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. 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:

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

where Re(ω) and Im(ω) are the real and imaginary parts of the transform, respectively. The *magnitude* of the transform is then:

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

and the *phase* is:

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

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, 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).

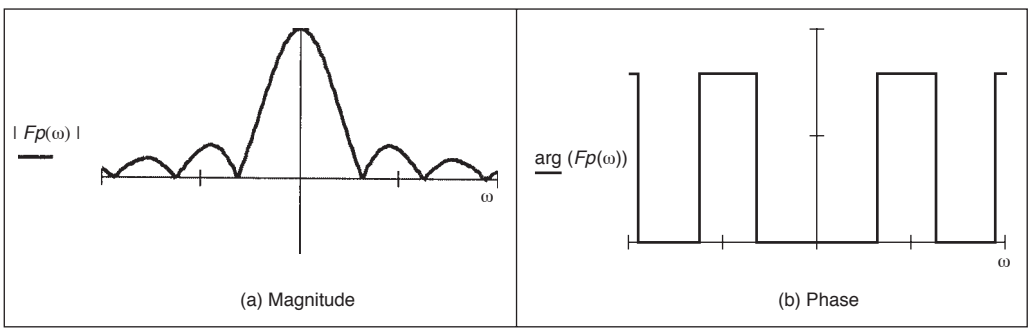


Figure 2.5 Magnitude and phase of Fourier transform of pulse

In order to return to the time domain signal, from the frequency domain signal, we require the *inverse Fourier transform*. Naturally, 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:

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

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, where the Fourier transformation is denoted by F, the Fourier transform of the convolution of two signals is

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

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

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

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). 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:

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

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**. First, 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. 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), then 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.

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

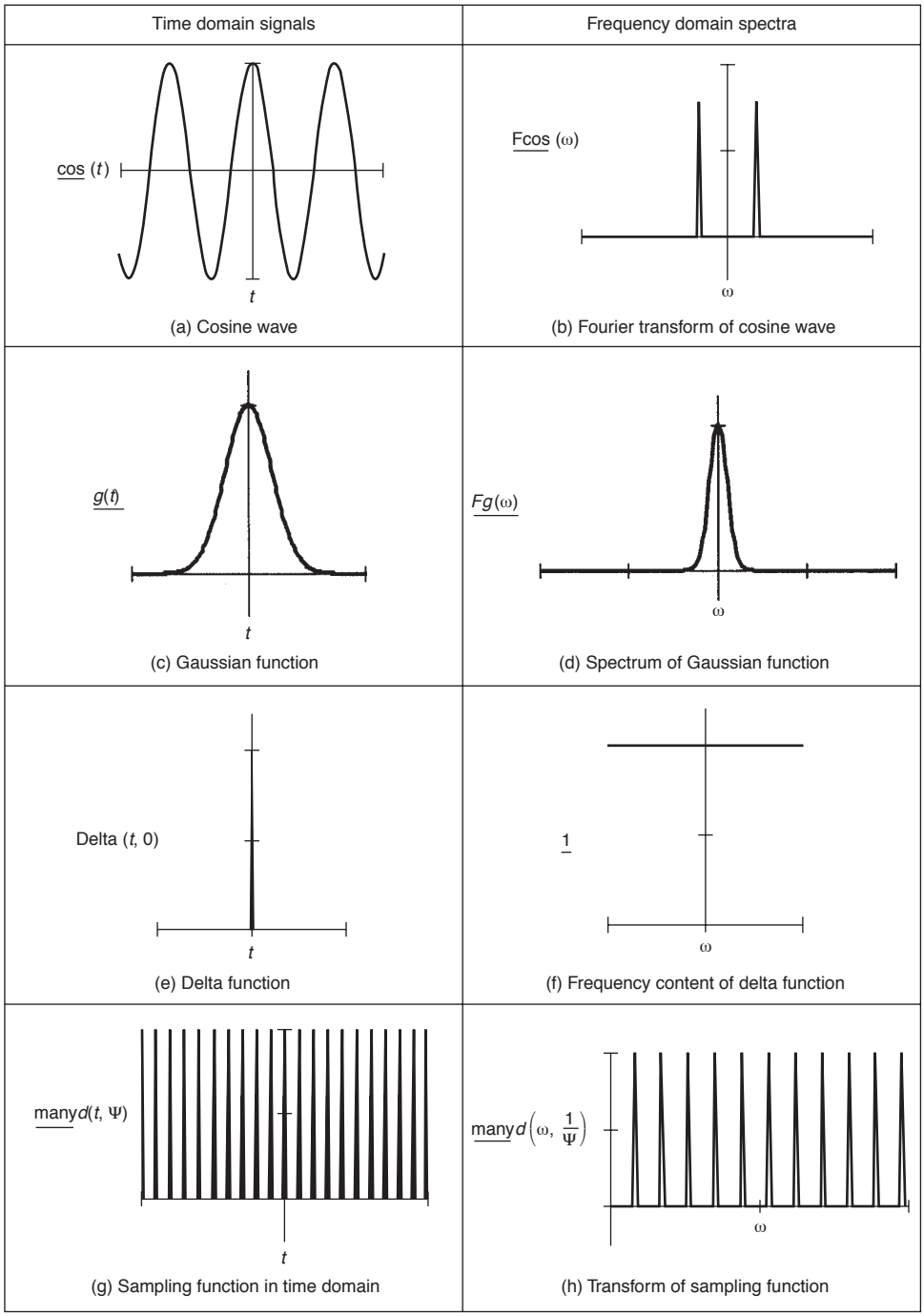


Figure 2.6 Fourier transform pairs

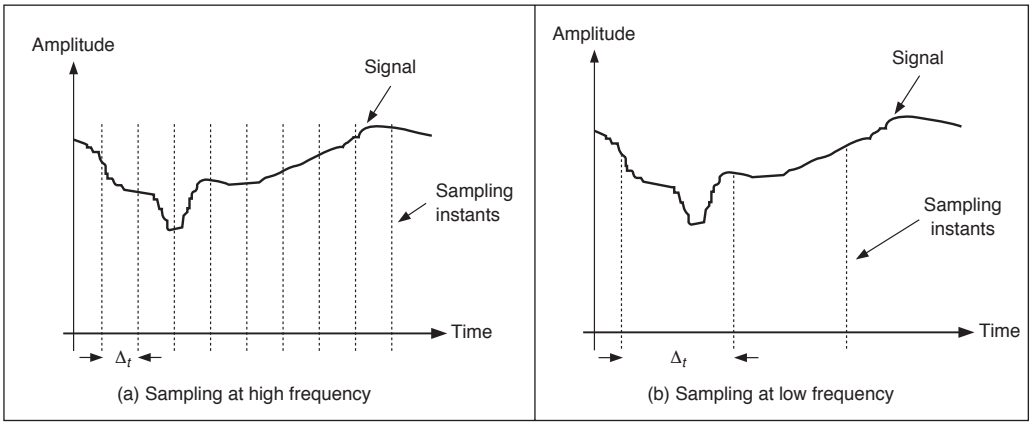


Figure 2.7 Sampling at different frequencies

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 Figures **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 *small*, 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 a texture image (a chain-link fence) taken at different spatial resolutions. The lines in an original version are replaced by indistinct information in the version sampled at low frequency. Indeed, it would be difficult to imagine what Figure **2.8**(c) represents, whereas it is much more clear in Figures **2.8**(a) and (b). Also, the texture in Figure **2.8**(a) appears to have underlying distortion (the fence appears to be bent) whereas Figures **2.8**(b) and (c) do not show this. 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 get the wrong signal.

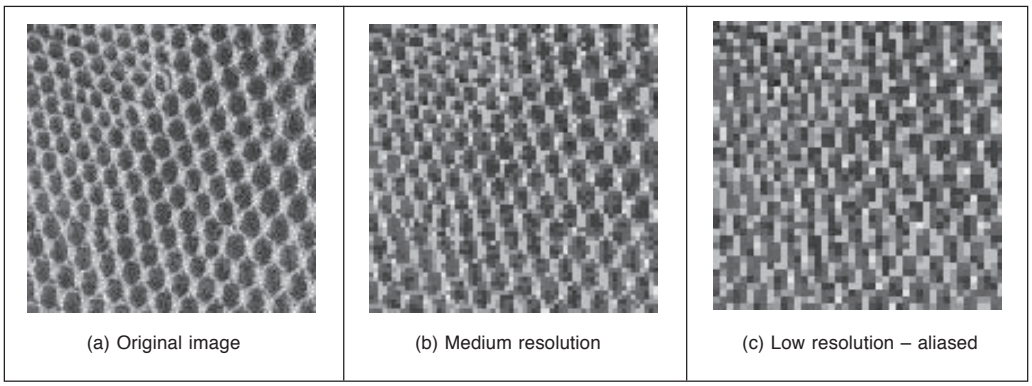


Figure 2.8 Aliasing in sampled imagery

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:

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.

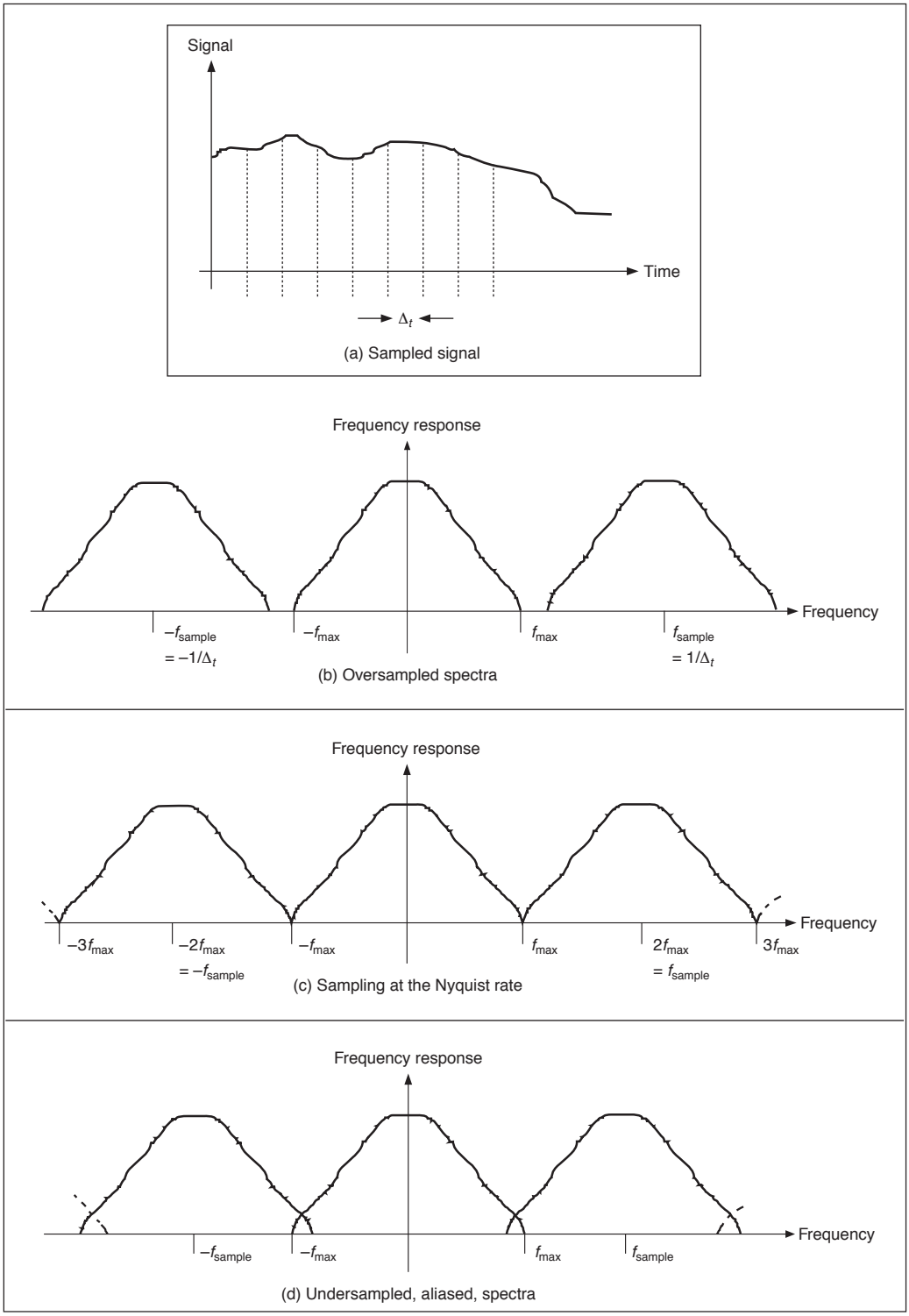


Figure 2.9 Sampled spectra

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 in 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 Figure **2.10**(d) then the wheel will appear to rotate the other way. If the wheel rotates by 360° between frames, then it will appear to be stationary. In order to perceive the wheel as rotating forwards, then the rotation between frames must be 180° at most. This is consistent with sampling at 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.

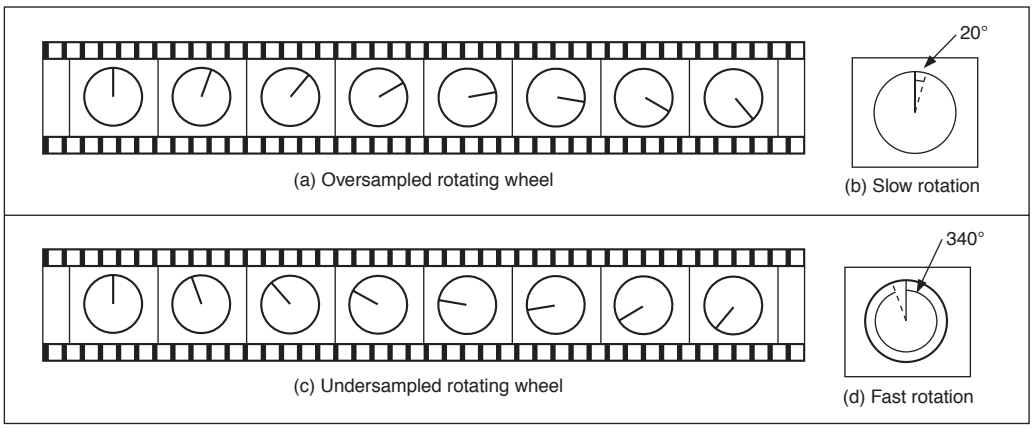


Figure 2.10 Correct and incorrect apparent wheel motion

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:

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

This 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), then the equation becomes:

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

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

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

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

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

By rearrangement, we obtain:

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

The modulus of the transform is:

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

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

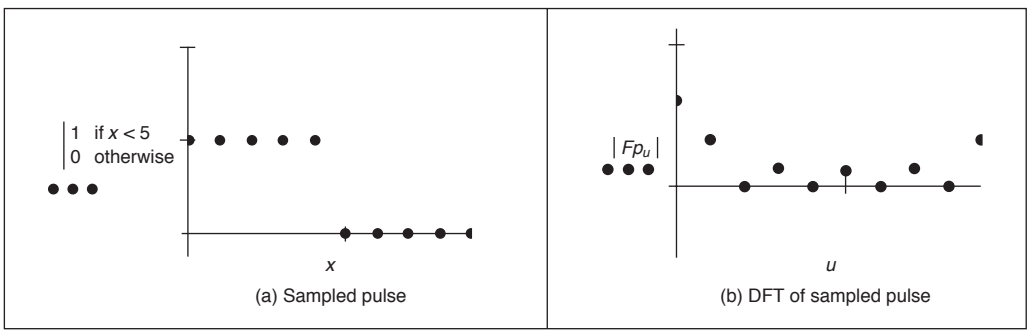


Figure 2.11 Transform pair for sampled pulse

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).

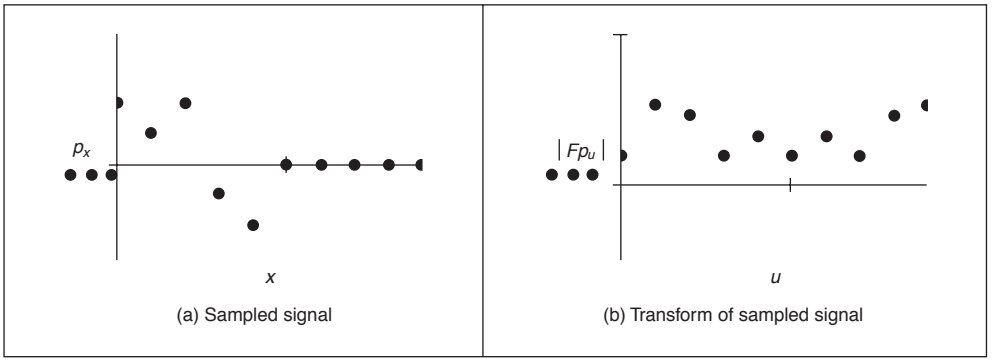


Figure 2.12 A sampled signal and its discrete transform

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 pulse. 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 corrupt.

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 co-ordinates x and y. This implies that we have two dimensions of frequency, u and v,

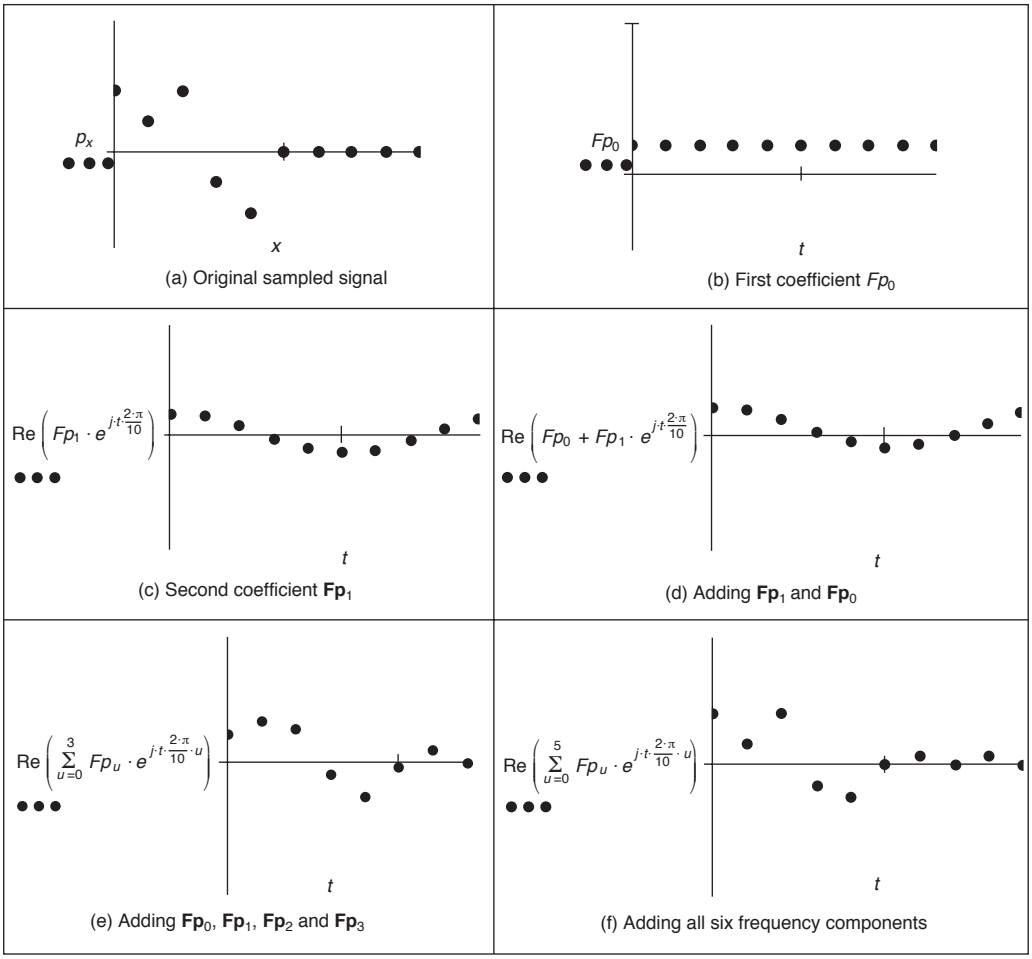


Figure 2.13 Signal reconstruction from its transform components

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:

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

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 to 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.

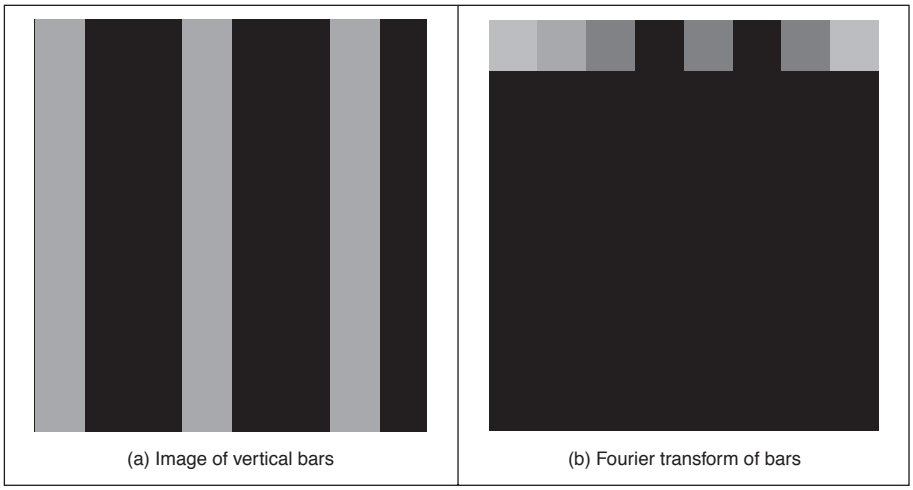


Figure 2.14 Applying the 2D discrete fourier transform

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

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

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:

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

so,

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

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 effort). 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

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

showing 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).

Since the computational cost of a 1D FFT of N points is $O(N \log(N))$, the cost (by separability) for the 2D FFT is $O(N^2 \log(N))$ 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 128 × 128 image, if the FFT takes minutes, the DFT will take days). The 2D FFT is available in Mathcad using the `icfft` function which gives a result equivalent to Equation 2.22. The inverse 2D FFT, Equation 2.23, can be implemented using the Mathcad `cfft` function. (The difference between many Fourier transform implementations essentially concerns the chosen scaling factor.) The Mathcad implementations of the 2D DFT, the inverse 2D DFT, are given in Code **2.1**(a) and Code **2.1**(b), respectively. The implementations using the Mathcad functions using the FFT are given in Code **2.1**(c) and Code **2.1**(d), respectively.

$$FP_{u,v} := \frac{1}{rows(P)}\sum_{y=0}^{rows(P)-1}\sum_{x=0}^{cols(P)-1}P_{y,x}\cdot e^{\frac{-j\cdot 2\cdot\pi\cdot(u\cdot y + v\cdot x)}{rows(P)}}$$

(a) 2D DFT, Equation 2.22

$$IFP_{y,x} := \sum_{u=0}^{rows(FP)-1}\sum_{v=0}^{cols(FP)-1}FP_{u,v}\cdot e^{\frac{j\cdot 2\cdot\pi\cdot(u\cdot y+v\cdot x)}{rows(FP)}}$$

(b) Inverse 2D DFT, Equation 2.23

```
Fourier(pic):=icfft(pic)
```

(c) 2D FFT

```
inv_Fourier(trans):=cfft(trans)
```

(d) Inverse 2D FFT

Code 2.1 Implementing Fourier transforms

For reasons of speed, the 2D FFT is the algorithm commonly used in application. One (unfortunate) difficulty is that the nature of the Fourier transform produces an image

which, at first, is difficult to interpret. The Fourier transform of an image gives the frequency components. The position of each component reflects its frequency: *low* frequency components are *near* the origin and *high* frequency components are further *away*. As before, the lowest frequency component – for zero frequency – the d.c. component represents the *average* value of the samples. Unfortunately, the arrangement of the 2D Fourier transform places the low frequency components at the *corners* of the transform. The image of the square in Figure **2.15**(a) shows this in its transform, Figure **2.15**(b). A spatial transform is easier to visualise 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 shifts the transform to the centre. Both operations result in the image in Figure **2.15**(c) wherein the transform is much more easily seen. Note that this is aimed to improve visualisation and does not change any of the frequency domain information, only the way it is displayed.

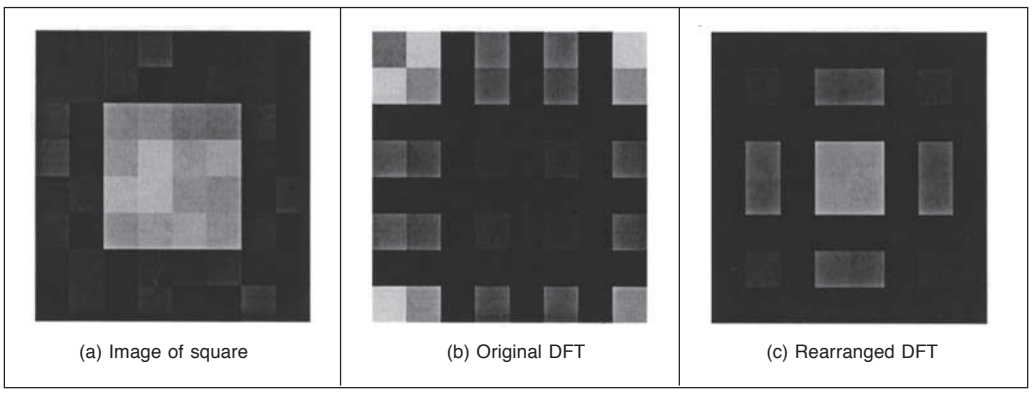


Figure 2.15 Rearranging the 2D DFT for display purposes

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 in the exponent keeps the analysis neat) so we obtain the transform of the multiplied image as:

$$\begin{aligned}\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)(ux+vy)}\times -1^{(x+y)} &= \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)(ux+vy)}e^{-j\pi(x+y)}\\ &= \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{x,y}e^{-j\left(\frac{2\pi}{N}\right)\left(\left(u+\frac{N}{2}\right)x+\left(v+\frac{N}{2}\right)y\right)} \qquad (2.28)\\ &= \mathbf{FP}_{u+\frac{N}{2},v+\frac{N}{2}}\end{aligned}$$

According to Equation 2.28, 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

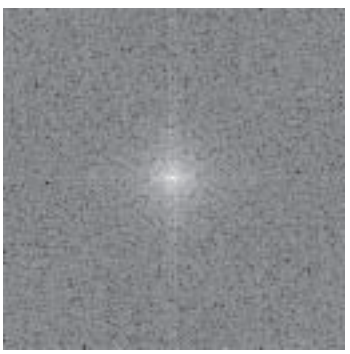
implies that the centre of a transform image will now be the d.c. component. (Another way of interpreting this is rather than look at the frequencies centred on where the image is, our viewpoint has been shifted so as to be centred on one of its corners – thus invoking the replication property.) The operator `rearrange`, in Code **2.2**, is used prior to transform calculation and results in the image of Figure **2.15**(c), and all later transform images.

```
rearrange(picture):=  for y∈0..rows(picture)−1
                        for x∈0..cols(picture)−1
                          rearranged_pic_{y,x}←picture_{y,x}·(−1)^(y+x)
                      rearranged_pic
```

Code 2.2 Reordering for transform calculation

The full effect of the Fourier transform is shown by application to an image of much higher resolution. Figure **2.16**(a) shows the image of a face and Figure **2.16**(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 face image has many regions where the brightness does not change a lot, such as the cheeks and forehead. The *high* frequency components reflect *change* in intensity. Accordingly, the higher frequency components arise from the hair (and that feather!) and from the borders of features of the human face, such as the nose and eyes.

(a) Image of face



(b) Transform of face image

Figure 2.16 Applying the Fourier transform to the image of a face

As with the 1D Fourier transform, there are 2D Fourier transform pairs, illustrated in Figure **2.17**. The 2D Fourier transform of a two-dimensional pulse, Figure **2.17**(a), is a two-dimensional sinc function, in Figure **2.17**(b). The 2D Fourier transform of a Gaussian function, in Figure **2.17**(c), is again a two-dimensional Gaussian function in the frequency domain, in Figure **2.17**(d).

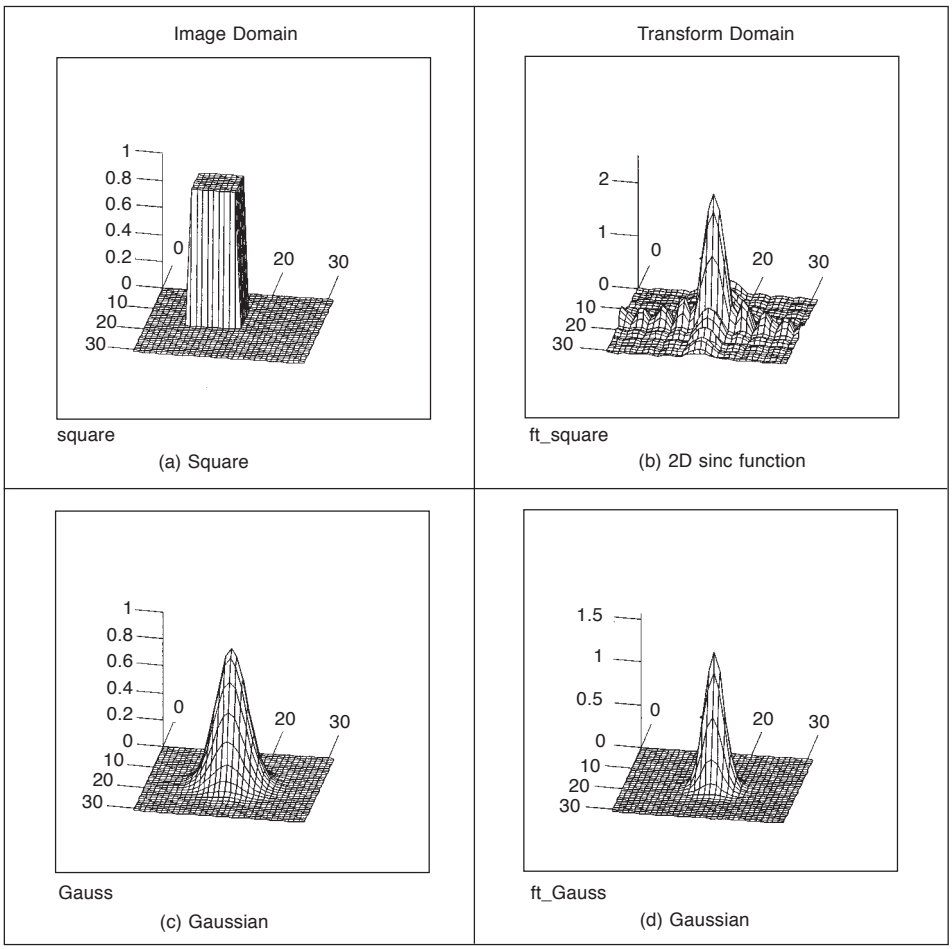


Figure 2.17 2D Fourier transform pairs

2.6 Other 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 is $F[\, p(t - \tau)]$, we obtain the relationship between a time domain shift in the time and frequency domains as:

$$F[\, p(t - \tau)] = e^{-j\omega\tau} P(\omega) \tag{2.29}$$

Accordingly, the magnitude of the Fourier transform is:

$$|F\,[\,p(t-\tau)]| = |e^{-j\omega\tau}\,P(\omega)| = |e^{-j\omega\tau}|\,|P(\omega)| = |P(\omega)| \tag{2.30}$$

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:

$$\langle F\,[\,p(t-\tau)] = \langle e^{-j\omega\tau}\,P(\omega) \tag{2.31}$$

The Mathcad implementation of a `shift` operator, Code **2.3**, uses the modulus operation to enforce the cyclic shift. The arguments fed to the function are: the image to be shifted (`pic`), the horizontal shift along the *x* axis (`x_val`), and the vertical shift down the *y* axis (`y_val`).

```
shift(pic,y_val,x_val):= NC←cols(pic)
                         NR←rows(pic)
                         for y∈0..NR–1
                           for x∈0..NC–1
                             shifted_{y,x}←pic_{mod(y+y_val, NR),mod(x+x_val,NC)}
                         shifted
```

Code 2.3 Shifting an image

This process is illustrated in Figure **2.18**. An original image, Figure **2.18**(a), is shifted by 30 pixels along the *x* and the *y* axes, Figure **2.18**(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 the shifted image are identical: Figure **2.18**(b) appears the same as Figure **2.18**(e). The phase differs: the phase of the original image, Figure **2.18**(c), is clearly different from the phase of the shifted image, Figure **2.18**(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 Fourier transform is used to analyse an image of a human face, to describe it by its spatial frequency, then we do not need to control the position of the camera, or the face, precisely.

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.

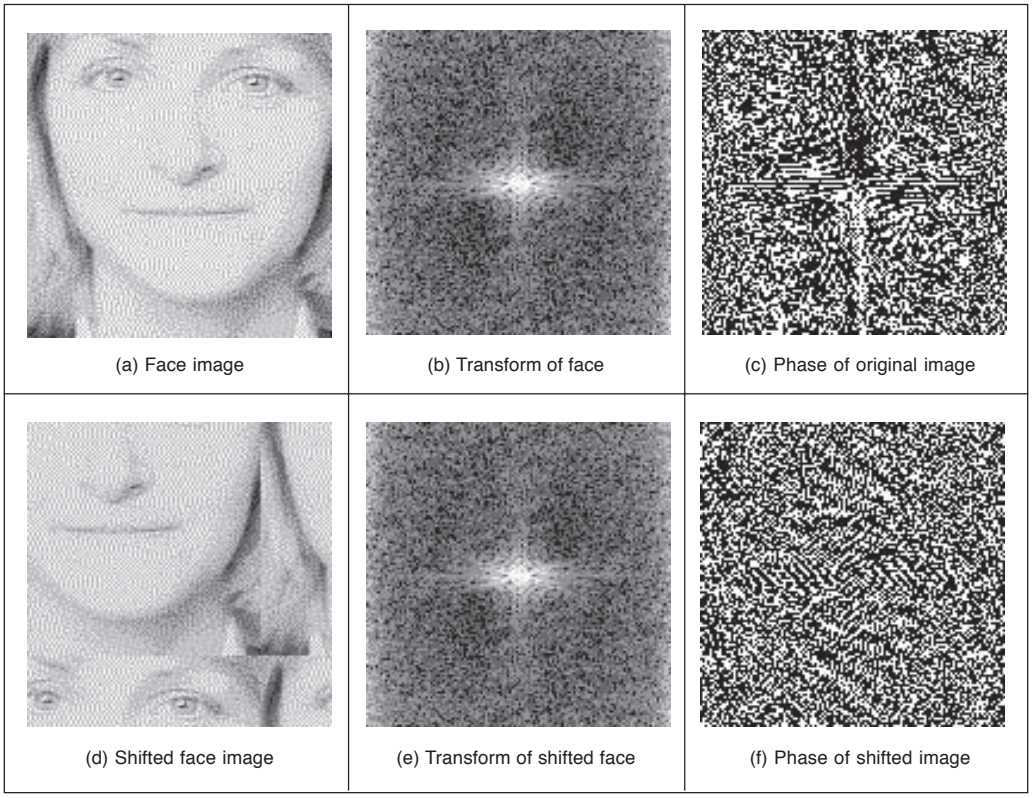


Figure 2.18 Illustrating shift invariance

This implies that if the frequency domain properties are to be used in image analysis, via the Fourier transform, then 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 **2.19**. A face image, Figure **2.19**(a), is rotated by 90° to give the image in Figure **2.19**(b). Comparison of the transform of the original image, Figure **2.19**(c), with the transform of the rotated image, Figure **2.19**(d), shows that the transform has been rotated by 90°, by the same amount as the image. In fact, close inspection of Figure **2.19**(c) shows that the major axis is almost vertical, and is consistent with the major axis of the face in Figure **2.19**(a).

2.6.3 Frequency scaling

By definition, time is the reciprocal of frequency. So if an image is compressed, equivalent to reducing time, then 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:

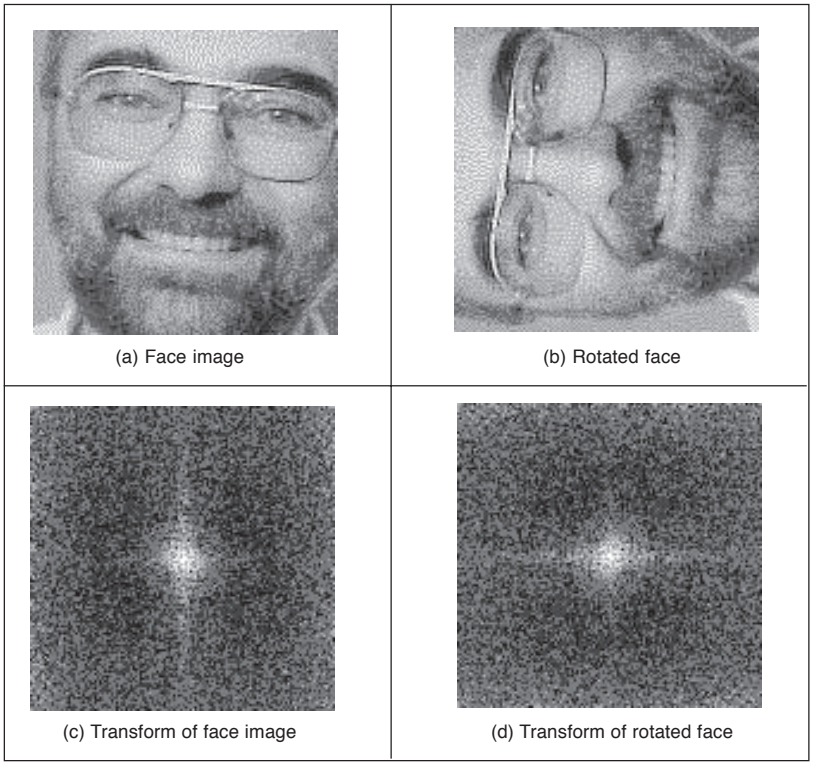


Figure 2.19 Illustrating rotation

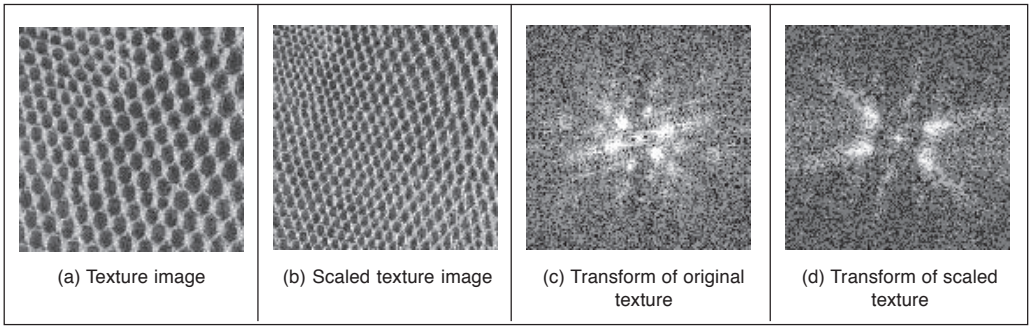
$$F[p(\lambda t)] = \frac{1}{\lambda} P\left(\frac{\omega}{\lambda}\right) \tag{2.32}$$

This is illustrated in Figure **2.20** where the texture image (of a chain-link fence), Figure **2.20**(a), is reduced in scale, Figure **2.20**(b), thereby increasing the spatial frequency. The DFT of the original texture image is shown in Figure **2.20**(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 **2.20**(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, then 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.

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



(a) Texture image | (b) Scaled texture image | (c) Transform of original texture | (d) Transform of scaled texture

Figure 2.20 Illustrating frequency scaling

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:

$$O(I_1 + I_2) = O(I_1) + O(I_2) \tag{2.33}$$

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:

$$F[p_1 + p_2] = F[p_1] + F[p_2] \tag{2.34}$$

In application this suggests that we can separate images by looking at their frequency domain components. 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 the texture (this is the spatial frequency of the cloth's pattern) and weaker, more scattered, components due to the fingerprint. If we suppress the frequency components due to the cloth's texture, and invoke the inverse Fourier transform, then the cloth will be removed from the original image. The fingerprint can now be seen in the resulting image.

2.7 Transforms other than Fourier

2.7.1 Discrete cosine transform

The *Discrete Cosine Transform* (DCT) (Ahmed, 1974) is a real transform that has great advantages in *energy compaction*. Its definition for spectral components $\mathbf{DP}_{u,v}$ is:

$$\mathbf{DP}_{u,v} = \begin{cases} \dfrac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathbf{P}_{x,y} & \text{if } u = 0 \text{ and } v = 0 \\ \dfrac{2}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathbf{P}_{x,y} \times \cos\left(\dfrac{(2x+1)u\pi}{2N}\right) \times \cos\left(\dfrac{(2y+1)v\pi}{2N}\right) & \text{otherwise} \end{cases} \tag{2.35}$$

The inverse DCT is defined by

$$\mathbf{P}_{x,y} = \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \mathbf{DP}_{u,v} \times \cos\left(\frac{(2x+1)u\pi}{2N}\right) \times \cos\left(\frac{(2y+1)v\pi}{2N}\right) \tag{2.36}$$