CHAPTER

Discrete-Time Analysis for Images and Video

3

CHAPTER OUTLINE

Communicating Pictures. http://dx.doi.org/10.1016/B978-0-12-405906-1.00003-9

The first thing to emphasize about this chapter is that it is not intended to be a course in discrete-time signal analysis. Many excellent texts on this topic already exist covering both 1-D [1] and 2-D [2] signals. We assume here that the reader is familiar with the basics of sampling, linear systems analysis, digital filters, transforms, information theory, and statistical signal processing. The purpose of this chapter is to provide an overview of these important topics as they form the basis of many of the compression techniques described in this book.

We firstly review the sampling theorem and consider the cases of 2-D (image) and 2-D + time (video) sampling. This provides the discrete-time samples that represent our image or video and that are processed during the various stages of compression and coding. Next in Section 3.2, we introduce the means of describing signals using statistical representations. We focus on second order statistics and we use these to characterize the redundancy contained in an image or video signal. It is this redundancy that is exploited during the compression process.

Filters and transforms form the basic work-horses of most compression systems and these are introduced in Section 3.3. The signal distortions introduced in most compression systems can be attributed to coefficient quantization after filtering or transformation; this important aspect is covered in Section 3.4. Prediction operators are also in common usage across many aspects of compression, from intra-prediction to motion estimation. The basic concepts of linear prediction are introduced in Section 3.5 and we pay particular attention to the mitigation of decoder drift. This architecture is important as it provides a framework for all current video compression systems. Finally, as a precursor to Chapter 7, we review the basics of information theory; this topic is key to all symbol encoding methods in use today.

3.1 Signal and picture sampling

The manner in which a signal is sampled has a profound effect on subsequent processing steps. Particularly in the context of sensory signals such as images or video, the samples must provide a suitable and realistic representation of the underlying (continuous) natural scene, when viewed on a screen. Sampling must thus be performed in a manner that avoids the introduction of perceptual spatio-temporal artifacts due to aliasing. The sampling process addresses three primary questions:

1. What spatial and temporal sampling rates are needed to avoid aliasing and to produce the desired signal fidelity?
2. How should the samples be distributed in space and time?
3. What pre-processing is required prior to sampling and what post-processing is needed prior to reconstruction?

These issues are influenced by a number of factors including: the frequency content of the signal being sampled, the perceptual thresholds of the observer as related to the prevailing viewing conditions, the characteristics of the capture and display devices, and the costs associated with storage and transmission. This section attempts to answer these questions.

3.1.1 The sampling theorem

In one dimension

Consider a continuous (let us assume time domain) signal $x_a(t)$ sampled using a sequence of delta functions $\delta(t - nT)$ to yield:

$$x_s(t) = x_a(t)s(t) \tag{3.1}$$

where:

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

Using the Fourier Transform we can obtain the frequency domain representation of these signals, thus:

$$X_a(\omega) = \int_{-\infty}^{\infty} x_a(t)\mathrm{e}^{-j\omega t}\,\mathrm{d}t \tag{3.2}$$

and

$$S(\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_s) \tag{3.3}$$

Invoking the modulation property of the Fourier Transform gives:

$$X_s(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a(\omega - k\omega_s) \tag{3.4}$$

where ω_s represents the continuous frequency variable.

The spectrum of a sampled signal thus comprises scaled replicated versions of the original (continuous time) spectrum. In this context, Shannon's sampling theorem captures the requirements for sampling in order to avoid aliasing [3]:

> *If a signal $x_a(t)$ is bandlimited with $X_a(\omega) = 0$ for $|\omega| > \omega_m$, then $x_a(t)$ is uniquely determined by its samples provided that $\omega_s \geq 2\omega_m$. The original signal may then be completely recovered by passing $x_s(t)$ through an ideal low-pass filter.*

The spectral characteristics of the sampling process as described by Shannon's theorem are shown in Figure 3.1. Paying particular attention to the bottom two sub-figures, the penultimate sub-figure shows the case where perfect reconstruction is possible with an ideal reconstruction filter. In contrast, the bottom sub-figure shows the case where ω_m exceeds the frequency $\omega_s/2$ and aliasing distortion occurs. The frequency $\omega_s/2$ is often referred to as the Nyquist frequency.

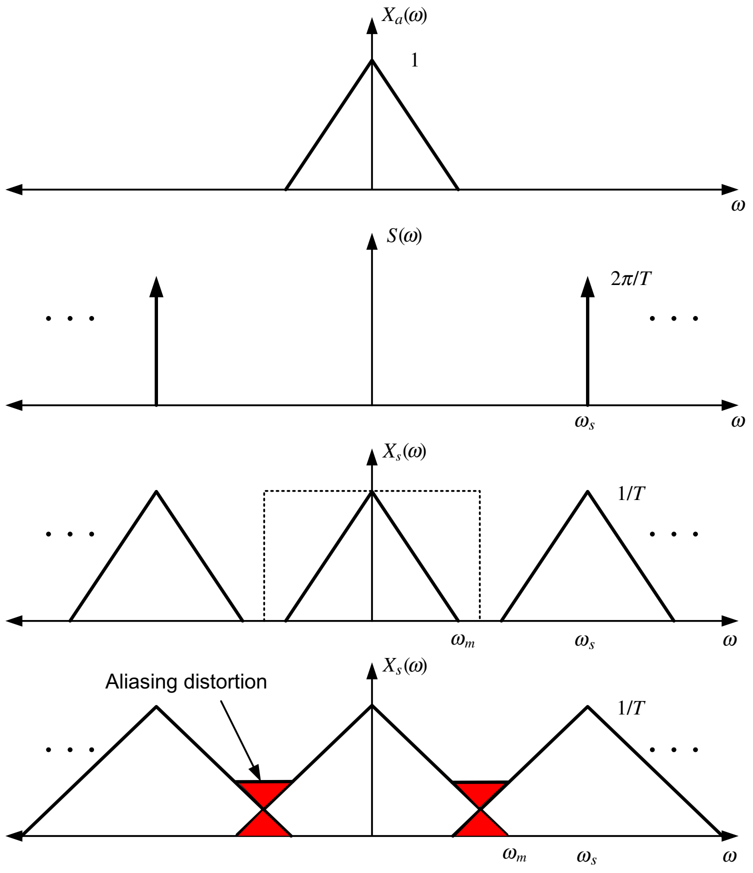


FIGURE 3.1
Spectral characteristics of sampling and aliasing.

Example 3.1 (1-D signal aliasing)
Consider a sinusoidal signal $x(t) = \cos(3\pi t/2)$ sampled at $\omega_s = \frac{4}{3}\omega_m$ with $T = 1$. Show that this sampling violates the sampling theorem and characterize the impact of the alias in the time and frequency domains.

Solution. The result is shown in Figure 3.2. The sinusoid $x(t) = \cos(3\pi t/2)$ is sampled at $\omega_s = \frac{4}{3}\omega_m$ with $T = 1$. As can be observed, an alias is created at a frequency of $\frac{\omega_s}{4}$ (to be accurate, this is actually an alias of the negative frequency component of the sinusoid at $-\omega_m$). This can be observed in both the signal domain plot (top) and the frequency domain plot (bottom). Here, as is conventional with discrete-time systems, we plot normalized frequency $\Omega = \omega T$, where the sampling is given by $\Omega = 2\pi$.

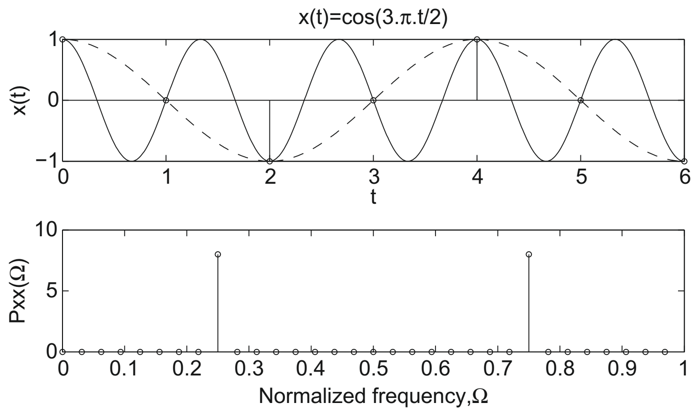


FIGURE 3.2

Demonstration of aliasing for a 1-D signal. Top: sinusoid sampled below Nyquist frequency. Bottom: Fourier plot showing spectral aliasing.

Extension to 2-D

Assuming a rectangular (orthogonal) sampling pattern in 2-D, we can extend our 1-D approach to the case of sampling of a 2-D signal. This gives:

$$X_s(\omega_1,\omega_2) = \frac{1}{T_1T_2}\sum_{k_1,k_2} X_a\left(\omega_1 - k_1\omega_s, \omega_2 - k_2\omega_s\right) \tag{3.5}$$

The effect of sampling and aliasing on the 2-D spectrum is shown in Figure 3.3. In 2-D images, aliasing errors will take the appearance of ringing in a direction perpendicular to high frequencies or sharp edges. It should be emphasized that while Figure 3.3 shows a circularly symmetric spectrum, in practice the spectrum could be skewed according to the prevailing horizontal and vertical frequencies or textures in the image.

Extension to 3-D

In the case of spatio-temporal sampling, it is conventional to sample at a fixed frame rate using either progressive or interlaced sampling (see Chapter 4). As we saw in Chapter 2, the visual sensitivities in the spatial and temporal dimensions are quite different. For example, a conventional 1080p/50 4:4:4 format will capture 103,680,000 samples per second. This however comprises 2,073,600 samples per picture but only 50 pictures per second. Thus there is a large difference in the number of samples per unit dimension between the spatial and temporal domains. It appears that the temporal domain gets a raw deal and to some extent this is true. We will come back to this in Chapter 13.

As described in Chapter 2, visual sensitivity depends on the mean brightness of the display. For a bright TV signal, it is generally considered, based on the temporal contrast sensitivity function and critical flicker frequencies, that a temporal update

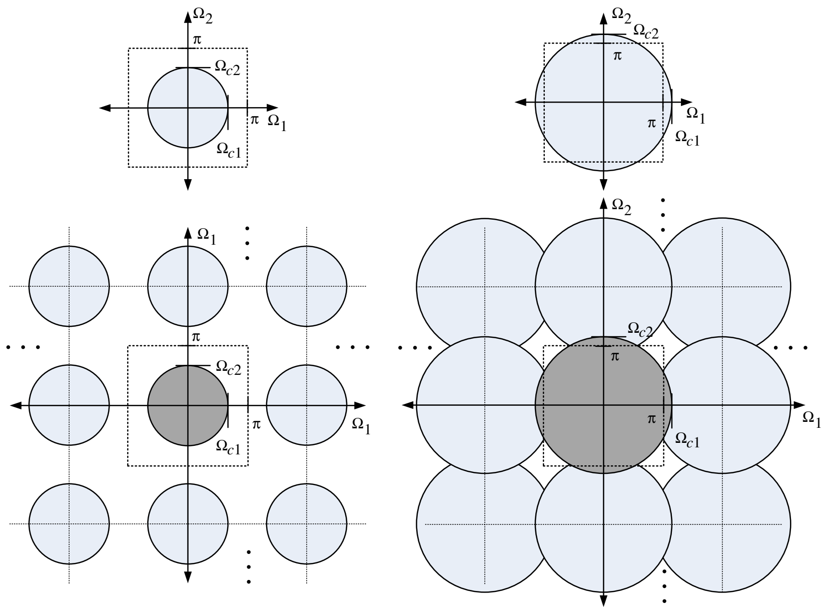


FIGURE 3.3

2-D spectral characteristics of sampling and aliasing. Left: Top—original signal spectrum; Bottom—sampled signal spectrum with no aliasing. Right: Top—original signal spectrum; Bottom—sampled signal spectrum with aliasing due to sub-Nyquist sampling.

rate of 50–70 Hz is sufficient, whereas for computer monitors which are viewed more closely in brighter environments, a higher update rate is normally specified.

Example 3.2 (How many spatial samples do we need?)

Consider the case of a 16:9 screen with a width of 1 m, viewed at a distance of $3H$ (where H is the height of the screen). How many horizontal samples are required to satisfy the contrast sensitivity limits of the human visual system?

Solution. For the case of a 16:9 screen with a width of 1 m, viewed at $3H$, we can compute the number of horizontal samples required from simple trigonometry. The height of the screen is 0.5625 m and the half viewing angle is then:

$$\theta = \tan^{-1}\left(\frac{0.5}{1.6875}\right)$$

Assuming a spatial resolution limit of 30 cpd (see Chapter 2), this yields 1980 horizontal samples per line, which is pretty close to the 1920 that are used in practice.

Although this calculation is useful, as screen sizes increase and viewing distances decrease, there will be a demand to take a fresh look at spatio-temporal sampling, but more of that later in Chapter 13.

3.1.2 Multidimensional sampling lattices

A sampling lattice, in real-valued K-dimensional space, is the set of all possible vectors that can be formed from linear combinations of a set of K linearly independent basis vectors, $\mathbf{v}_k \in \mathcal{R}^K$, $k = \{1, 2, 3, \ldots, K\}$. So the new vectors are formed from a weighted combination as follows:

$$\mathbf{w} = \sum_{k=1}^{K} c_k \mathbf{v}_k; \quad \forall c_k \in \mathcal{Z} \tag{3.6}$$

The matrix $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K]$ that combines the K basis vectors is called the *sampling matrix* or the *generating matrix*. For example, the generating matrix for our familiar rectangular lattice is:

$$\mathbf{V}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

However, we can explore more exotic sampling grids and their associated regions of support. Consider for example the generating matrix:

$$\mathbf{V}_2 = \begin{bmatrix} \sqrt{3}/2 & 0 \\ 1/2 & 1 \end{bmatrix} \tag{3.8}$$

The sampling lattice for this generating matrix is shown in Figure 3.4. As can be seen, this represents an hexagonal sampling grid. The lattice structure represents a simple and convenient tool for generating and analyzing sampling structures that are not necessarily hypercubic in structure. All the theorems for 1-D and 2-D sampling still apply and a generalized form of the Shannon sampling theory exists which dictates the structure and density of the sampling grid dependent on the signal spectrum. There is

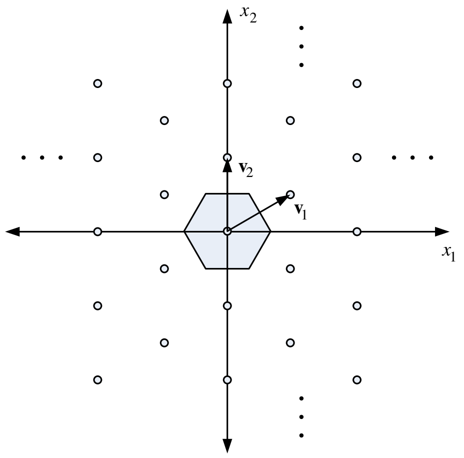


FIGURE 3.4
Hexagonal sampling lattice and its reciprocal as defined by equation (3.8).

a significant body of work relating to this more general approach to multidimensional sampling, which is relevant to image and video applications. The reader is referred to the work of Dubois [4] or Wang et al. [5] for excellent introductions to this topic.

3.2 Statistics of images

3.2.1 Histograms and distributions

A histogram is a discrete approximation of a probability density function based on the actual occurrences of quantized pixel values. It can provide important information about the distribution of sample amplitudes within the image, about the dynamic range of the signal as opposed to the wordlength of its representation, and about whether the image is under- or overexposed. An example histogram for the 256 × 256 image Stampe_SV4 is given in Figure 3.5.

Spatial and subband distributions

An understanding of spatial domain and frequency domain distributions can be of significant benefit in compression systems. For example, the Pyramid Vector Quantization method described in Chapter 11 exploits the Laplacian-like subband coefficient distribution while the Parametric Video Coding approaches outlined in Chapter 13 exploit the subband structure of Dual-Tree Complex Wavelet coefficients in providing segmentation into coherent texture regions.

3.2.2 Mean values

Mean and correlation provide a partial statistical characterization of a random process in terms of its averages and moments. They are useful as they offer tractable

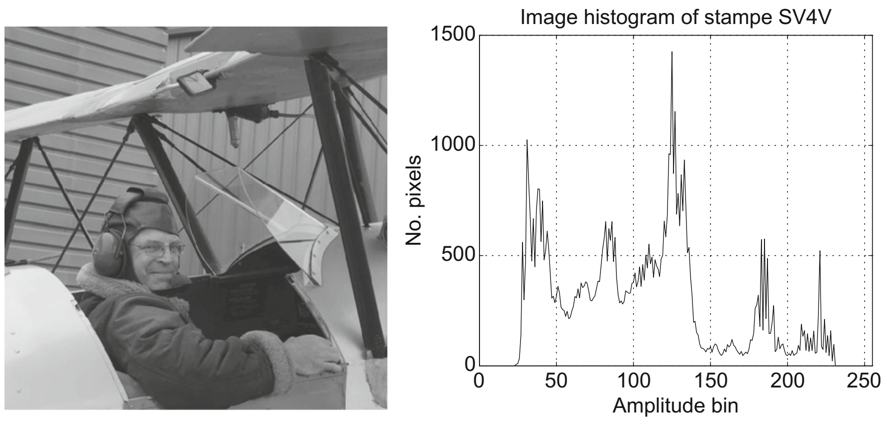


FIGURE 3.5

Example image histogram for 256 × 256 image Stampe_SV4.

mathematical analysis, they are amenable to experimental evaluation and they are well suited to the characterization of linear operations on random processes.

The mean for a stochastic process is given by:

$$\begin{aligned}\mu_n &= E\{x[n]\}\\ &= \int_{-\infty}^{\infty} x[n]f(x[n])\mathrm{d}x[n]\end{aligned} \tag{3.9}$$

where $f(\cdot)$ is the probability density function (pdf) of the process and $E\{\cdot\}$ is the expectation operator. In the case of a stationary process the pdf of the random variable is the same for all n and hence the mean is constant. Expectation can be interpreted as an average value obtained by repeating an experiment a number of times. Hence:

$$\mu_n = E\{x[n]\} = \lim_{N\to\infty}\left[\frac{1}{N}\sum_{i=1}^{N} x_i[n]\right] \tag{3.10}$$

3.2.3 Correlation in natural images

The autocorrelation (or autocovariance) of a sequence expresses the linear statistical dependencies between its samples. It is defined for a real-valued signal with a lag of m samples as:

$$R_{xx}[m] = E\left\{x[n]x[n+m]\right\} \tag{3.11}$$

where $x[n]$ is a stationary random process. In particular:

$$R_{xx}[0] = E\left\{|x[n]|^2\right\} \tag{3.12}$$

is the average power of the signal. In practice, the correlation is estimated based on a finite length sequence **x**, of N samples as follows:

$$R_{xx}[m] = \frac{1}{N}\sum_{n=0}^{N-|m|-1} x[n]x[n+m] \tag{3.13}$$

Equation (3.13) produces a biased estimate because of the attenuation for large lags. An unbiased estimate can also be used which is noisier, but preferable in certain situations:

$$R_{xx}[m] = \frac{1}{N-|m|}\sum_{n=0}^{N-|m|-1} x[n]x[n+m] \tag{3.14}$$

The autocovariance is computed in the same manner as the autocorrelation, but with the signal means removed. When the autocorrelation or autocovariance functions are normalized by their maximum value, they are generally referred to as autocorrelation coefficients or autocovariance coefficients respectively. These have values

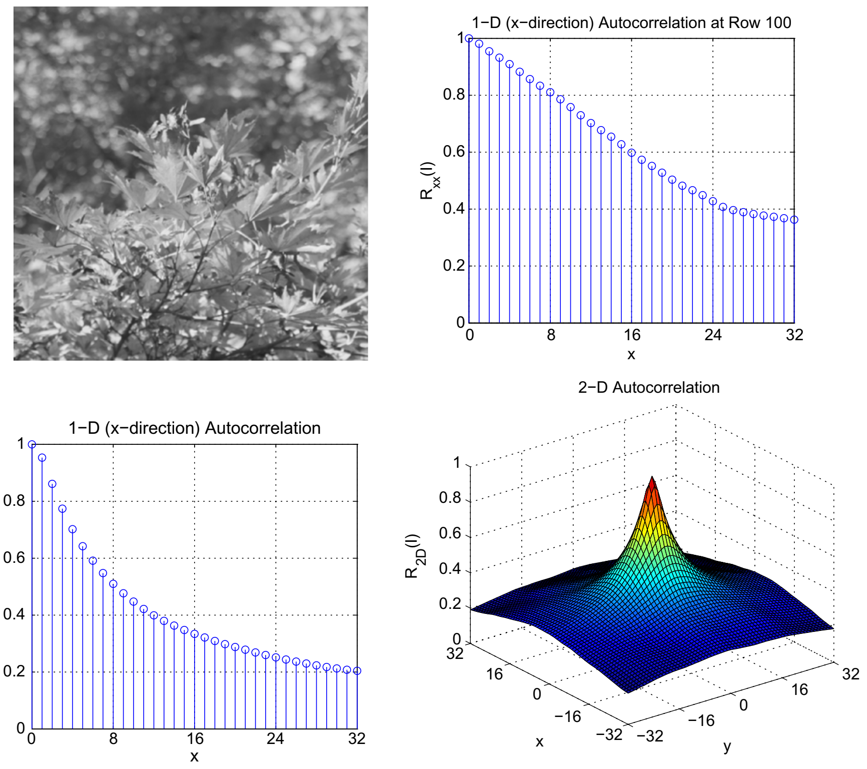


FIGURE 3.6

Autocorrelation plots for Acer image (512 × 512). Top left to bottom right: original image; autocorrelation function for row 100; autocorrelation function for whole image; 2-D autocorrelation surface.

between −1 and +1. For example, the autocorrelation coefficient is given by:[1]

$$\rho_x(k) = \frac{r_x(k)}{r_x(0)} \tag{3.15}$$

The autocorrelation matrix comprises autocorrelation values at various lags. The correlation matrix for a *wide sense stationary* (WSS) real-valued signal is the expectation of the outer product of the signal vector with itself. Thus:

$$\mathbf{R}_x = E\left\{\mathbf{x}\mathbf{x}^{\mathrm{T}}\right\} \tag{3.16}$$

[1] *Note:* We have dropped the double subscript as it is redundant and where no confusion results we will use lower case r to denote autocorrelation.

$$\mathbf{R}_x = \begin{bmatrix} r_x[0] & r_x[1] & \cdots & r_x[M-1] \\ r_x[1] & r_x[0] & & r_x[M-2] \\ \vdots & & & \vdots \\ r_x[M-1] & r_x[M-2] & \cdots & r_x[0] \end{bmatrix} \qquad (3.17)$$

The correlation matrix is Toeplitz and Hermitian. For the case of real-valued signals this means that it is transpose-invariant.

Spatial autocorrelation in natural images

Let us now consider the correlation properties of some images. Figures 3.6 and 3.7 show the autocorrelation properties of two 512×512 images—*Acer* and *Stampe_SV4*. The first has a highly textured, spatially active foreground with a blurred background and the second is highly structured with several plain areas. As well as the original

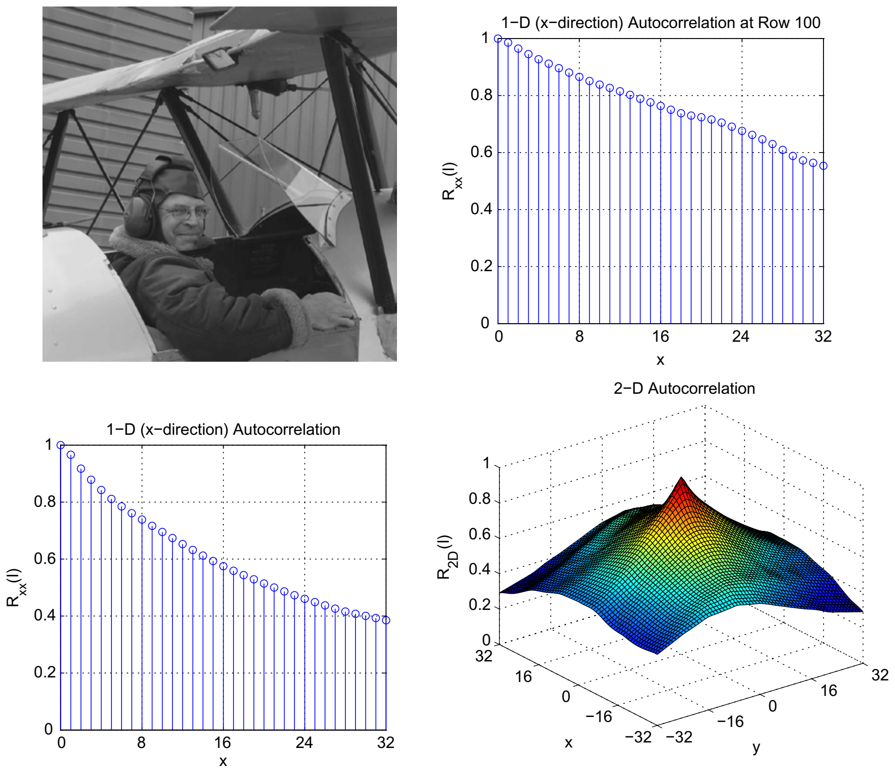


FIGURE 3.7

Autocorrelation plots for Stampe_SV4 image (512×512). Top left to bottom right: original image; autocorrelation function for row 100; autocorrelation function for whole image; 2-D autocorrelation surface.

image, each figure includes (i) an autocorrelation plot (unbiased with zero mean) for lags up to 32 pixels for a single row of pixels, (ii) a similar plot based on an average of all rows in the image, and (iii) a two-dimensional version. It can be observed that in both cases the correlation falls off gradually with increasing lag and that the values for the Acer image fall off more quickly that those for the Stampe_SV4 image. This is because of the structure in the latter compared to the more stochastic nature of the former. Figure 3.8 shows a similar series of plots, but this time for a 256×256 image, with only values up to a lag of 16 shown. As one would expect, the autocorrelation values fall off in a similar fashion to Figure 3.7 except that the corresponding lag values are halved.

Exploitation of this inter-sample correlation is the basis of prediction, transform, and filterbank compression methods. In each case, the purpose of the transform or predictor is to decorrelate the spatial information to make it more amenable to frequency-related perceptual quantization. As we can see from the figures, the autocorrelation

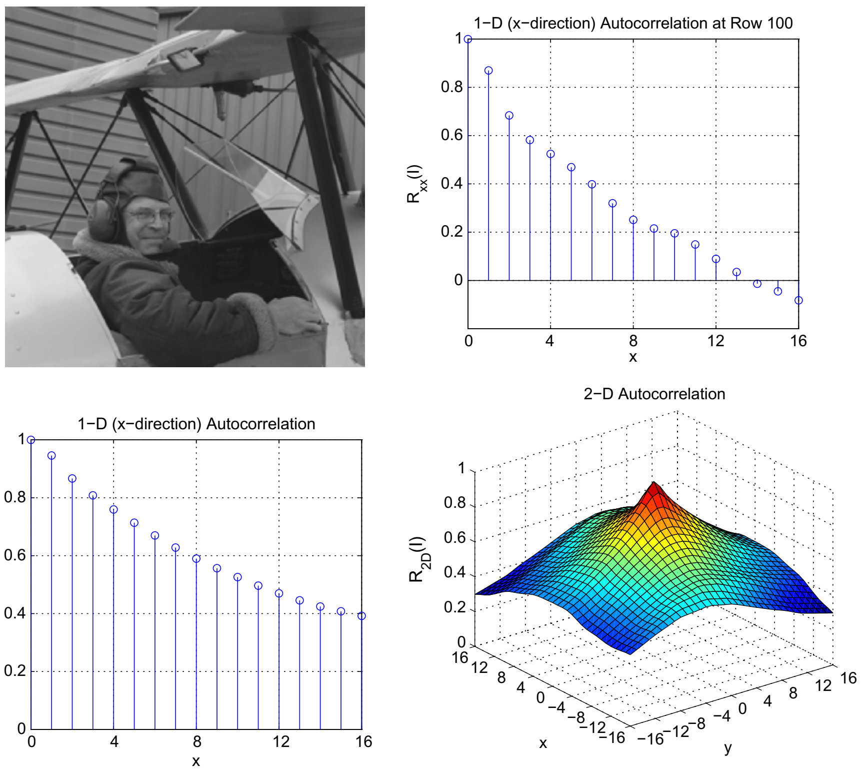


FIGURE 3.8

Autocorrelation plots for Stampe_SV4 image (256×256). Top left to bottom right: original image; autocorrelation function for row 100; autocorrelation function for whole image; 2-D autocorrelation surface.

values for large lags become increasingly small, while those for smaller lags are high. Because decorrelating transforms and predictors work better on regions with higher correlation, they are normally applied to small regions (typically 8 × 8 blocks) rather than to the whole image. An 8 × 8 block has been conventionally chosen as a good compromise between computational efficiency and coding gain. It is important to note, however, that the image resolution will have a significant impact on this—as lower resolution images will exhibit, like for like, lower correlations across the same size block.

Temporal autocorrelation in natural image sequences

Let us now consider the time axis. An example of temporal correlation, for the Foreman sequence, is shown in Figure 3.9. This indicates a similar characteristic to the spatial case considered above. Two plots are shown. The first is the time correlation for a single pixel computed using 300 frames of the Foreman sequence. The second is the same calculation, but this time averaged over the 16 × 16 block of pixels indicated in the top sub-figure. Again we can observe that significant temporal

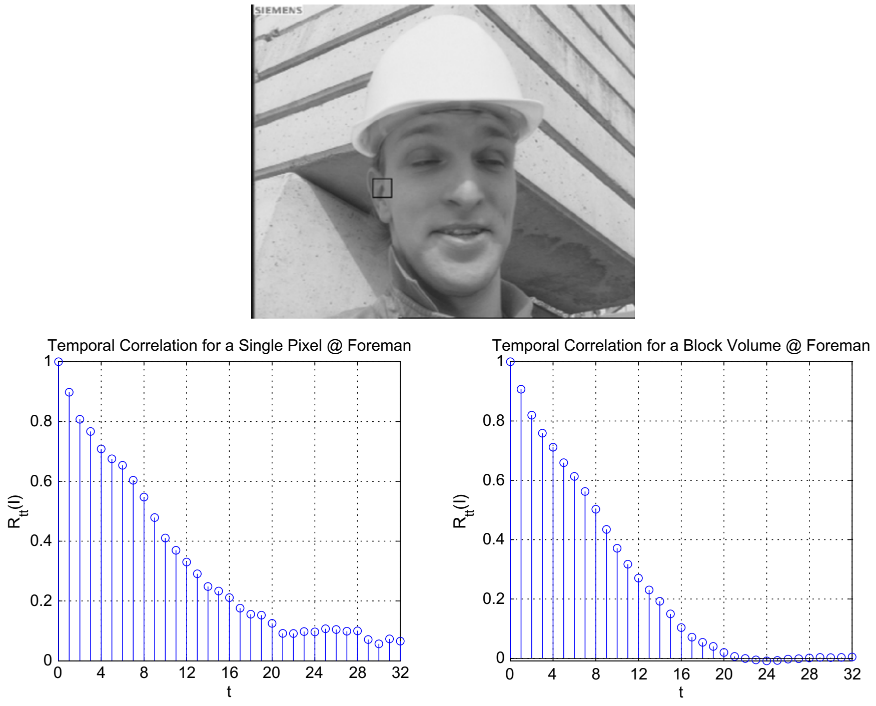


FIGURE 3.9

Temporal autocorrelation plots for Foreman (30 fps). Top to bottom right: sample frame showing selected 16 × 16 block used; temporal correlation for a single pixel; temporal correlation for a 16 × 16 block.

correlation exists between the adjacent video frames and indeed over 8 or so frames the correlation remains relatively high. This indicates that there is significant temporal redundancy present in most image sequences. As we will see later this is exploited in most compression methods, by applying motion compensated prediction, prior to transformation.

3.3 Filtering and transforms

We assume here that the reader is familiar with basic linear system theory, filters and transforms. For background reading, the reader is referred to Refs. [1,2,8,9].

3.3.1 Discrete-time linear systems

Many of the systems that we encounter in this book exhibit the properties of linearity and shift invariance.[2] Let us briefly describe these properties.

Shift invariance

A shift invariant system is one where a shift in the independent variable of the input signal causes a corresponding shift in the output signal. So if the response of a system to an input $x_0[n]$ is $y_0[n]$ then the response to an input $x_0[n-n_0]$ is $y_0[n-n_0]$.

Linearity

If the response of a system to an input $x_0[n]$ is $y_0[n]$ and the response to an input $x_1[n]$ is $y_1[n]$ then a system is referred to as linear if:

1. The response to $x_0[n]+x_1[n]$ is $y_0[n]+y_1[n]$ (additivity).
2. The response to $ax_0[n]$ is $ay_0[n]$, where a is any complex constant (scaling).

A corollary of this is the principle of superposition which states that if $x[n]=ax_0[n]+bx_1[n]$ then the output of a linear shift invariant system will be $y[n]=ay_0[n]+by_1[n]$.

3.3.2 Convolution

If the response of a linear time-invariant (LTI) system to a unit impulse, $\delta[n]$, is $h[n]$ then $h[n]$ is termed the *impulse response* of the system. The shift invariance, linearity and sifting properties of LTI systems give:

$$y[n]=\sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{3.18}$$

[2]Linear systems that are shift invariant are often referred to as linear time invariant (LTI). Although the independent variable is not always time, we will use this terminology here for convenience.

or:

$$y[n] = x[n] \star h[n]$$

This is known as the convolution sum and can be used to determine the response of a discrete-time LTI system to an arbitrary input sequence.

Example 3.3 (Convolution)
Using the convolution equation (3.18), determine the response of a linear system with impulse response $h[n] = \{1, 2, 1\}$ to an input $x[n] = \{1, 2, 0, 1, -1, 2\}$.

Solution. The convolution sum embodies the principle of superposition. Hence we can form the overall system output from the sum of the responses to each input sample as follows:

n	0	1	2	3	4	5	6	7
$y_0[n]$	1	2	1					
$y_1[n]$		2	4	2				
$y_2[n]$			0	0	0			
$y_3[n]$				1	2	1		
$y_4[n]$					-1	-2	-1	
$y_5[n]$						2	4	2
$y[n]$	1	4	5	3	1	1	3	2

3.3.3 Linear filters

The transfer function of a discrete-time linear system is given by:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}}{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}} \tag{3.19}$$

Since $x[n-k] \overset{Z}{\longleftrightarrow} z^{-k} X(z)$, the equivalent difference equation is (with b_0 scaled to unity):

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + \cdots + a_M x[n-M] \tag{3.20}$$

$$- b_1 y[n-1] - b_2 y[n-2] - \cdots - b_N y[n-N]$$

or:

$$y[n] = \sum_{k=0}^{M} a_k x[n-k] - \sum_{k=1}^{N} b_k y[n-k] \tag{3.21}$$

In cases where $\{b_k\} = 0$ then the filter is described as having a *finite impulse response* (FIR). In other cases it is known as *infinite impulse response* (IIR). FIR filters are much more common in image and video processing than their IIR counterparts. The reason for this is primarily due to their phase response. We saw in Chapter 2 that phase distortion has an important influence on perceptual quality, and it is straightforward to design FIR filters with a linear phase characteristic. Such a characteristic only introduces a simple shift in the filter response with no phase distortion. We will provide some examples of practical FIR filters that are used in video compression a little later, but first let us examine the filter frequency response in a little more detail.

Extension to 2-D

The 2-D convolution of a signal $x[m,n]$ with an impulse response $h[m,n]$ is given by:

$$y[m,n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x\left[m-k, n-1\right] h[k,l] = x[m,n] \star h[m,n] \tag{3.22}$$

2-D digital filters are used extensively in image and video coding, for example combined with sample-rate changing, in the analysis or synthesis stages of filter banks (Chapter 6), in interpolation filters for sub-pixel motion estimation (Chapters 8 and 9) or in more general pre- and post-processing operations. However, for reasons of flexibility and complexity these are, wherever possible, implemented separably; i.e. as a cascade of 1-D filtering stages.

Separability

Separability is an important property of digital filters and transforms that is exploited extensively in image and video processing. A system is separable when (for real-valued coefficients):

$$\mathbf{h} = \mathbf{h}_1 \mathbf{h}_2^{\mathrm{T}} \tag{3.23}$$

where $h_1[m]$ is a 1-D impulse response that operates across the rows of the signal and $h_2[n]$ is the corresponding impulse response that operates down the columns. We can now rewrite equation (3.22) as follows:

$$y[m,n] = \sum_{k=-\infty}^{\infty} h_1[k] \sum_{l=-\infty}^{\infty} x\left[m-k, n-1\right] h_2[l] \tag{3.24}$$

If we define:

$$y_2[m,n] = \sum_{l=-\infty}^{\infty} x\left[m, n-1\right] h_2[l]$$

then we can rewrite equation (3.24) as:

$$y[m,n] = \sum_{k=-\infty}^{\infty} y_2\left[m-k, n\right] h_1[k] \tag{3.25}$$

So this implies that if equation (3.23) holds, then we can achieve 2-D filtering by first filtering the columns of the 2-D signal with $h_2[n]$, followed by filtering the rows of the output from the first stage using $h_1[m]$. Due to the fact we are cascading linear operations, these two stages can be interchanged if necessary.

It should be noted that not all filters are separable and that separability does impose some orientation constraints on the filter characteristic. In general, a filter is separable if all its rows and all its columns are linearly dependent. This property can be observed in the solution of Example 3.4.

Example 3.4 (Filter separability)
Consider the two filter impulse responses below:

$$\mathbf{h}_1 = \begin{bmatrix} 0.5 & 0.3 & 0.2 \end{bmatrix}^{\mathrm{T}}$$
$$\mathbf{h}_2 = \begin{bmatrix} 0.6 & 0.1 & 0.3 \end{bmatrix}^{\mathrm{T}}$$

Form the equivalent 2-D digital filter.

Solution. The 2-D filter is given by:

$$\mathbf{h} = \mathbf{h}_1\mathbf{h}_2^{\mathrm{T}} = \begin{bmatrix} 0.30 & 0.05 & 0.15 \\ 0.18 & 0.03 & 0.09 \\ 0.12 & 0.02 & 0.06 \end{bmatrix}$$

As a further exercise, satisfy yourself that the 2-D filtering operation is identical to the use of separable filters.

3.3.4 Filter frequency response

From our understanding of convolution and digital filtering, we know that the response of a discrete linear system to a sampled sinusoidal input signal will be a number of weighted and shifted versions of the input signal. The output is thus also a sinusoidal signal at the same frequency, but with different amplitude and phase. We can therefore think of our filter in the frequency domain as a frequency-dependent modifier of amplitude and phase. If we generalize this with an input signal that is a complex exponential:

$$x[n] = \mathrm{e}^{j\Omega n}$$

then from the convolution equation (3.18) we have:

$$y[n] = \sum_k h[k]\mathrm{e}^{j\Omega(n-k)} = \mathrm{e}^{j\Omega n}\sum_k h[k]\mathrm{e}^{-j\Omega k}$$

Thus:

$$y[n] = x[n]H(\Omega)$$

where:

$$H(\Omega) = \sum_{k=-\infty}^{\infty} h[k]\mathrm{e}^{-j\Omega k} \tag{3.26}$$

The term $H(\Omega)$ describes the change in amplitude and phase experienced by the complex exponential as it passes through the system and it is referred to as the *frequency response* of the system.

3.3.5 Examples of practical filters

LeGall wavelet analysis filters

A common set of separable filters, that are employed to analyze a signal into low- and high-pass subbands prior to sub-sampling and quantization, are the 5/3 LeGall filters. The impulse responses of the low- and high-pass filters are:

$$\mathbf{h}_0 = [-1, 2, 6, 2, -1]/4$$

$$\mathbf{g}_0 = [1, -2, 1]/4$$

The frequency responses of these filters are shown in Figure 3.10. It is interesting to note that the low-pass and high-pass filters are not symmetrical and that there appears to be significant spectral leakage between them. However, as we will see in Chapter 6, these filters do perform well and offer the property of perfect reconstruction when combined in a filterbank.

H.264/AVC half-pixel interpolation filter

A second important filtering operation is interpolation, for example where a region of a picture is upsampled prior to performing sub-pixel motion estimation (see Chapters 8 and 9). In the currently deployed video coding standard, H.264/AVC, a six-tap FIR

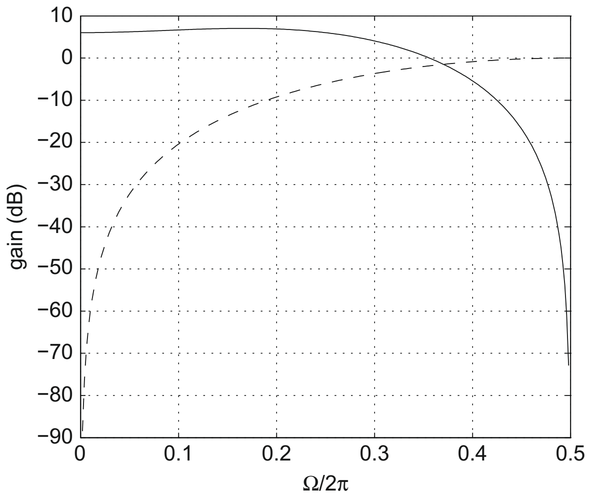


FIGURE 3.10

Filterbank responses for the LeGall low-pass and high-pass analysis filters.

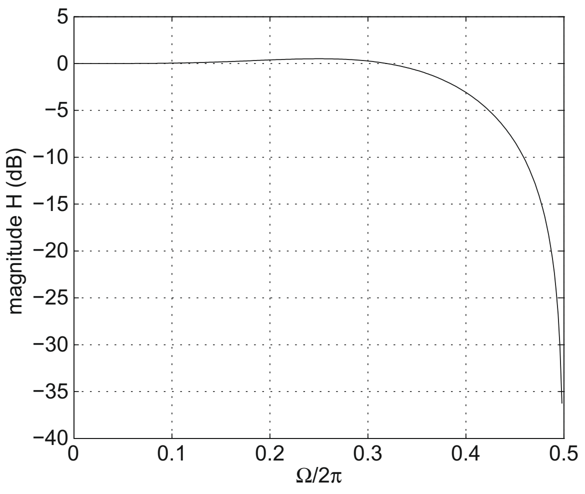


FIGURE 3.11

Filter response for H.264 half-pixel interpolation filter.

filter is employed:

$$\mathbf{h} = [1, -5, 20, 20, -5, 1]/32 \tag{3.27}$$

The frequency response of this filter is shown in Figure 3.11. It can be observed that the characteristic is not dissimilar to that of the LeGall low-pass analysis filter. However, in this case, the filter has no effect on the original integer-pixel resolution samples, it is simply used to generate the interpolated values. The filter characteristic is therefore a trade-off between obtaining a high cut-off frequency with rapid roll-off and low-pass band ripple. Such trade-offs are common in filter design [1].

3.3.6 Non-linear filters

Non-linear filters are not based on linear relationships between an input and an output via a system function. Instead they represent a much broader class of operations that do not have any explicit frequency domain transfer function. Non-linear filters find application in a range of image processing and coding applications such as: denoising, edge preserving operations and in some forms of prediction. For example, the deblocking filters used in the HEVC and H.264/AVC compression standards are content-adaptive non-linear filters as described in Chapter 9.

A common class of filter that is used in some motion vector prediction applications is the rank-order filter, in particular the median filter. We introduce these briefly below.

Rank-order and median filters

A rank-order filter is an operator that ranks the input samples and selects an output according to its position in the ranked list. The following are examples of rank-order filters:

$$y[m,n] = \min_{\mathcal{N}(m,n)} (x(i,j)) \tag{3.28}$$

$$y[m,n] = \max_{\mathcal{N}(m,n)} (x(i,j)) \tag{3.29}$$

where $\mathcal{N}$ represents the neighborhood, related to location (m,n) over which the operation is applied. The min operator will attenuate isolated peaks or ridges in an image, whereas the max operator will fill isolated troughs or holes. It can be seen that repeated application of this type of filter will have a continued effect of eroding the isolated regions described.

The median operator will rank all the N samples within the region and output the ranked value in position[3] $(N+1)/2$, thus:

$$y[m,n] = \operatorname*{med}_{\mathcal{N}(m,n)} (x(i,j)) \tag{3.30}$$

The median operator is better than a linear filter in removing outliers (e.g. salt and pepper noise). It will preserve straight edges, but will round corners and distort texture features. The extent of feature flattening will depend on the window size used.

We can of course combine these operations to produce non-linear difference filters such as that below:

$$y[m,n] = \min_{\mathcal{N}(m,n)} (x(i,j)) - \max_{\mathcal{N}(m,n)} (x(i,j)) \tag{3.31}$$

which produces a non-linear gradient of the image.

Morphological filters

The erosion and dilation operations referred to above can be generalized and combined to form a powerful toolbox of operators known as morphological filters. These shape-based operators support region erosion and dilation and geodesic reconstruction and have found applications in segmentation [6], contrast enhancement and in feature prioritization for low bit rate video coding [7].

3.3.7 Linear transforms and the DFT

Linear transforms, in particular the *Discrete Cosine Transform* (DCT), form the basis of most image intra-frame and residual coding and are covered in detail in Chapter 5. We provide an introduction to these here, focusing on the *Discrete Fourier Transform* (DFT). Fourier Transforms are used extensively across signal processing applications to perform spectral analysis based on discrete representations in both the signal domain and its associated frequency domain. An excellent introduction to Fourier Transforms is provided in Ref. [8].

[3]*Note:* Median filters are normally defined with an odd number of taps. In cases where the number of samples is even, it is conventional to take the average of the middle two terms.

While the Discrete Time Fourier Series and the Discrete Time Fourier Transform provide a basis for analyzing periodic and aperiodic signals respectively, the latter produces a continuous frequency domain function so is not well suited to digital implementation. The ubiquitous Discrete Fourier Transform overcomes this and enables discrete-time processing in both the signal and its associated frequency domain.

The Discrete Fourier Transform

Let us consider the 1-D version of the DFT initially. If $x[n]$ is a finite duration signal defined over the range $0 \leq n \leq N_1 - 1$, a periodic signal $\tilde{x}[n]$ can be constructed which equals $x[n]$ over one cycle. Let $N \geq N_1$ and let $\tilde{x}[n]$ be periodic in N. If we compute the Discrete Time Fourier Series (DTFS) of this new sequence:

$$c_k = \frac{1}{N} \sum_{n=\langle N \rangle} \tilde{x}[n] \mathrm{e}^{-jk(2\pi/N)n}$$

Letting $X(k) = N \cdot c_k$, this becomes the analysis equation of the DFT:

$$X(k) = \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-jk(2\pi/N)n}; \quad k = 0, 1, 2, \ldots, N-1 \tag{3.32}$$

The inverse DFT is given by:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \mathrm{e}^{jk(2\pi/N)n}; \quad n = 0, 1, 2, \ldots, N-1 \tag{3.33}$$

The N terms of the DFT represent samples of the continuous function $X(\Omega)$, equally spaced over a 2π interval. These can be easily shown to equal the samples of the z-transform on the unit circle in the z-plane.

Normally we use the DFT in matrix vector form:

$$\mathbf{X} = \mathbf{W}\mathbf{x} \tag{3.34}$$

where

$$W_N = \mathrm{e}^{-j(2\pi/N)}$$

Hence we have in matrix form:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & & W^{2(N-1)} \\ \vdots & & & & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix} \tag{3.35}$$

Similarly the inverse DFT is given by:

$$\mathbf{x} = \frac{1}{N} \mathbf{W}^* \mathbf{X} \tag{3.36}$$

The 2-D DFT

In a similar manner to that for filters in Section 3.3.3, transform operations can be separable and this is the conventional way of applying 2-D transforms, largely because of the associated complexity savings. We cover this in more detail in Chapter 5.

The DFT and compression

It may seem logical to attempt to exploit the ability of the DFT to analyze a signal in terms of its frequency domain components. There are however a number of significant issues with this which mean that this is not done in practice. These are:

1. Although images are invariably real-valued, the DFT coefficients are complex-valued so we are doubling the amount of information.
2. For short sequences or small 2-D regions, like those typical in compression, the discontinuities produced by the underlying assumptions of periodicity can have a major impact on performance as they produce ringing or spectral leakage in the frequency domain.
3. Ringing is normally addressed in spectral analysis applications by applying a non-rectangular window function (such as a Hamming window) to the input data prior to extension. These reduce spectral ringing but smear sharp frequency domain edges, distorting the characteristics of the underlying signal. Again, this is exactly what we don't want in a compression system!

Comparisons between the DFT and more appropriate transforms such as the DCT are made in Chapter 5.

3.4 Quantization

3.4.1 Basic theory of quantization

A quantizer maps an input value (or values in the context of a vector quantizer) to an output value. It can either take a continuous amplitude discrete-time signal and produce a quantized digital signal (with finite wordlength) or it can take a previously digitized sample and quantize it further to reduce its dynamic range. The former is common in acquisition devices such as A to D converters and the latter in compression systems where, for example, the coefficients of a transformed input signal might be mapped to a different range, with small values going to zero.

Once an image or video signal has been sampled to produce a discrete set of values with a given wordlength, the compression process normally involves some form of signal analysis (typically transformation) to decorrelate the content, followed by quantization to approximate the original signal in a way that retains as much perceptual quality as possible. Ignoring numerical imprecision, quantization is the stage in any compression process where loss is introduced. It is thus crucial to understand the impact of quantization on the perceptual quality of an image or video signal.

Uniform quantization

Uniform quantizers adopt a constant step size $\triangle$, between reconstruction levels. If the input to the quantizer is x, this will map to a reconstruction value, y_i, where y_i represents the center of the range as dictated by the step size parameter. We define a quantization error, $q[n] = x[n] - y[n]$, which is modeled as a zero mean, uniformly distributed signal within the range $[-\triangle/2, \triangle/2]$. This signal has energy or variance given by:

$$\sigma_q^2 = \frac{1}{\triangle}\int_{-\triangle/2}^{\triangle/2} q^2 \mathrm{d}q = \frac{\triangle^2}{12} \tag{3.37}$$

Two common uniform quantizer profiles are shown in Figure 3.12. On the left is the conventional midtread quantizer for positive valued signals, and on the right is a similar profile, but for two-sided input signals. Decision levels for x are indicated along with the corresponding reconstruction values for y.

For the case of the conventional midtread quantizer, the quantization operation on an input x, to produce a reconstruction level y_i, in terms of the decision levels x_i, is defined as:

$$\dot{x} = Q(x) = y_i; \quad x_{i-1} < x \leq x_i \tag{3.38}$$

This is typically achieved using a rounding operation, for example:

$$\dot{x} = \text{NINT}\left(\frac{x}{\triangle}\right) \tag{3.39}$$

where NINT is the operation of rounding to the nearest integer.

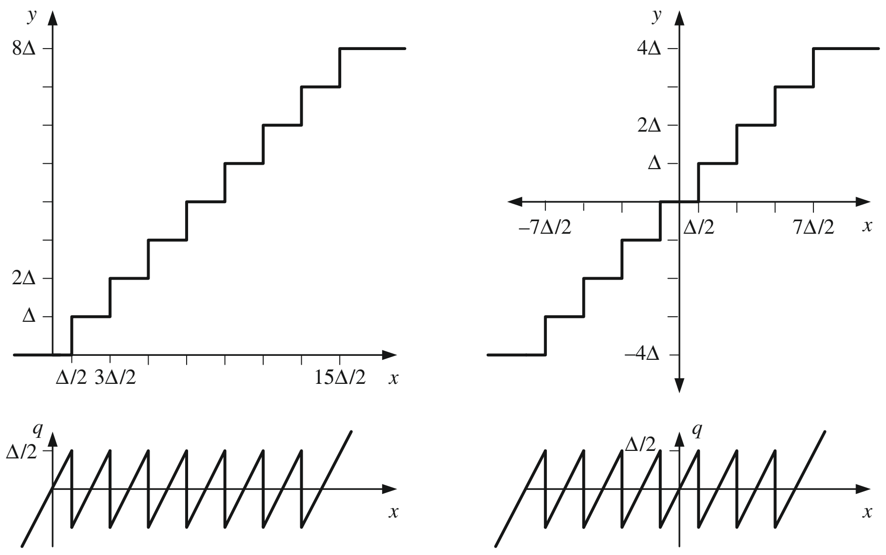


FIGURE 3.12
Common uniform quantizer characteristics.

In order to reconstruct a quantized signal, it will normally have to be rescaled to its original range. This rescaling operation is defined as:

$$\tilde{x} = \triangle \dot{x} \tag{3.40}$$

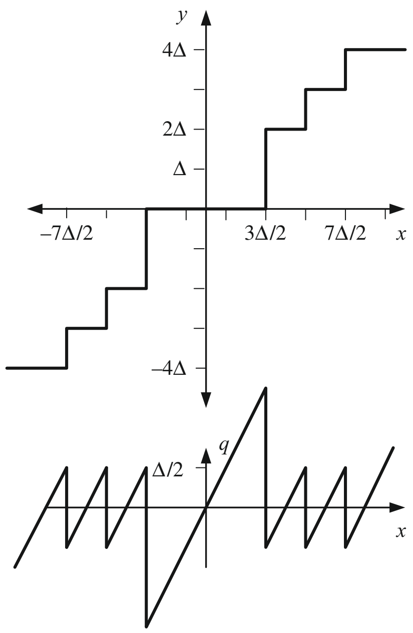
3.4.2 Adaptation to signal statistics

Quantizers with non-uniform step sizes can be beneficial in cases when:

- The signal being quantized does not exhibit a uniform PDF.
- Low level signals or other signal ranges are known to contain excess noise.
- There is a desire to map a range of low level signals to zero.
- There is a need to incorporate a non-linear amplitude mapping as part of the quantization process.

Deadzone quantizer

The most common form of non-uniform quantizer is the *Deadzone* quantizer. This has a broader decision range for the band of inputs close to zero. It has the benefit during compression of ensuring that noisy low level signals are not allocated bits unnecessarily. The characteristic of a deadzone quantizer is shown in Figure 3.13 (left).



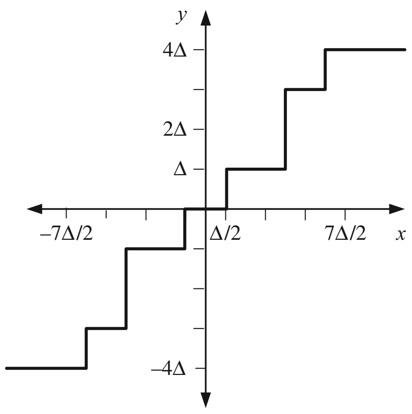


FIGURE 3.13

Common non-uniform quantizers. Left: center deadzone. Right: Lloyd Max quantizer.

Lloyd Max quantizer

The Lloyd Max algorithm [10] is a well-known approach to designing non-uniform quantizers optimized according to the prevailing PDF of the input signal. Rather than allocate a uniform step size, as would be optimum for a uniform PDF, the Lloyd Max approach identifies decision boundaries according to the mean values of equal area partitions of the PDF curve. The characteristic of a Lloyd Max quantizer is shown in Figure 3.13 (right).

3.4.3 HVS weighting

In practice for many compression applications, the quantizer step size is weighted according to some perceptual criterion, such as the spatial frequency of a coefficient produced by a decorrelating transform operation. This is considered in more detail in Chapter 5.

3.4.4 Vector quantization

Vector quantization (VQ) is a method which maps a group of samples to a single quantization index. It is a method that has been adapted in highly robust codecs such as those based on Pyramid Vector Quantization (PVQ) as discussed in Chapter 11. We will not cover this in detail here but the interested reader is referred to Refs. [9,11].

3.5 Linear prediction

3.5.1 Basic feedforward linear predictive coding

The idea of linear prediction is to exploit statistical relationships in random data sequences in order to decorrelate them prior to coding.

A feedforward analysis filter for linear prediction is shown in Figure 3.14 (top) and the corresponding synthesis filter is shown in Figure 3.14 (bottom). Ignoring the quantizer block for the time being, the transfer function of the analysis filter is given by:

$$\begin{aligned} E(z) &= X(z) - X_p(z) \\ &= X(z)(1 - P(z)) \end{aligned} \tag{3.41}$$

$$H(z) = \frac{E(z)}{X(z)} = 1 - P(z) \tag{3.42}$$

where the notation used is as shown in the figure. Assuming the predictor block in Figure 3.14 is an FIR filter, then in the sample domain we have:

$$e[n] = x[n] - \sum_{i=1}^{N} a_i x[n-i] \tag{3.43}$$

If we now consider the decoder or synthesis filter:

$$Y(z) = E(z) + Y(z)P(z) \tag{3.44}$$

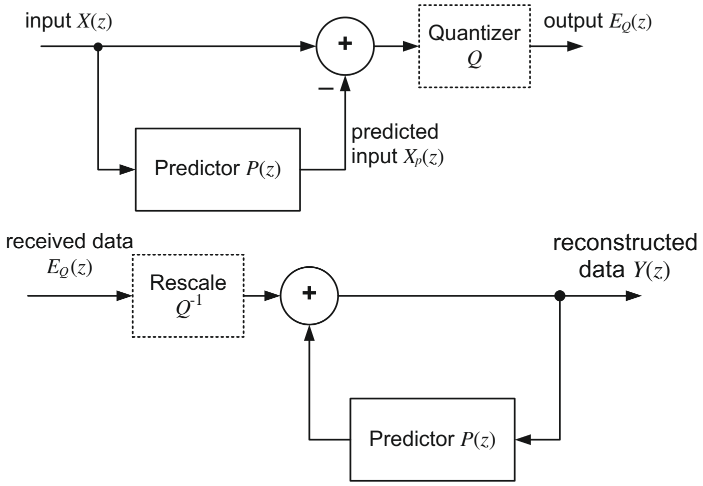


FIGURE 3.14

Feedforward linear prediction. Top: encoder. Bottom: decoder.

or:

$$Y(z) = \frac{E(z)}{1 - P(z)} = X(z) \tag{3.45}$$

Example 3.5 (Linear predictive coding)

Consider the case in Figure 3.14 when $P(z) = z^{-1}$ and the input to the predictive coder is $x[n] = \{1, 2, 3, 2, 5, 4, 2, 4, 5, 6\}$. Assuming that no quantization takes place, confirm that the decoder output will be identical to the input sequence.

Solution. We have $e[n] = x[n] - x[n-1]$ and $y[n] = e[n] + y[n-1]$. So, assuming zero initial conditions and working through for each input sample:

$x[n]$	1	2	3	2	5	4	2	5	6
$e[n]$	1	1	1	−1	3	−1	−2	3	1
$y[n]$	1	2	3	2	5	4	2	5	6

Hence $y[n] = x[n]$.

Predictor dynamic range

The benefit of prediction filtering as a basis for coding is that although the dynamic range of the transmitted error signal can be double that of the original signal, the variance of the error signal is significantly reduced compared to the original.

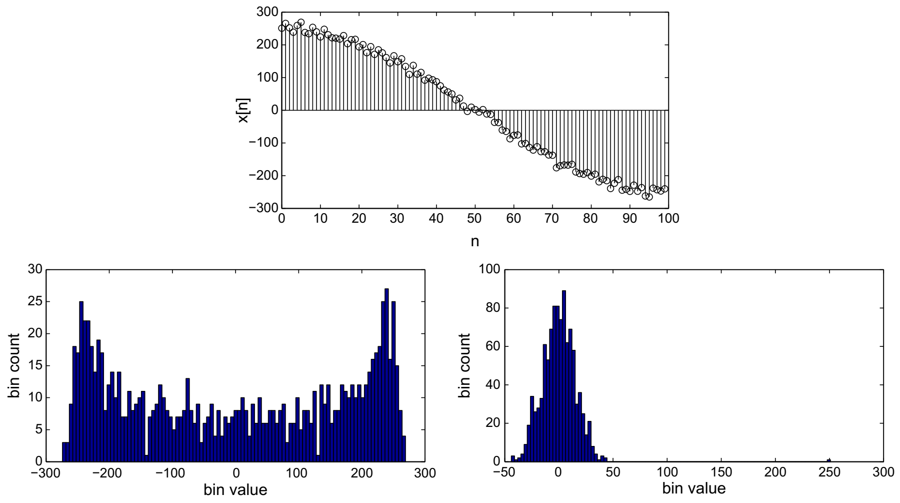


FIGURE 3.15

Prediction signal dynamic range. Top: input signal. Bottom left: distribution of 1000 samples of input signal. Bottom right: distribution of 1000 samples of prediction residual.

This is demonstrated in Figure 3.15, where a sinusoidal signal with additive Gaussian noise ($\sigma_v^2 = 0.05$):

$$x[n] = \cos(0.01\pi n) + v[n] \tag{3.46}$$

is filtered using a predictor $P(z) = z^{-1}$.

Linear predictive coding with quantization

The problem with the feedforward architecture arises when quantization is introduced in the process. In this case we have:

$$E_Q(z) = Q\{X(z) - X_p(z)\} \tag{3.47}$$

and

$$Y(z) = Q^{-1}\{E_Q(z)\} + Y(z)P(z) \tag{3.48}$$

or:

$$Y(z) = \frac{Q^{-1}\{E_Q(z)\}}{1 - P(z)} \neq X(z) \tag{3.49}$$

As one would expect, the quantization process introduces errors in the reconstructed output. This is to be expected since the operation is a lossy one. However, the situation is worse because there is the potential for drift between the encoder and the decoder. This is illustrated in Example 3.6.

Example 3.6 (Linear predictive coding with quantization)
Consider the case in Figure 3.14 where, as before, $P(z) = z^{-1}$ and the input to the predictive coder is $x[n] = \{1, 2, 3, 2, 5, 4, 2, 4, 5, 6\}$. Assuming that quantization takes place, where the quantization and rescaling operations take the form:

$$e_Q[n] = \mathrm{rnd}\left(\frac{e[n]}{2}\right); \quad e_R[n] = 2e_Q[n]$$

with rounding of 0.5 values toward 0, characterize the decoder output relative to the input signal.

Solution. Assuming 0 initial conditions and following the same approach as in Example 3.5, but this time including the quantization and rescaling operations:

$x[n]$	1	2	3	2	5	4	2	5	6
$e[n]$	1	1	1	−1	3	−1	−2	3	1
$e_Q[n]$	0	0	0	0	1	0	−1	1	0
$e_R[n]$	0	0	0	0	2	0	−2	2	0
$y[n]$	0	0	0	0	2	2	0	2	2

Hence $y[n]$ is not the same as $x[n]$. Furthermore significant drift has occurred due to the accumulation of rounding errors at the decoder.

3.5.2 Linear prediction with the predictor in the feedback loop

In order to avoid problems with drift, we must replicate the effects of the decoding operation within the encoder. This ensures that, in the absence of data corruption during transmission, both the encoder and the decoder process exactly the same data and operate in unison. This modified architecture is shown in Figure 3.16. The expression

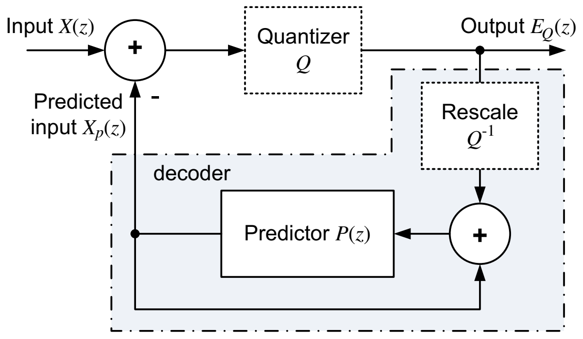


FIGURE 3.16
Feedback-based linear prediction.

for our encoded output (without quantization) is now given by:

$$\begin{aligned} E(z) &= X(z) - X_p(z) \\ &= X(z) - \frac{E(z)P(z)}{1 - P(z)} \end{aligned} \tag{3.50}$$

Therefore:

$$E(z) = X(z)(1 - P(z))$$

and, as before, at the decoder:

$$Y(z) = E(z) + Y(z)P(z) \tag{3.51}$$

or:

$$Y(z) = \frac{E(z)}{1 - P(z)} = X(z) \tag{3.52}$$

If we now include quantization in the process, the difference between the feedback and feedforward architectures will become clear. To ease analysis we model the quantization process as additive quantization noise, $V(z)$ with a variance of σ_N^2. Then we have, as illustrated in Figure 3.17:

$$\begin{aligned} E(z) &= X(z) - X_p(z) \\ &= X(z) - \frac{[E(z) + V(z)]P(z)}{1 - P(z)} \end{aligned} \tag{3.53}$$

Therefore:

$$E(z) = X(z)(1 - P(z)) - V(z)P(z) \tag{3.54}$$

and at the decoder:

$$Y(z) = E(z) + Y(z)P(z) + V(z) \tag{3.55}$$

or:

$$Y(z) = \frac{E(z) + V(z)}{1 - P(z)} \tag{3.56}$$

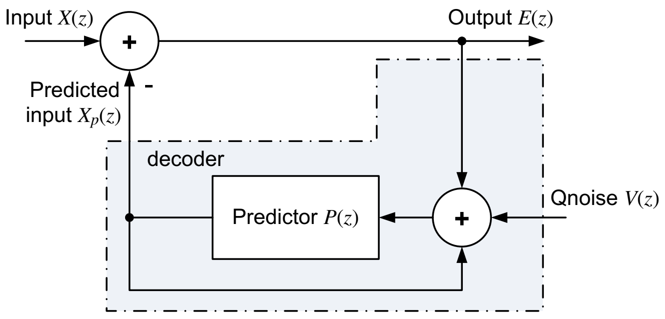


FIGURE 3.17
Feedback-based linear predictor with quantization noise modeling.

Substituting from equation (3.54) for $E(z)$ in equation (3.56), it is straightforward to see that:

$$Y(z) = X(z) + V(z) \tag{3.57}$$

So we no longer have any drift between encoder and decoder, we simply have a mismatch due to the additive noise of the quantization process. This is a well known but very important result as this prediction architecture is used extensively in video compression.

Example 3.7 (Feedback-based linear predictive coding with quantization)
Consider the case in Figure 3.16 where, as before, $P(z) = z^{-1}$ and the input to the predictive coder is $x[n] = \{1, 2, 3, 2, 5, 4, 2, 4, 5, 6\}$. Assuming that quantization takes place, where the quantization and rescaling operations take the form:

$$e_Q[n] = \text{rnd}\left(\frac{e[n]}{2}\right); \quad e_R[n] = 2e_Q[n]$$

with rounding of 0.5 values toward 0, characterize the decoder output relative to the input signal.

Solution. Assuming 0 initial conditions and following the same approach as in Example 3.6, but this time based on the architecture in Figure 3.16:

$x[n]$	1	2	3	2	5	4	2	5	6
$e[n]$	1	2	1	0	3	−1	−2	3	2
$e_Q[n]$	0	1	0	0	1	1	−1	1	1
$e_R[n]$	0	2	0	0	2	2	−2	2	2
$y[n]$	0	2	2	2	4	4	2	4	6

As expected, $y[n]$ is not identical to $x[n]$ because of the lossy quantization process. However, the variance of the predicted signal is significantly reduced, compared to the original, and also there is no drift between encoder and decoder.

3.5.3 Wiener Hopf equations and the Wiener filter

If we have a linear FIR predictor filter with impulse response, **h**, we wish the filter coefficients to be assigned values that force the error signal after prediction to be as small as possible. If we perform this optimization in terms of second order statistics (i.e. based on Mean Squared Error (MSE)) then the result is referred to as a Wiener filter—an optimum non-recursive estimator.

If we have a desired sequence (training sequence) **d**, then:

$$e[n] = d[n] - y[n] \tag{3.58}$$

and we can define a cost function as:

$$J = E\left\{|e[n]|^2\right\} \tag{3.59}$$

where, when the gradient $\nabla(J) = 0$ with respect to all filter coefficients, then the filter is said to be optimum in the MSE sense. It can be shown that a necessary and sufficient condition for J to attain a minimum is that the corresponding value of the estimation error is orthogonal to each sample that enters the estimation process at time n. This is known as the *principle of orthogonality*. Based on this observation, it is straightforward to show that the optimum filter coefficients can be computed from the autocorrelation of the filter input sequence $r[k]$ and the cross correlation of the filter input and the desired response $p[k]$ for a lag of k samples. This is known as the Wiener Hopf equation:

$$\sum_{i=0}^{\infty} h_{\mathrm{o}}[i]r[i-k] = p[k] \tag{3.60}$$

or in matrix vector form:

$$\mathbf{h}_{\mathrm{o}} = \mathbf{R}^{-1}\mathbf{p} \tag{3.61}$$

It is also possible to show, assuming stationarity, that the error surface is exactly a second order function of the filter coefficients. It will thus possess a unique minimum point corresponding to the optimum filter weights. For further details, the reader is referred to Ref. [12].

3.6 Information and entropy

The concept of generating a quantitative measure of information was originated by Shannon in 1948 [13]. He used entropy, a measure of the uncertainty in a random variable, to quantify the value of the information contained in a message.

Understanding the basics of information theory is important in the context of image and video compression, since it is used extensively for lossless symbol encoding after transformation and quantization. Typically, quantized transform coefficients are scanned and run-length encoded. Symbols representing runs of zeros and values of non-zero coefficients are coded, using variable length codewords, in a manner that reflects their information content. In this way a lossless reduction in bit rate can be achieved.

Before we look at symbol statistics, let us first review the basics of information theory. For more in-depth coverage, please refer to Ref. [14].

3.6.1 Self-information

Information is used to reduce the uncertainty about an event or signal. Shannon [13] defined the *self-information* of an event A, which could for example comprise a set of outcomes from an experiment, as:

$$i(A) = \log_2\left(\frac{1}{P(A)}\right) = -\log_2\left(P(A)\right) \tag{3.62}$$

The use of logarithms is intuitive, since $\log(1) = 0$ and $-\log(x)$ increase as x decreases from 1 to 0. Hence, if the probability of an event is low, then the information associated with it is high. For example, if $P(A) = 0.5$ then $-\log_2(0.5) = 1$ and if $P(A) = 0.25$ then $-\log_2(0.25) = 2$. Other logarithm bases can be used, but base 2 is the most common as it conveniently provides a measure of information in bits.

Independent events

The information obtained from two independent events is the sum of the information from each event. This can be seen as follows:

$$i(AB) = \log_2\left(\frac{1}{P(AB)}\right)$$

and, if A and B are independent, then:

$$P(AB) = P(A)P(B) \tag{3.63}$$

Therefore:

$$i(AB) = \log_2\left(\frac{1}{P(A)P(B)}\right) = \log_2\left(\frac{1}{P(A)}\right) + \log_2\left(\frac{1}{P(B)}\right)$$

or

$$i(AB) = i(A) + i(B) \tag{3.64}$$

3.6.2 Entropy

If we define a source that generates a series of random and independent events with outcomes from an alphabet $S = \{s_1, s_2, s_3, \ldots, s_N\}$, then the average self-information associated with an event is:

$$H = \sum_{i=1}^{N} P(s_i)i(s_i) = -\sum_{i=1}^{N} P(s_i)\log_2\left(P(s_i)\right)$$

The quantity H, the average self-information, is referred to as the entropy of the source and this tells us how many bits per symbol are required to code that source. Shannon demonstrated that no coding scheme can code the source losslessly at a rate lower than its entropy.

Entropy and first order entropy

The entropy of a source is given by:

$$H(S) = \lim_{N\to\infty} \frac{1}{N} G_N \tag{3.65}$$

where

$$G_N = -\sum_{i_1=1}^{N}\sum_{i_2=1}^{N}\cdots\sum_{i_M=1}^{N} P(d_1 = s_{i_1}, d_2 = s_{i_2}, \ldots, d_M = s_{i_M}) \\ \times \log P(d_1 = s_{i_1}, d_2 = s_{i_2}, \ldots, d_M = s_{i_M})$$

and $D = \{d_1, d_2, d_3, \ldots, d_M\}$ and M is the length of the sequence. However, if each symbol is independent and identically distributed (*iid*), then it can be shown that:

$$G_N = -M\sum_{i_1=1}^{N} P(d_1 = s_{i_1}) \log\left(P(d_1 = s_{i_1})\right) \tag{3.66}$$

and the entropy is then:

$$H(S) = -\sum_{i=1}^{N} P(s_i) \log_2\left(P(s_i)\right) \tag{3.67}$$

In most realistic cases equations (3.65) and (3.67) do not give the same result. Hence we differentiate them by referring to equation (3.67) as the first order entropy.

The relationship between symbol probability and the self-information of that symbol is given in Figure 3.18. The figure also illustrates the relationship between the probability of a symbol and the contribution that symbol makes to the overall entropy

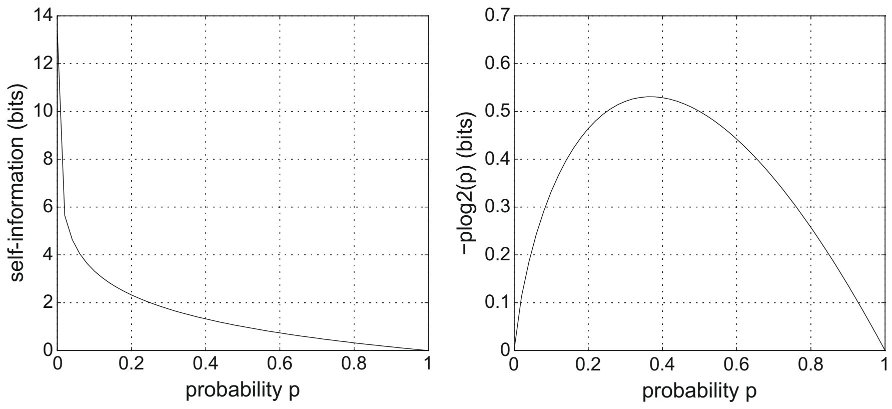


FIGURE 3.18
Self-information and probability. Left: plot of self-information vs probability for a single event. Right: plot of the self-information of an event weighted by its probability.

of the source. Figure 3.18 indicates that symbols with probabilities at either end of the scale contribute little to the source entropy while those in the middle of the range contribute values around 0.5.

Example 3.8 (Entropy)
An alphabet $S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}$. If the corresponding probabilities of occurrence of these symbols during transmission is $P(S) = \{0.06, 0.23, 0.3, 0.15, 0.08, 0.06, 0.06, 0.06\}$, calculate the first order entropy for the source.

Solution. First order entropy is given by:

$$\begin{aligned} H &= -\sum_S P(s_i)\log_2\big(P(s_i)\big) \\ &= -(0.06 \times \log_2 0.06 + 0.23 \times \log_2 0.23 + \cdots + 0.06 \times \log_2 0.06) \\ &= 2.6849 \text{ bits/symbol} \end{aligned}$$

3.6.3 Symbols and statistics

In practice, since equation (3.65) is impossible to compute, first order entropy is almost always used instead. We therefore have to estimate the entropy of the source and this depends heavily on the statistics of the source symbols. This point is emphasized in Example 3.9.

Example 3.9 (Source statistics and entropy)
Consider the sequence of symbols: $D = \{1, 2, 3, 2, 1, 2, 3, 4, 5, 6, 5, 6, 7, 8, 9, 10\}$. Compute the first order entropy for this sequence.

Solution. This seems like a straightforward question. We are given no information about the statistics of the source so let us estimate the probabilities of each symbol, based on relative frequencies of occurrence. In this way we have:

$$P(4) = P(7) = P(8) = P(9) = P(10) = \frac{1}{16}$$

$$P(1) = P(3) = P(5) = P(6) = \frac{2}{16}$$

$$P(2) = \frac{3}{16}$$

Thus the first order entropy is

$$H = -\sum_S P(s_i)\log_2\big(P(s_i)\big) = 3.2028 \text{ bits}$$

So this says that, on average, we need just over 3.2 bits per symbol for this source.

However, if we now look closer at the sequence we can observe that the symbols are correlated. So what happens if we process this sequence with a simple first order linear predictor? Our residual sequence, $E = \{1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1\}$.

So now we have a new set of probability estimates:

$$P(1) = \frac{13}{16}$$
$$P(-1) = \frac{3}{16}$$

In this case, we recalculate our entropy to be 0.6962 bits/symbol. So which is correct? In fact neither is right or wrong, it's just that in the latter case, we have better understood the structure of the source and provided a better estimate of its entropy.

For example, we can estimate the probabilities of occurrence of pixel values in an image, or a region of an image, through computing a histogram of its quantization levels and dividing the content of each quantization bin by the total number of pixels. Alternatively we can pre-compute them using a histogram of one or more similar training images. The latter is more common, although we rarely use actual pixel values as the input symbols in practice. More often we use some more efficient means based on prediction or run length coding of quantized spatial or transform domain data. More of this later in Chapter 7.

3.7 Summary

This chapter has reviewed some preliminary topics in signal processing and information theory that help to place the remainder of this book in context. We have not burdened the reader with series of proofs and theorems but have instead focused on the use of examples to aid understanding of some important topics. We examined some basic statistical measures and used these to characterize picture redundancy. Filters and transforms were introduced as they form the basis of most compression methods and these will be covered further in Chapters 5 and 6. We also revisit the area of quantization in the same chapters.

We spent some time here on prediction structures as these are used frequently in compression and can provide substantial coding gains over non-predictive methods, both for spatial and temporal processing. Intra-prediction will be covered in Chapters 9 and 12 and motion prediction in Chapters 8 and 9.

Finally, we reviewed the basics of information theory, explaining how its performance depends heavily on assumed source models and context. This forms the basis of symbol encoding as used in most compression standards. We specifically focus on entropy coding methods in Chapter 7 and follow this up in Chapter 12 in the context of standards.

References

[1] S. Mitra, Digital Signal Processing: A Computer Based Approach, fourth ed., McGraw Hill, 2011.

[2] J. Woods, Multidimensional Signal, Image and Video Processing and Coding, second ed., Academic Press, 2012.

[3] C. Shannon, Communication in the presence of noise, Proceedings of Institute of Radio Engineers 37 (1) (1949) 10–21.

[4] E. Dubois, The sampling and reconstruction of time-varying imagery with application in video systems, Proceedings of the IEEE 73 (1985) 502–522.

[5] Y. Wang, J. Ostermann, Y.-Q. Zhang, Video Processing and Communications, Prentice Hall, 2001.

[6] R. O'Callaghan, D. Bull, Combined morphological-spectral unsupervised image segmentation, IEEE Transactions on Image Processing 14 (1) (2005) 49–62.

[7] P. Czerepiński, D. Bull, Enhanced interframe coding based on morphological segmentation, IEE Proceedings — Vision, Image and Signal Processing 144 (4) (1997) 220–226.

[8] A. Oppenheim, A. Willsky, H. Nawab, Signals and Systems, second ed., Prentice Hall, 1999; A. Netravali, B. Haskell, Digital Pictures: Representation, Compression and Standards, second ed., Plenum Press, 1995.

[9] J.-R. Ohm, Multimedia Communication Technology, Springer, 2004.

[10] S. Lloyd, Least squares quantisation in PCM (1957), Reprint in IEEE Trans Communications 30 (1982) 129–137.

[11] Y. Linde, A. Buzo, R. Gray, An algorithm for vector quantiser design, IEEE Transactions on Communications 28 (1980) 84–95.

[12] M. Hayes, Statistical Digital Signal Processing and Modeling, Wiley, 1999.

[13] C. Shannon, A mathematical theory of communication, Bell System Technical Journal 27 (3) (1948) 379–423.

[14] T. Cover, J. Thomas, Elements of Information Theory, Wiley, 2006.