SPEECH CODING FOR THE MOBILE SATELLITE EXPERIMENT

Juin-Hwey Chen, Grant Davidson, Allen Gersho, and Kenneth Zeger

Communications Research Laboratory
Department of Electrical & Computer Engineering
University of California, Santa Barbara, CA 93106

ABSTRACT

This paper reviews the progress of the 4.8 kb/s speech coder project at UCSB for NASA's Mobile Satellite Experiment (MSAT-X) under the administration and technical management of the Jet Propulsion Laboratory. The purpose of MSAT-X is to develop the technologies needed to make feasible a nationwide mobile communications system using a satellite as the relay station. The UCSB project has led to two alternative coding algorithms, both of which use vector quantization (VQ) and offer a reasonably high quality of 'natural' speech reproduction at 4.8 kb/s with a moderate coding delay and a fairly high tolerance of transmission errors. A zero redundancy error control technique for VQ is described.

1. Introduction

In 1984, a three year research project was initiated at UC Santa Barbara (UCSB) to explore the feasibility of speech coding at the rate of 4.8 kb/s while retaining "nearly toll quality" and to implement a breadboard prototype that can operate in real-time at this bit rate while handling the rather severe conditions of a 5 kHz radio channel between a land mobile vehicle and a satellite in geosynchronous orbit. This project was sponsored by NASA and administered by the Jet Propulsion Laboratory (JPL) as part of NASA's Mobile Satellite Experiment (MSAT-X).

This was indeed a challenging project requiring a substantial advance beyond the 1984 state-of-the-art in speech coding. The prospect for meeting all of the demanding objectives for the coder seemed rather doubtful from the outset. Nevertheless, while cognizant of the high risk nature of the project, JPL chose to go ahead and awarded two separate three-year contracts for the speech coder work, one to UCSB and one to Georgia Tech. This paper reviews the progress and current status of the UCSB project. At this stage, the prospects for successfully meeting the key objectives are much higher than originally anticipated. Completion of the project is expected at the end of 1987 when a hardware prototype will be completed that can implement both of the coding algorithms developed under the contract.

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration.

1.1. Objectives

The speech coder must operate at a bit rate of 4.8 kb/s, have a total coding delay of at most 50 ms, have negligible degradation due to random bit errors with a somewhat bursty character and average bit error probability of 10^{-3}, have graceful degradation under short fades and rapid recovery from long fades, achieve good speech performance in the presence of typical vehicle acoustic background noise, and ultimately be implementable in a small low-power package.

Our first and major concern was the voice quality objective. Although "near toll quality" was not quantitatively defined, the intent was to achieve high intelligibility, the absence of any obvious degradations that could be annoying or uncomfortable to the listener, and an overall quality approaching that of a good long-distance telephone call. An important reference point is the widely used Dept. of Defense LPC-10 coder at 2.4 kb/s which suffers from a lack of naturalness having a distinct 'buzzy' quality, some loss of speaker recognizability, and very high sensitivity to environmental noise. Thus, we had two somewhat extreme reference points, toll telephone quality on the one hand and LPC-10 on the other hand. Perhaps the best coding algorithm available in 1984 that approached the quality objectives was adaptive transform coding at 16 kb/s and due to its complexity the only real-time implementation used a large array processor with a mainframe host computer. On the other hand, increasing the bit-rate of LPC to 4.8 kb/s was known to offer very little improvement in quality over LPC-10.

The first indication that the objectives might be at all feasible came in June 1984 at ICC in Amsterdam, when a paper by Schroeder and Atal [1] was presented that introduced a technique called stochastic coding (later called code excited linear prediction (CELP) which demonstrated a remarkably high quality that might be achievable at a projected rate of 4.8 kb/s. The catch was that the computer simulation of this coder required the use of a Cray I and operated in 125 times real-time. Furthermore, the bit-rate was still a 'projected' rate since the key filter parameters were left unquantized and it was not known how to do this quantization with the available bit allocation without severely degrading the voice quality.

Another milestone was the emergence of effective Multi-Pulse LPC coding algorithms which were beginning to

0756 CH2424-0/87/0000-0756 $1.00 © 1987 IEEE

show promise of achieving good quality at rates between 9.6 and 16 kb/s, albeit with a high complexity.

The focus of our prior work in speech coding at UCSB was in the application of Vector Quantization techniques to waveform coding. This work led to the development of Adaptive Vector Predictive Coding, a VQ based generalization of ADPCM. With this background we began our work on the MSAT-X project in the fall of 1984.

2. Project History

Our efforts in the first year of the project centered on parallel studies of several approaches to speech coding using VQ techniques. These included pitch synchronous transform coding using hierarchical vector quantization [2], and vector quantized Multi-Pulse LPC [3]. We quickly narrowed down the candidates to two new coding approaches, Vector Adaptive Predictive Coding (VAPC) [4,5] and Vector Excitation Coding (VXC) [6,7]. Both of these techniques continued to remain promising as the algorithm and simulation studies progressed so that we have chosen to continue with both in parallel and aim for a hardware prototype that can implement either technique with suitable firmware. In this paper we concentrate on presenting an overview of the two algorithms, the use of a pseudo-Gray error control technique for VQ, and briefly discuss the hardware implementation issues.

3. Vector Adaptive Predictive Coding (VAPC)

VAPC [4,5] is a new speech coding algorithm which greatly improves the performance of conventional Adaptive Predictive Coding (APC) [8] by using VQ techniques. Due to the power of VQ, VAPC has been able to reduce the bit-rate of APC from about 16 kb/s to 9.6 kb/s or 4.8 kb/s while maintaining roughly the same speech quality as in the older APC coders. In fact, VAPC evolved from Vector Predictive Coding, or Vector DPCM [9], although it is now almost completely different from the original Vector DPCM scheme.

3.1. Coding Algorithm

The structure of our first version of VAPC is similar to the APC coder in [8] except that the scalar quantizer and the LPC predictor in APC are replaced by a vector quantizer and a vector predictor, respectively [4]. The transmitter first uses a long-term predictor, or "pitch predictor", to remove the redundancy in speech waveforms due to pitch quasi-periodicity. A short-term vector predictor is then used to remove the short-term redundancy remaining in the pitch-prediction residual, and the final residual is quantized by a gain-adaptive vector quantizer [10]. In the receiver, the speech waveform is reconstructed by adding the short-term and long-term prediction back to the quantized residual.

In order to exploit the noise masking effect using noise spectral shaping, we replaced the short-term vector predictor by the usual LPC predictor in a more recent version of VAPC. To accommodate such a change, it was necessary to modify the operation of the gain-adaptive vector quantizer such that the codebook search is performed in a trial-and-error fashion. The resulting computational complexity, however, is an order of magnitude higher than that of our earlier VAPC structure if the trial-and-error approach is implemented directly without any complexity reducing technique. In [4], we showed the step-by-step evolution from a direct VAPC structure to an equivalent but computationally more efficient one. This efficient structure, shown in Fig. 1, requires only slightly more computation than our earlier VAPC structure which used a vector predictor.

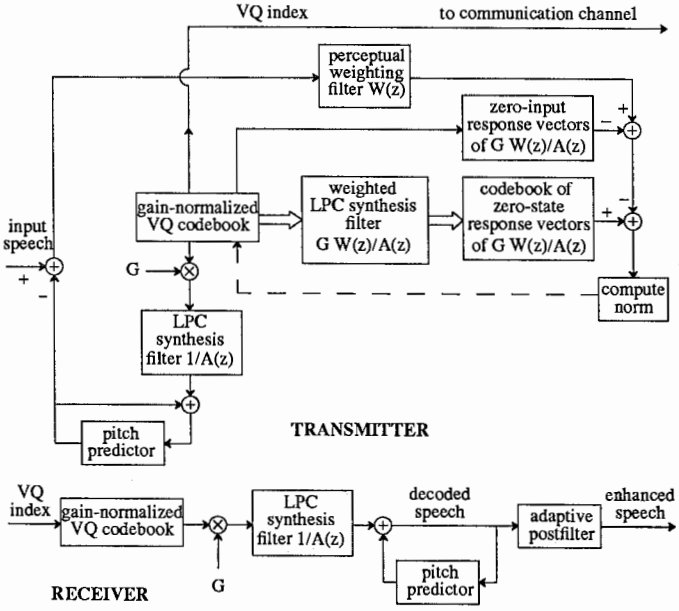

Figure 1. An efficient coder structure of VAPC

The speech analysis (not shown in Fig. 1) of 4.8 kb/s VAPC is performed in the following way. For each frame of 20 ms, or 160 samples (8 kHz sampling rate), the pitch period is first determined using a simple correlation-type pitch detector. Then, without computing the unquantized optimal three-tap pitch predictor, we directly search through the pitch-predictor VQ codebook and pick the candidate pitch predictor that minimizes the energy of the open-loop pitch-prediction residual. With direct inverse filtering, such an exhaustive search requires 2 million multiply-adds/second of computation for a codebook size of 64. We have developed a fast search scheme that also finds the same optimal pitch predictor but is at least 20 times faster. Once the pitch period and the quantized pitch predictor are determined, we compute the open-loop pitch-prediction residual and perform tenth-order LPC analysis on this residual. The LPC parameters are quantized by a switched-adaptive interframe vector predictive coding scheme [7]. Then, we use the quantized LPC predictor to inverse filter the open-loop pitch prediction residual and obtain the open-loop LPC prediction residual. The rms value of this final residual is quantized to 5 bits and the quantized value is used as the "gain" of gain-adaptive VQ in encoding the entire frame. The pitch period, the pitch predictor, the LPC predictor, and the gain constitute the side information of VAPC.

An important feature of the efficient structure in Fig. 1 is the *zero-state-response codebook*, or *ZSR codebook*, which is the key to complexity reduction. At 4.8 kb/s, we use a vector dimension of 20 and a codebook size of 128 for pred-

iction residual VQ. In a less efficient implementation of VAPC, the coder has to filter each of the 128 gain-normalized VQ codevectors for each input vector before a nearest-neighbor search can begin. The filter used in such filtering is a "weighted" LPC synthesis filter $GW(z)/A(z)$, where G is the quantized residual gain, $W(z)$ is a perceptual weighting filter, and $1/A(z)$ is the LPC synthesis filter using quantized LPC parameters. The repeated filtering of the 128 codevectors for each input vector is required because the memory of the filter $GW(z)/A(z)$ is different for each input vector. By separating the effect of the filter memory, we can perform the filtering of codevectors only once at the beginning of each frame and use the filtered vectors throughout the entire frame (see [4] for details).

The ZSR codebook is obtained in the following way. We pick one codevector at a time from the gain-normalized codebook of gain-adaptive VQ, set the memory of the filter $GW(z)/A(z)$ to zero, filter the codevector, and compute the first 20 samples of the filter output. The resulting vector is a codevector in the ZSR codebook, and this procedure is repeated until all the 128 codevectors are exhausted.

After the speech analysis and the ZSR codebook computation are finished, the transmitter in Fig. 1 operates as follows. For each input vector, the prediction generated by the pitch predictor is first subtracted from the original speech vector. The pitch-prediction residual vector is then filtered by the weighting filter $W(z)$. Then, the zero-input-response vector of the filter $GW(z)/A(z)$, which is the ringing of the filter from previously selected gain-normalized codevectors, is computed and is subtracted from the weighted pitch-prediction residual vector. The resulting vector is compared with the 128 codevectors in the ZSR codebook. The index of the nearest neighbor is sent to the receiver and is also used to extract the corresponding codevector from the gain-normalized codebook. The extracted codevector is used to reset the memory of the filter $GW(z)/A(z)$ (for computing the next zero-input-response vector), and a scaled version of it is also used to excite the LPC synthesis filter, whose output is then used in pitch prediction of the following speech vectors. The process continues until all eight 20-dimensional vectors in a frame are encoded.

At the receiver, the side information and the VQ index are first decoded. The decoded codevector is scaled up by the gain G, and the resulting vector is used to excite the LPC synthesis filter and the pitch synthesis filter to obtain the reconstructed speech. An adaptive postfilter is then used to enhance the quality of the reconstructed speech [5].

The adaptive postfilter has the following transfer function

$$H(z) = \frac{1 - \hat{P}(z/\delta)}{1 - \hat{P}(z/\gamma)} (1 - \mu z^{-1}),\ 0 < \delta < \gamma < 1,\ 0 < \mu < 1 \quad (1)$$

where $\hat{P}(z)$ is the LPC predictor obtained by performing LPC analysis on the decoded speech, and the first-order filter section $(1 - \mu z^{-1})$ is used to compensate for the muffling effect of the postfilter. Typical values of γ and δ are 0.8 and 0.5, respectively. The postfilter greatly improves the perceptual quality of VAPC-coded speech without introducing significant distortion. For clean speech, the postfilter has reduced the coding noise to an essentially inaudible level. With postfiltering, VAPC at 4.8 kb/s is capable of producing very good communication quality speech which is natural-sounding, clean, intelligible, and quite close to the original.

3.2. Effects of Noisy Speech and Channel Errors

The 4.8 kb/s VAPC coder has also been simulated using noisy speech as the input. Unlike vocoders, VAPC does not break down even at a high input noise level. For speech corrupted by truck noise at an SNR as low as 6 dB, the coded speech still maintains reasonably good intelligibility and quality. This is not surprising since VAPC is basically a waveform coder, which is inherently robust to background noise.

The effect of channel errors on speech quality has also been simulated. The transmitter output bit stream is corrupted by simulated channel errors that are generated by JPL's mobile satellite channel simulator. The average bit-error rate is 10^{-3}, but the errors are quite bursty. During 30 seconds of speech utterance, there are about 12 to 15 error bursts on average, but typically only 2 or 3 of them have audible effect in the decoded speech even without error protection. Thus, VAPC appears to be reasonably robust to channel errors. The VQ codebook addresses can be partially protected from errors through the use of pseudo-Gray coding, discussed in a later section. To further improve the robustness of VAPC, we are currently investigating error protection and interframe parameter smoothing schemes.

3.3. Real-Time Implementation

The VAPC and postfiltering algorithms have been implemented in real-time hardware using the AT&T DSP32 floating-point DSP chip. The full-complexity VAPC algorithm requires about 3.5 million multiply-adds/second of computation to implement. With system overhead, the overall complexity exceeds the capability of a 4-MIPS DSP32 chip. As a first step, we used a single DSP32 chip (4 MIPS capability) to implement a simplified version of VAPC which has a complexity of 3 million multiply-adds/second. The speech coded by this real-time VAPC coder has a somewhat higher level of coding noise than that obtained by computer simulation of the full-complexity VAPC; however, the speech still sounds very natural and is quite intelligible. We are currently implementing the full-complexity VAPC algorithm using a faster DSP32 chip which has an instruction cycle of 160 ns rather than 250 ns; we expect the quality of coded speech will be greatly enhanced after this is completed.

4. Vector Excitation Coding

Vector Excitation Coding (VXC) is a powerful new technique for digital encoding of analog speech signals for transmission or storage at medium and low bit rates. In a generic VXC coder, a vocal-tract model is used in conjunction with a set of excitation vectors (codevectors) and a perceptually-based error criterion to synthesize natural-sounding speech. The original embodiment of this concept was Code Excited Linear Prediction (CELP) [11, 12] in which

Gaussian random variables are used for the codevector components. CELP achieves very high reconstructed speech quality at the cost of high computational complexity (around 40 million multiply/add operations per second for versions utilizing efficient excitation codebook search procedures). The specific coder described here, Pulse Vector Excitation Coding (PVXC), achieves the same high reconstructed speech quality as comparable schemes, but requires significantly less computation. A variety of novel complexity reduction methods have been developed and combined, reducing optimal codevector selection computation to only 0.55 million multiply/adds per second. This feature has facilitated the implementation of a real-time 4.8 kb/s VXC coder using only one programmable digital signal processor chip [7].

Although PVXC employs some features of Multi-Pulse LPC (where excitation pulse amplitudes and locations are determined from the input speech) and CELP (where excitation vectors are selected from a fixed codebook), there are several important differences between them. PVXC is distinguished from other excitation coders by the use of a precomputed and stored set of pulse-like (sparse) codevectors. This form of vocal-tract model excitation is used together with an efficient error minimization scheme in the codebook search procedure. Finally, PVXC incorporates an excitation codebook which has been optimized using an iterative training procedure to minimize the perceptually-weighted error between original and reconstructed speech waveforms [7].

The coder is well-suited for operation in a mobile environment. Simulation studies have shown that coder performance is very robust to both white and vehicle noise-corrupted input speech. Furthermore, simulations have shown that the coder is inherently robust to the effects of channel errors.

4.1. PVXC Coding Algorithm

Fig. 2 presents the basic structure of a PVXC encoder and decoder. The encoder shown is a *conceptual* description only - details concerning the actual computation performed are described later.

The original speech input $\mathbf{s}_n$ is a vector with a dimension of $k = 40$ samples. The input vector $\mathbf{s}_n$ is preprocessed to generate $\mathbf{z}_n$, which is subtracted from each member of a ensemble of N weighted synthetic speech vectors $\{\hat{\mathbf{z}}_j\}$, $j \in \{1,...,N\}$. The ensemble $\{\hat{\mathbf{z}}_j\}$ is generated by filtering pulse-like codevectors $\mathbf{c}_j$ with two time-varying, cascaded LPC synthesis filters $H_l(z)$ and $\tilde{H}_s(z)$. $H_l(z)$ is a conventional three-tap all-pole filter which generates the spectral fine structure (due to pitch), and $\tilde{H}_s(z)$ consists of an all-pole LPC synthesis filter in cascade with a pole/zero perceptual weighting filter. The purpose of the weighting provided by $W(z)$ and incorporated into $\tilde{H}_s(z) = H_s(z)W(z)$ is to shape the spectrum of the error signal so that it is similar to the spectrum of $\mathbf{s}_n$, thereby masking distortion which would otherwise be perceived by the human ear.

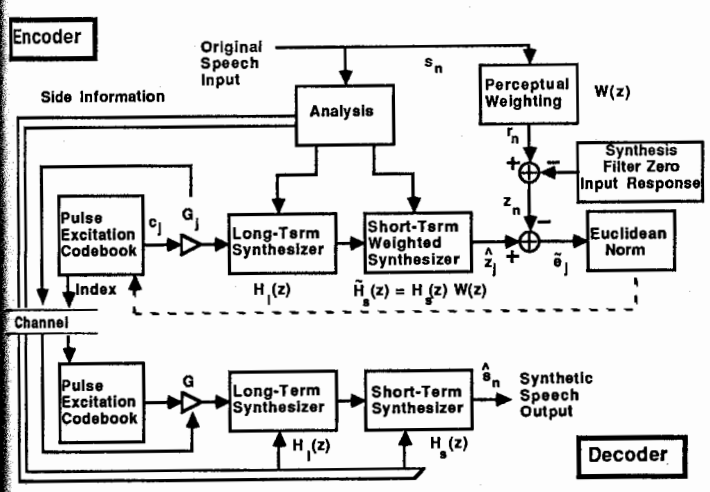

Figure 2. Block diagram of the PVXC speech coder

The input vector $\mathbf{s}_n$ is preprocessed to generate $\mathbf{z}_n$, which is subtracted from each member of a ensemble of N weighted synthetic speech vectors $\{\hat{\mathbf{z}}_j\}$, $j \in \{1,...,N\}$. The ensemble $\{\hat{\mathbf{z}}_j\}$ is generated by filtering pulse-like codevectors $\mathbf{c}_j$ with two time-varying, cascaded LPC synthesis filters $H_l(z)$ and $\tilde{H}_s(z)$. $H_l(z)$ is a conventional three-tap all-pole filter which generates the spectral fine structure (due to pitch), and $\tilde{H}_s(z)$ consists of an all-pole LPC synthesis filter in cascade with a pole/zero perceptual weighting filter. The purpose of the weighting provided by $W(z)$ and incorporated into $\tilde{H}_s(z) = H_s(z)W(z)$ is to shape the spectrum of the error signal so that it is similar to the spectrum of $\mathbf{s}_n$, thereby masking distortion which would otherwise be perceived by the human ear.

In synthesizing $\{\hat{\mathbf{z}}_j\}$, each pulse codevector is scaled by a variable gain G_j which is determined by minimizing the mean-squared error between the weighted synthetic speech signal $\hat{\mathbf{z}}_j$ and the weighted input speech vector $\mathbf{z}_n$. The gain is optimized for each codevector according to:

$$G_j = \frac{< \hat{\mathbf{z}}_j, \mathbf{z}_n >}{< \hat{\mathbf{z}}_j, \hat{\mathbf{z}}_j >}, \qquad (2)$$

where $< \cdot , \cdot >$ denotes a Euclidean inner product.

An exhaustive search is performed for every $\mathbf{s}_n$ to determine the excitation vector $\mathbf{c}_j$ which minimizes the squared weighted Euclidean distortion $|| \tilde{\mathbf{e}}_j ||^2$ between $\mathbf{z}_n$ and $\hat{\mathbf{z}}_j$. Once the optimal $\mathbf{c}_j$ is selected, a codebook index which identifies it is transmitted to the decoder together with the associated gain. The parameters of $H_l(z)$ and $H_s(z)$ are transmitted as side information once per input speech frame (every 20 ms).

The complexity of a standard VXC encoder can be reduced by removing the effect of synthesis filter memory on the choice of an optimal codevector. This is performed (in the encoder only) by finding the zero input response of the cascaded synthesis filters $H_l(z)$ and $\tilde{H}_s(z)$ and then subtracting this response from the weighted speech signal $\mathbf{r}_n$ to produce the input vector to the codebook search $\mathbf{z}_n$. Once this deterministic component has been removed, the initial memory values in $\tilde{H}_s(z)$ and $H_l(z)$ can be set to zero, reducing the arithmetic operations required to synthesize $\{\hat{\mathbf{z}}_j\}$. After the optimal codevector is selected, the filter memories are updated for encoding the subsequent vector.

A more significant complexity reduction technique, Sparse Vector Fast Search (SVFS) [6], exploits the fact that when the initial memory in the long and short-term IIR synthesis filters is set to zero, the output can be expressed as a matrix-vector product:

$$\hat{\mathbf{z}}_j = G_j \, \mathbf{H}\mathbf{c}_j, \qquad (3)$$

where $\hat{\mathbf{z}}_j$ is a k-dimensional weighted synthetic speech vector, and $\mathbf{H}$ is a $k \times k$ lower triangular matrix whose elements are from the impulse response $h(m)$ of the combined long-term, short-term, and weighting filters. Eq. (3) describes the

finite-length output of three cascaded IIR filters as a convolution of two sequences of length k, scaled by a gain. If we suitably choose each $\mathbf{c}_j$ to have only N_p pulses per vector (the other components are zero), then Eq. (3) can be computed very efficiently. Typically, N_p/k is 0.1. Like Multi-Pulse LPC, SVFS exploits the fact that only about four variable-height excitation pulses are required per pitch period to synthesize natural-sounding speech.

An enhanced version of this technique [7] combines the synthesis filter matrix formulation given above and pulse excitation model with the autocorrelation fast-search technique [13] to achieve substantially less computation per codebook search. When used with a designed pulse excitation codebook of size 256 and dimension $k = 40$ (instead of the more conventional Gaussian codebook of size 1024), the enhanced method requires only 0.55 million multiply/adds per second. This computational level, 800 times lower than that required for a standard VXC coder, is achieved while incurring only a slight drop in reconstructed speech quality.

4.2. Real-Time PVXC Implementation

In this section we briefly describe the features of a preliminary version real-time 4.8 kb/s PVXC coder which we have implemented using a single DSP32 floating-point digital signal processor. The real-time coder hardware currently consists of one general-purpose VMEbus/DSP32 development board with multiprocessing capability [14]. The final prototype speech coder will be based on our own architecture consisting primarily of three DSP32 chips and a microcontroller (see Section 6).

The short-term prediction coefficients are differentially encoded using a 27-bit switched-adaptive vector predictive coding scheme [7]. The three long-term prediction coefficients are vector quantized using a 6-bit codebook. Seven bits are used for the "pitch" term in the long-term filter, and eight bits are used to identify each excitation vector in the frame. A Lloyd-Max scalar quantizer is used to encode the four gain terms. Four bits are reserved for frame synchronization and bit error detection, for a total of 96 bits per frame. The decoder contains an adaptive postfilter for enhancing the decoded speech. Total memory usage is 1.5K words for instructions and 5K words for data, with 2.5K remaining for additional enhancements.

4.3. Performance in a Noisy Environment

A speech coder which operates in a mobile system must exhibit robust performance when the input speech is corrupted by vehicle (e.g. truck or helicopter) noise. It is well known that in these adverse conditions the performance of conventional 2400 bps LPC-10 is significantly degraded. Since PVXC also utilizes LPC techniques, its noisy speech performance is an important concern for MSAT-X.

Intuitively, one might expect hybrid speech coders such as VXC (which combine features of waveform and conventional LPC coders) to be robust to noisy speech inputs since the decoder creates a synthetic waveform with similar time-domain characteristics as the input signal. This hypothesis has been verified by PVXC computer simulations. Two classes of additive input noise were considered: vehicle (truck) noise and white noise. The signal-to-noise ratio at the coder input was 6 dB and 12 dB, respectively, for the two cases. These SNR figures are typical of those expected in the MSAT-X mobile environment.

Fig. 3 presents original and reconstructed speech signals of 150 ms duration for the case of truck noise. The original and decoded residual signals are also shown. The first 60 ms of the plots show signals corresponding to truck noise only (speaker is silent) followed by voicing onset. Note that both the speech and the truck noise are faithfully reproduced by the decoder. Informal listening tests show that very negligible degradation in speech quality is incurred due to the vehicle noise.

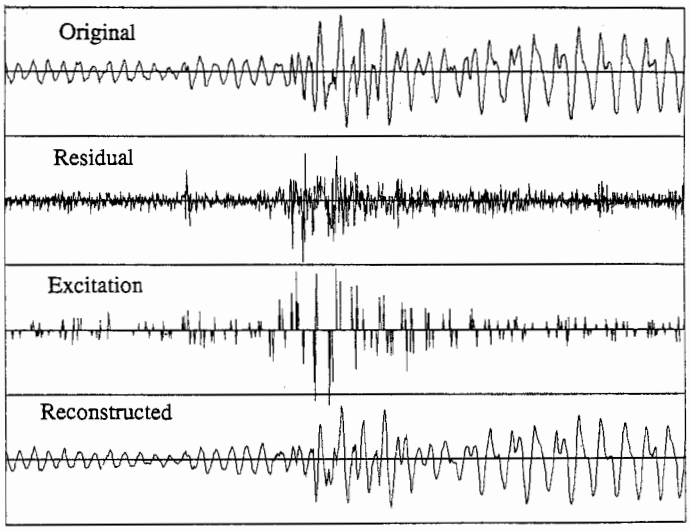

Figure 3. Vehicle noise corrupted original and reconstructed waveforms in the PVXC coder. SNR of input signal = 6 dB. Residual and excitation are amplified 15 dB relative to input signal.

Fig. 4 presents similar waveforms for the case of additive white noise at the input. In this case, the coder exhibits a "noise-suppression" characteristic - significant attenuation (6 dB) of the high-frequency white noise components can be observed in the figure. This is due in part to the fact that the pulse excitation models voiced (periodic) signals better than the high-frequency noise components. Although the decoded speech is slightly muffled compared to the original (noisy) speech, overall quality and intelligibility are actually *improved* due to the noise-suppression effect.

Another desirable characteristic for speech coders operating over a mobile satellite link is robustness to channel errors. Computer simulations using an MSAT-X channel model (described earlier) indicate that PVXC is reasonably robust to errors even in the absence of bit error detection. The robustness of the coder to burst channel errors can be improved by incorporating a frame-repeat algorithm in the decoder. When an error burst is detected in one frame, the decoder simply uses parameters from the last valid frame to synthesize the current one. Error protection can be applied to the most critical parameters - the short-term prediction coefficients and the pitch. Bit error sensitivities of the remaining three parameters are relatively small. The excita-

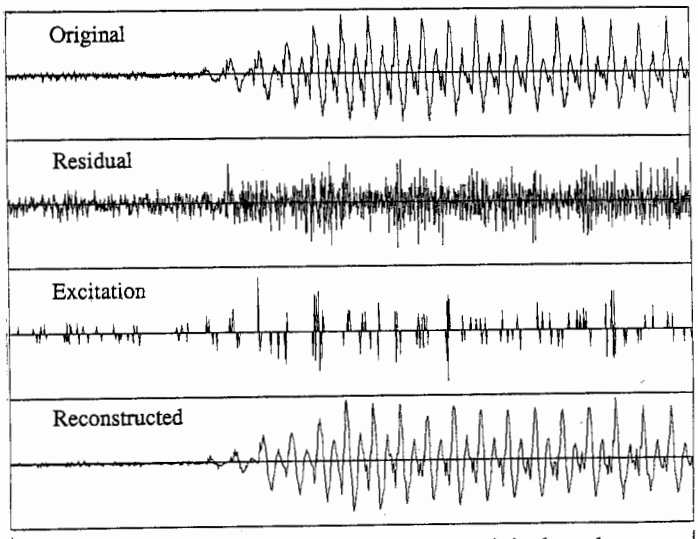

Figure 4. White noise corrupted original and reconstructed waveforms in the PVXC coder. SNR of input signal = 12 dB. Residual and excitation are amplified 10 dB relative to input signal.

tion codevector addresses can be partially protected without redundancy bits through the use of pseudo-Gray coding, discussed below.

5. Pseudo-Gray Coding

In both the VAPC and PVXC algorithms, a large fraction of the 4.8 kb/s rate is devoted to the transmission of codebook addresses to identify specific codevectors (i.e., for regenerating the residual in VAPC or the excitation in PVXC and for recovering parameter vectors). The distortion caused by an isolated channel error occasionally causes a very audible glitch and if this recurs frequently it can have a significant impact on the perceived speech quality. The low bit rate (4.8 kb/s) constraint gives little opportunity to allocate redundancy bits for error protection. In this section, we present a technique for reducing the degradation in speech quality when channel errors cause erroneous codevector addresses, or indices, to be received. The technique, called pseudo-Gray coding, does not require the addition of redundancy bits and thus provides an effective partial solution to the problem of channel errors.

The transmitted bit stream carries the binary indices of optimally chosen codevectors. The effect of channel errors is to cause the incorrect decoding of received codevector indices. Thus, a distortion is introduced due to the effect of errors in transmitted binary indices. The assignment of indices to vectors in a VQ codebook (i.e. the ordering of the codebook) plays a vital role in determining how significant channel errors will be in increasing overall system distortion. By assigning indices to codevectors in such a way that close indices (in the Hamming distance sense) correspond to close vectors (in terms of a distance function between vectors), the average distortion caused by channel error effects can be reduced.

The technique of pseudo-Gray coding was developed for a fixed metric type of distortion measure, such as MSE, the squared Euclidean distance between vectors. In some applications, however, the actual encoding is based on a time varying perceptually weighted distortion measure. Nevertheless, the coding method still provides effective protection in the event of a channel error by assuring that the incorrectly received index is likely to yield a decoded vector that has a relatively small distance from the optimal codevector selected at the transmitter. An important observation to make is that there always exists some reordering (possibly the original ordering) of a given codebook which provides a nonnegative decrease in overall average distortion in the VQ system. Hence it can only be an improvement in *any* VQ system to perform optimal pseudo-Gray coding.

5.1. Problem Formulation

Suppose an analog input vector in k dimensions is coded using a given codebook

$$CB = \{\mathbf{w}_0, \mathbf{w}_1, \cdots \mathbf{w}_{N-1}\}.$$

whose subscripts (indices) correspond to channel codewords, assuming a b-bit binary representation of the indices where the codebook size is $N = 2^b$. Hence, b represents the number of bits in each binary index i that corresponds to codevector w_i.

Let $d(\mathbf{x}, \mathbf{y})$ be a real-valued *distance function* (metric) between any two vectors $\mathbf{x}$ and $\mathbf{y}$ of $\mathbf{R}^k$. Let $p(\mathbf{w})$ be the probability that a particular codevector is selected by the encoder to represent the input vector.

5.2. Channel Errors

We assume a *memoryless binary symmetric channel* with *crossover probability* ε. An error in one or more of the bits of a transmitted index will result in an incorrectly received index and thus an incorrectly decoded codevector. The MSAT-X channel is in fact bursty, but interleaving is used to reduce the bursty character. Pseudo-Gray coding may not offer any advantage during a dense burst of errors but will be helpful during periods when errors are relatively sparse. In any case, there is no price to pay for the use of this technique.

5.3. Distortion Minimization

We seek a codebook with maximum protection against channel errors. The optimal arrangement of a codebook will in general depend upon the statistics of the input vectors. We instead introduce a minimization of an upper bound on the average distortion that is independent of the input statistics. We will attempt to minimize a quantity called the bit error distortion which is independent of the actual vectors contained in the codebook CB, and is affected only by the assignment of binary indices to those codevectors. That is, the order in which the vectors are placed in the codebook determines the resulting bit error distortion, whereas the actual choice of which vectors belong in a codebook will be completed during the codebook design process. There are at most a finite number of such index assignments. Thus, there is at least one "best" codebook arrangement, in the sense of yielding the minimal average distortion caused by bit errors inflicted upon indices due to channel noise. Hence it would

be advantageous to determine an algorithm that takes as input a codebook and yields a rearrangement of the codevectors that minimizes the expected time average bit-error distortion. This is the main idea behind what we call *Pseudo-Gray Coding* (analogous to scalar Gray Coding) which is discussed below.

Let the set of $N = 2^b$ vector indices be denoted by

$$I = \{0\ ,\ 1\ ,\ \cdots\ ,\ 2^b-1\}.$$

using decimal integers for notational brevity, but understanding that each integer represents a b-bit binary word. For any two binary words $i\ ,\ j \in I$, let $H(i\ ,\ j)$ denote the *Hamming distance* between i and j, i.e., the number of bit positions in which i and j differ. For each binary index $i \in I$ and each integer m with $0 \le m \le b$, define the m^{th} *neighbor set* of i as

$$N^m(i) = \{j \in I\ :\ H(i\ ,\ j) = m\},$$

the set of all integers whose binary representations differ from that of i in exactly m positions (i.e. with Hamming distance equal to m). If index j is transmitted, $N^m(j)$ is the set of all indices that can be received if exactly m bit errors occur.

In order to describe the assignment of binary indices to codevectors, let $f(\mathbf{w})$ be a one-to-one mapping or *permutation function*, from the codebook CB to the index set I. In other words, f specifies a permutation (or reordering) of the codebook CB by assigning new indices to its vectors.

Suppose an input vector $\mathbf{v}$ is approximated (vector quantized) by the codevector $\mathbf{w}_i$ and suppose that index j is received. The average distortion due to the combined effect of the quantization and channel index error (which causes index i to be improperly received as index j) can be upper bounded by the triangle inequality as

$$E[d(\mathbf{v}\ ,\ \mathbf{w}_j)] = E[d(\mathbf{v}\ ,\ \mathbf{w}_{i(\mathbf{v})})] + E[d(\mathbf{v}\ ,\ \mathbf{w}_j) - d(\mathbf{v}\ ,\ \mathbf{w}_{i(\mathbf{v})})]$$

$$\le E[d(\mathbf{v}\ ,\ \mathbf{w}_{i(\mathbf{v})})] + E[d(w_{i(\mathbf{v})}\ ,\ \mathbf{w}_j)].$$

where $i(\mathbf{v})$ denotes the index of the codevector selected by the encoder and is a function of the input $\mathbf{v}$, and j denotes the received index that depends on both $i(\mathbf{v})$ and on the channel bit errors.

The right hand side of the above inequality provides an upper bound for the average distortion of the VQ system. We seek to *minimize this upper bound* (rather than the actual average distortion) over all possible permutation functions. This avoids the need for explicit consideration of the input vector statistics in optimizing the choice of permutation. The first term of this bound, $E[d(\mathbf{v}\ ,\ \mathbf{w}_{i(\mathbf{v})})]$, the expected distortion in the absence of channel errors is independent of the assignment of indices to codevectors.

In order to minimize the upper bound in this case, we must find a permutation which minimizes the second term, which we call the *bit error distortion*:

$$D \equiv \{E[d(w_{i(\mathbf{v})}\ ,\ \mathbf{w}_j)]\}$$

where the minimization is taken over all possible codebook permutations f. by minimizing the upper bound rather than the exact distortion, *we have eliminated any need to consider the distribution of the input vector* $\mathbf{v}$ *in order to find an optimal permutation.*

Evaluating the bit error distortion gives the result

$$D = \sum_{r=0}^{2^b-1} \mathbf{C}(\mathbf{w}_r)$$

where $\mathbf{C}(\mathbf{w}_r)$ is the *total cost* associated with a codevector $\mathbf{w}_r$ and is given by

$$\mathbf{C}(\mathbf{w}_r) = \sum_{m=0}^{b} \varepsilon^m (1-\varepsilon)^{b-m}\, \mathbf{C}_m(\mathbf{w}_r)$$

where $\mathbf{C}_m(\mathbf{w}_r)$ is the m^{th} *cost* of codevector $\mathbf{w}_r$ which measures the relative contribution to the overall expected bit error distortion of the codebook when exactly m bit errors occur and $\mathbf{w}_r$ is selected by the encoder, defined as

$$\mathbf{C}_m(\mathbf{w}_r) = p(\mathbf{w}_r) \sum_{x \in N^m(r)} d(\mathbf{w}_r\ ,\ \mathbf{w}_x)$$

If we use notation involving permutation functions f, and note that for any index $q \in I$, $f^{-1}(q) = \mathbf{w}_q$, then we want to find

$$\min_f \left\{ \sum_{r=0}^{2^b-1} \mathbf{C}(f^{-1}(r)) \right\}$$

where the minimization ranges over all possible permutation functions f. We desire to minimize this quantity in the hope of reducing the expected distortion introduced to the VQ system from the combined effect of coding error and channel noise. In general, as can be seen in the above equation, this minimization involves the knowledge of the channel's error transition probability ε as well as the distance function d and the codevector probabilities $p(\mathbf{w}_r)$.

5.4. The Algorithm

We have developed an algorithm that rearranges a codebook such that the summation above is a local minimum. The main idea of the algorithm involves iteratively switching the positions of two codevectors, reducing the expected value of bit-error distortion at every switch. Thus a monotonic decrease in distortion results as the algorithm progresses. The choice of which pair of vectors to switch in the codebook at each iteration is determined by an ordering process. Each codevector $\mathbf{v}$ is assigned a number, $\mathbf{C}(\mathbf{v})$ as presented earlier. The codevectors are sorted in decreasing order of their cost values. The vector with the largest cost, say $\mathbf{v}_0$ is selected as a candidate to be switched first. A trial is conducted, where $\mathbf{v}_0$ is temporarily switched with each of the other codevectors to determine the potential decrease in the summation of costs of all the vectors following each switch. The codevector which yields the greatest decrease in summed cost when switched with $\mathbf{v}_0$ is then switched with $\mathbf{v}_0$. If no such vector exists (an unsuccessful switch attempt), then the second highest cost vector is used to check for its most cost efficient switch, and so on. If every codevector is such that when switched with every other vector, an increase in summed cost results, then the algorithm halts in a locally optimal state.

We have experimented with this algorithm on a variety of codebooks ranging in size from 16 to 256 and found that a useful drop in average distortion can be achieved by application of the algorithm on a codebook with a random initial index assignment.

6. Hardware Issues

From the outset, we recognized the need to develop the expertise and the hardware resources for the demanding requirements of implementing VQ based coders in real-time. We have been developing semi-custom IC chips to do the intensive pattern matching task of codebook search in VQ. A stepping-stone in learning about VQ based hardware implementation was the design of a real-time version of an Adaptive Vector DPCM. This coder used a Codebook Search Processor (CSP), a single board VQ processor that is centered around a VLSI pattern matching chip [15] and a single TMS32010 signal processor chip. This coder operates in real-time at rates in the neighborhood of 14 kb/s and is described in another paper at this conference [16]. Subsequently, as our algorithm development advanced, we recognized the need for a more powerful DSP processor and migrated to the AT&T DSP32. One of the advantages of selecting a floating point chip was the faster software development resulting from the avoidance of handling scaling and other problems that arise with the limited word length of fixed point DSP chips. It becomes very easy to transfer an algorithm from a general purpose computer simulation to a real-time implementation without degrading the performance due to word-length reduction.

Our final hardware implementation, to be completed in Fall 1987, will consist of a single board containing three AT&T WE® DSP32 chips, two with externally addressable memory, a microcontroller chip and an extended range AT&T 15-bit codec chip. Complexity reduction techniques for both algorithms has eliminated the need to use the CSP as a VQ coprocessor board. In fact, we expect that a second generation version of our speech coder board will be implementable with a single DSP32 chip running at the higher clock rate of 25 MHz. We have already achieved real-time implementation of somewhat simplified versions of VAPC and PVXC in the laboratory using a single DSP32 chip.

7. Conclusions

We have described two speech coders, VAPC and PVXC, both of which achieve very high quality at low bit-rates, and which require a tolerable level of computational complexity. Both are reasonably well-structured and in somewhat simplified form are suitable for real-time implementation using only one or two of today's programmable DSP chips.

The Jet Propulsion Laboratory is currently planning initial field trials of our prototype speech coder in 1988. In addition to the new understanding and new algorithmic techniques that are spinoffs of this project, it is possible that one of our algorithms may be adopted or modified for a future commercial mobile satellite service. It also appears that our coding algorithms will be very attractive for high-quality speech coding in other mobile communication applications at rates between 4000 and 9600 bits per second.

References

1. B. S. Atal and M. R. Schroeder, "Stochastic Coding of Speech Signals at Very Low Bit Rates," *Conf. Rec., IEEE Int'l Conf. Commun.*, pp. 1610-1613, Amsterdam, May 1984.
2. Y. Shoham and A. Gersho, "Pitch Synchronous Transform Coding of Speech at 9.6 kb/s Based on Vector Quantization," *Conf. Rec., IEEE Int'l Conf. Commun.*, pp. 1179-1182, Amsterdam, May 1984.
3. H. Koyama and A. Gersho, "Fully Vector-Quantized Multipulse LPC at 4800 bps," *Proc. ICASSP*, Tokyo, Japan, April 1986.
4. J. H. Chen and A. Gersho, "Vector Adaptive Predictive Coding of Speech at 9.6 kb/s," *Proc. ICASSP*, pp. 1693-1696, Tokyo, Japan, Apr. 1986.
5. J. H. Chen and A. Gersho, "Real-Time Vector APC Speech Coding at 4800 bps with Adaptive Postfiltering," *Proc. ICASSP*, Dallas, April 1987.
6. G. Davidson and A. Gersho, "Complexity Reduction Methods for Vector Excitation Coding," *Proc. ICASSP*, pp. 3055-3058, Tokyo, Japan, April 1986.
7. G. Davidson, M. Yong, and A. Gersho, "Real-Time Vector Excitation Coding of Speech at 4800 bps," *Proc. ICASSP*, Dallas, Apr. 1987.
8. B. S. Atal and M. R. Schroeder, "Adaptive Predictive Coding of Speech Signals," *Bell Syst. Tech. J.*, vol. 49, pp. 1973-1986, Oct. 1970.
9. V. Cuperman and A. Gersho, "Vector Predictive Coding of Speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. COM-33, pp. 685-696, July 1985.
10. J. H. Chen and A. Gersho, "Gain-Adaptive Vector Quantization for Medium-Rate Speech Coding," *Conf. Rec., IEEE Int'l Conf. Commun.*, pp. 1456-1460, Chicago, June 1985.
11. M. R. Schroeder and B. S. Atal, "Code-Excited Linear Prediction (CELP): High-Quality Speech at Very Low Bit Rates," *Proc. ICASSP* , Tampa, March 1985.
12. M. Copperi and D. Sereno, "CELP Coding for High-Quality Speech at 8 kbit/s," *Proc. ICASSP*, pp. 1685-1688, Tokyo, Japan, April 1986.
13. I. M. Trancoso and B. S. Atal, "Efficient Procedures for Finding the Optimum Innovation in Stochastic Coders," *Proc. ICASSP*, pp. 1681-1684, Tokyo, Japan, April 1986.
14. H. Alrutz, "Implementation of a Multi-Pulse Coder on a Single Chip Floating-Point Signal Processor," *Proc. ICASSP*, Tokyo, Japan, April 1986.
15. G. Davidson and A. Gersho, "Application of a VLSI Vector Quantization Processor to Real-Time Speech Coding," *IEEE J. Selected Areas in Commun.*, vol. SAC-4, January 1986.
16. K. Zeger and A. Gersho, "Real-Time Vector Predictive Coding of Speech," *Conf. Rec., IEEE Int'l Conf. Commun.*, Seattle, June 1987.