HST.582J / 6.555J / 16.456J Biomedical Signal and Image Processing
Spring 2007

HST582J/6.555J/16.456J Biomedical Signal and Image Processing Spring 2007

Chapter 9 - IMAGE PROCESSING

Introduction

Image processing represents an application of the more general field of two-dimensional (2-D) signal processing. In the same way that a one-dimensional (1-D) signal $x(t)$ can be sampled to form a discrete-time signal $x[n]$, an image can be digitized to form a 2-D discrete-space signal $x[n_1, n_2]$. The digitized samples of an image are referred to as *pixels.* Figure 1 (parts a-c) shows a sampled image with three different spatial resolutions.

Most commonly, the value of the pixel $x[n_1, n_2]$ is interpreted as the light intensity of the image at the point (n_1, n_2). This approach works very well for gray-scale images, and can easily be extended for color. Since any color image can be separated into combination of three mono-color images (usually red, green, and blue), a color image can be represented by three separate mono-color $x[n_1, n_2]$ images. While color images can convey more information to a human observer, they can greatly increase the complexity of the image processing task. In this chapter we will only consider gray-scale images.

Digital images are not only discrete-space, they are also quantized in the sense that each pixel can only take a finite number of values. Figure 1 (parts a,d,e) shows three versions of an image quantized to 256, 16, and 2 gray levels, respectively. While the 256-level (8-bit) image is of good quality, quantization is noticeable for 16 gray levels (4 bits). On the other hand, the image remains recognizable with as few as 2 gray levels (1 bit). In these notes, we will ignore quantization, assuming that the number of gray levels is large enough that these effects can be neglected.

Some image processing methods are simple extensions of their 1-D counterparts. For example, Fourier transforms and convolution/filtering have a natural extension from the 1-D to the 2-D case. However, some methods, such as histogram modification are specific to images, and arise because the results of image processing are viewed, not as graphs, but as as gray-scale images. The visual impact of $x[n_1, n_2]$ is usually very different when it is displayed as an image rather than a surface contour (see Figure 2). Other differences between 1-D and 2-D signal processing arise because certain mathematical properties of 1-D signals do not hold for 2-D signals. For example, as seen in Chapter 6, 1-D filters can be characterized by their poles and zeros. However, it is not in general possible to characterize 2-D filters by their poles and zeroes because 2-D polynomials do not always have roots.

9.1 The 2-D Continuous Space Fourier Transform

Although processing by digital computers requires discrete images, many basic concepts of image processing are most clearly introduced using continuous images because of the greater symmetry between spatial coordinates, and between the space and frequency domains.

The Fourier transform pair for a two dimensional signal $x(t_1, t_2)$ is given by [1]

$$X(F_1, F_2) \;=\; \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1, t_2)\; e^{-j2\pi\, t_1 F_1} e^{-j2\pi\, t_2 F_2} dt_1 dt_2 \tag{9.1a}$$

and

$$x(t_1, t_2) \;=\; \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(F_1, F_2)\; e^{j2\pi\, t_1 F_1}\; e^{j2\pi\, t_2 F_2} dF_1 dF_2 \tag{9.1b}$$

As an example of a 2-D transform, consider the simple rectangular function $x(t_1, t_2)$ which has unit amplitude for $|t_1| < T_1$ and $|t_2| < T_2$ and is 0 otherwise. The Fourier transform of $x(t_1, t_2)$ is given by

$$X(F_1, F_2) \;=\; \int_{-T_1}^{T_1}\int_{-T_2}^{T_2} e^{-j2\pi\, t_1 F_1}\; e^{-j2\pi\, t_2 F_2}\; dt_1 dt_2 \tag{9.2a}$$

$$X(F_1, F_2) \;=\; 4T_1T_2\, \frac{\sin(2\pi F_1 T_1)}{2\pi F_1 T_1}\, \frac{\sin(2\pi F_2 T_2)}{2\pi F_2 T_2} \tag{9.2b}$$

Thus, the Fourier transform of a 2-D rectangle is a 2-D *sinc* function. Figure 2 shows the graph of $|X(F_1, F_2)|$ for a rectangle with $T_1 = T_2$. The magnitude $|X(F_1, F_2)|$ is shown both in conventional 3-D perspective and as a gray-scale image. Figure 3 shows examples of other signals $x(t_1, t_2)$ which have unit amplitude in a part of the t_1, t_2 space; also shown are their Fourier transform magnitudes $|X(F_1, F_2)|$.

In displaying $|X(F_1, F_2)|$ as an image, it is common to transform the magnitude to image intensity using the function

$$I(F_1, F_2) \;=\; \log(1 + |X(F_1, F_2)|) \tag{9.3}$$

The use of Equation (9.3) helps compensate for the fact that the eye has a logarithmic response to intensity. If the intensity were left proportional to $|X(F_1, F_2)|$, most of the smaller features of $|X(F_1, F_2)|$ would not be visible. The Fourier transform images shown in Figure 2 and 9.3 have been adjusted using Equation (9.3).

9.1.1 Separability of the Fourier integral

The Fourier transform $X(F_1, F_2)$ in Equation (9.2b) can be written as the product of two separate sinc functions $X_1(F_1)$ and $X_2(F_2)$. Examining Equations (9.1a,b), we can see that this is to be expected. In general, if $x(t_1, t_2)$ is the product of a function of t_1 and a function of t_2, then $X(F_1, F_2)$ is the product of the 1-D transforms:

$$x(t_1, t_2) \;=\; x_1(t_1)\; x_2(t_2) \quad\longleftrightarrow\quad X(F_1, F_2) \;=\; X_1(F_1)\; X_2(F_2)$$

[1] We use the notation t_1 and t_2 to represent the two dimensions of the signal. These variables should not be confused with time. In the case of images, t_1 and t_2 represent spatial coordinates.

Even when $x(t_1, t_2)$ is not the product of a function of t_1 and a function of t_2, the evaluation of the Fourier transform can still be grouped as

$$X(F_1, F_2) \;=\; \int_{-\infty}^{\infty} (\int_{-\infty}^{\infty} x(t_1, t_2)\; e^{-j2\pi\, t_1 F_1}\; dt_1)\; e^{-j2\pi\, t_2 F_2}\; dt_2 \tag{9.4}$$

Equation (9.4) is important because it demonstrates that a 2-D Fourier transform can be evaluated as separate 1-D Fourier transforms.

9.1.2 Rotation theorem

If the signal $x(t_1, t_2)$ is rotated by and angle $\Delta\theta$ in the $(t_1,\ t_2)$ space, how does this change $X(F_1, F_2)$? In the simple case of a 90 degree rotation, the two indices are exchanged in both $x(t_1, t_2)$ and $X(F_1, F_2)$. Hence $X(F_1, F_2)$ is rotated by the same amount in the $(F_1,\ F_2)$ space as $x(t_1, t_2)$ in the (t_1, t_2) space. The same is easily seen to be true for 180 and 270 degree rotations. In order to see what happens for an arbitrary rotation $\Delta\theta$, let us transform $x(t_1, t_2)$ and $X(F_1, F_2)$ into polar coordinates, and define

$$t_1 \;=\; r\ \cos\ \theta, \qquad t_2 \;=\; r\ \sin\ \theta \tag{9.5}$$

$$F_1 \;=\; R\ \cos\ \phi, \qquad F_2 \;=\; R\ \sin\ \phi$$

Rewriting Equation (9.1a) in polar coordinates, we have

$$X(R, \phi) \;=\; \int_0^{\infty} \int_0^{2\pi} x(r, \theta)\; e^{-j2\pi\, rR\ \sin\theta\ \sin\phi}\; e^{-j2\pi\, rR\ \cos\theta\ \cos\phi}\; r\ dr\, d\theta \tag{9.6a}$$

Combining the trigonometric terms in the exponential, this expression simplifies to

$$X(R, \phi) \;=\; \int_0^{\infty} \int_0^{2\pi} x(r, \theta)\; e^{-j2\pi\, rRcos(\theta-\phi)}\; r\ dr\, d\theta \tag{9.6b}$$

If instead of taking the transform of $x(r, \theta)$, we took the transform of $x(r, \theta + \Delta\theta)$, we could still keep the right hand side of Equation (9.6b) unchanged as a function of R and ϕ if we substituted $\phi + \Delta\theta$ for ϕ. This means that the Fourier transforms $x(r, \theta)$ and $X(R, \phi)$ follow the rule

$$x(r, \theta + \Delta\theta) \quad \longleftrightarrow \quad X(R, \phi + \Delta\theta). \tag{9.7}$$

Regardless of the size of $\Delta\theta$, rotating $x(t_1, t_2)$ rotates $X(F_1, F_2)$ by the same amount. An example of a rotation is shown in Figure 4.

9.1.3 Radially symmetric signals

A 2-D signal $x(r, \theta)$ is said to be *radially symmetric* if it depends only on r, i.e. $x(r, \theta) \;=\; x_r(r)$, where $x_r(r)$ is a 1-D signal. The rotation theorem implies that $x(r, \theta)$ is radially symmetric if and only if its Fourier transform $X(R, \phi)$ is also radially symmetric:

$$x(r, \theta) \;=\; x_r(r) \quad \longleftrightarrow \quad X(R, \phi) \;=\; X_R(R) \tag{9.8a}$$

Note however, that the 1-D signals $x_r(R)$ and $X_R(R)$ do NOT form a Fourier transform pair. In fact, one has:

$$X_R(R) \;=\; \int_0^\infty \; x_r(r)\; J_0(2\pi r R)\; dr \tag{9.8b}$$

where $J_0(.)$ is the zeroth-order Bessel function of the first kind.

An important example of a radially symmetric signal is the ideal lowpass filter defined in the frequency domain by

$$H(R,\phi) \;\stackrel{\triangle}{=}\; \begin{cases} 1 & \text{if } R < W \\ 0 & \text{otherwise} \end{cases} \tag{9.9a}$$

The corresponding impulse response can be shown to be

$$h(r,\theta) \;=\; \frac{W}{r}\; J_1(2\pi r W) \tag{9.9b}$$

where $J_1(.)$ is the first-order Bessel function of the first kind. The function $J_1(r)/r$ resembles a *sinc* function, with the important difference that its zeroes do not occur at exactly regular intervals. Figure 5 shows a sketch of the function $J_1(r)/r$ and a perspective display of the impulse response of the ideal lowpass filter.

9.1.4 Projection-slice theorem

Consider what happens if we integrate $x(t_1,t_2)$ over t_2 to generate the 1-D projection

$$x_p(t_1) \;=\; \int_{-\infty}^{\infty} x(t_1,t_2)\; dt_2 \tag{9.10a}$$

and we then compute $X_p(F_1)$ as the 1-D Fourier transform of $x_p(t_1)$:

$$X_p(F_1) \;=\; \int_{-\infty}^{\infty} x_p(t_1)\; e^{-j2\pi\, t_1 F_1}\; dt_1 \;=\; \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1,t_2)\; e^{-j2\pi\, t_1 F_1}\; dt_1 dt_2 \tag{9.10b}$$

Comparing (9.10b) with (9.1a), we see that $X_p(F_1)$ is equal to $X(F_1,0)$. Hence $x_p(t_1)$ and $X(F_1,0)$ are Fourier transforms of each other.

The relationship between a projection and its Fourier transform can be easily generalized by application of the rotation theorem given by Equation (9.7). Since the Fourier transform of a rotated $x(t_1,t_2)$ is $X(F_1,F_2)$ rotated by the same amount, we can make a more general statement known as the *projection-slice theorem:*

> The Fourier transform of $x(t_1,t_2)$ projected onto a line that forms an angle θ_0 with respect to the horizontal axis $t_1 \;=\; 0$ is $X(R,\theta_0)$.

In the case of Equations (9.10a,b), the value of θ_0 is 0. The projection slice theorem is the basis for computerized tomography.

9.1.5 Magnitude and phase of images

It is common in 1-D signal processing to examine the magnitude and pay only passing attention to the phase of the Fourier transform. For typical 1-D signals, the structure of the phase is seldom as simple as that of the magnitude, so it cannot be characterized in some simple way. Yet the phase is critical for preserving the transitions in the level of the signal. For images, these transitions correspond to the boundaries between different elements in the image.

The information content of an image usually depends more critically on the preservation of edges and boundaries than on the absolute intensity level of a given region. For this reason, the phase information becomes all the more important. Figure 6 gives an example of two images which can be separated into magnitude and phase. Two new images were constructed by pairing the phase of the first image with the magnitude of the second, and the phase of the second with the magnitude of the first. In both of the composite images the phase dominates the information conveyed by the image.

9.2 The 2-D Discrete Space Fourier Transform

The Fourier transform pair for a 2-D discrete-space, stable signal $x[n_1, n_2]$ is given by

$$X(f_1, f_2) \;=\; \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x[n_1, n_2]\; e^{-j2\pi\, n_1 f_1}\; e^{-j2\pi\, n_2 f_2} \tag{9.11a}$$

and

$$x[n_1, n_2] \;=\; \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} X(f_1, f_2)\; e^{j2\pi\, n_1 f_1}\; e^{j2\pi\, n_2 f_2}\; df_1 df_2 \tag{9.11b}$$

$X(f_1, f_2)$ is periodic in both f_1 and f_2, with period 1 for both variables.

The 2-D transform pair satisfies relationships similar to its 1-D counterpart. If we define 2-D convolution as

$$x[n_1, n_2] \;**\; y[n_1, n_2] \;=\; \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x[k_1, k_2]\; y[n_1 - k_1, n_2 - k_2] \tag{9.12a}$$

then the discrete space Fourier transform (DSFT) pair satisfies the convolution theorem

$$x[n_1, n_2] \;**\; y[n_1, n_2] \;\longleftrightarrow\; X(f_1, f_2)\; Y(f_1, f_2) \tag{9.12b}$$

Similarly, if the 2-D cyclic convolution is defined by

$$X(f_1, f_2) \circledast\circledast Y(f_1, f_2) \;\triangleq\; \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} X(\phi_1, \phi_2) Y(f_1 - \phi_1, f_2 - \phi_2) d\phi_1 d\phi_2 \tag{9.13a}$$

the DSFT satisfies the product theorem

$$x[n_1, n_2] y[n_1, n_2] \longleftrightarrow X(f_1, f_2) \circledast\circledast Y(f_1, f_2) \tag{9.13b}$$

The transform pair also satisfies the initial value and DC value theorems

$$x[0,0] \;=\; \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} X(f_1,f_2)\; df_1 df_2 \tag{9.14a}$$

$$X(0,0) \;=\; \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} x[n_1,n_2] \tag{9.14b}$$

and Parseval's theorem

$$\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} |x[n_1,n_2]|^2 \;=\; \int_{-1/2}^{1/2}\int_{-1/2}^{1/2} |X(f_1,f_2)|^2\; df_1 df_2 \tag{9.15}$$

9.2.1 Sampling an image

Most 2-D discrete-space signals are obtained by sampling a continuous-space image. As in the 1-D case, the bandwidth has to be limited in order to avoid aliasing and subsequent loss of information. Specifically, assume that a continuous-space signal $x(t_1,t_2)$ is sampled at intervals of T_1 and T_2 in the horizontal and vertical dimensions respectively to form the discrete-space signal $x[n_1,n_2]$:

$$x[n_1,n_2] \;=\; x(n_1T_1, n_2T_2) \tag{9.16a}$$

The relationship between $X(f_1,f_2)$, the DSFT of $x[n_1,n_2]$, and $X(F_1,F_2)$, the CSFT of $x(t_1,t_2)$ is given by:

$$X(f_1,f_2) \;=\; \frac{1}{T_1T_2}\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} X\left(\frac{f_1-k_1}{T_1}, \frac{f_2-k_2}{T_2}\right) \tag{9.16b}$$

Thus, $X(f_1,f_2)$ is formed by repeating $X(F_1,F_2)$ indefinitely at intervals of $\frac{1}{T_1}$ and $\frac{1}{T_2}$ along the horizontal and vertical coordinates, respectively. In order to recover $X(F_1,F_2)$ from $X(f_1,f_2)$ (and therefore $x(t_1,t_2)$ from $x[n_1,n_2]$), the Nyquist condition must be satisfied for both coordinates: $W_1 < 1/2T_1$ and $W_2 < 1/2T_2$, where W_1 and W_2 are the bandwidths of $x(t_1,t_2)$ along the t_1 and t_2 dimensions respectively. If the Nyquist condition is verified, $x(t_1,t_2)$ can be derived from $x[n_1,n_2]$ by means of the 2-D interpolation formula:

$$x(t_1,t_2) \;=\; \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} x[n_1,n_2]\; \frac{\sin\,\frac{\pi}{T_1}(t_1-n_1T_1)}{\frac{\pi}{T_1}(t_1-n_1T_1)}\; \frac{\sin\,\frac{\pi}{T_2}(t_2-n_2T_2)}{\frac{\pi}{T_2}(t_2-n_2T_2)} \tag{9.17}$$

This formula is a straightforward extension of the 1-D case. Figure 7a shows an image that was digitized with an antialiasing lowpass filter. Figure 7b shows the noticeable aliasing which occurs when the antialiasing filter is omitted. In practice, many images have most of their energy at low frequencies, so that antialiasing filters are often unnecessary.

Even though it is, in principle, possible to recover a bandlimited 2-D continuous-space signal from its samples, certain properties of continuous-space signals do not hold for discrete-space signals. For example, we have seen that a continuous-space signal is radially symmetric if and only if its Fourier transform is radially symmetric. This is not true for discrete-space signals: A radially symmetric transform $X(f_1,f_2)$ implies that $x[n_1,n_2]$ is also radially symmetric, but the converse is not true, as shown by the simple counter examples of Equations (9.21) and (9.22). This result is significant because it is often desirable to design filters with radially symmetric

frequency responses. Choosing a discrete-space filter with a radially-symmetric impulse response (although necessary) will not suffice to ensure that the frequency response is radially symmetric. Another example of a property that holds for continuous, but not discrete images is the projection slice theorem. In this case, the difficulty is that the projection of a discrete image is not mathematically defined for every angle (i.e, one would have to interpolate the image to define the projection).

9.3 Convolution and 2-D Filters

A 2-D linear, shift-invariant (LSI) filter is a system that verifies the properties of superposition, scaling, and shift invariance. As in the 1-D case, a LSI filter is completely characterized by its unit-sample response $h[n_1, n_2]$. The 2-D unit sample $\delta[n_1, n_2]$ is defined by:

$$\delta[n_1, n_2] \triangleq \delta[n_1]\delta[n_2] = \begin{cases} 1 & \text{if } n_1 = n_2 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{9.18}$$

If $x[n_1, n_2]$ is passed through a filter with impulse response $h[n_1, n_2]$, the output $y[n_1, n_2]$ is given by

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h[k_1, k_2]\, x[n_1 - k_1, n_2 - k_2] = x[n_1, n_2] ** h[n_1, n_2] \tag{9.19}$$

In image processing, $h[n_1, n_2]$ is often referred to as the convolution *kernel*. While the unit-sample response of a linear system can be either finite or infinite, FIR discrete-space filters are by far the most important in practice.

In most applications the convolution kernel possesses a simple symmetry. Since the image is usually digitized and stored, non-causal zero-phase filters are simple to implement. Based on the discussion above, zero-phase filters are less likely to distort the features of the image, particularly the critical edges. A zero phase filter satisfies the condition

$$h[n_1, n_2] = h[-n_1, -n_2] \tag{9.20}$$

Figures 8 and 9 show examples of two simple zero-phase filters. The filter in Figure 8 has an impulse response given by

$$h[n_1, n_2] = \frac{1}{6}(2\delta[n_1, n_2] + \delta[n_1+1, n_2] + \delta[n_1-1, n_2] + \delta[n_1, n_2+1] + \delta[n_1, n_2-1]) \tag{9.21a}$$

Using Equation (9.11b), it is straightforward to show that the frequency response of this filter is

$$H(f_1, f_2) = \frac{1}{3}(1 + \cos\, 2\pi f_1 + \cos\, 2\pi f_2) \tag{9.21b}$$

This filter has unity gain at DC and has zeros for $|f_1| = |f_2| = 1/3$: It is a lowpass filter.

The filter in Figure 9 has impulse response is given by

$$h[n_1, n_2] = \frac{1}{4}(4\delta[n_1, n_2] - \delta[n_1 + 1, n_2] - \delta[n_1 - 1, n_2] - \delta[n_1, n_2 + 1] - \delta[n_1, n_2 - 1]) \tag{9.22a}$$

and has a frequency response given by

$$H(f_1, f_2) \;=\; \frac{1}{2}(2 \;-\; \cos\ 2\pi f_1 \;-\; \cos\ 2\pi f_2) \tag{9.22b}$$

The filter has zero gain at DC and rises to 2 as f_1 and f_2 both approach $\pm 1/2$: It is a highpass filter.

9.3.1 Separable filters

A filter is said to be separable if its impulse response can be written as the product of two 1-D impulse responses:

$$h[n_1, n_2] \;=\; h_1[n_1]\ h_2[n_2] \tag{9.23}$$

Separable 2-D filters are important because they can greatly reduce the computational burden of 2-D convolutions. For separable $h[n_1, n_2]$, Equation (9.19) can be written as

$$y[n_1, n_2] \;=\; \sum_{k_1=-\infty}^{\infty} h_1[k_1]\ (\ \sum_{k_2=-\infty}^{\infty} h_2[k_2]\ x[n_1 - k_1, n_2 - k_2]) \tag{9.24}$$

This formula reduces the 2-D convolution to a series of 1-D convolutions: Specifically, each row of the image is convolved with the 1-D filter $h_1[n_1]$, then each column is convolved with $h_2[n_2]$. (Alternatively, columns could be filtered first, followed by rows.) If $h[n_1, n_2]$ is an $N \times N$ function, the number of computations is reduced by a factor of $N/2$ compared to direct-form 2-D convolution (9.19).

9.4 The 2-D Discrete Fourier Transform

An $N \times N$ signal $x[n_1, n_2]$ is completely characterized by its $N \times N$ 2-D discrete Fourier transform $X[k_1, k_2]$, which is obtained by sampling the DSFT at intervals of $1/N$ along both frequency coordinates:

$$X[k_1, k_2] \;=\; \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x[n_1, n_2]\ e^{-j2\pi\, n_1 k_1/N}\ e^{-j2\pi\, n_2 k_2/N} \tag{9.25a}$$

$$x[n_1, n_2] \;=\; \frac{1}{N^2} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} X[k_1, k_2]\ e^{j2\pi\, n_1 k_1/N}\ e^{j2\pi\, n_2 k_2/N} \tag{9.25b}$$

The principal applications of the 2-D DFT are spectral analysis and the efficient implementation of convolutions. As in the 1-D case, the inverse DFT of the product of two DFT's gives the cyclic convolution rather than the linear convolution of the two signals:

$$x[n_1, n_2]\ \circledast\circledast_N\ h[n_1, n_2] \longleftrightarrow X[k_1, k_2]H[k_1, k_2] \tag{9.26a}$$

where

$$x[n_1, n_2]\ \circledast\circledast_N\ h[n_1, n_2] \stackrel{\triangle}{=} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \tilde{x}[k_1, k_2]\ \tilde{h}[n_1 - k_1, n_2 - k_2] \tag{9.26b}$$

and $\tilde{x}[n_1, n_2]$ and $\tilde{h}[n_1, n_2]$ are formed by repeating $x[n_1, n_2]$ and $h[n_1, n_2]$ respectively at intervals of N samples along both coordinates. If $x[n_1, n_2]$ is an $L \times L$ signal, and $h[n_1, n_2]$ is an $M \times M$ filter, it is necessary to use a DFT of length $N \geq L + M - 1$ for the cyclic convolution to give the same result as linear convolution. In practice, the differences between linear and cyclic convolutions are only visible near the borders of the filtered image, where there are usually few features of interest. For this reason, circular convolutions (without zero-padding) are often used in image processing.

The direct application of the DFT formula (9.25a) requires $N^2 \times N^2$ multiplications, which would be computationally prohibitive for all but the smallest images. Fortunately, it is possible to use 1-D fast Fourier algorithms to achieve considerable savings in the computation of 2-D DFT's. The key to such savings is the separability of the Fourier transform. The DFT formula (9.25a) can be rewritten as:

$$X[k_1, k_2] = \sum_{n_1=0}^{N-1} e^{-j2\pi n_1 k_1/N} \left(\sum_{n_2=0}^{N-1} x[n_1, n_2]\, e^{-j2\pi n_2 k_2/N} \right)$$

Let $z[n_1, k_2]$ be the expression inside the parentheses:

$$z[n_1, k_2] = \sum_{n_2=0}^{N-1} x[n_1, n_2]\, e^{-j2\pi n_2 k_2/N}$$

This represents the 1-D DFT of $x[n_1, n_2]$ (in which n_1 is a fixed parameter), i.e. the DFT of one column of the image. The 2-D DFT can now be written as

$$X[k_1, k_2] = \sum_{n_1=0}^{N-1} z[n_1, k_2]\, e^{-j2\pi n_1 k_1/N}$$

This is the 1-D DFT of $z[n_1, k_2]$, where now k_2 is a fixed parameter, i.e. the DFT of one row of the intermediate image $z[n_1, k_2]$.

Thus, an $N \times N$ DFT can be computed by the following sequence of operations:

1. Compute the 1-D DFT of each column in $x[n_1, n_2]$. This gives an intermediate image $z[n_1, k_2]$.
2. Compute the 1-D DFT of each row in $z[n_1, k_2]$, giving $X[k_1, k_2]$.

Obviously, it would also be possible to first transform each row, then each column of the intermediate image without changing the final result. If each 1-D DFT requires $N \log_2 N$ multiplications, either procedure will require a total of $2\, N^2 \log_2 N$ multiplications, which constitutes a decrease by a factor of $N^2/(2 \log_2 N)$ over the direct method (9.25a). For example, for a typical 512×512 image, the savings in computation would amount to a factor of nearly 15,000! This shows that fast Fourier algorithms are even more important in image processing than in 1-D signal processing.

9.5 2-D random signals

The theory of 2-D random signals is a straightforward extension of the 1-D case. The mean, or *space average* of a 2-D discrete random signal $x[n_1, n_2]$ is defined as:

$$< x[n_1, n_2] > \stackrel{\triangle}{=} \lim_{N\to\infty} \frac{1}{(2N+1)^2} \sum_{n_1=-N}^{N} \sum_{n_2=-N}^{N} x[n_1, n_2] \tag{9.27}$$

As in the 1-D case, this definition is only meaningful if $x[n_1, n_2]$ is stationary, i.e. if its statistical characteristics do not vary greatly over space. In practice, such infinite space averages are usually estimated from averages over a finite region of space. In contrast to 1-D signals, which are often zero-mean, 2-D signals usually have a non-zero mean because images take positive gray-scale values.

The 2-D autocorrelation is a function of two lag variables k_1 and k_2:

$$R_x[k_1, k_2] \stackrel{\triangle}{=} < x[n_1, n_2]\ x[n_1 + k_1, n_2 + k_2] > \tag{9.28a}$$

The autocorrelation function is symmetric with respect to the origin

$$R_x[-k_1, -k_2] \ = \ R_x[k_1, k_2], \tag{9.28b}$$

and maximum at the origin

$$R_x[k_1, k_2] \ \leq \ R_x[0, 0] \ = \ P_x \tag{9.28c}$$

The crosscorrelation function of two random signals $x[n_1, n_2]$ and $y[n_1, n_2]$ is

$$R_{xy}[k_1, k_2] \stackrel{\triangle}{=} < x[n_1, n_2]\ y[n_1 + k_1, n_2 + k_2] > = \ R_{yx}[-k_1, -k_2] \tag{9.29}$$

The power spectrum is the Fourier transform of the autocorrelation function

$$S_x(f_1, f_2) \stackrel{\triangle}{=} \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} R_x[k_1, k_2]\ e^{-j2\pi\, n_1 f_1}\ e^{-j2\pi\, n_2 f_2} \tag{9.30}$$

As in the 1-D case, it is always real and positive. The cross-spectrum $S_{xy}(f_1, f_2)$ is the Fourier transform of the crosscorrelation function.

If a random signal $x[n_1, n_2]$ is input to a linear filter $h[n_1, n_2]$, the output $y[n_1, n_2]$ verifies the following properties:

$$< y[n_1, n_2] > = \ H(0, 0)\ x[n_1, n_2] \tag{9.31a}$$

$$S_y(f_1, f_2) \ = \ |H(f_1, f_2)|^2\ S_x(f_1, f_2) \tag{9.31b}$$

$$S_{xy}(f_1, f_2) \ = \ H(f_1, f_2)\ S_x(f_1, f_2) \tag{9.31c}$$

The Wiener filter, which gives the linear, least-squares estimate of $y[n_1, n_2]$ given $x[n_1, n_2]$ is

$$H(f_1, f_2) \ = \ \frac{S_{xy}(f_1, f_2)}{S_x(f_1, f_2)} \tag{9.32}$$

9.6 Image enhancement

Image enhancement refers to image processing that makes an image more suitable for inspection by a human observer or automatic analysis by a digital computer. Image enhancement is usually needed when an image is captured under bad lighting conditions, or when specific aspects of the image, such as edges, need to be emphasized. Image enhancement techniques vary widely depending on the images that are being processed and the nature of the application.

9.6.1 Histogram modification

Perhaps the simplest enhancement technique is to change the distribution of the image brightness. Let us define $f_x(i)$ to be the probability that a given pixel $x[n_1, n_2]$ has a brightness corresponding to i, where i is normalized so that it ranges between 0 and 1. In practice, $f_x(i)$ is estimated empirically by computing a histogram of the individual pixels $x[n_1, n_2]$.

Figure 9.10a,b shows an image and its corresponding distribution $f_x(i)$. The image in Figure 10a has inadequate contrast, that is, all pixel values lie in a narrow range. The contrast can be improved by remapping the brightness levels of the original image to widen the distribution of intensities. Specifically, a new image $y[n_1, n_2]$ is formed such that

$$y[n_1, n_2] \;=\; g(x[n_1, n_2]) \tag{9.33}$$

where $g(.)$ is a monotonically increasing function applied to each pixel of the original image, shown in Figure 10c. Let us represent the distribution of the intensities in the remapped image by $f_y(j)$. Figure 10e shows $f_y(j)$ for a remapped version of the original image, and Figure 10d shows the enhanced image $y[n_1, n_2]$. Much more detail is visible in the enhanced image.

It is straightforward to show that the relationship between the new brightness level j and the original brightness level i is given by

$$j \;=\; g(i) \;=\; F_y^{-1}(\, F_x(i)\,), \tag{9.34}$$

where $F_y(j)$ and $F_x(i)$ are the cumulative probability density functions corresponding to $f_y(j)$ and $f_x(i)$, respectively:

$$F_x(i) \;=\; \sum_{k=0}^{i} f_x(k) \tag{9.35a}$$

and

$$F_y(j) \;=\; \sum_{k=0}^{j} f_y(k) \tag{9.35b}$$

In practice, for discrete-valued images, it is not in general possible to find a function $g(.)$ that will exactly map an input histogram $f_x(i)$ to a desired histogram $f_y(j)$, so that Equation (9.34) is only an approximation. In the special case when the enhanced image has a fairly uniform distribution (i.e. $f_y(j) \;\approx\; 1$), the approach is referred to as "histogram equalization". Histogram equalization makes maximal use of the eye's ability to distinguish between different gray levels.

9.6.2 Local Gray level modification

Sometimes an image may lack contrast, in spite of the fact that, on the whole, some parts of the image are much brighter than others. In such cases, the brightness distribution is sufficiently broad, so that global gray scale modification will not provide any real enhancement. The problem is not that the overall dynamic range of the brightness is too small, but that the local dynamic range in any particular region of the image is too small. The result is that the overall variation in brightness overshadows the local variations.

One solution is to adjust the intensity of each pixel based on the intensity of its neighbors. A simple approach is to change the value of a pixel $x[n_1, n_2]$ on the basis of the mean and variance of the brightness in the neighborhood of the pixel. If we choose a neighborhood of $\pm M$ pixels we can compute the mean and variance as

$$\mu[n_1,n_2] \quad = \quad \frac{1}{(2M+1)^2} \sum_{k_1=-M}^{M} \sum_{k_2=-M}^{M} x[k_1, k_2] \tag{9.36a}$$

$$\sigma^2[n_1,n_2] \quad = \quad \frac{1}{(2M+1)^2} \sum_{k_1=-M}^{M} \sum_{k_2=-M}^{M} (x[k_1, k_2] \; - \; \mu[k_1,k_2])^2 \tag{9.36b}$$

We can then create a new image $y[n_1, n_2]$ using the transformation given by

$$y[n_1, n_2] \quad = \quad \frac{A}{\sigma[n_1, n_2]}(x[n_1, n_2] \; - \; \mu[n_1,n_2]) \quad + \quad g(\mu[n_1,n_2]) \tag{9.37}$$

The first part of the transformation simply increases/decreases the deviation of $x[n_1, n_2]$ from the local mean depending on whether the local variance is low/high. This has the effect of making the local contrast more uniform. The constant A is chosen to provide the desired amount of contrast. The second half of the transformation represents a gray scale remapping of the local mean with the function $g(.)$. In the case of an image such as Figure 11a, the local mean is remapped to be more uniform over the entire image.

Other variations on this method exist. For example, instead of using the local mean and standard deviation, one can develop a related approach based on a lowpass filtered image (similar to the local mean) and a complementary highpass filtered image.

9.6.3 Image sharpening and softening

Certain kinds of image enhancement can be accomplished by the application of simple linear filters. For example, consider the situation when the image $x[n_1, n_2]$ can be approximately decomposed into low and high frequency components:

$$x[n_1, n_2] \quad = \quad x_L[n_1, n_2] \; + \; x_H[n_1, n_2] \tag{9.38}$$

where $x_L[n_1, n_2]$ represents the low frequency components of the image, while $x_H[n_1, n_2]$ represents the higher frequency components. As a generalization, $x_L[n_1, n_2]$ is usually associated with the smooth variations in the regional brightness of an image, whereas $x_H[n_1, n_2]$ is associated with the local contrast and actual details of the image.

Linear filters can be used to change the relative contributions of $x_L[n_1, n_2]$ and $x_H[n_1, n_2]$. A filter such as the one shown in Figure 8 reduces $x_H[n_1, n_2]$, and thereby reduces the apparent contrast of an image. A lowpass filter is a softening filter which makes an image appear more hazy. An example of the blurring due to lowpass filtering is shown in Figure 11. In contrast, a filter such as the one shown in Figure 9 removes $x_L[n_1, n_2]$ and increases $x_H[n_1, n_2]$. A filter which increases the relative contribution of $x_H[n_1, n_2]$ is a sharpening filter which emphasizes edges and makes an image appear more vibrant.

Figure 12 shows examples of highpass filtering in combination with histogram equalization. Figure 12a,b shows an original image of a chest X-ray and the image filtered with a highpass filter similar to the one shown in Figure 9. The filter removes the $x_L[n_1, n_2]$ component, eliminating much of the dark/light variation and emphasizing the higher-frequency features features such as the ribs. Figure 12c shows the result of processing similar to that of Figure 12b, except that the low frequency component $x_L[n_1, n_2]$ is not completely filtered out. Maintaining some $x_L[n_1, n_2]$ helps restore the larger features such as the heart and the arm. Figure 12d shows the image in Figure 12c after histogram equalization. The arm and heart are even more visible in the equalized image.

9.6.4 Homomorphic filtering

For many images, particularly those that are obtained photographically, the brightness of each pixel can be written in terms of the incident light intensity and a reflectivity coefficient

$$x[n_1, n_2] \quad = \quad i[n_1, n_2] \; r[n_1, n_2] \tag{9.39}$$

Here $i[n_1, n_2]$ represents the intensity of the light incident on the image object, and $r[n_1, n_2]$ represents the reflectivity of the object. The amount of light that would be recovered from the object is the product of the incident and reflected terms. Sometimes is desirable to eliminate the variation in the lighting intensity, either as a means of overcoming the poor lighting, or for compressing the dynamic range of the image. In such cases it is useful to operate on the logarithm of the image

$$\log x[n_1, n_2] \quad = \quad \log i[n_1, n_2] \; + \; \log r[n_1, n_2] \tag{9.40}$$

where the log function is evaluated individually for each pixel.

The use of this formalism assumes that the illumination component $i[n_1, n_2]$ varies slowly and is responsible for changes in the overall dynamic range of the image, while the reflectance component $r[n_1, n_2]$ varies more rapidly and is the responsible for local contrast. Under these conditions, $\log\ i[n_1, n_2]$ and $\log\ r[n_1, n_2]$ can be likened to $x_L[n_1, n_2]$ and $x_H[n_1, n_2]$, respectively. By selecting an appropriate sharpening filter which reduces $\log\ i[n_1, n_2]$, the filtered image can be computed from Equation (9.40) as

$$y[n_1, n_2] \quad = \quad e^{h[n_1,n_2] ** \log\ x[n_1,n_2]} \tag{9.41}$$

where the exponential is evaluated individually for each pixel of the 2-D convolution result. A system which performs a logarithmic operation, followed by a linear operation, followed by an exponentiation operation is referred to as a *homomorphic system for multiplication.*

Figure 13 shows both an original image and the image processed by a homomorphic filter. In practice, the filter $h[n_1, n_2]$ is chosen to have a gain less than 1 at frequencies corresponding primarily to $i[n_1, n_2]$ and a gain greater than 1 at frequencies corresponding primarily to $r[n_1, n_2]$.

9.6.5 Edge Detection

An *edge* in an image is a boundary or contour at which a significant change occurs in some property of the image, e.g. brightness, or texture. Edge detection consists in automatically detecting these boundaries in an image. It is often a first step in systems for image analysis and image recognition because edges contain important information for segmenting an image into different objects.

Techniques for edge detection can be based either on the gradient or the Laplacian of an image. The *gradient* is the generalization to 2 or more dimensions of the 1-D derivative of a signal. Specifically, the gradient of a 2-D, continuous image $x(t_1, t_2)$ is a vector defined by

$$\nabla x(t_1, t_2) \triangleq \begin{bmatrix} \frac{\partial x(t_1,t_2)}{\partial t_1} \\ \frac{\partial x(t_1,t_2)}{\partial t_2} \end{bmatrix} \tag{9.42a}$$

For edge detection, one is usually interested in the magnitude of the gradient:

$$|\nabla x(t_1, t_2)| \triangleq \sqrt{\left(\frac{\partial x(t_1, t_2)}{\partial t_1}\right)^2 + \left(\frac{\partial x(t_1, t_2)}{\partial t_2}\right)^2} \tag{9.42b}$$

Because the gradient emphasizes changes in the images, it is expected to show a local maximum at the edges of an image.

For discrete images, the gradient can be approximated in a number of ways. Perhaps the simplest approximation is:

$$\frac{\partial x(t_1, t_2)}{\partial t_1} \approx \frac{x[n_1 + 1, n_2] - x[n_1, n_2]}{T} \tag{9.43a}$$

and

$$\frac{\partial x(t_1, t_2)}{\partial t_2} \approx \frac{x[n_1, n_2 + 1] - x[n_1, n_2]}{T} \tag{9.43b}$$

where T is the sampling interval, assumed to be the same for both coordinates. In practice, T can be eliminated from the equations because, for edge detection, we are only interested in relative values of the gradient. The formulas (9.43ab) can be seen as a convolution of $x[n_1, n_2]$ with the two filters whose unit sample responses are shown in Figure 14ab. This simple approximation gives poor results for noisy images because it gives many spurious edges. In order to obtain smooth gradient estimates, it is preferable to use the 3x3 filters shown in Figure 14cd, which are known by the name of *Sobel's gradient.* To complete edge detection, the gradient magnitude is computed for every pixel, and then compared with a threshold. Pixels for which the gradient exceeds threshold are edge candidates. Figure 15 shows an example of edge detection based on Sobel's gradient approximation.

Another class of methods for edge detection is based on the Laplacian of the image. The *Laplacian,* which is a generalization of the second derivative, is defined by

$$\nabla^2 x(t_1, t_2) \triangleq \frac{\partial^2 x(t_1, t_2)}{\partial t_1^2} + \frac{\partial^2 x(t_1, t_2)}{\partial t_2^2} \tag{9.44}$$

At an edge, the gradient is maximum, and therefore the Laplacian, which is effectively a derivative of the gradient, is zero. Therefore, in edge detection, one seeks zero crossings of the Laplacian.

For discrete images, the second derivatives can be approximated by

$$\frac{\partial^2 x(t_1,t_2)}{\partial t_1^2} \;\approx\; x[n_1+1,n_2] \;-\; 2x[n_1,n_2] \;+\; x[n_1-1,n_2] \qquad (9.45a)$$

and

$$\frac{\partial^2 x(t_1,t_2)}{\partial t_2^2} \;\approx\; x[n_1,n_2+1] \;-\; 2x[n_1,n_2] \;+\; x[n_1,n_2-1] \qquad (9.45b)$$

Therefore, a reasonable Laplacian approximation is

$$\nabla^2 x(t_1,t_2) \;\approx\; x[n_1+1,n_2] \;+\; x[n_1-1,n_2] \;+\; x[n_1,n_2+1] \;+\; x[n_1,n_2-1] \;-\; 4x[n_1,n_2] \quad (9.46)$$

This operation is a convolution by the 3x3 kernel shown in Figure 14e (Note that this is the negative of the filter shown in Figure 9).

One problem with Laplacian-based methods for edge detection is that they generate many false edges in regions where the variance of the image is small. One way to circumvent this problem is to require that the local variance exceed a certain threshold before accepting a zero crossing of the Laplacian as an edge. The local variance can be estimated using Equation (9.36b).

Both gradient-based and Laplacian-based methods only provide candidate edges. In order to identify actual boundaries between objects, it is further necessary to connect edges that belong to the same contour, and to eliminate spurious edges. This *edge thinning* process, which requires a great deal of heuristics, is beyond the scope of these notes.

9.7 Image restoration

Although there is a substantial grey zone, a distinction is usually made between image enhancement and image restoration. Restoration refers to the elimination of some distortion which has degraded the image. The distortion can be as simple as additive or multiplicative noise, or as complex as stretching or dislocation of parts of the image.

9.7.1 Noise removal with linear filters and median filters

As in the 1-D case, linear filters are useful for separating 2-D signals from noise, particularly when the signal and the noise occupy different frequency bands. For examples, Figure 16a shows an image corrupted by additive 2-D sinusoidal noise. The Fourier transform magnitude of this noisy image is shown in Figure 16b. The bright dot at the center corresponds to the original image, while the symmetric dots on the diagonal represent the sinusoidal noise. A simple band-reject filter centered at the two symmetric dots, when applied to the image in Figure 16a yields the result in Figure 16c.

While linear filters are useful for deterministic, sinusoidal noise or for statistically uniform random noise, they do not work as well for impulsive noise known as salt-and-pepper noise. Salt-and-pepper noise is characterized by large spikes in isolated pixels. A typical example is random bit errors in a communication channel used to transmit images. Only a small fraction of the pixels in the image are affected, but the error in the affected pixels is often great. Median filters are well suited for this kind of noise.

A median filter picks the median value of a set of numbers. Whereas for a 1-D linear, rectangular (boxcar) filter of length N the result is the average of the N data points encompassed by the filter, for a 1-D median filter of length N the result is the median value of the N data points spanned by the filter. This principle is directly generalized to define a 2-D median filter.

Figure 17ab shows an original image and the same image contaminated by salt-and-pepper noise. Figure 17c shows the image filtered by a 5x5 linear boxcar filter. The linear filter does not do a very good job of restoring the image. Figure 17d shows the result of a 5x5 2-D median filter applied to the noisy image. The median filter introduces less blurring than the linear filter, while almost entirely eliminating the noise.

9.7.2 Reduction of image blur

In many situations the acquired image is a blurred version of the original image. The blurring can usually be modeled as convolution with a blurring function $b[n_1, n_2]$ whose Fourier transform is $B(f_1, f_2)$. This blurring function is sometimes referred to as the *point spread function.*

If $x[n_1, n_2]$ is the original image and $y[n_1, n_2]$ is the blurred image, then we have

$$y[n_1, n_2] = b[n_1, n_2] ** x[n_1, n_2], \tag{9.47a}$$

$$Y(f_1, f_2) = B(f_1, f_2)\, X(f_1, f_2). \tag{9.47b}$$

A simple solution for restoring $x[n_1, n_2]$ from $y[n_1, n_2]$ is to define the inverse filter

$$H(f_1, f_2) = \frac{1}{B(f_1, f_2)}, \tag{9.48}$$

which can then be used to filter $y[n_1, n_2]$. A problem with this approach is that $H(f_1, f_2)$ can take on extremely high values at points where $B(f_1, f_2)$ is close to zero. Even small amounts of noise (e.g quantization noise, or computation noise) can lead to huge errors in the reconstruction of $x[n_1, n_2]$. One way of avoiding this problem is to choose a suitable threshold γ and use the inverse filter given by

$$H(f_1, f_2) = \begin{cases} \frac{1}{B(f_1, f_2)} & \text{if} \frac{1}{|B(f_1, f_2)|} < \gamma \\ \gamma \frac{|B(f_1, f_2)|}{B(f_1, f_2)} & \text{otherwise} \end{cases} \tag{9.49}$$

The effect of Equation (9.49) is to cap the maximum value of $|H(f_1, f_2)|$ at γ.

Figure 18b shows an image blurred by a lowpass filter $b[n_1, n_2]$ with a low cutoff frequency. Figure 18cd show the image restored by inverse filtering with two different thresholds γ. While the value of γ in Figure 18c was chosen to exclude excessively small values of $B(f_1, f_2)$, this condition was not verified in Figure 18d, leading to a useless result. Figure 19 shows another example in which blurring due to uniform linear motion was eliminated by inverse filtering.

9.7.3 Wiener filtering

A very common situation is one in which the measured signal $x[n_1, n_2]$ consists of the true signal $y[n_1, n_2]$ contaminated with additive noise $\eta[n_1, n_2]$

$$x[n_1, n_2] = y[n_1, n_2] + \eta[n_1, n_2]. \tag{9.50}$$

We further assume that the noise is stationary, and uncorrelated with $y[n_1, n_2]$. The Wiener filter that gives the least-squares estimate of $y[n_1, n_2]$ from $x[n_1, n_2]$ is derived for this set of assumptions in the 1-D case. (See Chapter 12.) The result is identical in the 2-D case, and is given by

$$H(f_1, f_2) \;=\; \frac{S_{xy}(f_1, f_2)}{S_x(f_1, f_2)} \;=\; \frac{S_y(f_1, f_2)}{S_y(f_1, f_2) \;+\; S_\eta(f_1, f_2)} \tag{9.51}$$

In practice, $S_y(f_1, f_2)$ and $S_\eta(f_1, f_2)$ would be estimated either from a priori information, or from data giving the signal relatively free of noise and vice-versa.

9.7.4 Reduction of blurring and additive random noise

As we saw above, a simple inversion approach to the blurred image problem provides a poorly-behaved filter $H(f_1, f_2) \;=\; 1/B(f_1, f_2)$, especially in the presence of any amount of noise. In practice, we always encounter some kind of noise, even if it is only the quantization noise of image digitization. If we augment Equation (9.47a) to include an additive noise term $\eta[n_1, n_2]$, we get

$$x[n_1, n_2] \;=\; b[n_1, n_2] \;** y[n_1, n_2] \;+\; \eta[n_1, n_2] \tag{9.52}$$

where $\eta[n_1, n_2]$ is assumed to be uncorrelated with $y[n_1, n_2]$. The Wiener Filter for this more elaborate model can be easily derived using the results of Chapter 12. Let $z[n_1, n_2]$ be the result of the convolution $b[n_1, n_2] \;** y[n_1, n_2]$. Because $\eta[n_1, n_2]$ is uncorrelated with $y[n_1, n_2]$, and therefore with $z[n_1, n_2]$ which is derived linearly from $y[n_1, n_2]$, one has:

$$S_{xy}(f_1, f_2) \;=\; S_{zy}(f_1, f_2) \tag{9.53a}$$

and

$$S_x(f_1, f_2) \;=\; S_z(f_1, f_2) \;+\; S_\eta(f_1, f_2) \tag{9.53b}$$

Because $z[n_1, n_2]$ is derived from $y[n_1, n_2]$ by filtering through $b[n_1, n_2]$, on has, from (9.31):

$$S_{zy}(f_1, f_2) \;=\; S^*_{yz}(f_1, f_2) \;=\; B^*(f_1, f_2)\; S_y(f_1, f_2) \tag{9.54a}$$

where $B^*(f_1, f_2)$ is the complex conjugate of $B(f_1, f_2)$, and

$$S_z(f_1, f_2) \;=\; |B(f_1, f_2)|^2\; S_y(f_1, f_2) \tag{9.54b}$$

Thus, the Wiener restoration filter is

$$H(f_1, f_2) \;=\; \frac{S_{xy}(f_1, f_2)}{S_x(f_1, f_2)} \;=\; \frac{B^*(f_1, f_2)\; S_y(f_1, f_2)}{|B(f_1, f_2)|^2 S_y(f_1, f_2) \;+\; S_\eta(f_1, f_2)} \tag{9.55}$$

Three special cases are of interest: First, if $S_\eta(f_1, f_2) \;=\; 0$ (i.e. if there is no noise), $H(f_1, f_2)$ reduces to the inverse filter $1/B(f_1, f_2)$ in Equation (9.48). Second, if $B(f_1, f_2) \;=\; 1$ (i.e. if

there is no blur), $H(f_1, f_2)$ reduces to Equation (9.51) as expected. Third, if $|B(f_1, f_2)|$ becomes very small, the Wiener filter becomes:

$$H(f_1, f_2) \approx \frac{B^*(f_1, f_2)\ S_y(f_1, f_2)}{S_\eta(f_1, f_2)} \tag{9.56}$$

The value of this expression goes to zero rather than infinity, i.e. incorporating the noise $\eta[n_1, n_2]$ into the model automatically gives Equation (9.55) the desirable properties which needed to be artificially built into Equation (9.49).

Note that the Wiener filter can be rewritten as:

$$H(f_1, f_2) = \frac{1}{B(f_1, f_2)} \frac{S_z(f_1, f_2)}{S_z(f_1, f_2) + S_\eta(f_1, f_2)} \tag{9.57}$$

This can be seen as the cascade of the inverse filter $1/B(f_1, f_2)$ and a noise reduction filter for $z[n_1, n_2]$. The overall system is thus a cascade of a noise reduction filter and a deblurring filter.

Figures 20 and 21 show examples of Wiener filtering. Figure 20a shows three images that were both blurred and degraded by additive noise. Figure 20b show the corresponding Fourier spectra. The signal-to-noise ratio increases from top to bottom. Figure 20c shows these images processed by an inverse filter. The resulting images are completely masked by the noise. Figure 20d shows the three images processed by a Wiener filter as in (9.55). In this case, the dominos present in the original image are clearly visible, particularly for the bottom two images. Figure 21a shows an original image, and Figure 21b the image blurred and degraded by additive noise. Figure 21c shows the result of inverse filtering. Although the blur is removed, the noise is actually enhanced. Figure 21d shows the image processed by an approximation to the Wiener filter that removed the blur without excessively adding noise.

9.8 Further Reading

Lim, J.S., *Two-Dimensional Signal and Image Processing,* Prentice Hall, 1990 (Chapters 1, 8, 9).
Gonzalez, R.C., and Woods, R.E., *Digital Image Processing,* Addison-Wesley, 1993 (Chapters 3, 4, 5).

Photos removed due to copyright restrictions.
Five photos selected from Figs. 2.9, 2.10 and 2.11 in Gonzalez and Woods, 1993.

Figure 9.1: (From Gonzalez and Woods, pp. 35–37) Effects of spatial resolution and quantization. (a) 1024×1024 image displayed in 256 gray levels. (b) 128×128 image displayed in 256 gray levels. (c) 32×32 image displayed in 256 gray levels. (d) 1024×1024 image displayed in 8 gray levels. (e) 1024×1024 image displayed in 2 gray levels.

Figure removed due to copyright restrictions.
Fig 3.2 in Gonzalez and Woods, 1993.

Figure 9.2: (From Gonzalez and Woods, p. 85) (a) A 2-D rectagular (boxcar) filter and its 2-D Fourier transform displayed (b) in perspective and (c) as a gray-scale image.

Figure removed due to copyright restrictions.
Fig 3.3 in Gonzalez and Woods, 1993.

Figure 9.3: (From Gonzalez and Woods, p. 87) Three simple images (left) and their Fourier transform magnitudes (right).

Figure removed due to copyright restrictions.
Fig 3.10 in Gonzalez and Woods, 1993.

Figure 9.4: (From Gonzalez and Woods, p. 99) (a) A simple image, and (b) its Fourier transform magnitude. (c) Rotated image, and (d) its Fourier transform magnitude.

Figures removed due to copyright restrictions.
Fig 1.25 and 1.26 in Lim, 1990.

Figure 9.5: (From Lim, pp. 31-32) (a) The function $J_1(x)/x$, where $J_1(x)$ is the first order Bessel function of the first kind. (b) Impulse response of an ideal 2-D lowpass filter.

Figure removed due to copyright restrictions.
Fig 1.28 in Lim, 1990.

Figure 9.6: (From Lim, p. 35) (a) and (b) Two original images. (c) Image formed by combining the magnitude of the Fourier transform of (b) with the phase of the Fourier transform of (a). (d) Image formed by combining the magnitude of (a) with the phase of (b).

Figure removed due to copyright restrictions.
Fig 1.42 in Lim, 1990.

Figure 9.7: (From Lim, p. 48) (a) Image digitized with antialiasing lowpass filter. (b) Image with noticeable aliasing.

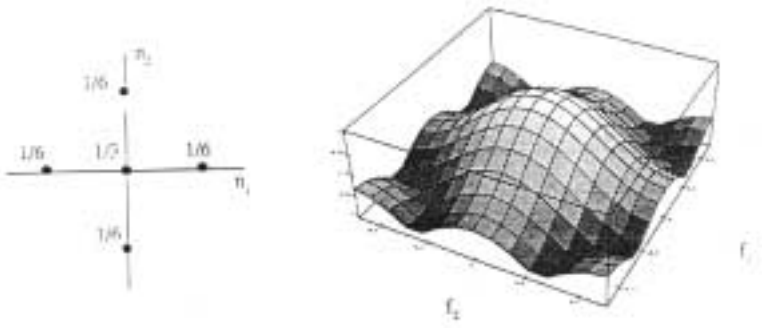

Figure 9.8: (a) Impulse-response of a simple 2-D digital lowpass filter. (b) Magnitude of the frequency response for the same filter.

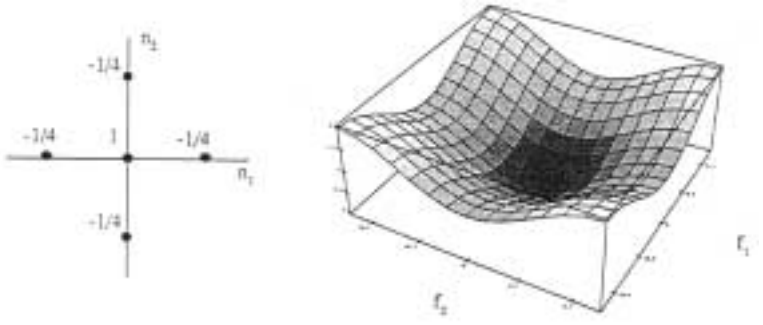

Figure 9.9: (a) Impulse-response of a simple 2-D digital highpass filter. (b) Magnitude of the frequency response for the same filter.

Figure removed due to copyright restrictions.
Fig 8.5 in Lim, 1990.

Figure 9.10: (From Lim, pp. 460–461) (a) An image with poor contrast. (b) Histogram of the gray-level distribution of the original image. (c) Transformation function used for gray-scale modification. (d) Image after gray-scale modification. (e) Histogram of the gray-level distribution after gray-scale modification.

Figure removed due to copyright restrictions.
Fig 4.22 in Gonzalez and Woods, 1993.

Figure 9.11: (From Gonzalez and Woods, p. 193) (a) Original image. (b)-(f) Results of lowpass filtering (blurring) the original image with a 2-D rectangular filter of size $N \times N$, where $N = 3, 5, 7, 15, 25$.

Figure removed due to copyright restrictions.
Fig 4.39 in Gonzalez and Woods, 1993.

Figure 9.12: (From Gonzalez and Woods, p. 214) (a) Original image. (b) Image sharpened by a highpass filter. (c) Image sharpened by a highpass filter that does not completely remove the DC component. (d) Image from (c) processed by histogram equalization.

Figure removed due to copyright restrictions.
Fig 4.42 in Gonzalez and Woods, 1993.

Figure 9.13: (From Gonzalez and Woods, p. 217) (a) Original image. (b) Image processed by homomorphic filtering.

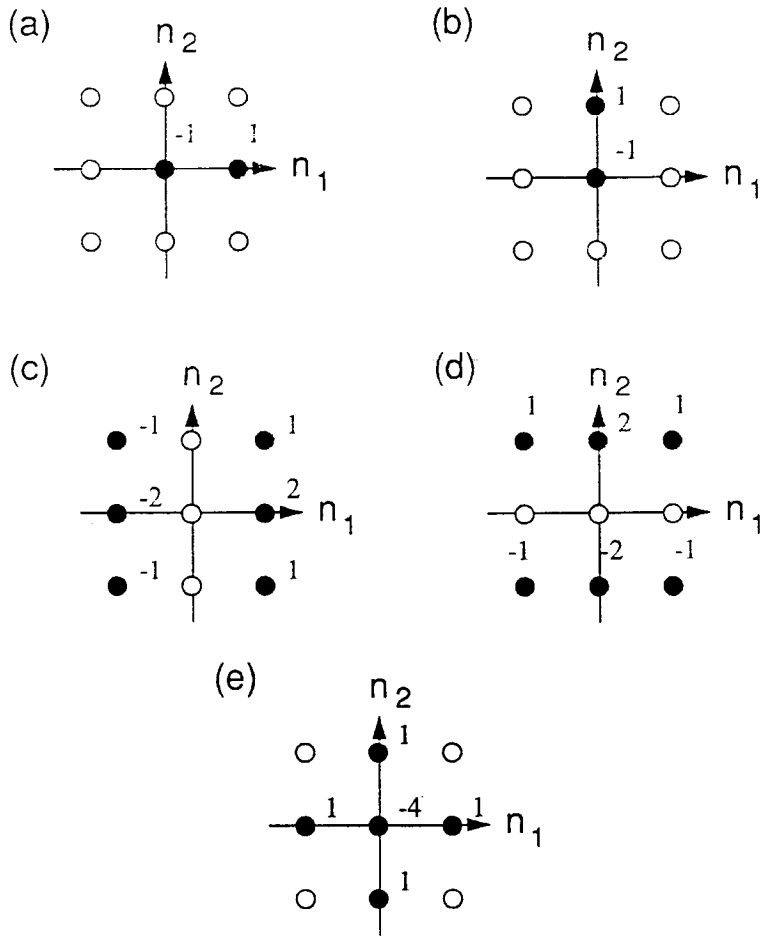

Figure 9.14: Filters used in edge detection. (a) Horizontal and (b) vertical components of a simple gradient approximation. (c) Horizontal and (d) vertical components of Sobel's gradient. (e) Laplacian.

Figures removed due to copyright restrictions.
Fig 8.30a and 8.31a in Lim, 1990.

Figure 9.15: (From Lim, pp. 484-485) (a) An image. (b) Result of applying Sobel's gradient-based edge detector to the image in (a).

Figure removed due to copyright restrictions.
Fig 5.7 in Gonzalez and Woods, 1993.

Figure 9.16: (From Gonzalez and Woods, p. 290) (a) Image corrupted by additive sinusoid and (b) the magnitude of its Fourier transform. (c) Image restored by linear bandstop filter.

Figure removed due to copyright restrictions.
Fig 4.23 in Gonzalez and Woods, 1993.

Figure 9.17: (From Gonzalez and Woods, p. 194) (a) Original image. (b) Image corrupted by additive salt-and-pepper noise. (c) Image processed by 5x5 rectangular boxcar filter. (d) Image processed by 5x5 median filter.

Figure removed due to copyright restrictions.
Fig 5.3 in Gonzalez and Woods, 1993.

Figure 9.18: (From Gonzalez and Woods, p. 273) (a) Original image. (b) Blurred image. (c) Image restored by capped inverse filter. (d) Image restored by inverse filter in which the cap was set higher than in (c).

Figure removed due to copyright restrictions.
Fig 5.4 in Gonzalez and Woods, 1993.

Figure 9.19: (From Gonzalez and Woods, p. 278) (a) Image blurred by uniform linear motion. (b) Image restored by inverse filtering.

Figure removed due to copyright restrictions.
Fig 5.5 in Gonzalez and Woods, 1993.

Figure 9.20: (From Gonzalez and Woods, p. 281) (a) Images blurred and degraded by additive noise and (b) the magnitudes of their Fourier transforms. (c) Images restored by inverse filtering. (d) Images restored by Weiner filtering, and (e) the magnitudes of their Fourier transforms.

Figure removed due to copyright restrictions.
Fig 5.6 in Gonzalez and Woods, 1993.

Figure 9.21: (From Gonzalez and Woods, p. 288) (a) Original image. (b) Image blurred and degraded by additive noise. (c) Image restored by inverse filtering. (d) Image restored by approximate Wiener filter.