

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 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.

Table 2.1 Overview of Chapter 2

Main topic	Sub topics	Main points
Images	Effects of differing <i>numbers</i> of points and of number <i>range</i> for those points.	Greyscale, colour, resolution, dynamic range, storage.
Fourier transform theory	What is meant by the <i>frequency domain</i> , how it applies to <i>discrete</i> (sampled) images, how it allows us to <i>interpret</i> images and the <i>sampling resolution</i> (number of points).	Continuous Fourier transform and properties, sampling criterion, discrete Fourier transform and properties, image transformation, transform duals. Inverse Fourier transform.
Consequences of transform approach	Basic <i>properties</i> of Fourier transforms, <i>other transforms</i> , frequency domain <i>operations</i> .	Translation (shift), rotation and scaling. Principle of superposition and linearity. Walsh, Hartley, discrete cosine and wavelet transforms. Filtering and other operations.

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 usually derived from the output of an analogue-to-digital (A/D) converter. The matrix of pixels, the image, is usually square and an image may be described 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 *greyscale image* of a walking man in Figure 2.1(a). Smaller values of m give fewer available levels, reducing the available contrast in an image.

The ideal value of m is related to the signal-to-noise ratio (bandwidth) of the camera. This is stated as approximately 45 dB for an analogue camera, and since there are 6 dB per bit, 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, although some applications use a different range. The relative influence of the 8 bits is shown in the image of the walking subject in Figure 2.1. Here, the least significant bit, bit 0 (Figure 2.1b), carries the least information (it changes most rapidly). 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 2.1i). Clearly, the fact that there is a walker in the original image can be recognized much more reliably from the high-order bits than it can from the other bits (notice too the odd effects in the bits, which would appear to come from lighting at the top of the image).

Colour images 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, which 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 8 bits for each of the three RGB components. These images are known as *24 bit true colour* and they can contain 16 777 216 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. A good compression algorithm is always helpful in these cases, particularly if images

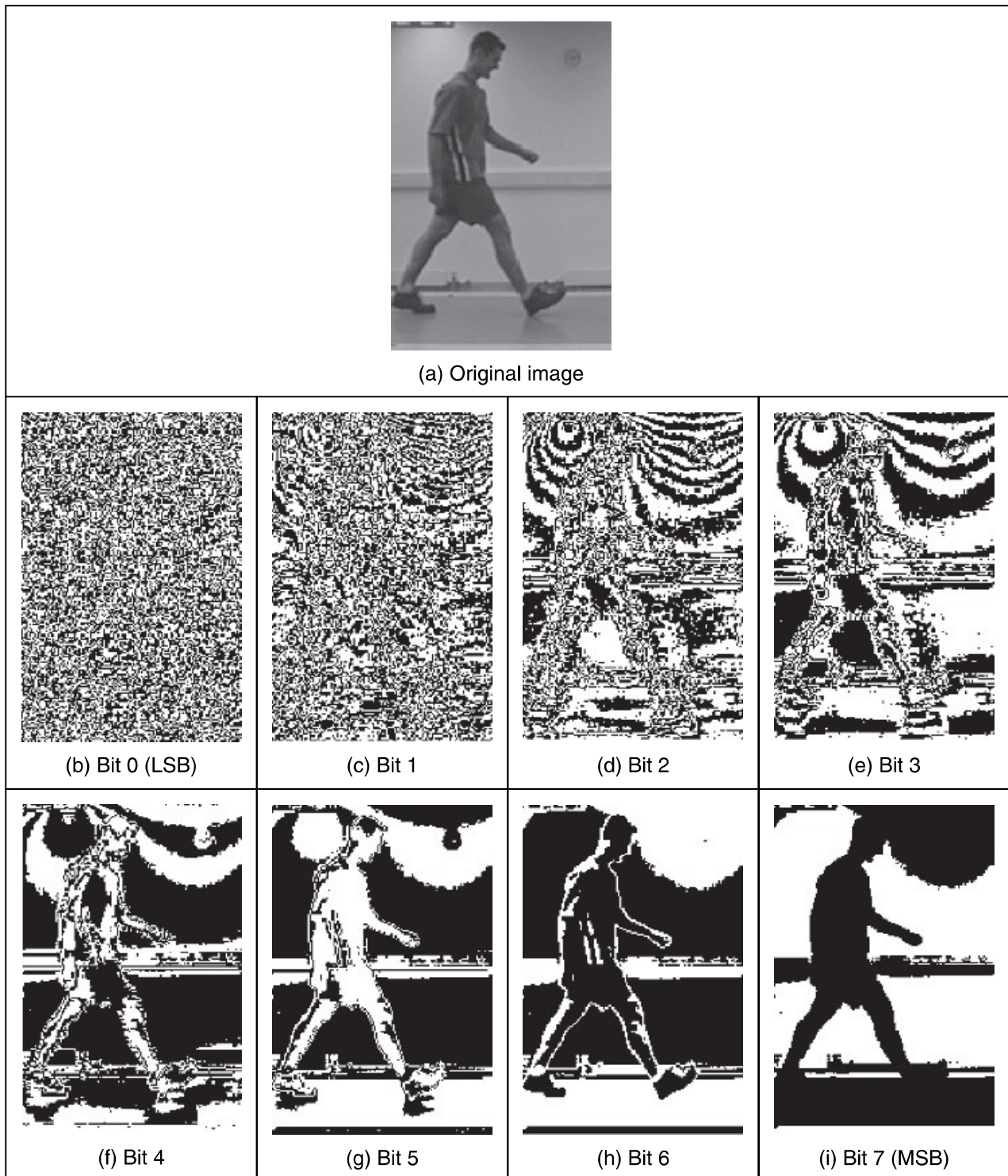


Figure 2.1 Decomposing an image into its bits

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. Should the image be originally 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.

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 quantized: 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 of the walking subject 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, which shows only the broad structure. It is impossible to see any detail in the 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 recognized from the image. (These images come from a research programme aimed to use computer vision techniques to recognize people by their gait; face recognition would have little potential for the low-resolution image, which is often the sort of image that security cameras provide.) If the image were a pure photographic image, some of the much finer detail like the hair would show up in much greater detail. This is because the grains in film are very much smaller than the pixels in a computer 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 emphasizes its blocky structure. The most common choices are for 256×256 images or 512×512 . These require 64 and 256 kbytes of storage, respectively. If we take a sequence of, say, 20 images for motion analysis, we will need more than 1 Mbyte to store the 20 256×256 images, and more than 5 Mbytes if the images were 512×512 . 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.

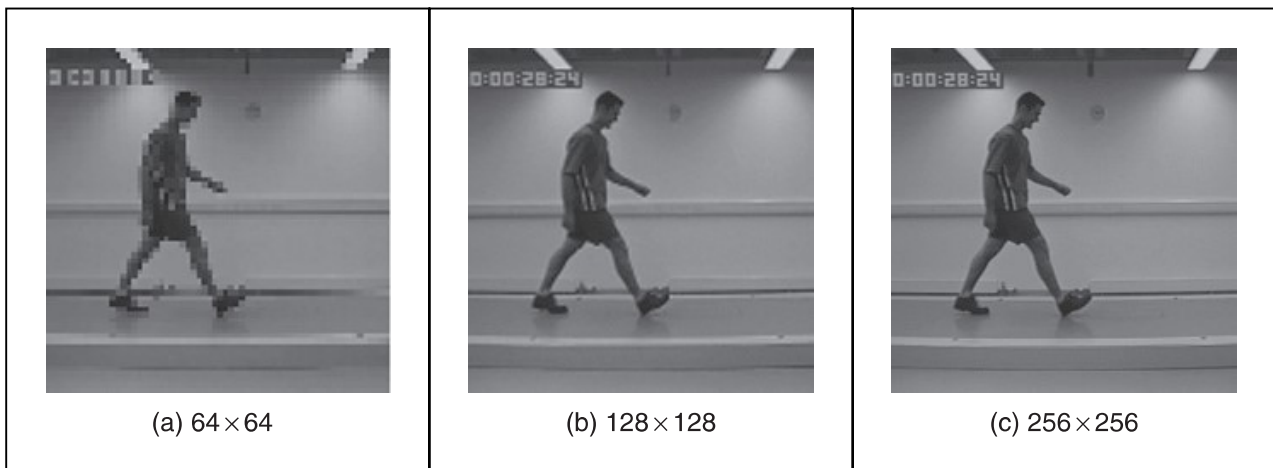


Figure 2.2 Effects of differing image resolution

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 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/second; s = 1/Hz).

Consider a music centre: the sound comes from a CD player (or a tape, etc.) 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 from the CD player, 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 visualize it in terms of its frequencies, rather than as a voltage which changes with time. But 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 equalizer, you have done this before. The graphic equalizer 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 equalizer. The equation that defines the *Fourier transform*, Fp , of a signal p , is given by a complex integral:

$$\mathfrak{F}(p(\omega)) = Fp(\omega) = \int_{-\infty}^{\infty} p(t)e^{-j\omega t} dt \quad (2.1)$$

where $Fp(\omega)$ is the Fourier transform; 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 (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!); $p(t)$ 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 when it ends at $t = T/2$, and is zero elsewhere, the pulse is:

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

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:

$$Fp(\omega) = \int_{-T/2}^{T/2} A e^{-j\omega t} dt \quad (2.3)$$

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

$$Fp(\omega) = -\frac{A e^{-j\omega T/2} - A e^{j\omega T/2}}{j\omega} \quad (2.4)$$

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

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

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 2.3b) 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 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 called the *spectrum* of the signal, which can be considered akin to the spectrum of light.

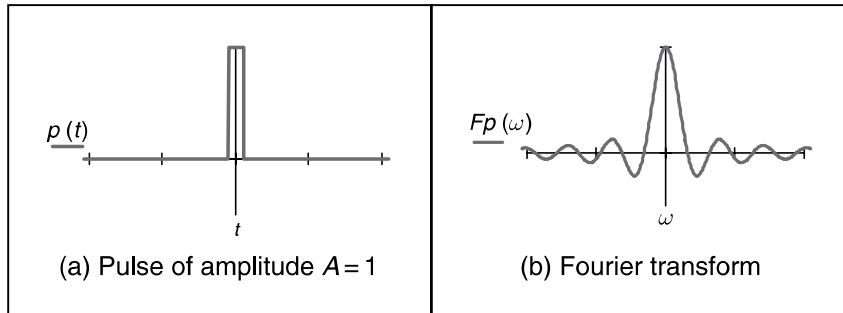


Figure 2.3 A pulse and its Fourier transform

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 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 2.4c), although there is more for $\omega = 4$ (Figure 2.4d). 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 the original pulse (Figure 2.4e). Here we have only considered frequencies from $\omega = -6$ to $\omega = 6$. If the

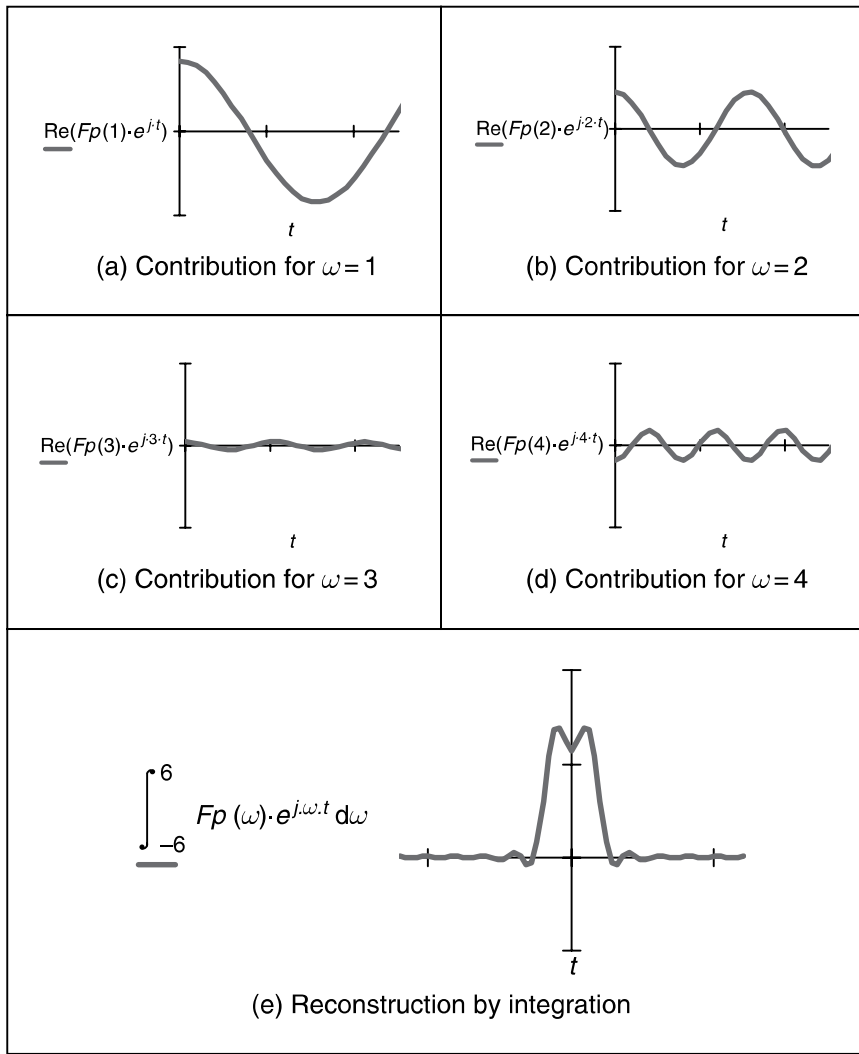


Figure 2.4 Reconstructing a signal from its transform

frequency range in integration were larger, more high frequencies would be included, leading to a more faithful reconstruction of the original pulse.

The result of the Fourier transform is a *complex* number. As such, it is usually represented in terms of its *magnitude* (or size, or modulus) and *phase* (or argument). The transform can be represented as:

$$\int_{-\infty}^{\infty} p(t)e^{-j\omega t} dt = \text{Re}[Fp(\omega)] + j \text{Im}[Fp(\omega)] \quad (2.6)$$

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

$$\left| \int_{-\infty}^{\infty} p(t)e^{-j\omega t} dt \right| = \sqrt{\text{Re}[Fp(\omega)]^2 + \text{Im}[Fp(\omega)]^2} \quad (2.7)$$

and the *phase* is:

$$\left\langle \int_{-\infty}^{\infty} p(t)e^{-j\omega t} dt \right\rangle = \tan^{-1} \frac{\text{Im}[Fp(\omega)]}{\text{Re}[Fp(\omega)]} \quad (2.8)$$

where the signs of the real and the imaginary components can be used to determine which quadrant the phase is in (since the phase can vary from 0 to 2π radians). The *magnitude* describes the *amount* of each frequency component, while 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).

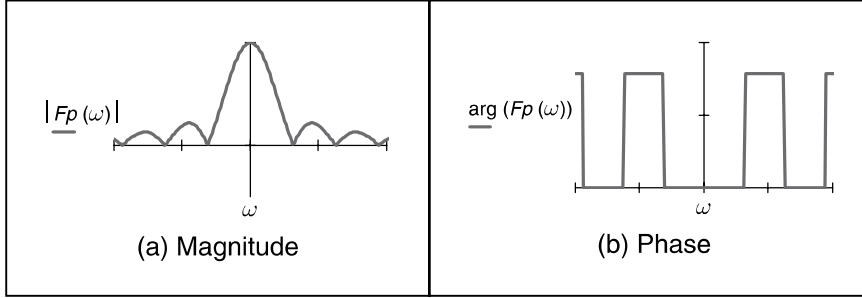


Figure 2.5 Magnitude and phase of the Fourier transform of a pulse

To return to the time domain signal, from the frequency domain signal, we require the *inverse Fourier transform*. This is the process by which we reconstructed the pulse from its transform components. The inverse FT calculates $p(t)$ from $Fp(\omega)$ according to:

$$p(t) = \mathfrak{F}^{-1}(Fp(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} Fp(\omega) e^{-j\omega t} d\omega \quad (2.9)$$

Together, Equations 2.1 and 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

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

This is 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, where the Fourier transformation is denoted by $\mathfrak{F}$, the Fourier transform of the convolution of two signals is

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

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

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

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). 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.

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:

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

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. First, Figure 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. Figure 2.6(c) and (d) show that the transform of the *Gaussian function* is another Gaussian function; this illustrates linearity (for linear systems it is 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 2.6f); 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, Figure 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

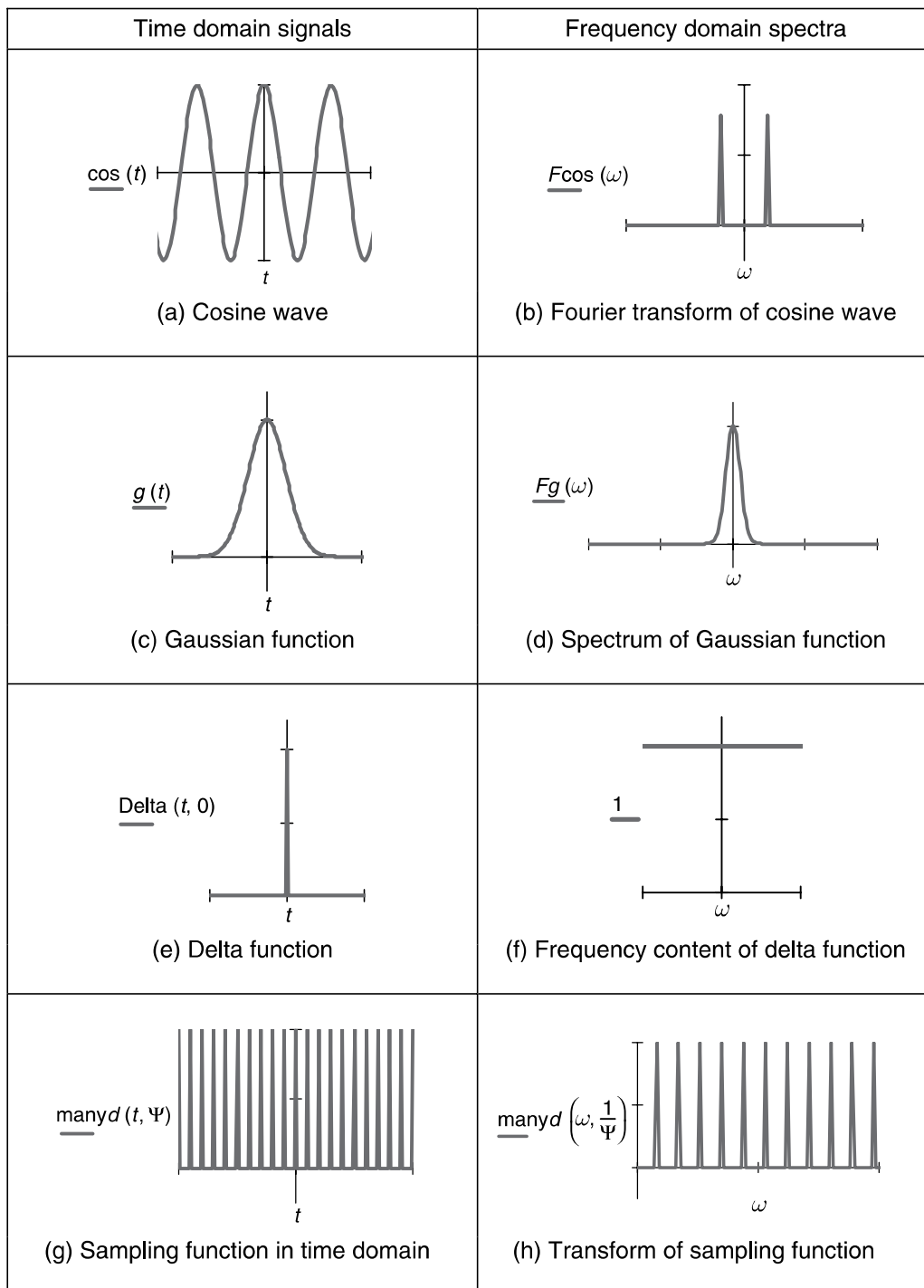


Figure 2.6 Fourier transform pairs

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 the basis of sampling theory (which we aim to use to find a criterion that 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. 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.

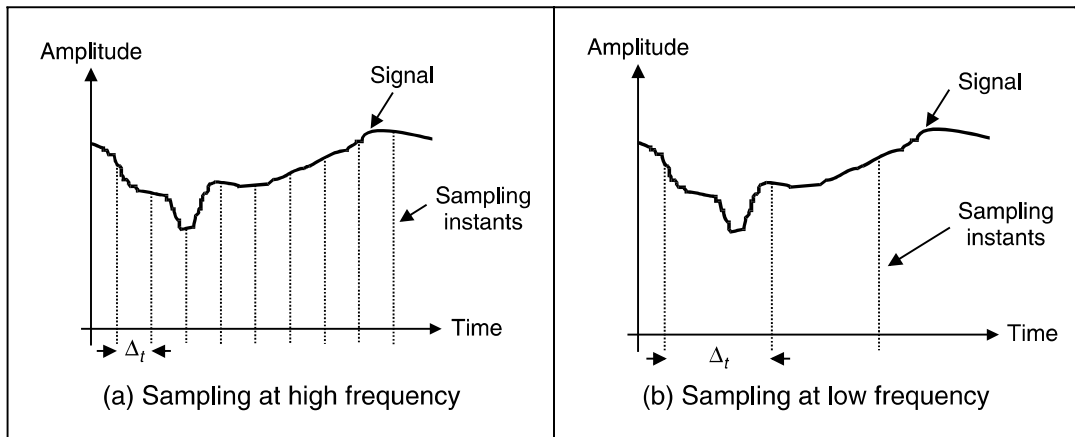


Figure 2.7 Sampling at different frequencies

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 2.8(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 $f_{\text{sample}} = 1/\Delta_t$ Hz. The Fourier pair of this transform was illustrated earlier (Figure 2.6g 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 2.8(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. Convolution of 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 2.8(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 2.8(b). If the sample spacing is small, then the time-variant signal's spectrum is replicated close together and the spectra *collide*, or interfere, as in Figure 2.8(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 2.8(c). If the

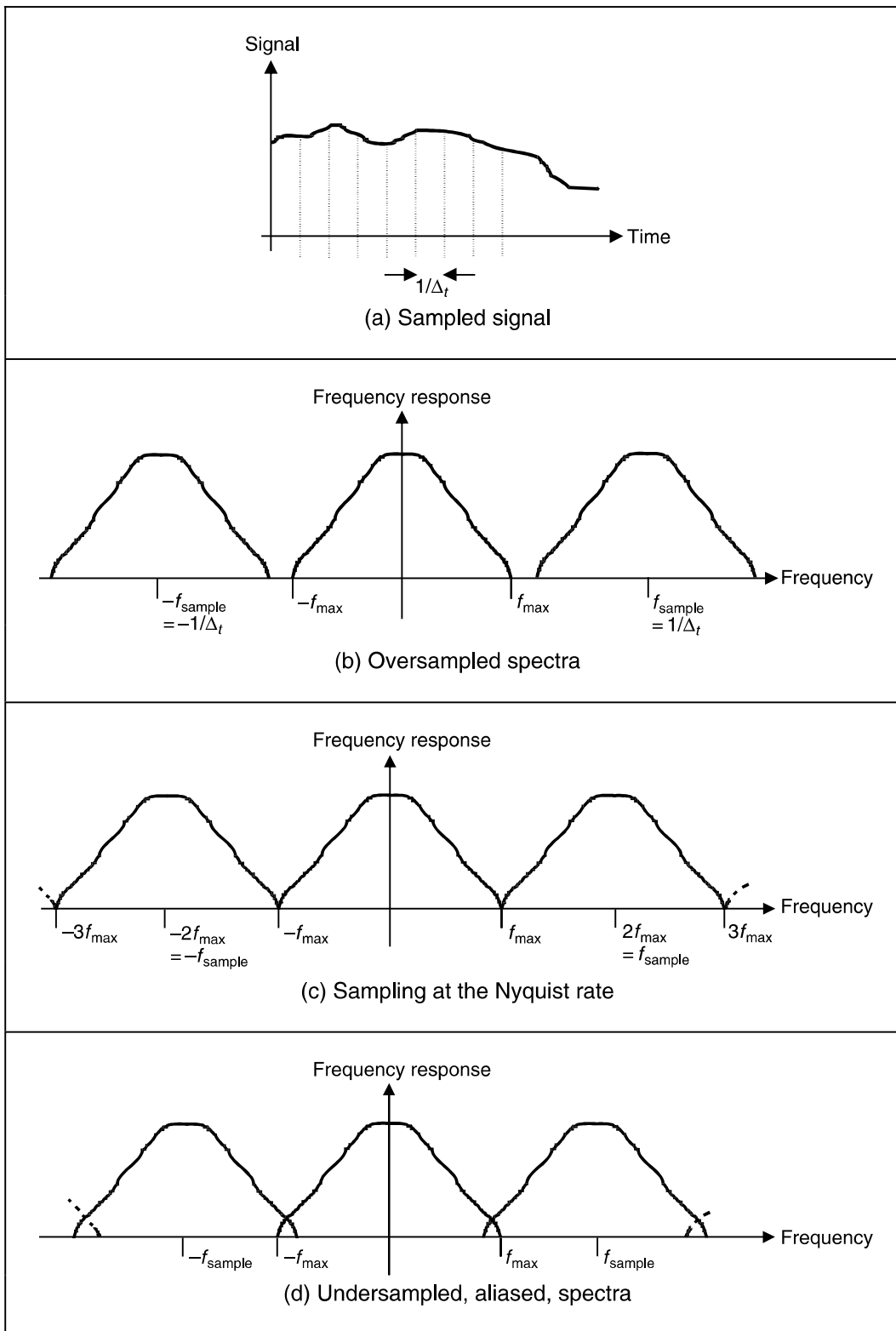


Figure 2.8 Sampled spectra

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 2.8(d) would lead to the wrong signal, whereas inverse Fourier transformation of the frequencies between $-f_{\max}$ and $f_{\max}$ in Figure 2.8(b) and (c) would lead back to the original signal. This can be seen in computer images as illustrated in Figure 2.9, 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 subimage, 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 2.9b). Now, look closely at the two sets of window blinds behind the people. At higher resolution, in Figure 2.9(a), these appear as normal window blinds. In Figure 2.9(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 as a 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 not to wear chequered clothes, or should not!)

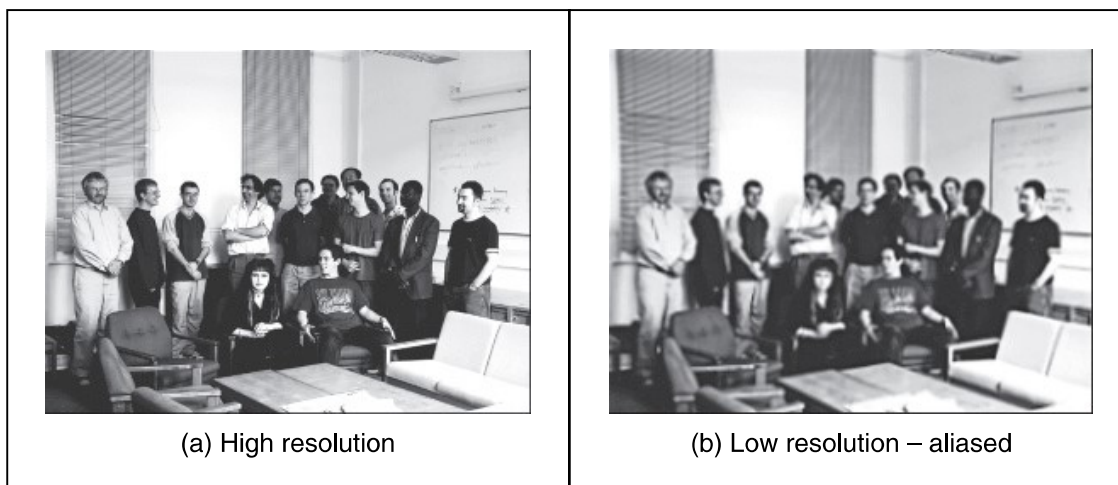


Figure 2.9 Aliasing in sampled imagery

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 many image processing systems have a maximum resolution of 512×512 , they must anticipate aliasing. This is mitigated somewhat by the observations that:

- 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
- 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, still applies.

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° between frames, as illustrated in Figure 2.10(b). If the wheel is rotating much faster, by 340° between frames, as in Figure 2.10(c) and (d), the wheel will appear to rotate the other way. If the wheel rotates by 360° between frames, it will appear to be stationary. To perceive the wheel as rotating forwards, then the rotation between frames must be 180° at most. This is consistent with sampling at least twice the maximum frequency. Our eye can resolve this in films (when watching a film, I 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.

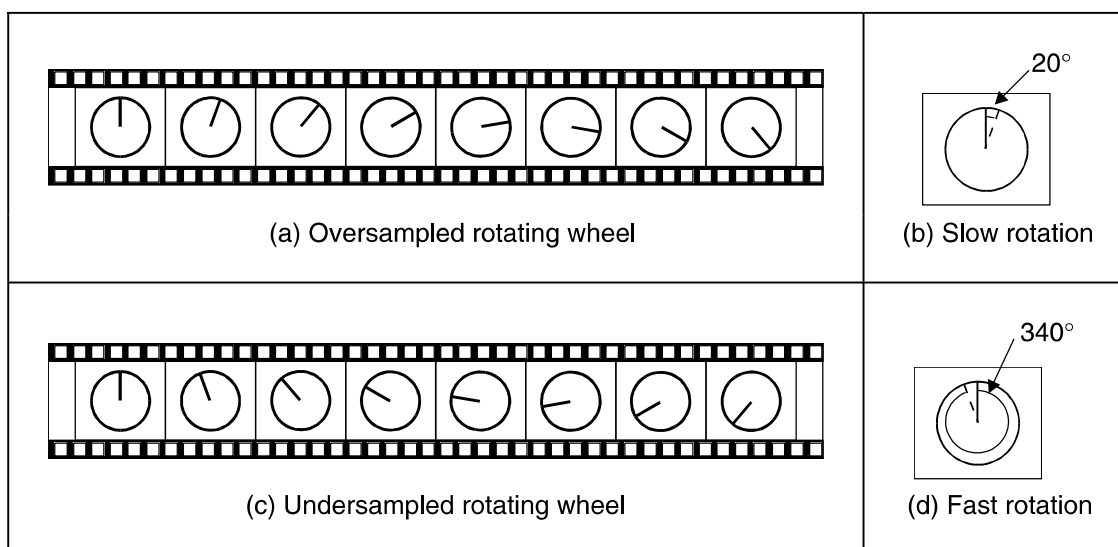


Figure 2.10 Correct and incorrect apparent wheel motion