DATA COMPRESSION CODES, LOSSY

In this article we introduce lossy data compression. We consider the overall process of converting from analog data to digital so that the data are processed in digital form. Our goal is to achieve the most compression while retaining the highest possible fidelity. First we consider the requirements of signal sampling and quantization. Then we introduce several effective and popular lossy data compression techniques. At the end of this article we describe the theoretical limits of lossy data compression performance.

Lossy compression is a process of transforming data into a more compact form in order to reconstruct a close approximation to the original data. Let us start with a description using a classical information coding system model. A common and general data compression system is illustrated in Fig. 1.

As shown in Fig. 1, the information *source data*, S, is first transformed by the *compression process* to *compressed signal,* which usually is a more compact representation of the source data. The compact form of data offers tremendous advantages in both communication and storage applications. For example, in communication applications, the compressed signal is transmitted to a receiver through a communication channel with lower communication bandwidth. In storage applications, the compressed signal takes up less space. The stored data can be retrieved whenever they are needed. After received (or retrieved) signal is received (retrieved), it is pro-

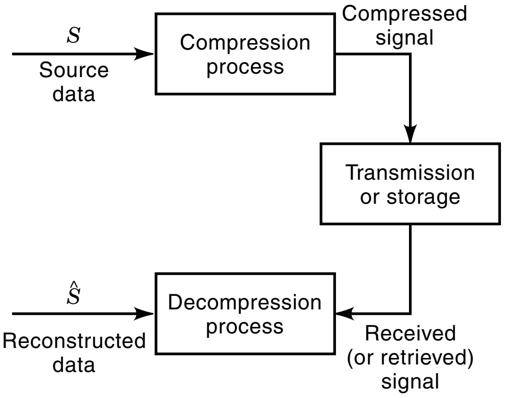

Figure 1. General data compression system.

cessed by the *decompression process,* which reconstructs the original data with the greatest possible fidelity. In lossy compression systems, the original signal, S, cannot be perfectly retrieved from the reconstructed signal, $\hat{S}$, which is only a close approximation.

LOSSY VERSUS LOSSLESS

In some applications, such as in compressing computer binary executables, database records, and spreadsheet or word processor files, the loss of even a single bit of data can be catastrophic. For such applications, we use *lossless data compression* techniques so that an exact duplicate of the input data is generated after the compress/decompress cycle. In other words, the reconstructed signal, $\hat{S}$, is identical to the original signal, S,

$$S = \hat{S}$$

Lossless data compression is also known as *noiseless data compression.* Naturally, it is always desirable to recreate perfectly the original signal after the transmission or storage process. Unfortunately, this requirement is difficult, costly, and sometimes infeasible for some applications. For example, for audio or visual applications, the original source data are analog data. The digital audio or video data we deal with are already an approximation of the original analog signal. After the compress/decompress cycle, there is no way to reconstruct an exact duplicate of the original continuous analog signal. The best we can do is to minimize the loss of fidelity during the compress/decompress process. In reality we do not need the requirement of $S=\hat{S}$ for audio and video compression other than for some medical or military applications. The *International Standards Organization* (ISO) has published the JPEG (Joint Photographic Experts Group) standard for still image compression (1) and the MPEG (Moving Pictures Expert Group) standard for moving picture audio and video compression (2, 3). Both JPEG and MPEG standards concern lossy compression, even though JPEG also has a lossless mode. The International Telecommunication Union (ITU) has published the H-series video compression standards, such as *H.261* (4) and *H.263* (5), and the G-series speech compression standards, such as *G.723* (6) and *G.728* (7). Both the H-series and G-series standards are also for lossy compression.

WHY LOSSY?

Lossy compression techniques involve some loss of source information, so data cannot be reconstructed in the original form after they are compressed by lossy compression techniques. However, we can generally get a much higher compression ratio and possibly a lower implementation complexity.

For many applications, a better compression ratio and a lower implementation complexity are more desirable than the ability to reconstruct perfectly the original data. For example, in audio-conferencing applications, it is not necessary to reconstruct perfectly the original speech samples at the receiving end. In general, telephone quality speech is expected at the receiver. By accepting a lower speech quality, we can achieve a much higher compression ratio with a moderated implementation complexity. Because of this, the conferencing speech signals can be transmitted to the destination through a lower bandwidth network at a reasonable cost. For music centric entertainment applications that require near CD-quality audio, the amount of information loss that can be tolerated is significantly lower. However, it is still not necessary to restrict compression to lossless techniques. The European MUSICAM and ISO MPEG digital audio standards both incorporate lossy compression yet produce high-fidelity audio. Similarly a perfect reconstruction of the original sequence is not necessary for most of the visual applications as long as the distortion does not result in annoying artifacts.

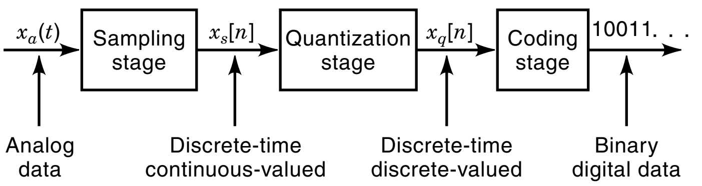

Figure 2. Analog-to-digital converter.

Most signals in our environment, such as speech, audio, video, radio, and sonar emissions, are analog signals. We have just discussed how lossy compression techniques are especially useful for compressing digital representations of analog data. Now let us discuss how to effectively convert an analog signal to digital data.

Theoretically converting an analog signal to the desired digital form is a three-stage process, as illustrated in Fig. 2. In the first stage, the analog data (continuous-time and continuous-valued) are converted to discrete-time and continuous-valued data by taking samples of the continuous-time signal at regular instants, $t=nT_1$,

$$x_s[n] = x_a(nT_1) \qquad \text{for } n = 0, \pm 1, \pm 2, \ldots$$

where T_1 is the *sampling interval.* In the *quantization stage,* the discrete-time continuous-valued signals are further converted to discrete-time discrete-valued signals by representing the value of each sample with one of a finite set of possible values. The difference between the unquantized sample $x_s[n]$ and the quantizer output $x_q[n]$ is called the *quantization error.* In reality quantization is a form of lossy data compression. Finally, in the *coding stage*, the quantized value, $x_q[n]$, is coded to a binary sequence, which is transmitted through the communication channel to the receiver. From a compression point of view, we need an analog-to-digital conversion system that generates the shortest possible binary sequence while still maintaining required fidelity. Let us discuss the signal sampling stage first.

PERIODIC SAMPLING

The typical method of converting a continuous-time signal to its discrete-time representation is through *periodic sampling,* with a sequence of samples, $x_s[n]$, obtained from the continuous-time signal $x_a(t)$ according to the following relationship

$$x_s[n] = x_a(nT_1) \qquad \text{for all integers } n$$

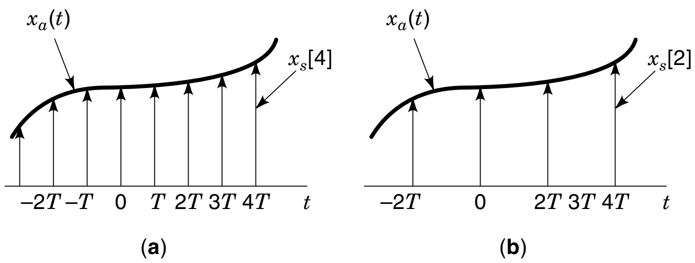

Figure 3. Continuous-time signal $x_a(t)$ sampled to discrete-time signals at the sampling period of (a) T, and (b) $2T$.

where n is an integer, T_1 is the *sampling period,* and its reciprocal $n_1=1/T_1$ is the *sampling frequency,* in samples per second. To visualize this process, consider embedding the samples in an idealized impulse train to form an idealized continuous time sampled waveform $x_s(t) = \sum_{n=-\infty}^{\infty} x_s[n]\delta(t-nT_1)$, where each impulse or Dirac δ function can be thought of as an infinitesimally narrow pulse of unit area at time $t = nT_1$ which is depicted as an arrow with height 1 corresponding to the area of the impulse. Then $x_s(t)$ can be drawn as a sequence of arrows of height $x_s[n]$ at time $t = nT_1$, as shown with the original signal $x_a(t)$ in Fig. 3 for sampling periods of T and $2T$.

The sampling process usually is not an invertible process. In other words, given a discrete-time sequence, $x_s[n]$, it is not always possible to reconstruct the original continuous-time input of the sampler, $x_a(t)$. It is very clear that the sampling process is not a one-to-one mapping function. There are many continuous-time signals that may produce the same discrete-time sequence output unless they have same bandwidth and sampled at Nyquist rate.

ALIASING

In order to get better understanding of the periodic sampler, let us look at it from frequency domain. First, consider the idealized sampling function, a periodic unit impulse train signal, $s(t)$:

$$s(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT_1)$$

where T_1 is the period of $s(t)$. The properties of impulse functions imply that the idealized sampled waveform is easily expressed as

$$\begin{aligned} x_s(t) &= x_a(t)s(t) \\ &= x_a(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_1) \\ &= \sum_{n=-\infty}^{\infty} x_a(nT_1)\delta(t - nT_1) \end{aligned} \tag{1}$$

To summarize, the idealized sampled data signal is defined as a product of the original signal and a samping function and is composed of a series of equally spaced impulses weighted by the values of the original continuous-time signal at the sampling instants, as depicted in Fig. 4.

Now let us make a Fourier analysis of $x_s(t)$. The Fourier transform pair (8) is defined as

$$x(t) = \int_{-\infty}^{+\infty} X(f)e^{j2\pi ft}\,df \tag{2}$$

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}\,dt \tag{3}$$

where $X(f)$ is the *Fourier transform* of $x(t)$, or symbolically, $X(f) = \mathscr{T}(x(t))$, and $x(t)$ is the *inverse Fourier transform* of $X(f)$, $x(t) = \mathscr{T}^{-1}(X(f))$. A standard result of generalized Fourier analysis is that

$$s(t) = \frac{1}{T_1} \sum_{n=-\infty}^{+\infty} e^{j2n\pi f_1 t} \tag{4}$$

After substitution of Eq. (4) into Eq. (1), the sampled data, $\chi_{s(t}$, yield

$$\begin{aligned} x_s(t) &= x_a(t)s(t) \\ &= \frac{1}{T_1} \sum_{n=-\infty}^{\infty} x_a(t)e^{j2n\pi f_1 t} \end{aligned} \tag{5}$$

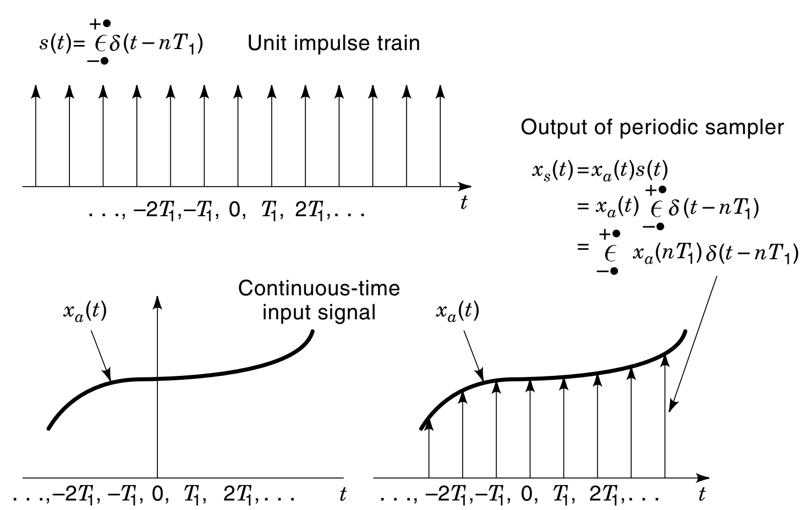

Figure 4. Periodic sampled continuous-time signal $x_a(t)$.

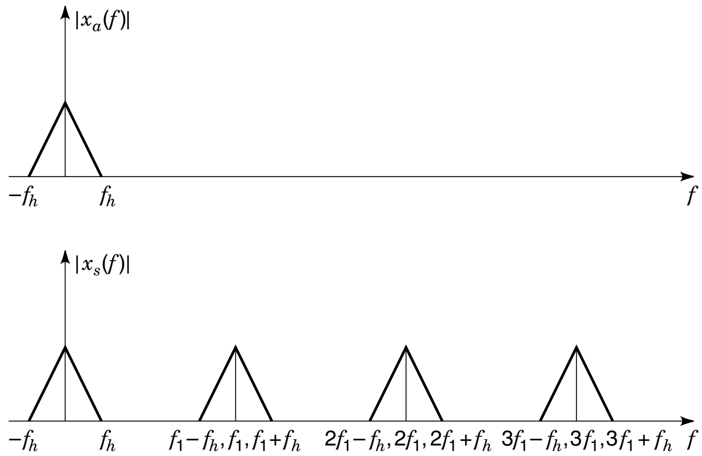

Figure 5. Spectrum of the sampled data sequence $x_s(t)$.

Now, taking the Fourier transform of $x_s(t)$ in Eq. (5), the result is

$$\begin{aligned} X_s(f) &= \int_{-\infty}^{+\infty} \left(\frac{1}{T_1} \sum_{n=-\infty}^{+\infty} x_a(t) e^{j2n\pi f_1 t} \right) e^{-j2\pi f t}\, dt \\ &= \frac{1}{T_1} \sum_{n=-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_a(t) e^{-j2\pi (f - n f_1) t}\, dt \qquad (6) \\ &= \frac{1}{T_1} \sum_{n=-\infty}^{+\infty} X_a(f - n f_1) \end{aligned}$$

We see from Eq. (6) that the spectrum of a sampled-data signal consists of the periodically repeated copies of the original signal spectrum. Each copy is shifted by integer multiples of the sampling frequency. The magnitudes are multiplied by $1T_1$.

Let us assume that the original continuous-time signal $x_a(t)$ is *bandlimited* to $0 \leq |f| \leq f_h$, then the spectrum of the sampled data sequence $x_s[n]$takes the form illustrated in Fig. 5. In the case where $f_h > f_1 - f_h$, or $f_1 < 2f_h$, there is an overlap between two adjacent copies of the spectrum as illustrated in Fig. 6. Now the overlapped portion of the spectrum is different from the original spectrum, and therefore it becomes impossible to recover the original spectrum. As a result the reconstructed output is distorted from the original continuous-time input signal. This type of the distortion is usually referred to as *aliasing*.

To avoid aliasing a bandlimited continuous-time input, it is necessary to sample the input at the sampling frequency $f_1 \geq 2f_h$. This is stated in the famous *Nyquist sampling theorem* (10).

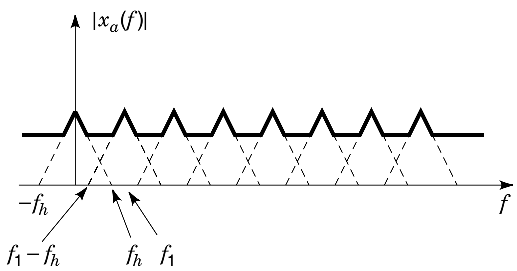

Figure 6. Spectrum of the sampled data sequence $x_s(t)$ for the case of $f_h > f_1 - f_h$.

Nyquist Sampling Theorem. If $x_a(t)$ is a bandlimited continuous-time signal with $X(f) = 0$ for $|f| > f_h$, then $x_a(t)$ can be uniquely reconstructed from the periodically sampled sequence $x_a(nT)$, $-\infty < n < \infty$, if $1/T > 2f_h$.

On the other hand, if the signal is not bandlimited, theoretically there is no avoiding the aliasing problem. All real-life continuous-time signals, such as audio, speech, or video emissions, are approximately bandlimited. A common practice is to get a close approximation of the original signals by filtering the continuous-time input signal with a low-pass filter before the sampling stage. This low-pass filter ensures that the filtered continuous-time signal meets the bandlimited criterion. With this *presampling filter* and a proper sampling rate, we can ensure that the spectral components of interest are within the bounds for which the signal can be recovered, as illustrated in Fig. 7.

QUANTIZATION

In the quantization stage discrete-time continuous-valued signals are converted to discrete-time discrete-valued signals. In the quantization process, amplitudes of the samples are quantized by dividing the entire amplitude range into a finite set of amplitude ranges. Each amplitude range has a representative amplitude value. The representative amplitude value for the range is assigned to all samples falling into the given range. Quantization is the most important step to removing the irrelevant information during lossy compression process. Therefore the performance of the quantizer plays a major role of overall performance of a lossy compression system.

There are many different types of quantizers. The simplest and most popular one is the *uniform quantizer,* in which the quantization levels and ranges are distributed uniformly. In general, a signal with amplitude x is specified by index k if x falls into the interval

$$I_k : \{x : x_k \leq x < x_{k+1}\}, \qquad k = 1, 2, 3, \ldots, L \qquad (7)$$

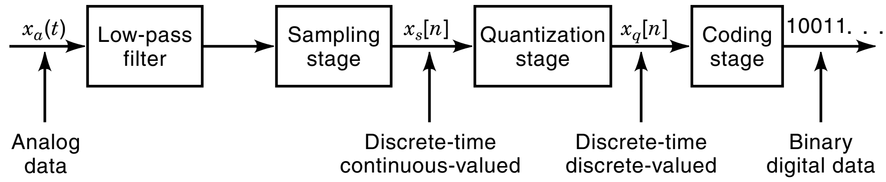

Figure 7. Sampling a continuous-time signal that is not bandlimited.

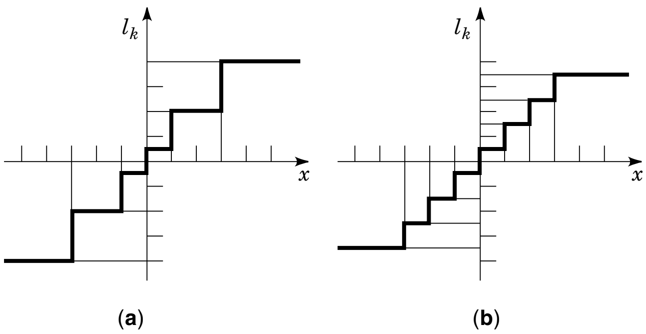

Figure 8. Examples of (a) a nonuniform quantizer, (b) an 8-level uniform quantizer.

In this process, the continuous valued signal with amplitude x is mapped into an *L-ary* index k. In most cases the *L-ary* index, k, is coded into binary numbers at the *coding stage* and transmitted to the receiver. Often, at the coding stage, efficient entropy coding is incorporated to generate variable length codewords in order to reach the entropy rate of quantized signals. Figure 8(a) and 8(b) gives examples of a nonuniform quantizer and an 8-level ($L = 8$) uniform quantizer.

At the receiver, the index k is translated into an amplitude I_k that represents all the amplitudes of signals fall into the interval I_k, namely

$$\hat{x}_k = l_k \qquad \text{if } x \in I_k \tag{8}$$

wherê x_k is the output of the decoder. The amplitude l_k is called the *representation level,* and the amplitude x_k is called the *decision level.* The difference between the input signal and the decoded signal, $x_k - x$, is called the *quantization error,* or *quantization noise.* Figure 9 gives an example of a quantized waveform and the corresponding quantization noise.

Quantization steps and ranges can be changed adaptively during the compression process. As an example, for video conferencing application, the compressed audio and video bit streams are transmitted through a network to the destination. Under the condition that the network is out of bandwidth, one cannot possibly transmit all the compressed data to the decoder in a timely manner. One easy solution is to increase the quantization step, such that quantizer generates a lower-quality output, and the bandwidth requirement is lower accordingly. This quantizer which changes adaptively is called an *adaptive quantizer.*

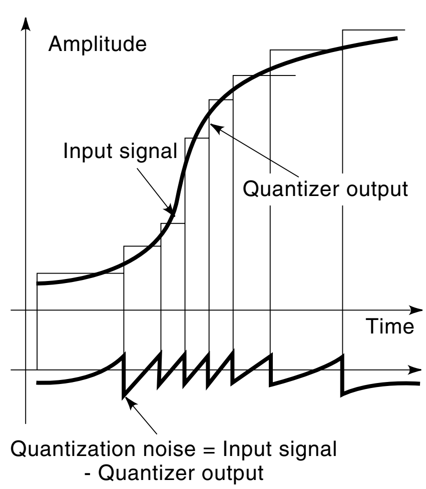

Figure 9. Quantization and quantization noise.

VECTOR QUANTIZATION

We have just introduced different ways of quantizing the output of a source. In all cases we discussed, the quantizer inputs were scalar values. In other words, the quantizer takes a single output sample of the source at a time and converts it to a quantized output. This type of quantizer is called a *scalar quantizer.*

Consider a case where we want to encode a consecutive sequence of samples from a stationary source. It is well-known from Shannon information theory that encoding a block of samples is more efficient than encoding each individual sample separately. In other words, during the quantization stage we wish to generate a representative index for a block of samples instead of for each separate sample. The basic concept is to generalize the idea from quantizing one sample at a time to quantizing a set of samples at a time. The set of the samples is called a *vector,* and this type of quantization process is called *vector quantization.*

Vector quantization is one of the most popular lossy data compression techniques. It is widely used in image, audio, and speech compression applications. The most popular vector quantization is *fixed-length vector quantization.* In the quantization process, consecutive input samples are grouped into fixed-length vectors first. As an example, we can group L samples of input speech as one L-dimensional vector, which forms the input vector to the vector quantizer. For a typical vector quantizer, both the encoder and the decoder share a common codebook, $\boldsymbol{C} = \{\boldsymbol{c}_i;\, i = 1, \ldots, N\}$, which can be predefined, fixed, or changed adaptively. Each entry of the codebook, $\boldsymbol{c}_i$, is called a *code-vector,* which is carefully selected as one of N representatives of the input vectors. Each code vector, $\boldsymbol{c}_i$, is also assigned an index, i. During the quantization stage the input vector, $\boldsymbol{x}$, is compared against each code-vector, $\boldsymbol{c}_i$, in the codebook. The "closest" code-vector, $\boldsymbol{c}_k$, is then selected as the representative code-vector for the input vector, and the corresponding index, k, is transmitted to the receiver. In other words, $\boldsymbol{c}_k$ is selected as the representative code-vector if

$$d(\boldsymbol{x}, \boldsymbol{c}_k) \le d(\boldsymbol{x}, \boldsymbol{c}_i) \qquad \text{for all } \boldsymbol{c}_i \in \boldsymbol{C} \tag{9}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_L)$ is the L-ary input vector and $\boldsymbol{C} = \{\boldsymbol{c}_i;\, i = 1, \ldots, N\}$ is the shared codebook, with ith code-vector, $\boldsymbol{c}_i$. The idea of vector quantization is identical to that of scalar quantization, except the distortion is measured on an L-dimensional vector basis. In Fig. 10 we show an example of a two-dimensional vector space quantized by a vector quantizer with $L = 2$, and $N = 16$. The code-vector $\boldsymbol{c}_k$ represents the input vector if it falls into the shaded vector space where Eq. (9) is satisfied. Since the receiver shares the same codebook with the encoder, and with received index, k, the decoder can easily retrieve the same representative code-vector, $\boldsymbol{c}_k$.

How do we measure the closeness, $d(\boldsymbol{x}, \boldsymbol{y})$, or distortion, between two L-ary vectors, $\boldsymbol{x}$ and $\boldsymbol{y}$, during the vector quantization process? The answer is dependent on the application. A *distortion measure* usually quantifies how well a vector quantizer can perform. It is also critical to the implementa-

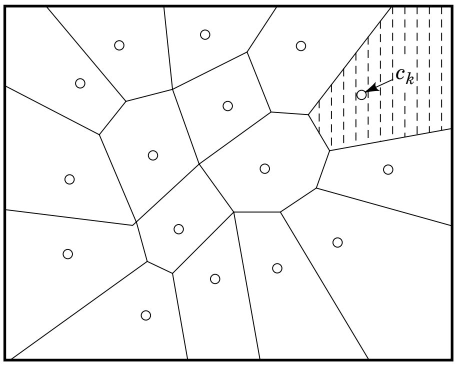

Figure 10. Two-dimensional vector space quantized by a vector quantizer.

tion of the vector quantizer, since measuring the distortion between two L-dimensional vectors is one of the most computationally intensive parts of the vector quantization algorithm. There are several ways of measuring the distortion. The most widely used distortion measure is the *mean square error* (MSE), which is defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{L} \sum_{i=1}^{L} (x_i - y_i)^2$$

Another popular distortion measure is the *mean absolute difference* (MAD), or *mean absolute error* (MAE), and it is defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{L} \sum_{i=1}^{L} |x_i - y_i|$$

There are various ways of generating the vector quantization codebook. Each method generates the codebook with different characteristics. The *LBG algorithm* (11) or the generalized Lloyd algorithm, computes a codebook with minimum average distortion for a given training set and a given codebook size. Tree-structured VQ (vector quantitization) imposes a tree structure on the codebook such that the search time is reduced (12,13,14). *Entropy-constrained vector quantization* (ECVQ) minimizes the distortion for a given average codeword length rather than a given codebook size (15). *Finite-state vector quantization* (FSVQ) can be modeled as a finite-state machine where each state represents a separate VQ codebook (16). *Mean/residual VQ* (M/RVQ) predicts the original image based on a limited data set, and then forms a residual by taking the difference between the prediction and the original image (17). Then the data used for prediction are coded with a scalar quantizer, and the residual is coded with a vector quantizer.

TRANSFORM CODING

We just considered the vector quantization, which effectively quantizes a block of data called a vector. Suppose that we have a reversible orthogonal transform, T, that transforms a block of data to a transform domain with the transform pair as

$$\boldsymbol{y} = T(\boldsymbol{x})$$
$$\boldsymbol{x} = T^{-1}(\boldsymbol{y})$$

where $\boldsymbol{x}$ is the original data block and T^{-1} is the inverse transform of T. In the transform domain we refer to the components of $\boldsymbol{y}$ as the transform coefficients. Suppose that the transform T has the characteristic that most of the transform coefficients are very small. Then the insignificant transform coefficients need not to be transmitted to decoder and can be eliminated during the quantization stage. As a result very good compression can be achieved with the transform coding approach. Figure 11 shows a typical lossy transform coding data compression system.

In Fig. 11 the input data block, $\boldsymbol{x}$, passes through the forward transform, T, with transform coefficients, $\boldsymbol{y}$, as its output. T has the characteristics that most of its output, $\boldsymbol{y}$, are small and insignificant and that there is little statistical correlation among the transform coefficients, which usually results in efficient compression by simple algorithms. The transform coefficients, $\boldsymbol{y}$, are quantized by the quantizer, Q. Small and insignificant coefficients have a zero quantized value; therefore only few nonzero coefficients need to be coded and transmitted to the decoder. For the best compression ratio, efficient entropy coding can be applied to the quantized coefficients at the coding stage. After receiving the signal from the network, the decoder decodes and inverse quantizes the received signal and reconstructs the transform coefficients, $\hat{\boldsymbol{y}}$. The reconstructed transform coefficients passes through the inverse transform, T^{-1}, which generates the reconstructed signal, $\hat{\boldsymbol{x}}$.

In general, transform coding takes advantage of the linear dependency of adjacent input samples. The linear transform actually converts the input samples to the transform domain for efficient quantization. In the quantization stage the transform coefficients can be quantized with a scalar quantizer or a vector quantizer. However, bit allocation among transform coefficients is crucial to the performance of the transform coding. A proper bit allocation at the quantization stage can achieve the output with a good fidelity as well as a good compression ratio.

There are quite a few transform coding techniques. Each has its characteristics and applications. The discrete Fourier transform (DFT) is popular and is commonly used for spectral analysis and filtering (18). Fast implementation of the DFT, also known as fast Fourier transform (FFT), reduces the transform operation to $n(n \log_2 n)$ for an n-point transform (19). The Karhunen–Loeve transform (KLT) is an optimal transform in the sense that its coefficients contain a larger fraction of the total energy compared to any other transform (20). There is no fast implementation of the KLT, however,

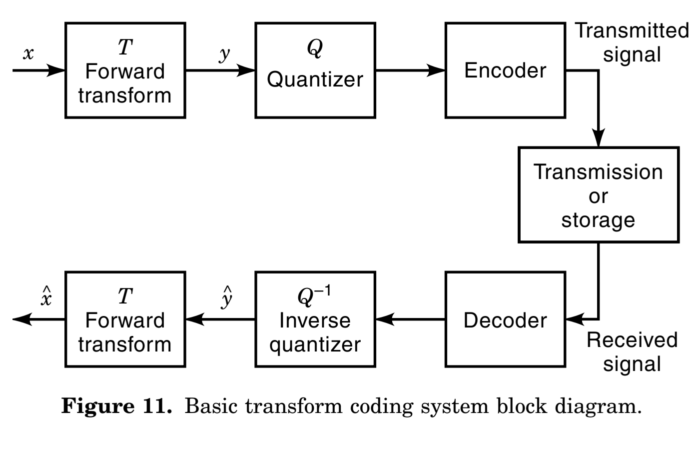

Figure 11. Basic transform coding system block diagram.

and its basis functions are target dependent. Because of this the KLT is not widely used. The Walsh–Hadamard transform (WHT) offers a modest decorrelating capability, but it has a very simple implementation (21). It is quite popular, especially for hardware implementation.

Transform coding plays a very important role in the recent lossy compression history. In the next section we will introduce the discrete cosine transform (DCT), which is the most popular transform for transform coding techniques.

DISCRETE COSINE TRANSFORM

The most important transform for transform coding is the discrete cosine transform (DCT) (22). The one-dimensional DCT F of a signal f is defined as follows (23,24):

$$F(k) = \sqrt{\frac{2}{N}}\, c(k) \sum_{j=0}^{N-1} f(j) \cos\left[\frac{(2j+1)k\pi}{2N}\right], \quad k - 0, 1, 2, 3, \ldots, N-1$$

where $c(0) = 1/\sqrt{2}$ and $c(k) = 1$ for $k \neq 0$. The inverse DCT (IDCT) is given by

$$f(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} c(k)F(k) \cos\left[\frac{(2n+1)k\pi}{2N}\right], \quad n = 0, 1, 2, 3, \ldots, N-1$$

A two-dimensional DCT for an image is formed by first taking the one-dimensional DCT of all rows of an image, and then taking the one-dimension DCT of all columns of the resulting image.

The DCT has fast implementations with a computational complexity of $O(n \log n)$ for an n-point transform. It has higher compression efficiency, since it avoids the generation of spurious spectral components. The DCT is the most widely used transform in transform coding for many reasons. It has superior energy compaction characteristics for most correlated source (25), especially for Markov sources with high correlation coefficient ρ,

$$\rho = \frac{E[\mathbf{x}_n \mathbf{x}_{n+1}]}{E[\mathbf{x}_n^2]}$$

where E denotes expectation. Since many sources can be modeled as Markov sources with a high correlation coefficient value, the superior energy compaction capability has made the DCT the most popular transform coding technique in the field of data compression. The DCT also tends to reduce the statistical correlation among coefficients. These properties make DCT-based lossy compression schemes very efficient. In addition the DCT can be implemented with reasonably low complexity. Because of this the DCT transform coding technique is widely used for both image and audio compression applications. The JPEG (1) and MPEG (2,3) published by ISO, and H.261 (4) and H.263 (5) published by ITU, are based on DCT transform coding compression techniques.

SUBBAND CODING

In the last section we introduced transform coding, which converts the input samples to the transform domain. Quantization and bit allocation are applied to the transform coefficients in the transform domain. One of the drawbacks of transform coding is that it has high computational complexity. Now we introduce another compression technique—*subband coding,* which usually has lower complexity than transform coding.

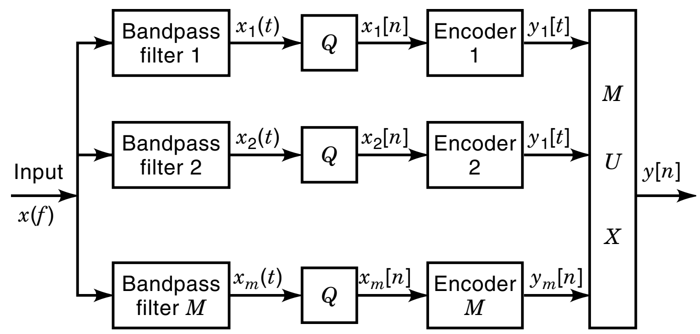

Figure 12. Block diagram of a typical subband coder.

Just like transform coding, subband coding uses a frequency domain approach. The block diagram of a typical subband encoder is illustrated in Fig. 12. The input signal, $x(t)$, is first filtered by a bank of M bandpass filters. Each bandpass filter produces a signal, $x_k(t)$, with limited ranges of spatial frequencies. Each filtered signal is followed by a quantizer and a bandpass encoder, which encodes the signal, $x_k(t)$, with different encoding techniques according to the properties of the subband. It may be encoded with different bit rates, quantization steps, entropy codings, or error distributions. The coding techniques we introduced in the previous sections, such as the vector quantization and entropy coding, are often used at the encoder. Finally the multiplexer combines all the subband coder output, $y_k[n]$, together and sends it through the communication channel to the decoder.

A subband decoder has the inverse stages of its encoder, as shown in Fig. 13. When a signal, $\hat{y}[n]$, is received from the communication channel, it goes through demultiplexing, decoding, and bandpass filtering prior to subband addition.

Subband coding has many advantages over other compression techniques. By controlling the bit allocations, quantization levels, and entropy coding separately for each subband, we can fully control the quality of the reconstructed signal. For this reason we can fully utilize the bandwidth of the communication channel. With an appropriate subband coding technique, we can achieve a good reconstruction signal quality, along with good compression. To take an example, for

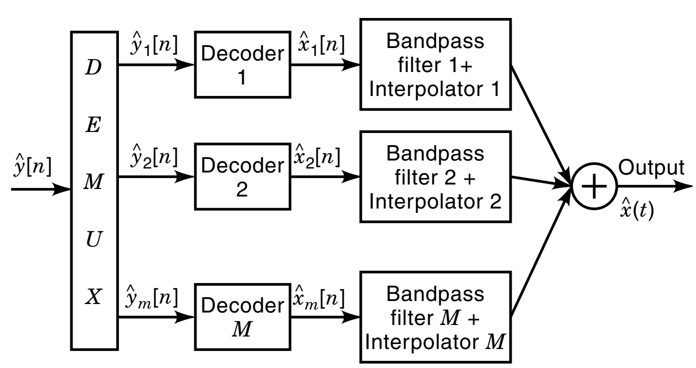

Figure 13. Subband decoder.

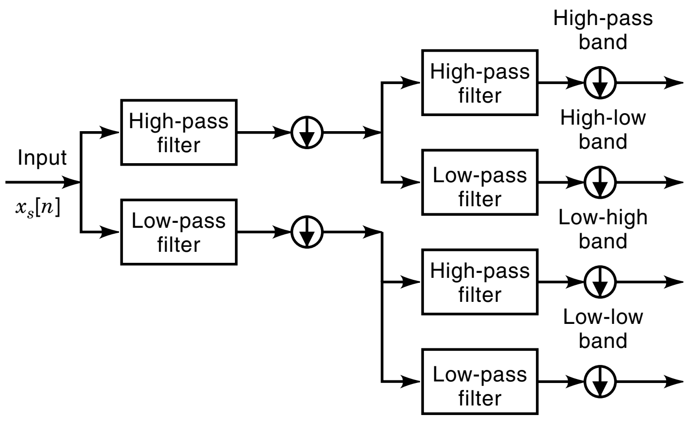

Figure 14. Four-band filter bank for uniform subband coding.

audio and speech applications low-frequency components are usually critical to the reconstructed sound quality. The subband coding technique enables the encoder to allocate more bits to lower subbands, and to quantize them with finer quantization steps. As a result the reconstructed data retains higher fidelity and higher signal-to-noise ratio (SNR).

A critical part of subband coding implementation is the filter bank. Each filter in the filter bank isolates certain frequency components from the original signal. Traditionally the most popular bandpass filter used in subband coding consisted of cascades of *low-pass filters* (LPFs) and *high-pass filters* (HPFs). A four-band filter bank for uniform subband coding is shown in Fig. 14. The filtering is usually accomplished digitally, so the original input is the sampled signal. The circled arrows denote down sampled by 2, since only half the samples from each filter are needed. The total number of samples remains the same. An alternative to a uniform subband decomposition is to decompose only the low-pass outputs, as in Fig. 15. Here the subbands are not uniform in size. A decomposition of this type is an example of a critically sampled *pyramid decomposition* or *wavelet decomposition* (26). Two-dimensional wavelet codes are becoming increasingly popular for image coding applications and include some of the best performing candidates for JPEG-2000.

Ideally the filter bank in the encoder would consist of a low-pass and a high-pass filter set with nonoverlapping, but contiguous, unit gain frequency responses. In reality the ideal filter is not realizable. Therefore, in order to convert the full spectrum, it is necessary to use filters with overlapping frequency response. As described earlier, the overlapping frequency response will cause aliasing. The problem is resolved by using exact reconstruction filters such as the *quadrature mirror filters* (QMF), as was suggested by Princey and Bradley (27), Croisier, Easteban, and Galand (28), Johnson (29), and Smith and Barnwell (30).

The idea of QMF is to allow the aliasing caused by overlapping filters in the encoder (analysis filter) canceled exactly by the filter banks in the decoder (synthesis filter). The filters are designed such that the overall amplitude and phase distortion is minimized. Then overall subband coding system with QMF filter bank is almost aliasing-free.

PREDICTIVE CODING

In this section we introduce another interesting compression technique—predictive coding. In the predictive coding systems, we assume a strong correlation between adjacent input data, which can be scalar, vector, or even block samples. There are many types of predictive coding systems. The most popular one is the linear predictive coding system based on the following linear relationship:

$$\hat{x}[k] = \sum_{i=0}^{k-1} \alpha_i x[i] \tag{10}$$

where the $x[i]$ are the input data, the α_i are the prediction coefficients, and $\hat{x}[k]$ is the predicted value of $x[k]$. The difference between the predicted value and the actual value, $e[k]$, is called the *prediction error:*

$$e[k] = x[k] - \hat{x}[k] \tag{11}$$

It is found that the prediction error usually has a much lower variance than the original signal, and is significantly less correlated. It has a stable histogram that can be approximated by a Laplacian distribution (31). With linear predictive coding, one can achieve a much higher SNR at a given bit rate. Equivalently, with linear predictive coding, one can reduce the bit rate for a given SNR. There are three basic components in the predictive coding encoder. They are predictor, quantizer, and coder, as illustrated in Fig. 16.

As shown in Fig. 16, the predicted signal, $\hat{x}[k]$, is subtracted from the input data, $x[k]$. The result of the subtraction is the prediction error, $e[k]$, according to Eq. (11). The prediction error is quantized, coded, and sent through communication channel to the decoder. In the mean time the predicted signal is added back to quantized prediction error, $e_q[k]$, to create reconstructed signal, $\tilde{x}$. Notice that the predictor makes the prediction according to Eq. (10), with previously reconstructed signal, $\tilde{x}$'s.

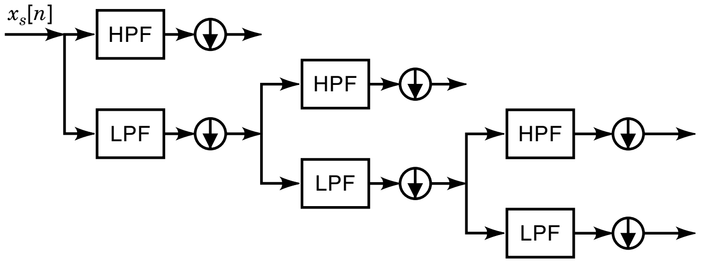

Figure 15. Filter bank for nonuniform subband coding.

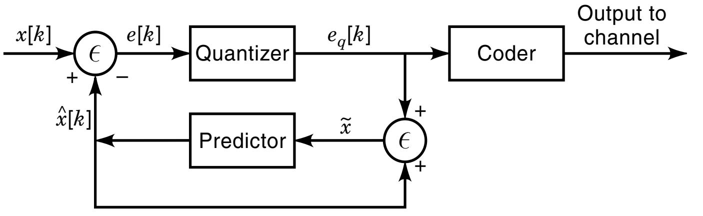

Figure 16. Block diagram of a predictive coder.

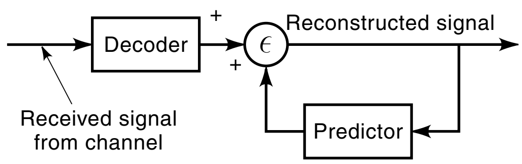

Figure 17. Predictive coding decoder.

Just like the encoder, the predictive coding decoder has a predictor, as shown in Fig. 17, which also operates in the same way as the one in the encoder. After receiving the prediction error from the encoder, the decoder decodes the received signal first. Then the predicted signal is added back to create the reconstructed signal. Even though linear prediction coding is the most popular predictive coding system, there are many variations. If the predictor coefficients remain fixed, then it is called *global prediction*. If the prediction coefficients change on each frame basis, then it is called *local prediction*. If they change adaptively, then it is called *adaptive prediction*. The main criterion of a good linear predictive coding is to have a set of prediction coefficients that minimize the mean-square prediction error.

Linear predictive coding is widely used in both audio and video compression applications. The most popular linear predictive codings are the *differential pulse code modulation* (DPCM) and the *adaptive differential pulse code modulation* (ADPCM).

RATE DISTORTION THEORY

In the previous sections we have briefly introduced several lossy data compression techniques. Each of them has some advantages for a specific environment. In order to achieve the best performance, one often combines several techniques. For example, in the MPEG-2 video compression, the encoder includes a predictive coder (motion estimation), a transform coder (DCT), an adaptive quantizer, and an entropy coder (run-length and Huffman coding). In this section we consider how well a lossy data compression can perform. In other words, we explore the theoretical performance trade-offs between fidelity and bit rate.

The limitation for lossless data compression is straightforward. By definition, the reconstructed data for lossless data compression must be identical to the original sequence. Therefore lossless data compression algorithms need to preserve all the information in the source data. From the lossless source coding theorem of Shannon information theory, we know that the bit rate can be made arbitrarily close to the entropy rate of the source that generated the data. So the entropy rate, defined as the entropy per source symbol, is the lower bound of size of the compressed data.

For lossy compression, distortion is allowed. Suppose that a single output X of a source is described by a probability density source function $f_x(x)$ and that X is quantized by a quantizer q into an approximate reproduction $\hat{x} = q(x)$. Suppose also that we have a measure of distortion $d(x, \hat{x}) \geq 0$ such as a square error $|x - \hat{x}|^2$ that measures how bad $\hat{x}$ is as a reproduction of x. Then the quality of the quantizer q can be quantized by the *average distortion*

$$D(q) = Ed(x, q(x)) = \int f_x(x) d(x, q(x))\, dx$$

The *rate* of the quantizer $R(q)$ has two useful definitions. If a fixed number of bits is sent to describe each quantizer level, then

$$R(q) = \log_2 M$$

where M is the number of possible quantizer outputs. On the other hand, if we are allowed to use a varying number of bits, then Shannon's lossless coding theorem says that

$$R(q) = H(q(x))$$

The entropy of the discrete quantizer output is the number of bits required on the average to recover $q(x)$. Variable length codes can provide a better trade-off of rate and distribution, since more bits can be used on more complicated data and fewer bits on low-complexity data such as silence or background. Whichever definition is used, we can define the *optimal performance* at a given bit rate by

$$\Delta(r) = \min_{q\,:\,R(q)\leq r} D(q)$$

By the *operational distortion-rate function,* or by the dual function,

$$R(d) = \min_{q\,:\,D(q)\leq d} R(q)$$

That is, a quantizer is *optimal* if it minimizes the distortion for a given rate, and vice versa. In a similar fashion we could define the optimal performance $\Delta_k(r)$ or $\mathbf{R}_k(d)$ using vector quantizers of dimension k as providing the optimal rate-distortion trade-off. Last we could ask for the optimal performance, say $\Delta_\infty(r)$ or $\mathbf{R}_\infty(d)$, when one is allowed to use quantizers of arbitrary length and complexity:

$$\Delta_\infty(r) = \min_k \Delta_k(r)$$
$$\mathbf{R}_\infty(d) = \min_k \mathrm{R_k}(d)$$

where the Δ_k and $\mathbf{R}_k$ are normalized to distortion per sample (pixel) and bits per sample (pixel). Why study such optimizations? Because they give an unbeatable performance bound to all real codes and they provide a benchmark for comparison. If a real code is within 0.25 dB of $\Delta_\infty(r)$, it may not be worth any effort to further improve the code.

Unfortunately, Δ_∞ and $\mathbf{R}_\infty$ are not computable from these definitions, the required optimization is too complicated. Shannon rate-distortion theory (32) shows that in some cases Δ_∞ and $\mathbf{R}_\infty$ can be found. Shannon defined the (Shannon) rate-distortion function by replacing actual quantizers by random mappings. For example, a first-order rate-distortion function is defined by

$$R(d) = \min I(X, Y)$$

where the minimum is over all conditional probability density functions $f_{Y|X}(y|x)$ such that

$$Ed(X, Y) = \int\int f_{Y|X}(y|x) f_X(x) d(x, y)\, dxdy$$
$$\leq d$$

The dual function, the Shannon distortion-rate function $D(r)$ is defined by minimizing the average distortion subject to a constraint on the mutual information. Shannon showed that for a memoryless source that

$$\mathbf{R}_{\infty}(d) = R(d)$$

That is, $R(d)$ provides an unbeatable performance bound over all possible code, and the bound can be approximately achieved using vector quantization of sufficiently large dimension.

For example, if the source is a memoryless Gaussian source with zero mean and variance σ^2, then

$$R(d) = \frac{1}{2}\log\frac{\sigma^2}{d}, \qquad 0 \le d \le \sigma^2$$

or equivalently,

$$D(r) = \sigma^2 e^{-2R}$$

which provides an optimal trade-off with which real systems can be compared. Shannon and others extended this approach to sources with memory and a variety of coding structures.

The Shannon bounds are always useful as lower bounds, but they are often over conservative because they reflect only in the limit of very large dimensions and hence very complicated codes. An alternative approach to the theory of lossy compression fixes the dimension of the quantizers but assumes that the rate is large and hence that the distortion is small. The theory was developed by Bennett (33) in 1948 and, as with Shannon rate-distortion theory, has been widely extended since. It is the source of the "6 dB per bit" rule of thumb for performance improvement of uniform quantizers with bit rate, as well as of the common practice (which is often misused) of modeling quantization error as white noise.

For example, the Bennett approximation for the optimal distortion using fixed rate scalar quantization on a Gaussian source is (34)

$$\delta(r) \cong \frac{1}{12} 6\pi\sqrt{3}\sigma^2 2^{-2R}$$

which is strictly greater than the Shannon distortion-rate function, although the dependence of R is the same. Both the Shannon and Bennett theories have been extremely useful in the design and evaluation of lossy compression systems.

ACKNOWLEDGMENTS

The author wishes to thank Professor Robert M. Gray of Stanford University for providing valuable information and enlightening suggestions. The author also wishes to thank Allan Chu, Chi Chu, and Dr. James Normile for reviewing his manuscript.

BIBLIOGRAPHY

1. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard,* New York: Van Nostrand Reinhold, 1993.
2. J. L. Mitchell, et al., *MPEG Video Compression Standard,* London: Chapman & Hall, 1997.
3. B. B. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2,* London: Chapman & Hall, 1997.
4. Recommendation H.261: *Video Codec for Audiovisual Services at $p \times 64$ kbits/s.* ITU-T (CCITT), March 1993.
5. Draft Recommendation H.263: *Video Coding for Low Bitrate Communication,* ITU-T (CCITT), December 1995.
6. Draft Recommendation G.723: *Dual Rate Speech Coder for Multimedia Communication Transmitting at 5.3 and 6.3 Kbits/s,* ITU-T (CCITT), October 1995.
7. Draft Recommendation G.728: *Coding of Speech at 16 Kbit/s Using Low-Delay Code Excited Linear Prediction (LD-CELP),* ITU-T (CCITT), September 1992.
8. R. N. Bracewell, *The Fourier Transform and Its Applications,* 2nd ed., New York: McGraw-Hill, 1978, pp.6–21.
9. R. N. Bracewell, *The Fourier Transform and Its Applications,* 2nd ed., New York: McGraw-Hill, 1978, pp. 204–215.
10. H. Nyquest, Certain topics in telegraph transmission theory, *Trans. AIEE,* **47**: 617–644, 1928.
11. Y. Linde, A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Commun.,* **28**: 84–95, 1980.
12. R. M. Gray, Vector quantization, *IEEE Acoust. Speech Signal Process.,* **1** (2): 4–29, 1984.
13. J. Makhoul, S. Roucos, and H. Gish, Vector quantization in speech coding, *Proc. IEEE,* **73**: 1551–1588, 1985.
14. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression,* Norwell, MA: Kluwer, 1992.
15. P. A. Chou, T. Lookabaugh, and R. M. Gray, Entropy-constrained vector quantization, *IEEE Trans. Acoust., Speech Signal Process.,* **37**: 31–42, 1989.
16. J. Foster, R. M. Gray, and M. O. Dunham, Finite-state vector quantization for waveform coding, *IEEE Trans. Inf. Theory,* **31**: 348–359, 1985.
17. R. L. Baker and R. M. Gray, Differential vector quantization of achromatic imagery, *Proc. Int. Picture Coding Symp.,* 1983, pp. 105–106.
18. W. K. Pratt, *Digital Image Processing,* New York: Wiley-Interscience, 1978.
19. E. O. Brigham, *The Fast Fourier Transform,* Englewood Cliffs, NJ: Prentice-Hall, 1974.
20. P. A. Wintz, Transform Picture Coding, *Proc. IEEE,* **60**: 809–820, 1972.
21. W. K. Pratt, *Digital Image Processing,* New York: Wiley-Interscience, 1978.
22. W. H. Chen and W. K. Pratt, Scene adaptive coder, *IEEE Trans. Commun.,* **32**: 224–232, 1984.
23. N. Ahmed, T. Natarajan, and K. R. Rao, Discrete cosine transform, *IEEE Trans. Comput.,* **C-23**: 90–93, 1974.
24. N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing,* New York: Springer-Verlag, 1975.
25. N. S. Jayant and P. Noll, *Digital Coding of Waveforms,* Englewood Cliffs, NJ: Prentice-Hall, 1984.
26. M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding,* Upper Saddle River, NJ: Prentice-Hall PTR, 1995.
27. J. Princey and A. Bradley, Analysis/synthesis filter bank design based on time domain aliasing cancellation, *IEEE Trans. Acoust. Speech Signal Process.,* **3**: 1153–1161, 1986.
28. A. Croisier, D. Esteban, and C. Galand, Perfect channel splitting by use of interpolation/decimation techniques, *Proc. Int. Conf. Inf. Sci. Syst.,* Piscataway, NJ: IEEE Press, 1976.
29. J. D. Johnson, A filter family designed for use in quadrature mirror filter banks, *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.,* Piscataway, NJ: IEEE Press, 1980, pp. 291–294.

30. M. J. T. Smith and T. P. Barnwell III, A procedure for designing exact reconstruction filter banks for tree structured subband coders, *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.,* Piscataway, NJ: IEEE Press, 1984.
31. A. Habibi, Comparison of *n*th-order DPCM encoder with linear transformation and block quantization techniques, *IEEE Trans. Commun. Technol.,* **19**: 948–956, 1971.
32. C. E. Shannon, Coding theorems for a discrete source with a fidelity criterion, *IRE Int. Convention Rec.,* pt. 4, **7**: 1959, 142–163.
33. A. Gersho, Asymptotically optimal block quantization, *IEEE Trans. Inf. Theory,* **25**: 373–380, 1979.
34. A. Gersho, *Principles of Quantization, IEEE Trans. Circuits Syst.,* **25**: 427–436, 1978.

Reading List

1. R. M. Gray and D. L. Neuhoff, Quantization, *IEEE Trans. Inf. Theory,* 1998.
2. M. Rabbani and P. W. Jones, *Digital Image Compression Techniques,* Tutorial Texts in Optical Engineering, vol. 7, Bellingham, WA: SPIE Optical Eng. Press, 1991.
3. J. L. Mitchell, et al., *MPEG Video Compression Standard,* London: Chapman & Hall, 1997.
4. B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2,* London: Chapman & Hall, 1997.
5. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard,* New York: Van Nostrand Reinhold, 1993.
6. T. M. Cover and J. A. Thomas, *Elements of Information Theory,* New York: Wiley, 1991.

KEN CHU
Apple Computers Inc.

DATA CONVERTERS. See ANALOG-TO-DIGITAL CONVERSION.

DATA DETECTION. See DEMODULATORS.