required level of spatial detail in the image. If N is too small, the image will be coarsely *quantised*: lines will appear to be very 'blocky' and some of the detail will be lost. Larger values of N give more detail, but need more storage space and the images will take longer to process, since there are more pixels. For example, with reference to the image in Figure 2.1(a), Figure 2.2 shows the effect of taking the image at different resolutions. Figure 2.2(a) is a 64×64 image, that shows only the broad structure. It is impossible to see any detail in the sitting

subject's face, or anywhere else. Figure 2.2(b) is a 128×128 image, which is starting to show more of the detail, but it would be hard to determine the subject's identity. The original image, repeated in Figure 2.2(c), is a 256×256 image which shows a much greater level of detail, and the subject can be recognised from the image. Note that the images in Figure 2.2 have been scaled to be the same size. As such, the pixels in Figure 2.2(a) are much larger than in Figure 2.2(c) which emphasises its blocky structure. Common choices are for 512×512 or 1024×1024 8-bit images which require 256 KB and 1 MB of storage, respectively. If we take a sequence of, say, 20 images for motion analysis, we will need more than 5 MB to store twenty 512×512 images. Even though memory continues to become cheaper, this can still impose high cost. But it is not just cost which motivates an investigation of the appropriate image size, the appropriate value for *N*. The main question is: are there theoretical guidelines for choosing it? The short answer is 'yes'; the long answer is to look at digital signal processing theory.

The choice of sampling frequency is dictated by the *sampling criterion*. Presenting the sampling criterion requires understanding of how we interpret signals in the *frequency domain*. The way in is to look at the Fourier transform. This is a highly theoretical topic, but do not let that put you off (it leads to image coding, like the JPEG format, so it is very useful indeed). The Fourier transform has found many uses in image processing and understanding; it might appear to be a complex topic (that's actually a horrible pun!) but it is a very rewarding one to study. The particular concern is number of points per unit area or the appropriate sampling frequency of (essentially, the value for *N*), or the rate at which pixel values are taken from, a camera's video signal.

2.3 The Fourier Transform

The *Fourier transform* is a way of mapping a signal into its component frequencies. *Frequency* measures in Hertz (Hz) the rate of repetition with time, measured in seconds (s); time is the reciprocal of frequency and vice versa (Hertz = 1/seconds; s = 1/Hz).

Consider a music centre: the sound comes from a computer, a CD player (or a tape, whatever) and is played on the speakers after it has been processed by the amplifier. On the amplifier, you can change the bass or the treble (or the loudness which is a combination of bass and treble). Bass covers the low frequency components and treble covers the high frequency ones. The Fourier transform is a way of mapping the signal, which is a signal varying continuously with time, into its frequency components. When we have transformed the signal, we know which frequencies made up the original sound.

So why do we do this? We have not changed the signal, only its representation. We can now visualise it in terms of its frequencies, rather than as a voltage which changes with time. However, we can now change the frequencies (because we can see them clearly) and this will change the sound. If, say, there is hiss on the original signal then since hiss is a high frequency component, it will show up as a high frequency component in the Fourier transform. So we can see how to remove it by looking at the Fourier transform. If you have ever used a graphic equaliser, you have done this before. The graphic equaliser is a way of changing a signal by interpreting its frequency domain representation; you can selectively control the frequency content by changing the positions of the controls of the graphic equaliser. The equation which defines the *Fourier transform*, *Fp*, of a signal *p*, is given by a complex integral:

$$Fp(\omega) = \Im\left(p(t)\right) = \int_{-\infty}^{\infty} p(t) e^{-j\omega t} dt \tag{2.1}$$

where: $Fp(\omega)$ is the Fourier transform, and $\Im$ denotes the Fourier transform process;

ω is the angular frequency, $\omega = 2\pi f$ measured in radians/s (where the frequency f is the reciprocal of time t, $f = 1/t$);

j is the complex variable $j = \sqrt{-1}$ (electronic engineers prefer j to i since they cannot confuse it with the symbol for current; perhaps they don't want to be mistaken for mathematicians who use $i = \sqrt{-1}$)

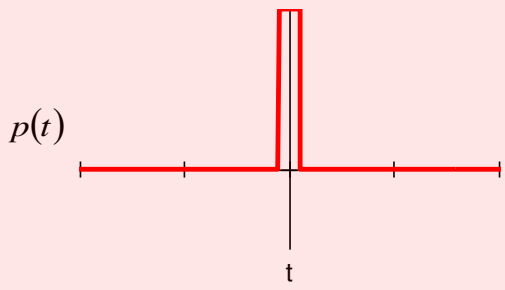
$p(t)$ is a *continuous signal* (varying continuously with time); and

$e^{-j\omega t} = \cos(\omega t) - j\sin(\omega t)$ gives the frequency components in $p(t)$.

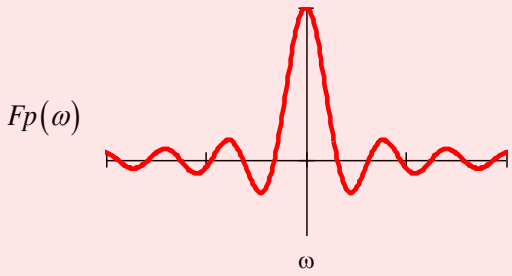
We can derive the Fourier transform by applying Equation 2.1 to the signal of interest. We can see how it works by constraining our analysis to simple signals. (We can then say that complicated signals are just made up by adding up lots of simple signals.) If we take a pulse which is of amplitude (size) A between when it starts at time $t = -T/2$ and it ends at $t = T/2$, and is zero elsewhere, the *pulse* is:

$$p(t) = \begin{cases} A & \text{if} \quad -T/2 \le t \le T/2 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

To obtain the Fourier transform, we substitute for $p(t)$ in Equation 2.1. $p(t) = A$ only for a specified time so we choose the limits on the integral to be the start and end points of our pulse (it is zero elsewhere) and set $p(t) = A$, its value in this time interval. The Fourier transform of this pulse is the result of computing:



(a) pulse of amplitude $A = 1$



(b) Fourier transform

Figure 2.3 A Pulse and its Fourier Transform

$$Fp(\omega) = \int_{-T/2}^{T/2} Ae^{-j\omega t} dt \tag{2.3}$$

When we solve this we obtain an expression for $Fp(\omega)$:

$$Fp(\omega) = -\frac{Ae^{-j\omega T/2} - Ae^{j\omega T/2}}{j\omega} \tag{2.4}$$

By simplification, using the relation $\sin(\theta) = (e^{j\theta} - e^{-j\theta})/2j$, then the Fourier transform of the pulse is:

$$Fp(\omega) = \begin{cases} \dfrac{2A}{\omega}\sin\left(\dfrac{\omega T}{2}\right) & \text{if} \quad \omega \neq 0 \\ AT & \text{if} \quad \omega = 0 \end{cases} \tag{2.5}$$

This is a version of the *sinc* function, $\text{sinc}(x) = \sin(x)/x$. The original pulse, and its transform are illustrated in Figure 2.3. Equation 2.5 (as plotted in Figure 2.3(b)) suggests that a pulse is made up of a lot of low frequencies (the main body of the pulse) and a few higher frequencies (which give us the edges of the pulse). (The range of frequencies is symmetrical around zero frequency; negative

frequency is a necessary mathematical abstraction.) The plot of the Fourier transform is actually called the *spectrum* of the signal, which can be considered akin with the spectrum of light.

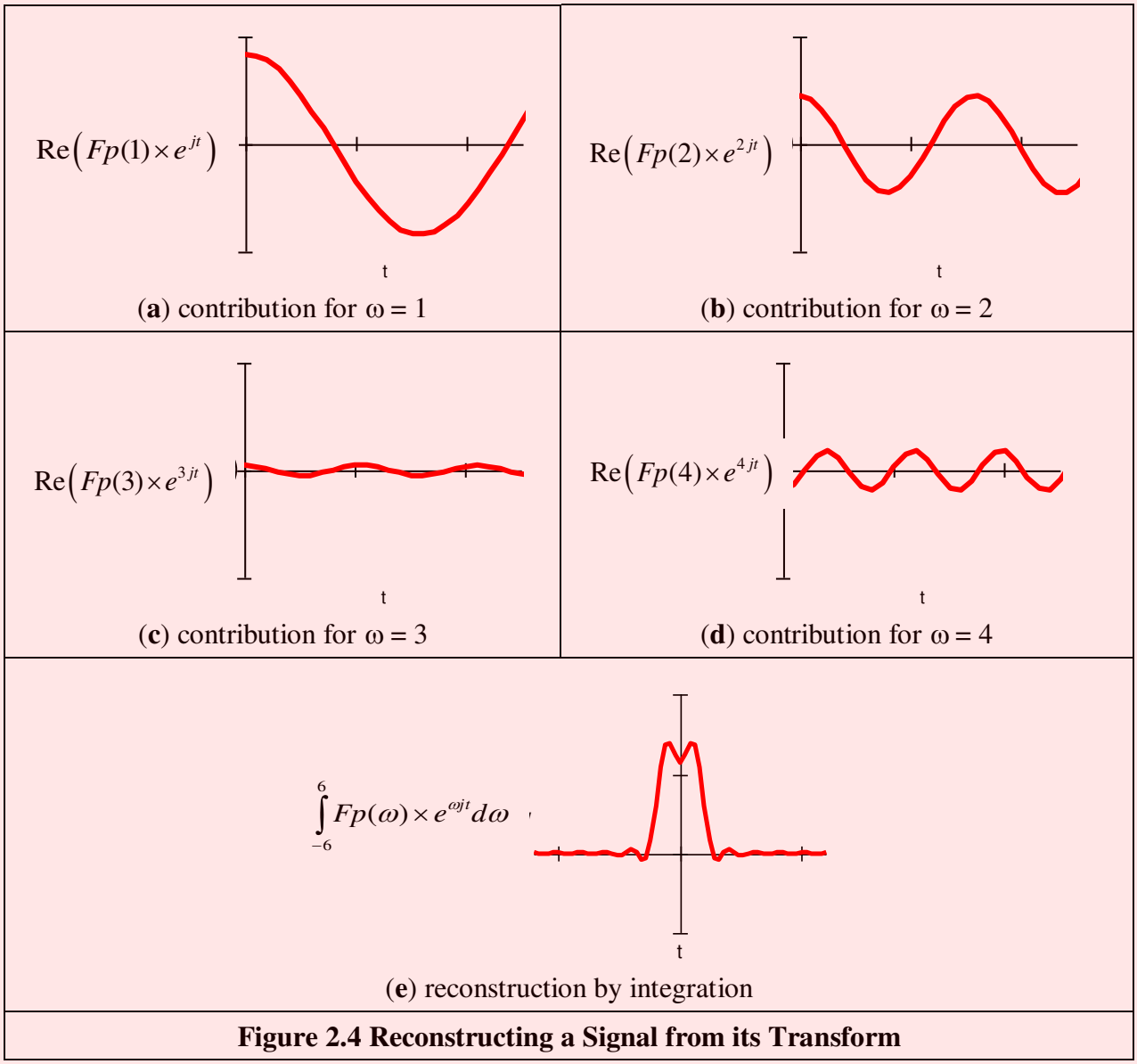


Figure 2.4 Reconstructing a Signal from its Transform

So what actually is this Fourier transform? It tells us what frequencies make up a time domain signal. The magnitude of the transform at a particular frequency is the amount of that frequency in the original signal. If we collect together sinusoidal signals in amounts specified by the Fourier transform, we should obtain the originally transformed signal. This process is illustrated in Figure 2.4 for the signal and transform illustrated in Figure 2.3. Note that since the Fourier transform is actually a complex number it has real and imaginary parts, and we only plot the real part here. A low frequency, that for $\omega = 1$, in Figure 2.4(a) contributes a large component of the original signal; a higher frequency, that for $\omega = 2$, contributes less as in Figure 2.4(b). This is because the transform coefficient is less for $\omega = 2$ than it is for $\omega = 1$. There is a very small contribution for $\omega = 3$, Figure 2.4(c), though there is more for $\omega = 4$, Figure 2.4(d). This is because there are frequencies for which there is no contribution, where the transform is zero. When these signals are integrated together, we achieve a signal that looks similar to our original pulse, Figure 2.4(e). Here we have only considered frequencies from $\omega = -6$ to $\omega = 6$. If the frequency range in integration was larger, more high frequencies would be included, leading to a more faithful reconstruction of the original

pulse.

The result of the Fourier transform is actually a complex number: each value $Fp(w)$ is represented by a pair of numbers that represent its real and imaginary parts. Rather than use the real and imaginary parts, the values are usually represented in terms of *magnitude* (or size, or modulus) and *phase* (or argument). The transform can be represented as:

$$Fp(\omega) = \int_{-\infty}^{\infty} p(t)e^{-j\omega t}dt = \mathrm{Re}\left(Fp(\omega)\right) + j\,\mathrm{Im}\left(Fp(\omega)\right) \tag{2.6}$$

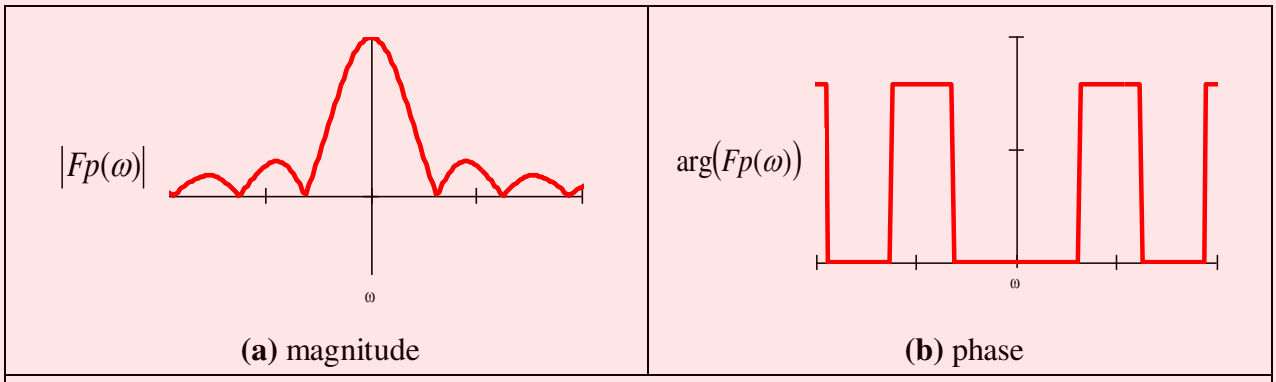
where $\mathrm{Re}(\ )$ and $\mathrm{Im}(\ )$ are the real and imaginary parts of the transform, respectively. The magnitude of the transform is then:

$$\left|Fp(\omega)\right| = \sqrt{\mathrm{Re}\left(Fp(\omega)\right)^2 + \mathrm{Im}\left(Fp(\omega)\right)^2} \tag{2.7}$$

and the phase is:

$$\arg\left(Fp(\omega)\right) = \tan^{-1}\left(\frac{\mathrm{Im}\left(Fp(\omega)\right)}{\mathrm{Re}\left(Fp(\omega)\right)}\right) \tag{2.8}$$

where the signs of the real and the imaginary components can be used to determine which quadrant the phase (argument) is in, since the phase can vary from 0 to 2π radians. The magnitude describes the amount of each frequency component, the phase describes timing, when the frequency components occur. The magnitude and phase of the transform of a pulse are shown in Figure 2.5 where the magnitude returns a positive transform, and the phase is either 0 or 2π radians (consistent with the sine function).



(a) magnitude **(b)** phase

Figure 2.5 Magnitude and Phase of Fourier Transform of Pulse

In order to return to the time-domain signal, from the frequency domain signal, we require the *inverse Fourier transform*. This was the process illustrated in Figure 2.4, the process by which we reconstructed the pulse from its transform components. The inverse FT calculates $p(t)$ from $Fp(\omega)$ by the inverse transformation $\Im^{-1}$:

$$p(t) = \Im^{-1}\left(Fp(\omega)\right) = \frac{1}{2\pi}\int_{-\infty}^{\infty} Fp(\omega)e^{j\omega t}d\omega \tag{2.9}$$

Together, Equation 2.1 and Equation 2.9 form a relationship known as a *transform pair* that allows us to transform into the frequency domain, and back again. By this process, we can perform operations in the frequency domain or in the time domain, since we have a way of changing between them. One important process is known as *convolution*. The convolution of one signal $p_1(t)$ with another signal $p_2(t)$, where the convolution process denoted by $*$ is given by the integral

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

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

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

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

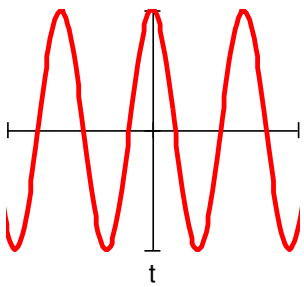
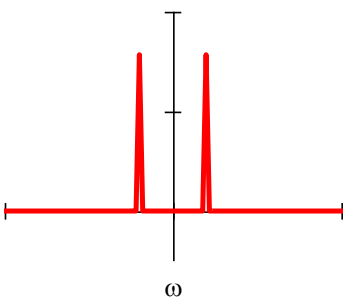
$$\begin{aligned} \Im\left[p_1(t) * p_2(t)\right] &= \int_{-\infty}^{\infty} Fp_2(\omega) p_1(\tau) e^{-j\omega\tau}\, d\tau \\ &= Fp_2(\omega) \int_{-\infty}^{\infty} p_1(\tau) e^{-j\omega\tau}\, d\tau \\ &= Fp_2(\omega) \times Fp_1(\omega) \end{aligned} \tag{2.12}$$

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

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

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
	

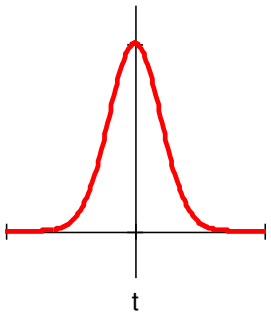
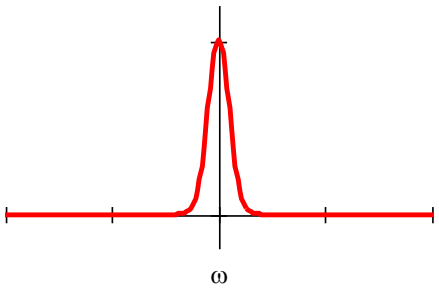
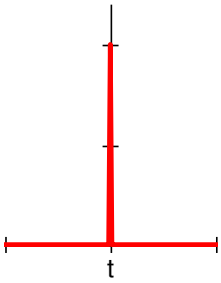
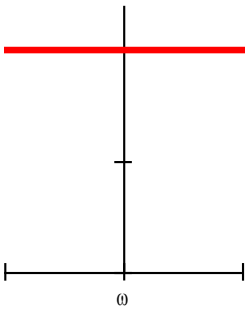
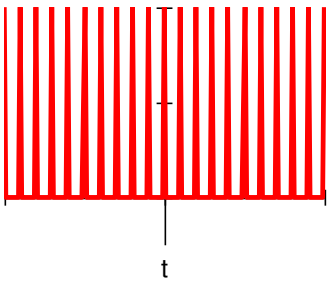
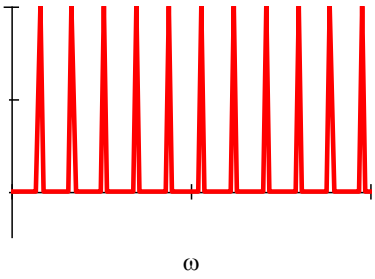
(a) cosine wave	**(b)** Fourier transform of cosine wave
 **(c)** Gaussian function	 **(d)** spectrum of Gaussian function
 **(e)** delta function	 **(f)** frequency content of delta function
 **(g)** sampling function in time domain	 **(h)** transform of sampling function

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:

$$delta(t-\tau) = \begin{cases} 1 & \text{if} \quad t = \tau \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

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.

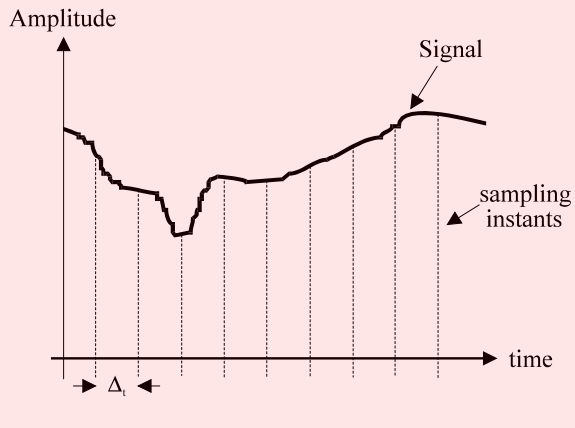
There are other Fourier transform pairs, as illustrated in Figure 2.6. Firstly, Figures 2.6(a) and (b) show that the Fourier transform of a cosine function is two points in the frequency domain (at

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.

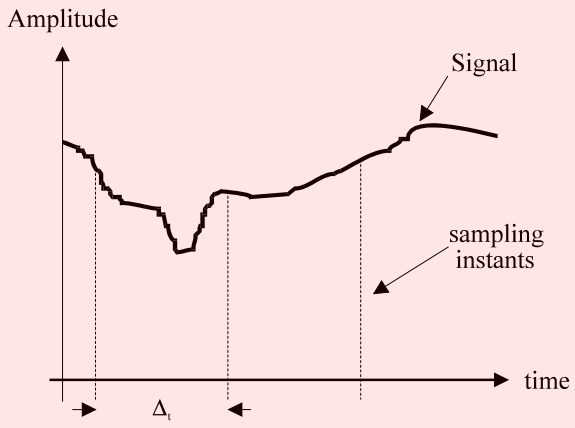
Finally, Figures 2.6(g) and (h) show that the transform of a set of uniformly-spaced delta functions is another set of uniformly-spaced delta functions, but with a different spacing. The spacing in the frequency domain is the reciprocal of the spacing in the time domain. By way of a (non-mathematical) 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