2

Vector Quantization Techniques in Speech Coding

ALLEN GERSHO and SHIHUA WANG University of California, Santa Barbara, California

KENNETH ZEGER University of Hawaii, Honolulu, Hawaii

1. VECTOR QUANTIZATION

It has become widely recognized in the past decade that a vector, i.e., an ordered set of signal samples or parameters, can be efficiently coded by matching the input vector to a similar pattern or *code vector* in a codebook. For a given input vector, the encoder simply identifies the address, or index, of the best matching code vector. The index, as a binary word, is then transmitted and the decoder replicates the corresponding code vector by a table lookup from a copy of the same codebook. Since the codebook contains a finite set of entries, the matching code vector is one of a finite set of possible approximations. In this sense, analog-to-digital conversion has been performed. The index is a digital code that allows the receiver to reproduce an approximation to the original. Hence, the operation of quantization has taken place directly on a vector. The vector components are not coded individually as in scalar quantization, but rather all at once. Considerable efficiency is achieved, fractional bit rates (bits per vector component) become possible, and the average distortion (e.g., average squared error per component) for a given bit rate is generally much reduced in comparison to scalar quantization of the components. Figure 1 illustrates the basic idea of vector

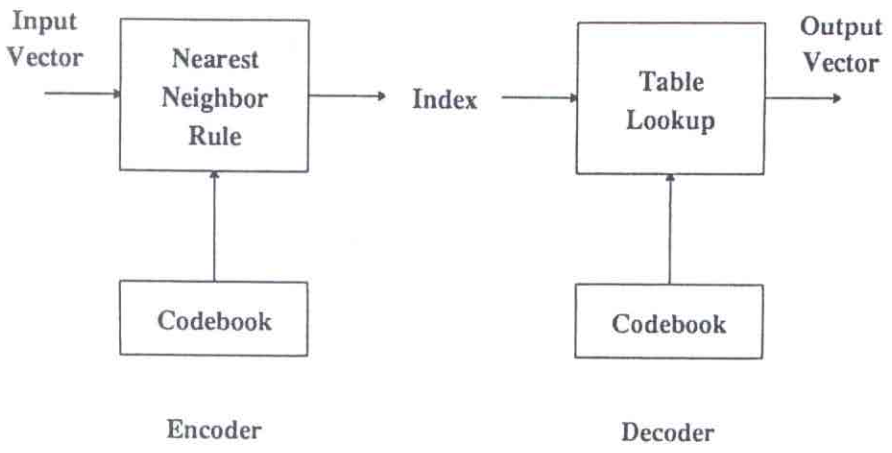

Figure 1 Vector quantization.

quantization (VQ). From this description, it is evident that VQ is in essence a pattern-matching algorithm; such an approach to pattern recognition was in use long before the age of digital electronics.

Perhaps the first application of pattern matching to speech coding dates back to Dudley (1958). The first major application of VQ to speech coding was reported by Buzo et al. (1980), who substantially reduced the bit rate of a linear predictive coding (LPC) vocoder by applying VQ to the LPC parameters. Subsequently VQ found its way into waveform coding as well and a generalization of DPCM using vector prediction together with VQ was developed. Today VQ is a well established and widely used technique. It has been applied to the efficient coding of LPC parameters, pitch predictor filter parameters, gain parameters, block waveform coding and coding of the excitation or residual signal in analysis-by-synthesis predictive coding techniques, such as VXC, CELP, and VAPC.

In this chapter we first review the fundamentals of vector quantization, including optimality and design considerations. Current applications of VQ to speech coding are then discussed, including waveform coding, analysis-by-synthesis predictive coding, and parameter coding. Important complexity issues inherent in VQ are examined, and finally, a discussion is presented of various recent techniques that are incorporated in VQ to combat some of the detrimental effects of channel noise.

1.1. Vector Quantization Fundamentals

Quantization plays a fundamental role in the coding of speech signals. Vector quantization is a generalization of scalar quantization to the quantization of a vector, an ordered set of real numbers. (For comprehensive discussions of VQ and its applications to speech coding, see Gray, 1984; Gersho, 1982; Abut et al., 1982; Gray et al., 1980, 1981; Jayant and Noll, 1984.) The jump from one dimension to multiple dimensions is a major step and allows a wealth of new ideas, concepts, techniques, and applications to arise that often have no counterpart in scalar quantization. While scalar quantization is primarily associated with analog-to-digital conversion, VQ deals with sophisticated digital signal processing in which the relevant input signals already have some form of digital representation and VQ is usually, but not exclusively, used for the purpose of data compression. Nevertheless, there are interesting parallels with scalar quantization and many of the structural models and analytical techniques used in VQ are natural generalizations of the scalar case.

A vector can be used to describe almost any type of *pattern*, such as a segment of a speech waveform, by forming a vector of samples from the waveform. An important example in speech coding arises when a set of parameters (forming a vector) is used to represent the spectral envelope of a speech sound. Vector quantization can be viewed as a form of pattern recognition in which an input pattern is "approximated" by one of a predetermined set of standard patterns; or in other words, the input pattern is matched with one of a stored set of templates or code words.

Vector quantization can also be regarded as a building block for a variety of complicated signal processing tasks, including classification and linear transformation. In such applications VQ can be viewed as a complexity-reducing technique because the reduction in bits can simplify the subsequent computations, sometimes permitting complicated digital signal processing to be replaced by simple table lookups. Thus VQ is far more than a generalization of scalar quantization. Its scope and implications are vast. In the last few years it has become an important technique in speech recognition as well as in speech compression. Recently, it has been used in speech enhancement and other speech processing applications. The importance of VQ continues to grow and the range of applications continues to expand.

A *vector quantizer* Q of dimension k and size N is formally a mapping from k-dimensional Euclidean space $\mathbf{R}^k$ into a finite set $\mathbf{C}$, containing N output or reproduction vectors from $\mathbf{R}^k$. Thus,

$$Q : \mathbf{R}^k \longrightarrow \mathbf{C}$$

where $\mathbf{C} = \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{N-1}\}$ and $\mathbf{y}_i \in \mathbf{R}^k$ for each $i \in \mathbf{J} = \{0, 1, \ldots, N-1\}$. The set $\mathbf{C}$ is called the *codebook* and has N elements, each a distinct vector in $\mathbf{R}^k$. The codebook is typically implemented as a table in a digital memory. The *resolution* or *rate* of the quantizer is $r = (\log_2 N)/k$, which measures the number of bits per vector component used to represent the input vector (and thus the degree of accuracy). It is important to recognize that for a fixed dimension k the resolution is determined by the size N of the codebook and not by the number of bits used to describe the code vectors stored in the codebook.

Associated with every N point vector quantizer is a *partition* of $\mathbf{R}^k$ into N regions or *cells*, R_i for $i \in \mathbf{J}$. The ith cell is defined by

$$R_i = \left\{ \mathbf{x} \in \mathbf{R}^k : Q(\mathbf{x}) = \mathbf{y}_i \right\}$$

or in terms of inverse image notation as $R_i = Q^{-1}(\mathbf{y}_i)$. From the definition of the cells it follows that the R_i are pairwise disjoint and their union covers all of $\mathbf{R}^k$.

A vector quantizer can be decomposed into two component operations, the vector *encoder* and the vector *decoder*. The encoder $\mathbf{E}$ is a mapping from $\mathbf{R}^k$ to the index set $\mathbf{J}$, and the decoder $\mathbf{D}$ maps the index set $\mathbf{J}$ into the reproduction set $\mathbf{C}$. Thus,

$$\mathbf{E} : \mathbf{R}^k \longrightarrow \mathbf{J} \quad \text{and} \quad \mathbf{D} : \mathbf{J} \longrightarrow \mathbf{R}^k$$

and the overall operation of VQ can be regarded as the cascade or composition of two operations:

$$Q(\mathbf{x}) = \mathbf{D}(\mathbf{E}(\mathbf{x}))$$

In the context of a speech coding system, the encoder of a vector quantizer performs the task of selecting an appropriately matching code vector $\mathbf{y}_i$ to approximate, or in some sense to describe or represent, an input vector $\mathbf{x}$. The vector $\mathbf{x}$ may represent a number of different possible speech coding parameters including, among others, consecutive speech samples, LPC coefficients, and prediction residual samples. The index i of the selected code vector is transmitted (as a binary word) to the receiver, where the decoder performs a table lookup procedure and generates the reproduction $\mathbf{y}_i$, the quantized approximation of the original input vector.

To conveniently assess the performance of a particular vector quantizer, we use a *distortion function* $d(\mathbf{x}, \mathbf{y})$ that indicates the distortion between any two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{R}^k$ where $\mathbf{x}$ is the original and $\mathbf{y}$ the reproduced vector. The "overall" measure of performance for a given quantizer is given by the average distortion between an input vector and the quantized output vector, $E[d(\mathbf{x}, Q(\mathbf{x}))]$, where E represents the expectation operator. As an example, the mth power distortion measure is defined by

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^m$$

The most common and perhaps natural distortion measure is the squared error function, defined by setting $m = 2$ above. In this case the average quantizer distortion is the *mean squared error*, $E\|\mathbf{x}-Q(\mathbf{x})\|^2$. Another useful (and more general) distortion function is the *weighted squared error* function given by

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^t\mathbf{W}(\mathbf{x}-\mathbf{y})$$

where $\mathbf{W}$ is a symmetric nonnegative definite matrix. In some cases $\mathbf{W}$ may be a function of the original vector, $\mathbf{x}$, to be quantized.

The *Itakura–Saito* distance function (Gray et al., 1980; O'Shaughnessy, 1987) computes a distortion between two input random vectors by using their spectral densities. It is defined by the following L_1 norm:

$$d(\mathbf{x},\mathbf{y}) = \left\| \frac{f_x}{f_y} - \ln\left(\frac{f_x}{f_y}\right) - 1 \right\|_1$$

where f_x and f_y are the spectral densities of the input random vectors $\mathbf{x}$ and $\mathbf{y}$, respectively. This measure can also be conveniently computed using autocorrelation functions.

An important issue for speech coding with vector quantization is the design and implementation of quantizers to meet performance objectives. That is, for a given source, distortion function, and constraint on the bit rate, what are the best vector quantizer encoding and decoding rules? In general there is no known method for explicitly specifying the optimal quantizer, although certain optimality conditions do exist.

For a general class of distortion functions, it is known that the best vector quantizer encoder for a given decoder satisfies the *nearest-neighbor rule*; that is, each input vector $\mathbf{x}$ is quantized to an output vector $Q(\mathbf{x})$ which is at least as close to $\mathbf{x}$ as any other vector in the codebook. If there happen to be two or more code vectors which are at least as close to $\mathbf{x}$ as any other code vector, then the value of $Q(\mathbf{x})$ can be chosen arbitrarily from among the set of nearest code vectors to $\mathbf{x}$.

If the squared-error distortion function is assumed, then for a given quantizer encoder, an optimal decoder must satisfy the *centroid condition*; that is, each code vector $\mathbf{y}$ must be the conditional mean, or centroid, of its corresponding cell R_i.

The problem of designing a vector quantizer for a particular source is of fundamental importance. If one assumes the nearest neighbor and centroid rules, then the design problem reduces to determining the best codebook for a given source (the nearest-neighbor rule is implicitly assumed). The generalized Lloyd algorithm (GLA) (Linde et al., 1980) utilizes these two necessary rules to perform an iterative descent procedure based on a finite training set representation of the source. During each iteration in the GLA each vector in the training set is compared with every vector in the current codebook to find the closest code vector, thus partitioning the training set into nearest-neighbor cells. The centroid of each partition cell is then computed and serves as a code vector for the next iteration, and the process is repeated.

For a fixed transmission rate r (bits per vector component) the size of a VQ codebook is 2^{kr}, where k is the dimension of the vector. Hence the complexity of a full codebook search increases exponentially as the vector dimension grows. This is one major drawback of using the GLA for VQ design and for using a full-search VQ encoder in general.

The GLA yields locally optimal codebooks that perform well in practice but may not perform as well as is theoretically possible. Some probabilistic methods have been developed that can find near–globally optimal VQ codebooks in a reasonable amount of computation time (Vaisey and Gersho, 1988; Cetin and Weerackody, 1988; Flanagan et al., 1989; Zeger and Gersho, 1989). One of the most successful of these probabilistic search techniques is *simulated annealing*, which iteratively "perturbs" a given codebook in a random manner. Each perturbed codebook is "accepted" if the quantizer improves or is probabilistically accepted if the quantizer performance degrades. The probability of accepting degraded codebooks is gradually reduced as the algorithm progresses, which eventually settles the codebook into a good stationary point, avoiding many poor local minima.

Other VQ design techniques have been developed with the purpose of reducing the computational complexity of the GLA and achieving better local minima. The splitting method (Gray, 1984) begins by constructing a codebook containing one code vector. It then "splits" the code vector into two new code vectors and designs a new codebook of size two with the GLA, using these two vectors as initial code vectors. At each stage in the algorithm, all of the previous code vectors are split and a new codebook is designed which is twice the size of the previous codebook. The design terminates when the desired power-of-two codebook size is reached. Other VQ design techniques include using a gradient descent (Chang and Gray, 1986), a conjugate gradient descent (Yair et al., 1990), and a learning algorithm related to neural networks (Yair et al., 1991; Nasrabadi and Feng, 1988).

Because of the large search complexity of ordinary VQ, there have been several research efforts to develop fast algorithms for the nearest-neighbor search process. Algorithms exploiting the constrained geometry of the encoding regions find nearest-neighbor code vectors within a time whose expected value is a logarithmic function of the codebook size (Friedman et al., 1977; Cheng and Gersho, 1986). Other fast search techniques are given in Cheng et al. (1984), Fukunaga and Narendra (1975), Soleymani and Morgerai (1987), and Vidal (1986).

For a given codebook size N, the transmission rate of a VQ system is $\log_2 N$ bits per vector. The output indexes of a given VQ encoder constitute a finite ensemble and have a certain associated probability distribution. In general, this distribution is nonuniform and hence entropy coding can be exploited. If a VQ encoder is followed by a variable-rate entropy coder, such as a Huffman coder, the output average bit rate of the encoder can be reduced while maintaining the same average distortion. This increases the performance of the system in the rate-distortion sense. An algorithm for optimal design of a vector quantizer with a constraint on the entropy of the VQ indexes is given in Chou et al. (1989a). The algorithm, known as *entropy-constrained VQ (ECVQ)*, closely resembles the GLA and assumes that the quantizer is followed by an ideal entropy coder, which in practice works well.

In principle, if the probability distribution $f_{\mathbf{X}}(\mathbf{x})$ of the input vector $\mathbf{X}$ is known, it is possible to evaluate the average distortion achieved for a particular vector quantizer. In practice, it is sufficient to estimate this figure from the training set as part of the codebook design process. On the other hand, it is of great interest to know the minimum average distortion attainable for an optimal vector quantizer for a statistically specified input and a given codebook size. Unfortunately, no general theory is available to provide an answer. However, there exists a useful and relatively simple formula for the squared Euclidean distance measure in the asymptotic case of *high resolution*, that is, as the codebook size N approaches infinity. It turns out that this formula is reasonably accurate for moderately high-resolution applications, and it provides considerable insight into the gain that VQ offers compared to scalar quantization. Specifically, we have:

$$D \approx N^{-2/k} C_k H_k(f_{\mathbf{x}})$$

where C_k is the coefficient of quantization and depends only on the vector dimension and

$$H_k(f_{\mathbf{X}}) = \|f_{\mathbf{X}}(\mathbf{x})\|_{k/(k+2)} = \left[\int_{-\infty}^{\infty} (f_{\mathbf{X}}(\mathbf{x}))^{k/(k+2)}\, d\mathbf{x}\right]^{(k+2)/k}$$

The constant C_k drops relatively little as the dimension increases from one to infinity. The equation clearly shows how the minimum distortion decreases with increasing codebook size. The dependence on the joint probability density of the input vector is not so easy to see, but it can be shown that as the statistical interdependence of the vector component grows, the distortion decreases as expected. In particular, in the case of a jointly Gaussian random vector, we get

$$H_k(f_{\mathbf{X}}) = 2\pi \left(\frac{k+2}{k}\right)^{(k+2)/2} (\det R_{\mathbf{x}})^{1/k}$$

where $R_{\mathbf{x}}$ is the covariance matrix of X. It is well known that the kth root of the determinant of the covariance matrix is a useful measure of the degree of correlation between the vector components. When the vector is a block of consecutive scalar samples of a stationary random process, $(\det R_{\mathbf{x}})^{1/k}$ approaches the minimum mean squared prediction error for one-step prediction with infinite memory.

Perhaps the simplest form of VQ for speech coding is block waveform coding or vector PCM (VPCM), as it is sometimes called. The digitized speech waveform is divided into consecutive blocks (or vectors) of length k, and each vector is encoded to the index of one of the k-dimensional vectors in a codebook. The speech waveform is thus converted into a sequence of binary data (code vector indexes) which can be transmitted or stored. The performance of this simple VQ technique is generally much lower than can be achieved when VQ is embedded in a more sophisticated system.

If the digitized speech waveform is immediately followed by a waveform predictor, then the output residual waveform can be vector quantized and an increase in performance generally results. The most common predictor is one employing scalar linear predictive coding but another possibility is vector linear prediction. A generalized version of scalar predictive quantization (as in DPCM) combines the use of vector prediction and VQ (Gersho and Cuperman, 1982; Gersho and Cuperman, 1985). A pth-order vector linear prediction $\hat{\mathbf{x}}_n$ of a k-dimensional vector $\mathbf{x}_n$, at time instant n, is of the form

$$\hat{\mathbf{x}}_n = \sum_{i=1}^{p} \mathbf{A}_i \mathbf{x}_{n-i}$$

where each $\mathbf{A}_i$ is a constant $k \times k$ matrix. The *prediction gain ratio* (in dB) in this case is defined as

$$10 \log_{10} \frac{E\|\mathbf{x}_n\|^2}{E\|\mathbf{x}_n - \hat{\mathbf{x}}_n\|^2}$$

where a zero-mean input is assumed. Vector predictive coding has also found applications to interframe coding of LPC parameters and to image coding.

2. GAIN-ADAPTIVE VECTOR QUANTIZATION

The wide dynamic range of a speech waveform contributes to the difficulty of directly applying VQ to blocks of samples. Vectors formed in this way will require an excessively large codebook to accommodate not only the wide variety of shapes of the waveform segment represented by the vector but also the wide range of amplitude levels that can arise. In scalar quantization methods for speech, the technique of gain-adaptive quantization has become widely used to reduce the inefficiency of directly coding samples that have a wide dynamic range. Its use in ADPCM is widely known.

The same concept has been extended to the vector case and has proved to be an effective means of combating the dynamic range problem for waveform coding (Chen and Gersho, 1987). In various forms, it is applicable not only for direct waveform coding of speech but also for coding subband signals generated by an analysis filter bank as in subband coding (Gersho et al., 1984).

In a typical application of gain-adaptive VQ, a codebook for gain-normalized speech vectors is first designed by a codebook design algorithm optimized specifically for this application. Input vectors are scaled in amplitude by a gain-normalizing factor which is adaptively generated by either forward or backward adaptation. Each normalized vector is then vector encoded and the received index is decoded to produce a corresponding code vector which is then scaled back to the normal level by using the same adaptive gain value as in the encoder. This effectively allows a much greater dynamic range for signal vectors than would be possible with conventional VQ with a reasonable codebook size.

If the gain is produced via forward adaptation, the input speech is buffered and an average gain value is measured, quantized, and then used for an entire frame consisting of several signal vectors. Side information is transmitted to the receiver to identify the needed gain value. Because one gain value serves several successive vectors, a relatively small bit rate is needed to specify the gain.

In the more interesting case of backward adaptation, no side information is needed and the gain control is generated as an *estimate* of the norm of the current input vector by operating on the past of the reconstructed vectors. The gain estimation operation can be viewed as a linear or nonlinear prediction operation and, in particular, LPC techniques can be applied to the random process consisting of the sequence of vector norms. Often it is convenient to work on the logarithm of the vector norm, the *log-norm*, in which case linear prediction methods can be applied to predict the current log-norm value from the past of the sequence of log-norms of the reconstructed vectors. One particularly important application of backward gain adaptive VQ is in low-delay speech coding (Cuperman et al., 1989; Chen, 1989), which will be discussed later in this chapter. It is expected that a forthcoming CCITT standard for 16 kb/s speech coding will incorporate a form of gain-adaptive VQ.

3. STRUCTURALLY CONSTRAINED VECTOR QUANTIZATION

Although it is not possible to quantize a vector better than by using an optimal full-search codebook, there are serious drawbacks to this owing to computational complexity and storage requirements. For a fixed transmission rate, both the storage and the search complexity of a vector quantizer grow exponentially with the vector dimension. For example, in simple VPCM coding of 8-kHz sampled speech at a rate of 1 bit/s, the size of the codebook is 2^k, where k is the vector dimension. The amount of time available to search the codebook grows linearly in proportion to k, whereas the codebook size grows exponentially in k. If we choose $k = 20$ then at most 2.5 ms would be available to search over one-million 20-dimensional vectors for a best match, requiring over 8 billion floating-point arithmetic operations per second, an infeasible task with existing signal processor chips.

There are, however, several suboptimal VQ coding schemes which provide reduced-complexity alternatives to full-search optimal VQ. Each such technique to be described in this section introduces some structural constraint on the VQ system. This means that the set of code vectors cannot have arbitrary locations as points in k-dimensional space but are distributed in a restricted manner that allows lower-complexity search methods for nearest neighbors. Some of the structurally constrained techniques do not even find the true nearest-neighbor code vectors. Hence the performance of these VQ codes are suboptimal due to both the constrained locations of the code vectors and the suboptimal search procedure. The constrained VQ techniques discussed in this section include tree-structured VQ, multistage VQ, and gain/shape VQ. All of these are applicable to speech compression and have been used in various speech coding schemes in the past.

3.1. Gain/Shape VQ

Gain/Shape VQ (*GSVQ*) is another structurally constrained VQ technique that can help reduce the inherent computational complexity of VQ. The basic premise of GSVQ is that many different input vectors will have similar "shapes" but will have varying "norms." This leads to the idea of storing in a codebook a finite set of typical shapes and storing in a scalar codebook a finite set of typical "gains." GSVQ was introduced in Buzo et al. (1980) and later optimized in Sabin and Gray (1984).

In the most basic form of GSVQ, the norm (or *gain*) $g = \|\mathbf{X}\|$ of each input vector $\mathbf{X}$ is extracted and encoded into $\hat{g}$ by a scalar quantizer, and the resulting unit norm *shape* vector $\mathbf{S} = \mathbf{X}/g$ is encoded into $\hat{\mathbf{S}}$ by a vector quantizer. The decoder generates the product $\hat{g}\hat{\mathbf{S}}$ as its approximation to the input vector $\mathbf{X}$. One method of quantizing the gain and shape is to find independently nearest-neighbor matches for them in their codebooks. It turns out that this encoding technique is not optimal for GSVQ. The optimal encoding scheme consists of first finding which shape vector $\hat{\mathbf{S}}$ in a unit-norm shape codebook maximizes the scalar product $\mathbf{X}^t\mathbf{S}$, where the superscript t denotes the transpose, and then finding which gain $\hat{g}$ in the scalar codebook best approximates $\mathbf{X}^t\hat{\mathbf{S}}$. This solution, however, assumes a shape codebook with unit-norm code vectors.

In the more general setting, where the shape code vectors need not have unit norm, slightly better performance can be attained but the optimal encoding strategy involves a full search of all possible products of gain and shape code vectors, thus eliminating the complexity reduction advantage.

3.2. Multistage VQ

Another complexity reduction VQ technique that can be used when codebook storage size must also be reduced is *multistage VQ (MSVQ)*, In MSVQ an input vector $\mathbf{X}$ is successively encoded by a series of quantizers. The input to the first quantizer is $\mathbf{X}$ and the output is $\hat{\mathbf{X}}$. The residual vector $\mathbf{X} - \hat{\mathbf{X}}$ is the input to the second quantizer. At each stage, the input to a quantizer is the residual vector from the previous quantizer. The decoder output is a reconstructed vector obtained as the sum of $\hat{\mathbf{X}}$ plus all of the residual code vectors from each stage. The basic structure of MSVQ is illustrated in Fig. 2.

If the ith stage has a codebook with N_i code vectors, then both the search complexity for each input vector and the storage requirement are equal to the sum of the N_i's, whereas the size of the equivalent codebook (i.e., size of a full search) is equal to the product of the N_i's. Thus the complexity and storage requirements are greatly reduced using MSVQ.

Generally, the components of the quantization error vector at each stage of MSVQ tend to be less statistically dependent on one another than those of the input vector at the same stage. Thus, the coding gain of the quantizers

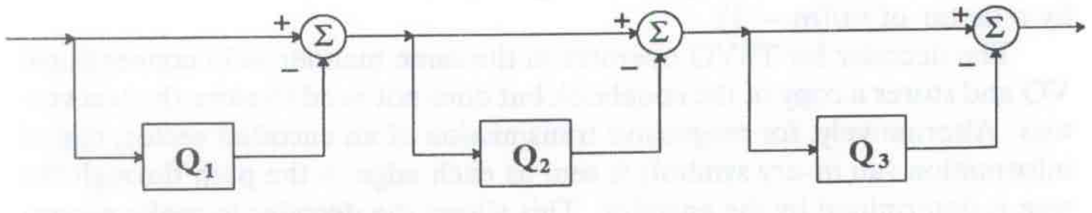

Figure 2 Multistage VQ.

tends to diminish in the later stages. In practice, usually only two or three stages are used in MSVQ because of diminishing returns.

3.3. Tree-Structured VQ

One of the most effective complexity reduction techniques for VQ, called tree-structured VQ (TSVQ), is based on a specially designed codebook that allows a successive approximation search procedure with a tree structure. The search for the best code vector is performed in stages; each stage eliminates a substantial number of candidate code vectors from consideration. The outcome of each stage determines how the search will proceed in the next stage.

An m-ary tree search begins at the root node of the tree and continues along a sequence of branches until a terminal node or *leaf* is reached. At each nonterminal node, the input vector is compared with m predesigned test vectors, each of which is associated with a particular child node connected by an edge from the parent node. A unique path through the tree is determined by following the edge at each stage associated with the test vector that is nearest to the input vector. The number of remaining candidate code vectors at each stage is equal to $1/m$ of those remaining at the previous stage.

An m-ary (balanced) tree of length d corresponds to a codebook of length $N = m^d$. In encoding an input vector, a particular path down the tree is determined that ends at a leaf. Each leaf node is associated with a particular code vector, which also serves as a test vector for the final stage of the search process. An index identifying the selected code vector is transmitted by the encoder to the decoder. Note that an m-ary tree with length $d = 1$ is equivalent to a full-search vector quantizer of size m.

The number of vector comparisons performed for a tree search is d, which equals $\log_m N$, a significant reduction from the N comparisons necessary in full-search VQ. The total search complexity for an m-ary tree is md because each of the d stages requires m vector comparisons. This is a substantial reduction from m^d, the total number of comparisons required in full-search VQ. In contrast, all of the test vectors must be stored in addition to the code vectors, increasing the storage requirement over conventional VQ by a factor of $m/(m-1)$.

The decoder for TSVQ operates in the same manner as in conventional VQ and stores a copy of the codebook but does not need to store the test vectors. Alternatively, for *progressive* transmission of an encoded vector, digital information (an m-ary symbol) is sent as each edge in the path through the tree is determined by the encoder. This allows the decoder to make successively improved approximations to the ultimate code vector by using the test vectors associated with the traversed edges as successive approximations to the input vector. In this case, the decoder must also store the test vectors and utilize the tree structure.

A standard method of designing tree-structured vector quantizers is somewhat analogous to the "splitting method" for designing ordinary vector quantizers. Initially, the entire training set is used in the generalized Lloyd algorithm to design a codebook with m code vectors. These m code vectors are the test vectors for the first level of the tree being constructed. The subset of the training set lying in the nearest-neighbor region of each test vector is then used by the GLA to design an m-vector codebook for the subregions. The m new code vectors at each of the m branches of the root node become the test vectors for the second stage of the tree. As this process is repeated, the tree grows and the quantization error of the resulting codebook decreases. TSVQ has been successfully used in speech coding applications in Makhoul et al. (1985), Buzo et al. (1980), and Gray and Abut (1982).

Another successful approach in tree-structured VQ is to use unbalanced trees, in which some paths from the root node to a leaf node are longer than others. Unbalanced trees offer a natural way to obtain variable-rate transmission codes, which can be useful in some applications. For example, in a binary tree one bit can be allocated for the choice between the two edges at any given node. Any path through the tree is then uniquely specified by a binary word with length in bits equal to the number of edges traversed to reach the leaf. The rate-distortion performance of an unbalanced tree-structured VQ can be significantly better than that of a balanced tree. The most common technique for designing unbalanced TSVQ coders is to grow a large balanced tree and then prune away edges. An efficient algorithm that prunes a tree in an optimal rate-distortion sense has been given by Chou et al. (1989b). An improved technique is given in Riskin and Gray (1991), in which an unbalanced tree is grown and then pruned back.

4. WAVEFORM VECTOR QUANTIZATION

The most direct and primitive application of VQ to speech coding is to perform block waveform coding, or VPCM, on the speech signal vector with a codebook of size N and dimension k (Abut et al., 1982; Gersho and Cuperman, 1983). VPCM is illustrated in Fig. 3, where the box labeled S/P performs serial-to-parallel conversion on each group of k samples and the box P/S performs the corresponding parallel-to-serial conversion.

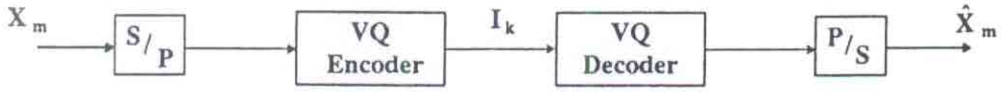

Figure 3 Vector PCM.

Although VPCM indeed offers a substantial coding gain over PCM (which does not exploit redundancy at all), it is nevertheless by itself a primitive coding technique and not of practical significance. For example, a bit rate of 8 kb/s based on $k = 8$ and $N = 256$ results in speech with a signal-to noise ratio in the neighborhood of 10 dB and with a substantial level of granular noise. The quality is roughly similar to that of delta modulation coding at 16 kb/s. This poor quality (by contemporary standards) is due largely to the limited vector dimension, implying that only local correlation among a small number of samples within a block is considered and no correlation between adjacent blocks is being exploited.

From distortion-rate theory we know that increasing the block size (vector dimension) for the same rate will asymptotically approach the ultimate performance limits. If we use dimension $k = 40$ and a rate of 1 bit per sample, we would indeed be exploiting substantial correlation and could expect a significant performance improvement. Indeed, it is reasonable to speculate that this would provide a very high-quality speech coding scheme. However, this implies the use of a VQ codebook of size N_2^{40}, which has an impractically high search complexity as well as storage requirement.

How do we circumvent this complexity barrier? Constrained codebook schemes such as tree structures or multiple stages of VQ can certainly reduce the complexity, but the price in performance to obtain the drastic reduction in complexity that is needed in the above example can be severe and may be overshadowed by alternative coding methods that do not use VQ.

Alternatively, we can greatly simplify the problem by making use of parametric modeling of speech (specifically linear prediction methods) as a redundancy removal technique. In particular, we can improve on direct waveform coding by performing *residual coding*, where the unpredictable residual is coded as a waveform and the prediction parameters are separately coded as side information. This idea is described below.

5. RESIDUAL CODING WITH VECTOR QUANTIZATION

When we subtract from the current speech sample a prediction of this sample based on past values of the speech signal, the prediction error waveform or *residual* is much less correlated and has much lower power. It also has a statistical distribution that is relatively invariant to changes in speech statistics. The residual signal is therefore an easier waveform to quantize with VPCM than is the original speech signal.

There are two primary ways to use VQ for coding the residual. One way is to extend the DPCM concept from sample-by-sample processing to process vectors of samples at a time. This approach leads to *vector DPCM* or *vector predictive coding*, where vector prediction is combined with VQ to code successive blocks of prediction residuals with the usual feedback structure corresponding to scalar DPCM (Gersho and Cuperman, 1985). An alternative approach is to use an open-loop configuration as shown in Fig. 4. Although both methods offer enhanced performance over direct VQ of the speech waveform, we focus here only on the second, which can be viewed as an evolutionary step toward analysis-by-synthesis predictive coding methods of particular importance in speech coding today.

In Fig. 4, the buffer stores a frame of speech samples so that analysis can be performed to determine the predictor parameters. The quantized parameters are used to control the time-varying prediction error filter A and, after transmission to the decoder, the corresponding synthesis filter B, which is the inverse of A. These filters may be based on long-term or pitch-based correlation as well as on short-term correlation. It is reasonable to expect that a codebook of dimension 40 with size $N = 1024$ might be adequate for coding this residual with reasonable accuracy. Thus, we may expect that 10 bits (rather than 40 bits) per vector will be sufficient for high-quality speech coding.

Although this coding method improves on VPCM of the speech waveform itself, there is one significant weakness to the residual encoding scheme of Fig. 4. The VQ encoder, as used here, performs a nearest-neighbor search to minimize the distortion between the input residual vector and the code

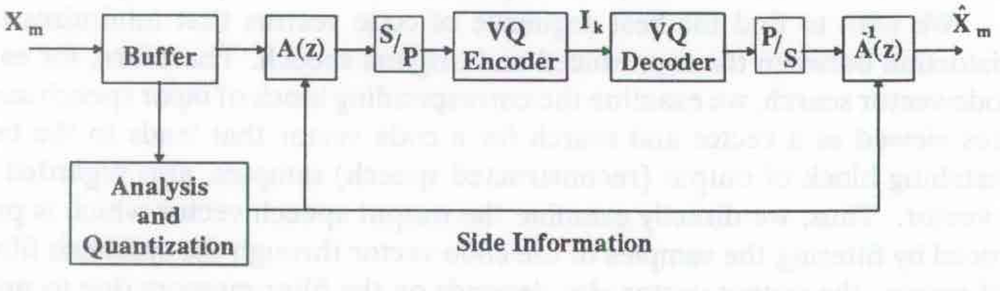

Figure 4 Residual coding structure with VQ.

vector. Regardless of the particular distortion measure used, there is something fundamentally wrong with this method: reducing the quantization error in the residual does not necessarily result in a corresponding reduction in the error between the reconstructed speech and the original speech. In fact, the quantization error produced by the VQ will be filtered by the synthesis filter so that the final synthesized output speech will have a spectrally altered quantization noise and may have a substantially higher power level than the error produced by the quantizer itself. We next describe a better way to search for the best code vector.

6. ANALYSIS-BY-SYNTHESIS PREDICTIVE CODING

Vector excitation coding (VXC) and code-excited linear prediction (CELP) are different names for a powerful and effective family of speech coding algorithms that are increasingly being adopted for a variety of applications at bit-rates ranging from 4.8 up to 16 kb/s. The term CELP is becoming associated with the specific coding algorithm adopted by the US government as a standard for 4.8 kb/s (Campbell et al., 1990). Henceforth we refer to the generic family of algorithms that perform predictive coding using analysis-by-synthesis techniques as VXC. Since this family of algorithms is described elsewhere in this book (see Chapter 5), in this section we focus only on the role of VQ in VXC algorithms and provide some insight into the operation of VXC by giving an evolutionary interpretation of the approach starting with basic waveform VQ, or vector PCM.

We now reexamine the residual coding structure and take an important conceptual leap that leads from the residual coding structure of Fig. 4 to VXC. Suppose that the codebook used in the scheme of Fig. 4 has an adequate set of candidate code vectors. In other words, if the encoder magically knew the right indexes to transmit to identify the sequence of code vectors to be used, the codebook would be adequate to provide the desired speech quality.

We wish to find the best sequence of code vectors that minimizes the distortion between the reproduced and original speech. Therefore, for each code vector search, we examine the corresponding block of *input* speech samples viewed as a vector and search for a code vector that leads to the best matching block of output (reconstructed speech) samples, also regarded as a vector. Thus, we directly examine the output speech vector which is produced by filtering the samples of the code vector through the synthesis filter. Of course, the output vector also depends on the filter memory due to prior inputs and this must also be taken into account. For each candidate code vector, the output speech vector is compared to the original input speech vector. The search process then identifies the best possible code vector.

It is striking to note that with this approach the residual VQ encoder and the residual signal itself are no longer needed! The new encoding system requires a replica of the decoder in order to compute what output speech vector would be produced for any given candidate code vector and given state (memory) of the synthesis filter. The resulting search process is called analysis by synthesis and the overall coding scheme is VXC.

In VXC it is helpful first to examine the operation of the VXC decoder, since it determines how the speech is synthesized (reconstructed) from transmitted data. The encoder is in a sense a servant of the decoder, since its job is to examine the original speech and determine what data must be delivered to the decoder. The VXC decoder structure remains the same as the decoder already shown in Fig. 4 for the residual VQ scheme. The VXC decoder receives and demultiplexes the data needed to specify the synthesis filter parameters and the excitation code vector. Each frame is divided into subframes of k samples corresponding to the duration of one excitation vector.

For each subframe, the decoder receives an index consisting of c excitation code bits which identifies the index that specifies one of 2^c excitation code vectors by means of a table-lookup procedure. This vector is serialized as k successive samples and is applied to the synthesis filter. The filter is clocked for k samples, feeding out the next k samples of the synthesized speech; then the filter is "frozen" until the next scaled excitation vector is available as the next input segment to the synthesis filter.

In many applications an adaptive postfilter is added to the decoder as a final postprocessing stage to enhance the quality of the recovered speech. This filter is usually adapted to correspond to the short-term spectrum of the speech (Chen and Gersho, 1986).

6.1. The VXC Encoder

The VXC encoder structure is shown in Fig. 5. We describe its operation in the simplest way, ignoring the many shortcuts and tricks which greatly reduce the complexity involved in the search process. For simplicity, the S/P and P/S blocks are omitted from Fig. 5; they are easily determined from the context. Also omitted are the frame buffer and the analysis and quantization of the synthesis filter parameters.

The encoder forms input speech vectors $\mathbf{V}(n)$ from successive blocks of k input speech samples. The task of the encoder is to determine the next c bits of data to be transmitted to the decoder so that the decoder will then be able

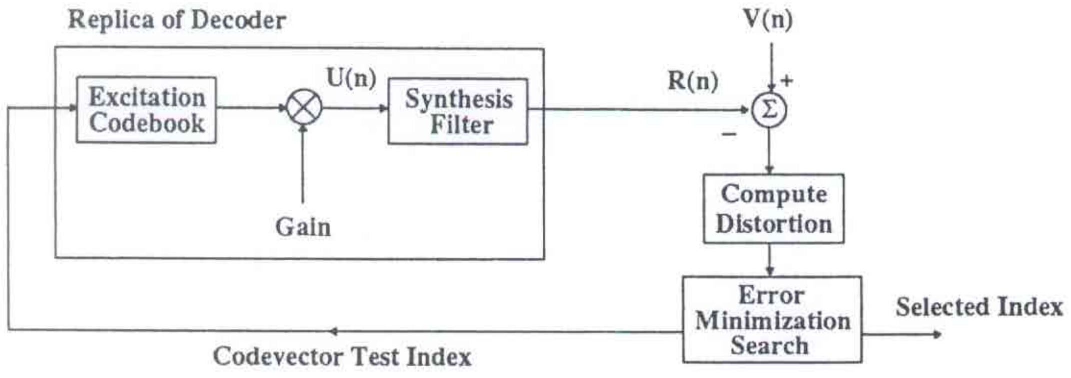

Figure 5 VXC encoder.

to synthesize a reconstructed output speech vector that closely approximates the original input speech vector.

This implies that the decoder must embody a replica of the decoder, as shown in Fig. 5, which can locally generate each of the $N = 2^c$ possible speech vector candidates that the decoder would produce for the same transmitted data values.

The error minimization search module sequentially generates new test indexes corresponding to particular code vectors. Each test index is fed to the replica of the decoder, which generates a synthesized speech vector that would be produced by the actual decoder if this index were actually transmitted. The replica decoder is initialized by setting the synthesis filter memory to the initial conditions that were determined after the prior search process was completed. Then the test index is applied to the excitation codebook, yielding an excitation vector. The excitation vector is then applied to the synthesis filter to produce the output vector $\mathbf{R}(n)$. The vector $\mathbf{R}(n)$ is then subtracted from the input speech vector $\mathbf{V}(n)$, and the distortion between these two vectors, i.e., the sum of the squares of the components of the difference vector, is computed by the distortion computation module. This error value is applied to the search module, which stores the distortion value, compares it with the lowest distortion value obtained so far in the current search process, and, if appropriate, updates the lowest distortion value and the corresponding vector index.

Speech samples emerging from the synthesis filter are configured into corresponding vectors of k contiguous samples.

Because the replica decoder is operating repeatedly in the search process, we must ensure that each candidate output speech vector, corresponding to a candidate index being tested, is produced under the conditions that will be present when the actual decoder generates the next output vector. After each test of a candidate index, the memory state of the replica decoder has changed and is no longer at the correct initial condition for the next test. Therefore, before generating each of these candidates, the memory in the replica decoder must also be reset to the correct initial conditions.

Several important modifications to this basic scheme are important to practical implementations but have not been described here. Among them is the use of a perceptual weighting filter to compute a more meaningful measure of distortion. Another is the use of GSVQ, with separate gain and shape codebooks instead of a single standard VQ codebook.

6.2. Vector Sum Excitation Codebooks

A question of practical importance is how the quality of a given VXC coder can be improved if more bits are made available and to which components of the coder these bits should be assigned for the maximum benefit. Generally, the best performance gain comes from increasing the codebook size. However, adding just one bit per code vector doubles the codebook size and the corresponding search complexity. Thus, computational and storage constraints quickly force one to limit the codebook size and lead to alternative designs in which the vector dimension is reduced and more bits are given to synthesis filter parameters. On the other hand, the use of specially constrained codebook structures offers the possibility of larger codebooks and significant performance improvements while maintaining tolerable complexity.

A novel and powerful technique for reducing the complexity of the excitation codebook search procedure was introduced by Gerson and Jasiuk (1990). Rather than have each of M code vectors be independently generated either randomly or by a design procedure, they design b "basis" vectors and then generate the $M = 2^b$ code vectors by taking binary linear combinations of the basis vectors. The resulting coding algorithm, a derivative of VXC, is called vector sum excited linear prediction (VSELP), and an 8-kb/s version of this algorithm has been adopted as a standard for the U.S. cellular mobile telephone industry. We next explain the basic idea of this technique for fast codebook search.

Let $\mathbf{v}_i$ denote the ith basis vector of a given set of b basis vectors. The code vectors are then formed as

$$\mathbf{u}_\theta = \sum_{i=1}^{b} \theta_i \mathbf{v}_i \cdot$$

by taking all possible linear combinations where $\theta_i = \pm 1$ for each i. Thus each binary-valued vector θ determines a particular code vector $\mathbf{u}_\theta$. Naturally, the b-bit binary word transmitted over the channel can simply correspond to a mapping of θ values with $+1$ being a binary 1 and -1 being a binary 0. Since the code vectors are so simply generated, only b basis vectors, rather than an entire codebook of M code vectors, need be stored.

This special codebook structure can be searched very efficiently. Instead of finding the vector output of the weighted synthesis filter for each of the M code vectors, only the filtered output of the b basis vectors needs to be determined because from these any synthesized output can be readily obtained by addition. Furthermore, the search for the optimal code vector is computationally simplified by noting that the mean squared error between the weighted input vector and a filtered code vector depends in a simple manner on the values of θ_i. By ordering the b-bit binary words in a Gray code, only 1 bit changes from one word to the next. This means that only a simple change is needed to compute the mean squared error for the next candidate code vector from the previous candidate code vector.

The basis vectors are designed from a training set of speech data by solving for each component of each basis vector in a large set of simultaneous equations. These equations are obtained by minimization of the total weighted squared error, which is a function of the basis vector components. The design is iterated in a closed-loop manner, starting with Gaussian random numbers for the components (Gerson and Jasiuk, 1990).

The vector sum approach can be augmented by combining it with MSVQ. MSVQ was applied to VXC in Davidson and Gersho (1988). It becomes more effective when each stage consists of a vector sum codebook and joint optimization of the gains for each stage is performed. The joint optimization becomes easy to implement by an orthogonalization procedure (Gerson and Jasiuk, 1990)

6.3. Low-Delay VXC

Vector excitation coding (VXC) combines techniques such as vector quantization, analysis-by-synthesis codebook searching, perceptual weighting, and linear predictive coding to successfully achieve good speech quality at low bit rates. However, one important aspect of coding has been ignored in the development of VXC or other conventional low-bit-rate excitation coding schemes; that is the *coding delay*. In fact, most existing speech coders with rates at or below 16 kb/s require high delays in their operation and cause various problems when they are applied to practical communication systems. In

VXC, a large net coding delay, excluding computational delays, results from the use of buffering needed to perform the LPC and open-loop pitch analysis. New methods have been proposed to adapt synthesis filters without the high coding delay mentioned above while maintaining the quality of encoded speech.

With the conventional VXC scheme described above, the synthesis filter is adaptively updated every frame by what is sometimes called forward adaptation, the process of recomputing and updating the desired filter parameters from the input speech. The use of forward adaptation has two disadvantages: it requires transmission of side information to the receiver to specify the filter parameters and it leads to a large encoding delay of at least one analysis frame due to the buffering of input speech samples. The input buffering and other processing typically result in a one-way codec delay of 50 to 60 ms. In certain applications in the telecommunications network environment, coding delays as low as 2 ms per codec are required. Recently, the CCITT adopted a performance requirement of less than 5 ms delay with a desired objective of less than 2 ms for candidate 16-kb/s speech coding algorithms to be considered for a new standard intended to achieve essentially the same quality as the 32-kb/s ADPCM standard, G.721. Such a low delay is not feasible with the established coders that are based on forward adaptive prediction coding systems. Although ADPCM satisfies the low-delay requirement, it cannot give acceptable quality when the bit rate is reduced to 16 kb/s.

An alternative solution is based on proposed backward adaptation configurations (Iyengar and Kabal, 1988; Watts and Cuperman, 1988; Cuperman et al., 1989). In a backward adaptive analysis-by-synthesis configuration, the parameters of the synthesis filter are not derived from the original speech signal but are computed by backward adaptation, extracting information only from the sequence of transmitted codebook indexes. Since both the encoder and decoder have access to the past reconstructed signal, side information is no longer needed for synthesis filters and the low-delay requirement can be met with a suitable choice of vector dimension. The remarkable feature of these low-delay coders is that only the indexes that specify excitation code vectors need to be transmitted. All side information usually associated with VXC-based algorithms is completely eliminated.

A particular version of low-delay coding with backward adaptation, known as low-delay CELP (LD-CELP), is under consideration for the 16-kb/s CCITT standardization (Chen 1989). This method uses *block* rather than *recursive* adaptation for updating a 50th-order LPC-based synthesis filter.

7. VECTOR QUANTIZATION OF LINEAR PREDICTOR COEFFICIENTS

Linear prediction is perhaps the most widely used technique in speech coding. It can remove near-sample or distant-sample correlations in a speech waveform. The former, which we usually call short-term or "spectral" prediction, effectively matches the spectral envelope of the signal. The latter, often referred to as "pitch" prediction, removes distant-sample redundancy in quasi-periodic speech segments, typical in voiced sounds.

In high-bit-rate waveform coding schemes, such as ADPCM and APC, short-term linear prediction results in an error signal of significantly lower power than the original waveform, and it can thus be encoded more accurately. In contrast, in low-bit-rate vocoding systems the short-term predictor is the basis of a time-varying filter modeling the vocal tract characteristics. In vector excitation systems this is augmented by long-term prediction to more fully remove redundancies in the error signal.

Since speech is a nonstationary process, it is necessary to update the prediction coefficients periodically. With forward adaptation, the predictor coefficients need to be transmitted to the decoder for synthesis. Therefore, the efficient quantization of these parameters becomes an important task in any adaptive prediction coding system.

LPC parameters have traditionally been individually (scalar) quantized. To obtain an acceptable quantization distortion with scalar quantization, usually 30 to 40 bits are needed to quantize a 10th order predictor (Tremain, 1982).

Since linear prediction can be viewed as a spectral matching process, distortion measures used for LPC quantization can be as effectively defined in the frequency domain as in the time domain. A number of spectral distortion measures have been discussed by Gray and Markel (1976). One of the most meaningful distortion measures, which has the property of being consistent with the residual energy minimization concept of the LPC analysis process, is the likelihood ratio:

$$d_{LR}\left(\frac{1}{A_M}, \frac{1}{A}\right) = \frac{r_x(0)}{\alpha_M} r_a(0) + 2\sum_{i=1}^{M} \frac{r_x(i)}{\alpha_M} r_a(i) - 1$$

where $r_x(i)$ and $r_a(i)$ are the autocorrelation sequences of the input frame and of the polynomial coefficients of $A(z)$, respectively, where $1/A(z)$ is any Mth order all-pole filter and $1/A_M(z)$ is the optimal LPC model of input frame $X(z)$. The term α_M is the residual energy resulting from inverse filtering

$X(z)$ with $A_M(z)$. The likelihood ratio is closely related to the Itakura–Saito distortion measure defined earlier.

Studies of this distortion measure (Juang et al., 1982) illustrate that the amount of parameter deviation allowed for a specified level of distortion is highly dependent on the entire input vector. This coupling effect is called *nonlinear dependence*. This property cannot be fully incorporated into the sensitivity approach for scalar quantization, in which the distortion is measured approximately by a sum of incremental distortions due to individual LPC parameters, without accounting for the coupling effect.

Vector quantization can remove redundancy in a vector effectively by making use of four interrelated properties of vector parameters: linear dependence (correlation), nonlinear dependence, the shape of the probability density function, and vector dimensionality itself. It is the existence of such significant nonlinear dependencies in the space of spectral parameters for speech which makes vector quantization truly attractive in this application. Kang and Coulter (1976) made the first effort in introducing the technique of vector quantization into LPC parameter quantization. Their work was mainly based on intuition, rather than on a well-defined mathematical distortion and methodology for designing an optimal codebook. Buzo et al. (1980) developed a mathematical framework and procedure for vector quantization of LPC parameters. Their experimental results showed that vector quantization permits a significant bit-rate savings over scalar quantization at the same distortion level, making efficient utilization of the statistics of coefficient vectors (clustering property) and of the coupling effect between coefficients. Wong et al. (1982) applied this framework to develop an 800-b/s vector-quantized LPC vocoder. Subjective evaluation showed that, overall, the 800-b/s vocoder approached a quality quite close to that of a 2400-b/s LPC vocoder. Their work has demonstrated that vector quantization appears to preserve more continuity from frame to frame than scalar quantization. Kang and Fransen (1984) also applied VQ to quantize the line spectral pairs in an 800-b/s vocoder which achieved a DRT (diagnostic rhyme test) score only 1.4 below that of a 2400-b/s LPC vocoder.

Most applications of vector quantization in speech coding have until recently focused on very low bit-rate coding, where the quantization resolution of LPC parameters with 10–12-bit vector quantizers are adequate to maintain acceptable speech intelligibility and quality. For some medium- and high-bit-rate coding systems, high-resolution LPC quantization, say 30 to 40 bits per frame, is demanded. If we improve the quantization resolution with straightforward VQ, the codebook search complexity and codebook storage will increase exponentially. For example, in a 10 bit/frame VQ for an Mth order linear prediction with the d_{LR} measure, $2^{10} \times (M + 1)$ multiply-adds are needed to quantize one input vector. If we use a 30 bit/frame LPC VQ, the computation for quantizing one input vector would be $2^{30} \times (M + 1)$ multiply-adds which is 10^6 times as many as the computations in the 10 bit/frame VQ. The complexity in both computation and storage has thwarted the direct application of vector quantization to high-resolution LPC parameter quantization.

The simplest solution for overcoming the difficulties in complexity is to use TSVQ. The major advantage of TSVQ is the substantial decrease in computational cost compared to full search with a relatively small decrease in performance. However, the storage requirements are increased and may be prohibitive. On the other hand, MSVQ reduces the storage requirement as well as search complexity but leads to a somewhat greater degradation in performance than does TSVQ, particularly when more than two stages are used. Nevertheless, MSVQ has been found to be a useful way to quantize log-area ratio (LAR) parameters.

An alternative to MSVQ is a product-code approach to VQ. In this type of VQ, the input space is partitioned into two or more subspaces; the vector dimension in each subspace is usually reduced and often considerably reduced. In fact, GSVQ is a special case of a product code. The subvector for each subspace is then quantized separately with a relatively small codebook designed for that subspace.

There are several ways to partition LPC parameter vectors, including vector splitting and band splitting. In the vector-splitting method, a vector is split into two or more subvectors and each subvector is quantized independently. For example, a vector with 14 LARs is divided into two vectors and two codebooks are used: a 6-dimensional, 10-bit codebook for parameters 1 to 4 and an 8-dimensional, 5-bit codebook for parameters 5 to 14. With a band-splitting method, proposed in Copperi and Sereno (1984), the spectral vector is split into two frequency bands and separate LPC model parameters for each band are vector quantized separately. Besides the advantage of reducing complexity in these methods, we gain the flexibility of controlling the bit allocation to the subvectors according to their perceptual importance in speech coding. For example, in the band-splitting method, we can make use of the fact that the human ear is generally less sensitive to high-frequency distortions than to low-frequency distortions and assign more bits for the lower-order vector.

In another VQ technique the LPC polynomial is represented as a cascade of two lower-order polynomials, a separate codebook is used for each polynomial, and quantization of the two components is jointly optimized using a likelihood-ratio distortion measure (Shoham, 1989). The LPC all-pole filter is represented as a cascade of two lower-order all-pole filters. The quantizer uses two codebooks to quantize each of the lower-order filters. Splitting the LPC filter into two filters reduces the coding complexity, while the efficiency of VQ is largely preserved.

7.1. Interframe Coding of LPC Parameters

So far, we have discussed only memoryless VQ for LPC parameter quantization, in which each input vector is quantized independently of any other input vector. Because of the nonstationary characteristics of speech, the LPC parameter set used for modeling the vocal tract transfer function needs to be updated during successive speech frames, with a frame size of 20 to 30 ms. However, for some segments of speech, especially for sustained vowels, the speech spectral envelope is actually a slow time-varying process, and spectra of adjacent frames are highly correlated. Figure 6 illustrates a typical evolution of log-magnitude speech spectral envelopes for successive frames, spaced 20 ms apart. This time dependence inherent in the LPC parameter sets suggests that any quantization method which exploits interframe redundancy will further reduce the coding bits for a given average distortion.

One way to exploit interframe correlation is to use vector linear prediction (VLP), introduced in Gersho and Cuperman (1985). Vector prediction of a frame's parameter set from previous frames is essentially equivalent to predicting the spectrum of the frame from the evolving pattern of spectra of the prior frames. By applying VLP, each spectral parameter in the

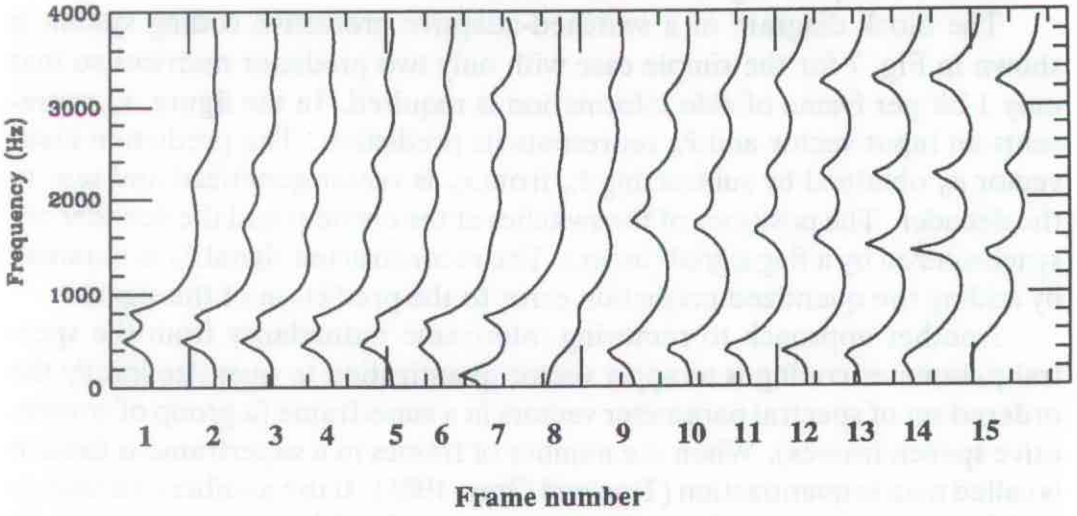

Figure 6 Log-spectral envelopes versus frames.

current frame is predicted not only from corresponding parameters of previous frames but also from the other spectral parameters of previous frames. In this way, correlation between different speech frames can be maximally exploited. Works which combine VLP with VQ for LPC parameter quantization (Davidson et al., 1987; Yong et al., 1988; Shoham, 1987) have shown that the coding performance is greatly enhanced by exploiting interframe correlation.

In Yong et al. (1988), a switched-adaptive interframe vector prediction (SIVP) scheme tracks statistical changes in the time-varying spectral parameters while maintaining reasonably low complexity. The line spectral frequencies (LSFs) were chosen as the spectral parameter set in this study, and it was found that very little gain is offered with higher than first-order prediction. Thus, a single predictor matrix is used to predict the LSF parameter vector $\mathbf{x}_n$ in the current frame from the corresponding vector $\mathbf{x}_{n-1}$ for the prior frame, according to $\hat{\mathbf{x}}_n = \mathbf{A}\mathbf{x}_{n-1}$. With switched adaptation, the predictor matrix $\mathbf{A}$ is updated vector by vector. For each input vector to be predicted, a predictor matrix is selected from a fixed set of such matrices via a statistical classification of the input vector. An index identifying the selected predictor matrix is transmitted to the receiver as side information. For a statistically specified source of vectors, the optimal predictor matrix is given by $\mathbf{A} = \mathbf{C}_{01}\mathbf{C}_{11}^{-1}$, where the covariance matrices are defined by $\mathbf{C}_{ij} = E\{\mathbf{x}_{n-i}\mathbf{x}_{n-j}^T\}$. To design the predictor matrices, a training sequence of LSF vectors is used to compute the covariance matrices for each class. Thus, if the current vector $\mathbf{x}_m$ is assigned to class i, the running estimate of C_{0j} is updated with the product $\mathbf{x}_m\mathbf{x}_{m-j}^T$. After passing through the training sequence in this way, the prediction matrix for each class is then computed from the corresponding covariance matrices.

The block diagram of a switched-adaptive predictive coding system is shown in Fig. 7 for the simple case with only two predictor matrices so that only 1 bit per frame of side information is required. In the figure, x_n represents an input vector and $\hat{x}_n$ represents its prediction. The prediction error vector e_n obtained by subtracting $\hat{x}_n$ from x_n is vector quantized and sent to the decoder. The positions of the switches at the encoder and the decoder are synchronized by a flag signal (index). The reconstructed signal $\tilde{x}_n$ is obtained by adding the quantized prediction error to the prediction of the signal.

Another approach to removing interframe redundancy from the spectral parameter coding is to apply vector quantization to quantize jointly the ordered set of spectral parameter vectors in a superframe (a group of consecutive speech frames). When the number of frames in a superframe is fixed, it is called matrix quantization (Tsao and Gray, 1985). If the number of frames is variable (according to a dynamic segmentation algorithm), the term segment quantization (Roucos et al., 1983; Wong et al., 1983) is used.

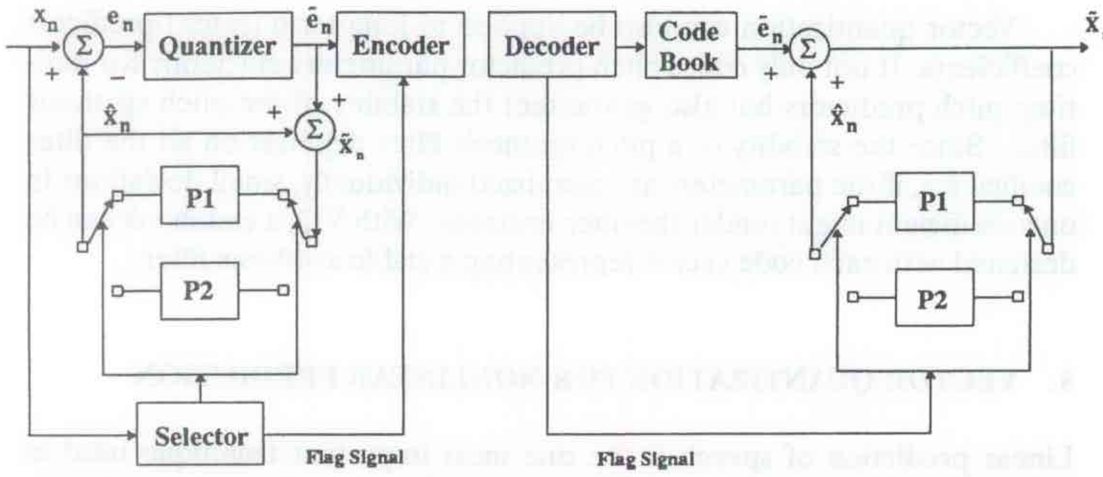

Figure 7 Switched-adaptive predictive coding system.

Motivated by rate-distortion theory, matrix quantization extends VQ techniques to matrices of n spectral parameter vectors. Matrix quantizers, designed with the GLA and a summed Itakura–Saito distortion measure, are locally optimal quantizers. For a given quality of reproduction, they generally require substantially lower coding rates than standard VQ. A matrix quantizer operating at 150 b/s with a 9-bit codebook and $N = 3$ is claimed to be comparable to a standard VQ operating at 350 b/s (Tsao and Gray, 1985).

In segment quantization, a segment which contains a number of LPC parameter sets corresponding to a spectral steady-state region in the input speech is quantized as a single entity to the nearest template in a codebook. Assuming the segmentation to the input is given, a segment $X = [x_1 x_2 \cdots x_l]$ is defined as a variable-length sequence of k-dimensional LPC vectors x_i. In order to simplify the codebook design and distortion measure in the codebook search, a time-warping transformation is performed on X prior to the quantization, which converts a variable-length segment to a fixed-length segment. The converted segment is denoted as $Y = [y_1 y_2 \cdots y_m]$ in the transformed space, where m is the segment length after the time-warping transformation. The segment quantizer uses a Euclidian distance to search for the nearest template in a codebook. The codebook given in Roucos et al. (1983) has 8192 segment templates where each template is a $k \times M$ matrix.

This type of quantization indeed removes most of the intra- and inter-frame redundancy from the LPC parameters but usually requires a large time delay and high complexity. Therefore, it is more attractive for very low bit-rate vocoding applications.

Vector quantization can also be applied to long-term (pitch) predictor coefficients. It not only codes pitch predictor parameters efficiently for multitap pitch predictors but also guarantees the stability of the pitch synthesis filter. Since the stability of a pitch synthesis filter depends on all the filter coefficients, if the parameters are quantized individually, small deviations in one coefficient might render the filter unstable. With VQ, a codebook can be designed with each code vector representing a stable synthesis filter.

8. VECTOR QUANTIZATION FOR NONLINEAR PREDICTION

Linear prediction of speech is the one most important technique used in speech coding. Yet speech is non-Gaussian, so linear prediction is not the optimal way to estimate a sample from prior values of the waveform. Nonlinear prediction of speech in principle offers a way to enhance the redundancy-removing feature of linear prediction and provide a superior model of the speech production mechanism. Recently, a method for performing nonlinear prediction was proposed based on the use of VQ (Wang et al., 1990). The method is based on a general theory of optimal nonlinear interpolation from vector-quantized observables, which includes nonlinear prediction as a special case (Gersho, 1990). The nonlinear prediction method, simply stated, is to vector quantize the set of past samples and then obtain the prediction of the new sample from a lookup table of prestored values. The table is addressed by the index of the quantized vector that approximates the past history.

In theory, the optimal predictor is the conditional expectation of the new value given the set of prior values. With this method, two approximations are made. One is to replace the past with a finite-resolution quantization of the past values using VQ. The accuracy of this approximation grows with the size of the VQ codebook. The second approximation is to compute the conditional expectation of the new value given the quantized past by using a training set of speech data rather than the "true" analytical model of the speech statistics.

The approach has been validated by testing it on a moving-average random process for which the optimal nonlinear predictor is known. When it is applied to open-loop prediction of speech, the results show that rather large codebook sizes are needed for representation of the past and very large training sets are necessary for designing the predictor table. Typically, the segmental prediction gain in this way exceeds that of linear prediction by only a very small margin. Nevertheless, the most striking indication of the effectiveness of nonlinear prediction is that the prediction residual is much less intelligible than the corresponding linear prediction residual. This result was observed with a fixed nonlinear predictor which was designed using speech training data different from the speech on which the effect was tested.

Figure 8 shows the spectrum of one frame of the residual for both linear and nonlinear prediction with a fourth-order predictor in each case. The

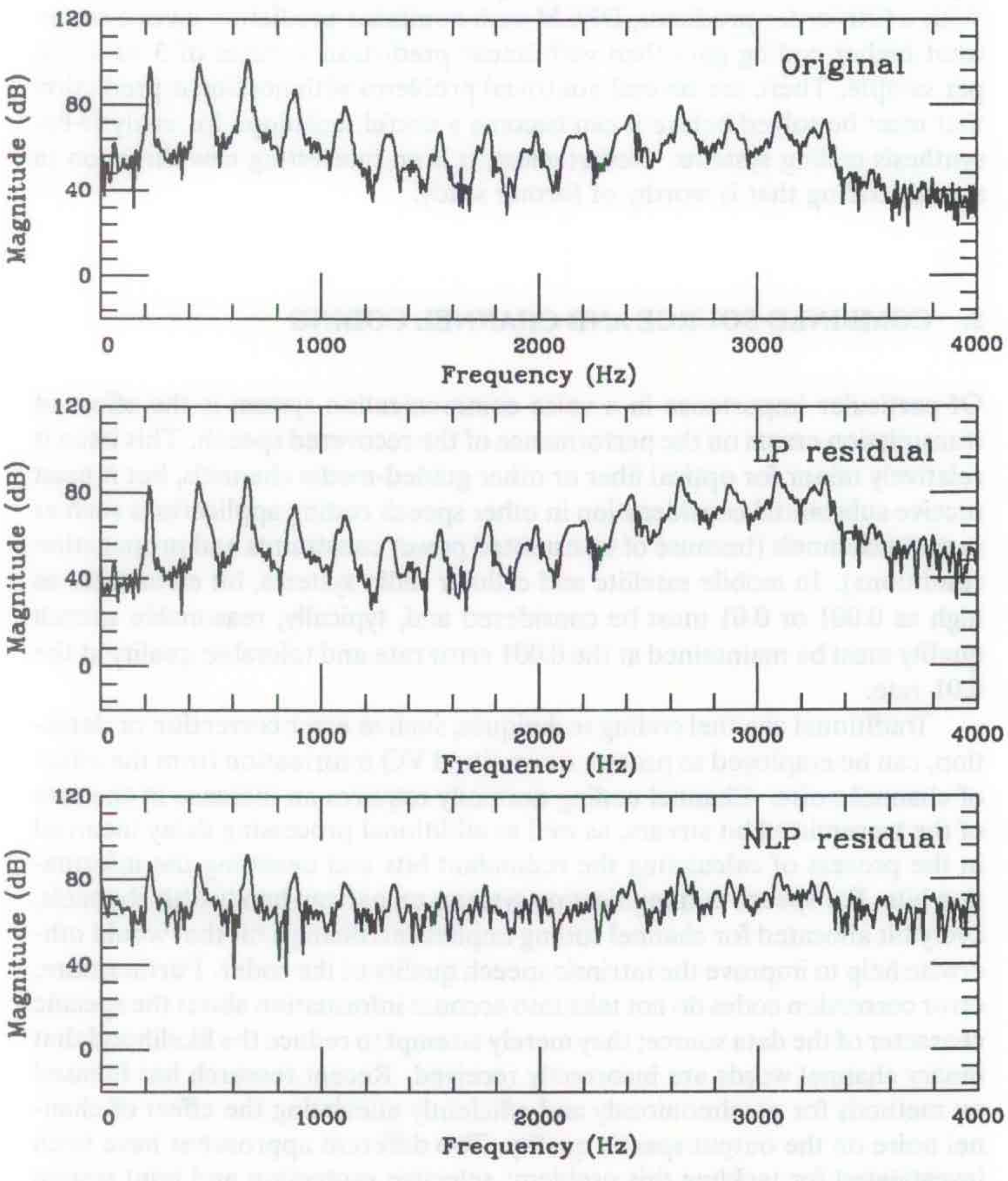

Figure 8 Spectra of original speech and linear and nonlinear prediction residuals.

spectrum of the original speech frame is also shown. It is remarkable that the spectral envelope has been notably flattened and the higher pitch harmonics have almost disappeared.

These results motivate further study of nonlinear prediction, but practical application to speech coding may be premature. The only speech coding algorithm tested so far with nonlinear prediction was (nonadaptive) DPCM. With a fifth-order predictor, DPCM with nonlinear prediction gives a somewhat higher coding gain than with linear prediction at rates of 3 or 4 bits per sample. There are several nontrivial problems with nonlinear prediction that must be solved before it can become a useful technique for analysis-by-synthesis coding systems. Nevertheless, it is an interesting new direction in speech coding that is worthy of further study.

9. COMBINED SOURCE AND CHANNEL CODING

Of particular importance in a voice communication system is the effect of transmission errors on the performance of the recovered speech. This issue is relatively minor for optical fiber or other guided-media channels, but it must receive substantial consideration in other speech coding applications such as in radio channels (because of transmitted power constraints and propagation conditions). In mobile satellite and cellular radio systems, bit error rates as high as 0.001 or 0.01 must be considered and, typically, reasonable speech quality must be maintained at the 0.001 error rate and tolerable quality at the 0.01 rate.

Traditional channel coding techniques, such as error correction or detection, can be employed to protect transmitted VQ information from the effect of channel noise. Channel coding normally requires an increase in the rate of the transmitted bit stream, as well as additional processing delay incurred in the process of calculating the redundant bits and decoding the information bits. For speech communication systems on narrow-bandwidth channels, every bit allocated for channel coding implies sacrificing a bit that would otherwise help to improve the intrinsic speech quality of the coder. Furthermore, error correction codes do not take into account information about the specific character of the data source; they merely attempt to reduce the likelihood that binary channel words are incorrectly received. Recent research has focused on methods for parsimoniously and efficiently alleviating the effect of channel noise on the output speech quality. Two different approaches have been investigated for tackling this problem: selective protection and joint source channel coding.

The first approach consists of determining which transmitted bits are the most important in terms of speech quality and selectively protecting them against channel noise using known channel coding methods. A typical data frame of encoded speech may contain several binary words representing VQ indexes for different vectors of parameters, and the more sensitive indexes can be selectively protected. By understanding the relative sensitivity of different speech coding parameters to bit errors, techniques for maintaining robustness to such errors can be developed. This form of selective channel coding can protect the transmitted speech information in a limited way without greatly increasing the overall bit rate (Cox et al., 1988).

The other approach is one that incorporates combined source and channel coding techniques with VQ. Combined source and channel coding can significantly improve the performance of the reconstructed speech signal on poor channels.

It is also possible to reduce the sensitivity of the bit stream to errors without adding redundant bits, simply by carefully allocating codewords to the signal parameter values. In particular, *redundancy-free* error protection codes for VQ have been introduced (Zeger and Gersho, 1991, 1987; Chen et al., 1987; De Marca and Jayant, 1987; Farvardin, 1988) and effective design algorithms for this purpose have been developed. An important feature of these techniques is that no added delay occurs in the coding system. The only cost is some fairly intensive off-line computation during the design process.

The basic idea of the technique, known as pseudo-Gray coding, is to determine carefully which channel words should be associated with which VQ code vector indexes, by way of an *index assignment function*. In the past, source code designers using VQ generally made the assignment arbitrary. Intuitively, vectors that are "close" to each other should be assigned indexes which differ in as few bit positions as possible. In this way, channel errors cause an index to be decoded as a vector which approximates the code vector that was supposed to be correctly decoded (i.e., without channel noise).

Another approach to joint source-channel coding with vector quantization is to design the quantizer's encoder and decoder taking into account the statistics of the channel (Kumazawa et al., 1984; Zeger and Gersho, 1988). For a given noisy channel, the problem is to find the best encoder for a fixed decoder and then find the best decoder for a fixed encoder. Equivalently, for a fixed partition of the input space, the goal is to find the best codebook, and vice versa. It has been shown that the optimal encoder and decoder of a quantizer of a noisy channel must satisfy two necessary conditions, known as the *weighted nearest neighbor* and *weighted centroid* rules, which generalize the nearest neighbor and centroid conditions that are necessary for quantizer optimally on a noiseless channel.

10. CONCLUDING REMARKS

The concept of vector quantization is clearly a general and powerful one with many specialized techniques that have emerged for combating the complexity obstacle. This chapter has given only a brief overview of the use of VQ in speech coding. There remain many coding applications where VQ techniques are still too costly in complexity or storage to be incorporated into hardware products. Research in VQ techniques is continuing with the aim of expanding its utility and applicability for the next generation of speech coding algorithms and implementations.

REFERENCES

Abut, H., Gray, R. M., and Rebolledo, G. (1982). Vector quantization of speech and speech-like waveforms, *IEEE Trans. Acoust., Speech, Signal Process. ASSP-30*: 423–435.

Buzo, A., Gray, A. H., Gray, R. M., and Markel, J. D. (1980). Speech coding based upon vector quantization. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-28*: 562–574.

Campbell, J., Tremain, T., and Welch, V. (1990). The DoD 4.8 kbps standard (proposed federal standard 1016). In *Advances in Speech Coding*, B. S. Atal, V. Cuperman, and A. Gersho (Eds.). Kluwer Academic Publishers, Norwell.

Cetin, A. E., and Weerackody, B. (1988). Design vector quantizers using simulated annealing, *IEEE Trans. Circuits Syst. CAS-35*:1550.

Chang, P. C., and Gray, R. M. (1986). Gradient algorithms for designing predictive vector quantizers. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-34*:679–690.

Chen, J. H. (1989). A robust low-delay CELP speech coder at 16 kb/s. *Proc. IEEE Global Communications Conf.*, November, Dallas.

Chen, J. H., and Gersho, A. (1986). Vector adaptive predictive coding of speech at 9.6 kb/s. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1693–1696, Tokyo.

Chen, J. H., and Gersho, A. (1987). Gain-adaptive vector quantization with application to speech coding. *IEEE Trans. Commun. COM-35*:918–930.

Chen, J. H., Davidson, G., Gersho, A., and Zeger, K. (1987). Speech coding for the mobile satellite experiment. *Proc. IEEE Int. Conf. on Communications*, June, Seattle.

Cheng, D. Y. and Gersho, A. (1986). A fast codebook search algorithm for nearest-neighbor pattern matching. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing* vol. 1, pp. 265–268, Tokyo.

Cheng, D. Y., Gersho, A., Ramamurthi, B., and Shoham, Y. (1984). Fast search algorithms for vector quantization and pattern matching. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 911.1–911.4, March, San Diego.

Chou, P. A., Lookabaugh, T., and Gray, R. M. (1989a). Entropy-constrained vector quantization. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-37*:31–42.

Chou, P. A., Lookabaugh, T., and Gray, R. M. (1989b). Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Inform. Theory, IT-35*:299–315.

Copperi, M., and Sereno, D. (1984). 9.6 Kb/s piecewise LPC residual excited coder using multiple-stage vector quantization. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 10.5.1–10.5.4, March, San Diego.

Cox, R. V., Hagenauer, J., Seshadri, N., and Sundberg, C.-E. (1988). A sub-band-coder designed for combined source and channel coding. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 235–238, New York.

Cuperman, V., Gersho, A., Pettigrew, R., Shynk, J. S., and Yao, J.-H. (1989). Backward adaptation techniques for low delay vector excitation coding of speech. *Conf. Record IEEE Global Communications Conference*, pp. 1242–1246, November, Dallas.

Davidson, G., and Gersho, A. (1988). Multiple-stage vector excitation coding of speech waveforms. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 163–166, April, New York.

Davidson, G., Yong, M., and Gersho, A. (1987). Real-time vector excitation coding of speech at 4800 bps. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 2189–2192, April, Dallas.

Dudley, H. (1958). Phonetic pattern recognition vocoder for narrow-band speech transmission. *J. Acoust. Soc. Am.* 30(8):733–739.

Farvardin, N. (1988). Optimal Binary Code Word Assignment for Vector Quantization over a Noisy Channel—An Application of Simulated Annealing. *IEEE Int. Symp. on Information Theory*. June, Kobe, Japan.

Flanagan, J., Morrell, D., Frost, R., Read, C., and Nelson, B. (1989). Vector quantization codebook generation using simulated annealing. *Proc. ICASSP'89*, P32.M7.8, May, Glasgow.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, *3*(3):209–226.

Fukunaga, K., and Narendra, P. M. (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Trans. Comput. C-24*:750–753.

Gersho, A. (1982). On the structure of vector quantizers. *IEEE Trans. Inform. Theory IT-28*:157–166.

Gersho, A. (1990). Optimal nonlinear interpolative vector quantization. *IEEE Trans. Commun. COM-38*:1285–1288.

Gersho, A., and Cuperman, V. (1982). Adaptive differential vector coding of speech. *Conf. Record GlobeCom 82*, December, pp. 1092–1096.

Gersho, A., and Cuperman, V. (1983). Vector quantization: A pattern matching technique for speech coding, *IEEE Commun. Mag.*, December, pp. 15–21.

Gersho, A., and Cuperman, V. (1985). Vector predictive coding of speech at 16 b/s. *IEEE Trans. Commun. COM-33*:685–696.

Gerson, I., and Jasiuk, M. (1990). Vector sum excited linear prediction (VSELP) speech coding at 8 kb/s. *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, pp. 461–464, April, Albuquerque.

Gersho, A., Ramstad, T., and Versvik, I. (1984). Fully vector-quantized subband coding with adaptive codebook allocation, *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, p. 10.7, March, San Diego.

Gray, A. H., Jr., and Markel, J. D. (1976). Distance measures for speech processing. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-24*:380–391.

Gray, R. M. (1984). Vector quantization. *IEEE ASSP Mag.* 1:4–29.

Gray, R. M., and Abut, H. (1982). Full search and tree searched vector quantization of speech waveforms. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 593–596, May, Paris.

Gray, F. M., Buzo, A., Gray, A. H., Jr., and Matsuyama, Y. (1980). Distortion measures for speech processing. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-28*:367–376.

Gray, R. M., Gray, A. H., Jr., Rebolledo, G., and Shore, J. E. (1981). Rate distortion speech coding with a minimum discrimination information distortion measure. *IEEE Trans. Inform. Theory IT-27*:708–721.

Iyengar, V., and Kabal, P. (1988). A low delay 16 kbits/sec speech coder. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 243–246, April, New York.

Jayant, N. S., and Noll, P. (1984). *Digital Coding of Waveforms*. Prentice-Hall, Englewood Cliffs, N. J.

Kang, G. S., and Coulter, D. C. (1976). 600 bps voice digitizer. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 91–94, April, Philadelphia.

Kang, G. S., and Fransen, L. J. (November 1984). Low-bit-rate speech encoders based on line-spectrum frequencies (LSFs). Naval Research Laboratory Report No. 8857.

Kumazawa, H., Kasahara, M., and Namekawa, T. (1984). A construction of vector quantizers for noisy channels. *Electron. Eng. Jpn.* 67-B(4):39–47.

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design, *IEEE Trans. Commun. COM-28*:84–95.

Makhoul, J., Roucos, S., and Gish, H. (1985). Vector quantization in speech coding. *Proc. IEEE 73*:1551–1588.

Marca, J. De, and Jayant, N. (1987). An algorithm for assigning binary indices to the codevectors of a multi-dimensional quantizer. *Proc. IEEE Int. Conf. on Communications*. June, Seattle.

Nasrabadi, N. M., and Feng, Y. (1988). Vector quantization of images based upon the Kohonen self-organization feature maps. *Proc. 2nd ICNN Conf.*, vol. I, pp. 101–105.

O'Shaughnessy, D. (1987). *Speech Communication*. Addison-Wesley, Reading, Mass.

Riskin, E. A., and Gray, R. M. (1991). A greedy tree growing algorithm for the design of variable rate vector quantizer. *IEEE Trans. Acoust., Speech, Signal Process.*, in press.

Roucos, S., Schwartz, R. M., and Makhoul, J. (1983). A segment vocoder at 150 b/s. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 61–64, April, Boston.

Sabin, M. J., and Gray, R. M. (1984). Product code vector quantizers for waveform and voice coding. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-32*:474–488.

Shoham, Y. (1987). Vector predictive quantization of the spectral parameters for low rate speech coding. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2181–2184, April, Dallas.

Shoham, Y. (1989). Cascaded likelihood vector coding of the LPC information. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 160–163, May, Glasgow.

Soleymani, M. R., and Morgerai, S. D. (1987). A high-speed search algorithm for vector quantization, *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 45.6.1–3, April, Dallas.

Tremain, T. E. (1982). The government standard linear predictive coding algorithm: LPC-10. *Speech Technol.* April, pp. 40–49.

Tsao, C., and Gray, R. M. (1985). Matrix quantizer design for LPC speech using the generalized Lloyd algorithm, *IEEE Trans. Acoust., Speech, Signal Process. ASSP-33*:537–545.

Vaisey, J., and Gersho, A. (1988). Simulated annealing and codebook design. *Proc. ICASSP'88*, pp. 1176–1179, April, New York.

Vidal, E. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time, *Pattern Recogn. Lett. 4*:145–157.

Wang, S., Paksoy, E., and Gersho, A. (1990). Performance of nonlinear prediction of speech. *Proc. Int. Conf. on Spoken Language Processing*, November, Kobe.

Watt, L., and Cuperman, V. (1988). A vector ADPCM analysis-by-synthesis configuration for 16 kbit/s speech coding, *Conf. Record IEEE Global Communications Conference*, November, pp. 275–259.

Wong, D. Y., and Gray, A. H. (1982). Distortion performance of vector quantization for LPC voice coding. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-30*:294–303.

Wong, D. Y., Juang, B. H., and Gray, A. H. (1982). An 800 b/s vector quantization LPC vocoder. *IEEE Trans. Acoust., Speech, Signal Process. ASSP-30*: no. 5, pp. 770–780.

Wong, D. Y., Juang, B. H., and Cheng, D. Y. (1983). Very low data rate speech compression with LPC vector and matrix quantization. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 65–68, April, Boston.

Yair, E., Zeger, K., and Gersho, A. (1992). Competitive learning and soft competition for vector quantizer design, *IEEE Trans. Signal Process.*, in press.

Yair, E., Zeger, K., and Gersho, A. (1990). Conjugate gradient methods for designing vector quantizers. *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, April, Albuquerque.

Yong, M., Davidson, G., and Gersho, A. (1988). Encoding of LPC spectral parameters using switched-adaptive interframe vector prediction. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 402–405, April, New York.

Zeger, K., and Gersho, A. (1987). Zero redundancy channel coding in vector quantisation. *Electron. Lett. 23*:654–656.

Zeger, K., and Gersho, A. (1988). Vector quantization design for memoryless noisy channels, *Proc. IEEE Int. Conf. on Communications*, pp. 1593–1597, June, Philadelphia.

Zeger, K., and Gersho, A. (1989). A stochastic relaxation algorithm for improved vector quantizer design, *Electron. Lett. 25*(14):896–898.

Zeger, K., and Gersho, A. (1990). Pseudo-Gray coding. *IEEE Trans. Commun., COM-38*:2147–2158.

Advances in Speech Signal Processing

edited by

Sadaoki Furui
NTT Human Interface Laboratories
Tokyo, Japan

M. Mohan Sondhi
AT&T Bell Laboratories
Murray Hill, New Jersey

Marcel Dekker, Inc. New York · Basel · Hong Kong