

A SOFTWARE BASED, 13 KBITS/S REAL-TIME INTERNET CODEC

A Thesis

by

MARC A. RANDOLPH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 1996

Major Subject: Electrical Engineering

A SOFTWARE BASED, 13 KBITS/S REAL-TIME INTERNET CODEC

A Thesis

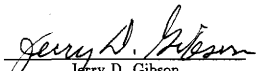
by

MARC RANDOLPH

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

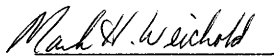
MASTER OF SCIENCE

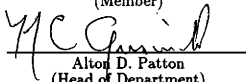
Approved as to style and content by:


Jerry D. Gibson
(Chair of Committee)


Jay Livingston
(Member)


Maurice Rahe
(Member)


Mark Weichold
(Member)


Alton D. Patton
(Head of Department)

August 1996

Major Subject: Electrical Engineering

ABSTRACT

A Software Based, 13 kbits/s Real-Time Internet Codec. (August 1996)

Marc A. Randolph, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Jerry D. Gibson

Bandwidth usage is a prime concern to many on the Internet, especially for users on low bit rate channels. As video conferencing becomes more popular, the need for efficient software based compression of video and audio becomes more important. This work develops a scalable, real-time, software based speech codec for use on desktop computers. The system is based on subband coding, adaptive prediction, and Huffman coding, and is capable of bit rates below 13 kbits/s for communications quality audio. The quality may be "scaled" up by allocating additional bits to the subbands. This coder has been successfully implemented in real-time on a Sun Sparc 10 platform.

To my parents, for their unfailing encouragement and support.

ACKNOWLEDGMENTS

First and foremost, one thousand thank you's are owed to Dr. Jerry Gibson. Not only for his excellent guidance, but for also believing in me enough to make this all possible through moral and financial¹ support. He was always there to answer my questions, and (thankfully) came to me when I wasn't smart enough to ask him.

Roderick Maddox, my officemate and good friend: one of these days, we will actually hit on an idea to make us rich and famous. I can't wait. And we won't hire anyone that scuffs their feet!

Many thanks are due to the members of the Multimedia Communications and Networking Lab, especially Stan McClellan for taking the time to explain many aspects of speech coding when I was getting started, and Tom Brown for attempting to integrate the coder I developed into a usable tool.

Lastly, my parents, Beth and Verle Randolph, deserve credit for too many things to list. A more perfect pair of parents could not be dreamed of.

This work is in memory of Jill Randolph and Dr. Carl Erdman.

¹This research was supported by the Texas Advanced Technology Program, Project No. 999903-017, and by the National Science Foundation Grant No. NCR-9318337 under Research Agreement No 25429-5498 with Cornell University.

TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION	1
	A. Motivation	1
	B. Outline	3
II	SPEECH CODING AND COMPRESSION	4
	A. Background	4
	1. Bit Rate	6
	2. Quality	6
	3. Delay	8
	4. Complexity	8
	B. Justification for choosing ADPCM	9
III	ITU-T ADPCM STANDARD	10
	A. Introduction	10
	B. Coder Structure	12
	C. Predictor	14
	D. Quantizer	16
IV	SUBBAND CODING	19
	A. Introduction	19
	B. Quadrature Mirror Filters	20
	C. Polyphase quadrature mirror filters	25
	D. Infinite impulse response based QMF	26
	E. Subband choice	26
V	SYSTEM DESIGN	27
	A. Overview	27
	B. Quadrature Mirror Filter	27
	C. Speech coder	30
	1. Predictor	30
	a. Pole-Zero predictor	30
	b. Pitch predictor	31
	2. Quantizer	34

CHAPTER	Page
D. Huffman coder	35
E. Packet loss issues	37
VI RESULTS	39
A. Overview	39
B. Bit Rate	39
C. Complexity	40
D. Quality	42
E. Delay	47
F. Summary	47
G. Areas for Future Study	48
REFERENCES	50
APPENDIX A	54
APPENDIX B	56
APPENDIX C	57
VITA	58

LIST OF TABLES

TABLE		Page
I	Speech Coders	7
II	Filterbank output	29
III	Adaptive quantizer multipliers	35
IV	Example Huffman code	36
V	Subband Huffman code	38
VI	Selected subband bit allocations	40
VII	Time spent filtering	41
VIII	Total delay time	47

LIST OF FIGURES

FIGURE		Page
1	Block diagram of the ADPCM encoder	11
2	Block diagram of the ADPCM decoder	13
3	Example speech samples	14
4	An example uniform eight level quantizer	17
5	Quadrature mirror filter pair	20
6	A complete two band quadrature mirror filterbank with coding	21
7	A four band tree structured QMF with coder	23
8	Uncancelled aliasing in a QMF	24
9	Block diagram of implemented system	28
10	Example pitch redundancy in speech	32
11	Typical pitch predictor structure	33
12	Uncoded speech sample	44
13	Speech sample after suffering packet loss	45
14	Speech sample after packet loss with double transmitted low band	46
15	Two-band QMF without coding	54

CHAPTER I

INTRODUCTION

A. Motivation

As the general public flocks to the Internet, the underlying network is becoming increasingly saturated with traffic. The network has a wide range of uses, including the transfer of sound, still images, motion pictures, files, and more recently, the World Wide Web and real-time applications. Although some parts of the Internet backbone run at DS3 rates (45 Mbits/s) or above, many links are only DS1 (1.544 Mbits/s), ISDN (128 kbits/s), or even lower. At these rates, a handful of users, each simultaneously requesting what they believe to be a nominal amount of information, could saturate the network. Two examples of uncompressed data streams include full color video, requiring well in excess of 50 Mbits/s, and toll-quality audio, needing 64 kbits/s. The most common method of reducing the likelihood of saturation is the use of compression. On the Internet, algorithms such as JPEG (Joint Picture Experts Group), GIF (Graphical Interchange Format), and MPEG (Motion Picture Experts Group) are all very widely accepted and supported compression standards for the interchange of image and video data, even when sent to drastically different computer systems. This "cross-platform" ability is becoming increasingly important in the rapidly growing heterogeneous computing environment. Unfortunately however, there is no such widely accepted format for audio transmission over the Internet.

Some might argue that available bandwidth on the Internet is continually increasing, hence the need for data compression and further research is minimal. Although it is true bandwidth has increased over time, and is predicted to continue to do so,

The journal model is *IEEE Transactions on Automatic Control*.

history has shown that excess bandwidth is always put to use, either by new applications, by an increase in the number of clients served, or by an increase in the quality of services provided. Their argument also ignores the growing popularity of wireless connections, where bandwidth is usually at a premium and sometimes highly variable. Alternately, some channels are fixed at a certain rate, a portion of which is dedicated to voice information, and the remainder to video. At any moment, a person on one end of a video conferencing system might be silent, so little speech data needs to be transmitted during that period. A good conferencing system will be able to dynamically lower its speech bit rate and allow the video coder to utilize that additional bandwidth in order to obtain the best possible video quality. When the speaker resumes, the system transparently takes back the bandwidth and reallocates it to voice coding. Any of these items alone justify the continued use and further research in the area of compression and efficient coding methods. The transmission of speech is no different, as it is just one of an increasing number of data streams.

There are currently several incompatible speech compression systems for the Internet, including the Unix audio tool Vat, the PC based I-Phone, and the CU-SeeMe video conferencing system. These systems suffer from various problems such as low audio quality, high bit rates, high complexity, and poor ability to recover from packet losses. All of these problems stem from deficiencies in the algorithms being used.

The motivation for this work is to develop a speech compression algorithm which will provide an acceptable compromise between the issues of quality, bit rate, complexity, and data loss. Each of these problems have been well researched and are complete works of study on their own. Rather than completely developing new theories, we shall attempt to adapt several existing algorithms and combine them in such a way that the system will have acceptable quality while maintaining a complexity

level low enough to run in real-time on common workstations and personal computers, without using a DSP or other dedicated hardware.

B. Outline

First, an overview of speech coding and commonly used terms is presented in Chapter II, followed by a more detailed description of the ITU-T ADPCM speech standard in Chapter III. We will move on to give some details of various subband filter structures in Chapter IV, followed by Chapter V detailing the new system and its theory of operation. Chapter VI will conclude this work with results and a discussion of possible future areas of study.

CHAPTER II

SPEECH CODING AND COMPRESSION

A. Background

Speech is the most common and widely used method of communications due to its natural effectiveness and instant interactivity, allowing the speaker to insure he is being understood. This will continue into the foreseeable future as video is combined with speech to form multimedia/videoconferencing systems. To increase the efficiency of audio transmission, a great deal of research time, money, and effort has gone into digital speech coding and compression.

Although initially developed for military secure communications, commercial enterprise now constitutes the driving force behind digital speech coding. With the recent explosion of cellular phone usage, and the expected widespread acceptance of personal communications systems (PCS), this research will continue well into the future, searching for higher quality and increased bandwidth efficiency.

The term "speech coding" refers to the method of reducing the amount of information needed to transmit or store a speech signal. There are two ways to achieve this reduction: using either lossless or lossy techniques. Lossless refers to coding a signal such that no information is thrown away. Although this preserves the exact original, there are no known methods of removing most of the redundancy of speech in a lossless manner. This is why, for over twenty years, research has concentrated on using lossy compression techniques since a great deal of information can be missing while keeping the perceptible quality high.

In the late seventies, speech compression followed two main routes: if high quality was desired, a "waveform coder" could be used. If instead, the goal was low bit

rates, a "vocoder" (a contraction of "voice coder") was used. During this period of time, low bit rate and high quality were mutually exclusive terms when referring to real-time systems. A waveform coder attempts to preserve the overall shape of the time domain speech waveform through the use of representative quantized samples. The ITU-T Adaptive Differential Pulse Code Modulation (ADPCM) standard is a waveform coder. A vocoder, at the opposite extreme, makes no attempt at preserving the time domain speech waveform, rather it attempts to artificially model the human vocal tract. The most well known vocoder is the U.S. Government Federal Standard 1015, also known as LPC-10e.

The pace of coder development increased in the eighties, however. It did not take researchers long to begin experimenting with other methods of speech coding, usually starting with a vocoder or waveform coder and mutating it. These new coders fall into a broad class of what is now called "hybrid" coders. These hybrid coders overcame the synthetic sounds of the vocoder, often producing a much more natural sounding reproduction of speech while maintaining a lower bit rate.

Complex techniques were put to use, including taking advantage of the natural masking abilities of the human ear (called perceptual masking), and using newly available high speed digital signal processing to quantize parameters in blocks (vector quantization) rather than individually.

Speech coders are designed with the following tradeoffs in mind: bit rate, quality, delay, and complexity. While one would hope to have the best of all of these attributes, it is not generally possible. In order to increase quality, for example, the bit rate or complexity must generally go up as well. Lower delay generally means higher complexity as well. Other tradeoffs can occur in situations, such as a rate limited wireless channel or complexity limitations due to power constraints. These tradeoffs are defined and explored in more detail in the following subsections.

1. Bit Rate

The bit rate provides an indication of how well the data stream was compressed, usually compared to a telephone bandwidth stream which is sampled at 8 kHz with an 8 bit logarithmic quantizer (i.e., 64 kbits/s total). Rates vary from 64 kbits/s for long distance and international telephone networks, down to 800 bits/s for some secure communications applications. Table I shows the bit rates for many common coders.

It should be noted that the bit rate of a coder does not have to be constant. In fact, a great deal of compression can be had by allowing the encoding rates to vary according to how well a predictor is performing, or by not transmitting data during periods of silence in the conversation.

2. Quality

Many people view quality as the most important part of a speech coder because if the quality is too low, the consumer may shy away from using the product when it is not absolutely necessary. Low quality can be caused by many things, including poor speech coding or modeling, channel errors, packet losses, or background noise effects.

The most widely used measure of perceptual quality of a speech coder is the mean opinion score (MOS), whereby blind tests are performed using a specified set of conditions. The "graders" give a rating of 1 to 5, corresponding, in order, to a rating of bad, poor, fair, good, or excellent. The tests typically consist of several different talkers and a number of reference coders. MOS scores for some common coders can also be seen in Table I. SNR and SEGSNR are popular absolute measurements, but they often produce results counter to perceptual results, hence are not nearly as widely quoted.

Table I. Speech Coders

Year of Introduction	Bit Rates kbits/s	Description	MOS
1972	64	PCM (for PSTN)	4.4
1976	2.4	LPC-10e (Fed Std 1015)	2.7
1984	32	G.721 ADPCM (for PSTN)	4.1
1987	24	G.723 ADPCM	4.0
1990	16	G.726 ADPCM	3.9
1990	4.15	Inmarsat (Satellite)	≈ 3.2
1991	13	GSM (European Cellular)	3.6
1991	4.8	CELP (Fed Std 1016)	3.2
1992	16	G.728 (Low delay-CELP)	4.0
1992	8	VSELP (US Cellular)	3.5
1993	1-8	QSELP (US CDMA)	≈ 3.4
1993	6.8	VSELP (Japanese Cellular)	≈ 3.3
1995	8	G.729 (new toll-quality)	≈ 4.2
1995	6.3	G.723.1 (in H.323 & H.324)	3.98
1995	5-6	Half-Rate GSM	≈ 3.4
1996	2.4	New low rate Fed std	≈ 3.3

3. Delay

Although delay could be viewed as a component of quality, the design methodology and complexity of designing the coder can often be quite different if the system is required to have low delay, such as for a real-time system. This is in contrast to an application such as offline storage of speech, where the impact of delay is minimal. We will now discuss several important aspects of delay: algorithmic, computation, and transmission.

Most lower bit rate coders operate on blocks of speech rather than on a sample by sample basis. This allows them to encode more efficiently, but at a price; it takes extra time. The sum of time for accumulating a block (sometimes called a frame) plus other inherent data preparation delays make up the *algorithmic delay*. To code a specific frame of speech, some form of processing is required, creating a *computation delay*, which is dependent on the implementation of the algorithm on a specific processor. Finally, a *transmission delay*, the time it takes for a block to propagate from transmitter to receiver, must be taken into account. The sum of all these component delays form the overall system delay, which must be lower than a pre-specified target to be considered usable for a particular application. In real-time communications, most speakers find delays larger than about 250 msec to be quite disruptive to typical conversation patterns.

4. Complexity

The complexity of a speech compression system often dictates where it can be used. For example, a system which is too complex will consume too much power, thereby making it unusable in a battery powered environment. Highly complex systems also generally require components which cost more.

After a system is designed and simulated on host processors, they are often implemented on DSP chips, and possibly on VLSI devices if demand is high enough. Algorithmic speed and memory requirements are the main driving force behind the complexity (and therefore cost) issue since they usually dictate the processing requirements of the DSP. An additional factor in the complexity equation is floating point vs. fixed point implementations. Algorithmically, it is often easier to use floating point computations, but the cost and power requirements of floating point DSP chips is considerably higher than that of fixed point DSP's.

B. Justification for choosing ADPCM

Although there have been tremendous low rate advancements in recent years, they all require DSP technology to operate in real time. This is a problem for computer users wishing to use speech to communicate over modem channels or over the Internet since such schemes would not run on existing CPU's in real-time.

The ADPCM style coder was chosen for this work because it has acceptable quality while maintaining a low complexity level so that it does not requiring additional hardware to run on the desired platforms.

CHAPTER III

ITU-T ADPCM STANDARD

A. Introduction

The Adaptive Differential Pulse Code Modulation (ADPCM) standard was set by the Telephony branch of the International Telecommunications Union (ITU-T) (formally the CCITT) in response to the need for an international standard for speech coding and compression. The original recommendations, called G.721 and G.723, took 64 kbits/s speech and compressed it to either 40, 32, or 24 kbits/s in a lossy manner. In 1990, recommendation G.726 was created, encompassing G.721, G.723, and adding a new 16 kbits/s compressed rate [1].

When the original need for the G.721 standard was identified, a CCITT Committee Study Group was assigned the task of developing a compression algorithm which would maintain the performance of standard 64 kbits/s PCM as closely as possible. This included many items, but concentrated on having a simple enough scheme to be implementable on the hardware available at that time while having minimal encoding and decoding delays, allowing for real-time speech communication [2].

The system is designed to take voice band signals and compress them for efficient transmission over the telephone network. A "tone and transition detector" is also included in the design to allow for adequate performance of frequency shift keying (FSK) modems when operating on links which are coded with ADPCM.

This chapter will provide a somewhat detailed description of the ITU-T ADPCM standard. Additional information can be obtained from [1] and [3].

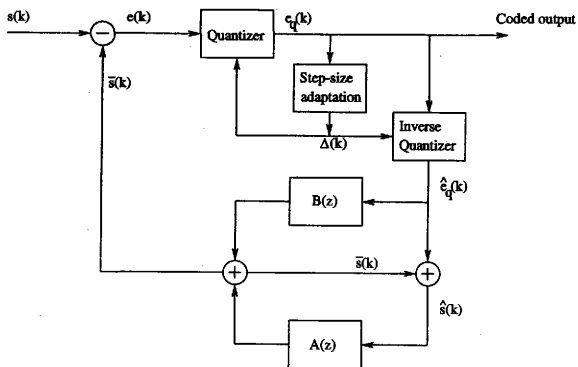


Fig. 1. Block diagram of the ADPCM encoder

B. Coder Structure

A block diagram of the ADPCM encoder is shown in Fig. 1. The system is based on two major components: an adaptive predictor and an adaptive quantizer. The decoder, shown in Fig. 2, is actually a subset of the encoder, simply outputting $\hat{s}(k)$ instead of $e_q(k)$.

The basic algorithm is to take a speech sample at time instant k , denoted by $s(k)$, and subtract it from the predicted value $\bar{s}(k)$. The prediction error, called $e(k)$, is the difference between the actual and the predicted values. $e(k)$ is then quantized to $e_q(k)$ and transmitted to the receiver. At the same time, this quantized value is decoded and added to $\bar{s}(k)$ to form the reconstructed value $\hat{s}(k)$. This reconstructed sample is related to the original sample by

$$\hat{s}(k) = s(k) + q(k) \quad (3.1)$$

where $q(k)$ is the error introduced by the quantizer,

$$q(k) = e_q(k) - e(k) \quad (3.2)$$

As can be seen above, if the quantizer were taken out of the system, the error introduced by the quantizer would be zero and the reconstructed value would exactly match the original speech sample. The receiver can produce an identical reconstruction assuming no transmission errors occur.

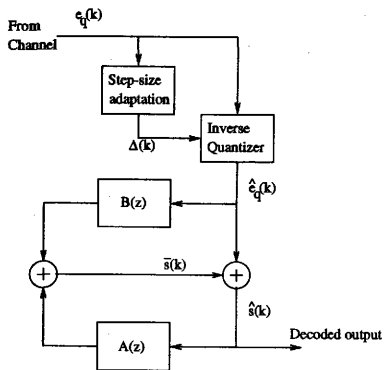


Fig. 2. Block diagram of the ADPCM decoder

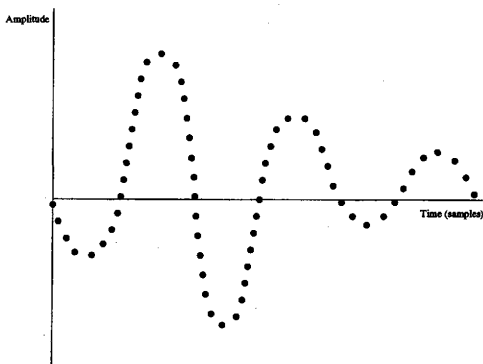


Fig. 3. Example speech samples

C. Predictor

The predictor section within the ADPCM standard consists of two components, one with a short memory, and one with a relatively longer memory, both working together to reduce short term redundancies. The predictor is responsible for lowering the dynamic range as much as possible so the quantizer can use smaller step sizes to more finely quantize the prediction error, thereby increasing the signal to noise ratio.

The predictor lowers the dynamic range by removing the majority of the correlation between consecutive samples. As can be seen from Fig. 3, it would be highly inefficient to code each sample independent of every other sample. Instead, the input stream is run through two filters which attempt to decorrelate the samples. The first one is an infinite impulse response (IIR) filter with two taps, given by

$A(z) = \sum_{i=1}^2 a_i z^{-i}$. In the time domain, this can be viewed as a weighted linear combination of previous reconstructed output samples, as shown in equation 3.3,

$$s_e(k) = \sum_{i=1}^2 a_i (k-1) \hat{s}(k-i) \quad (3.3)$$

where a simplified adaptive gradient algorithm is used to generate the a_i terms.

The second filter is a finite impulse response (FIR), all-zero predictor, $B(z) = \sum_{j=1}^6 b_j z^{-j}$, which has 6 taps to help improve predictor performance. Its time domain equivalent is given in equation 3.4 as a weighted linear combination of decoded previous prediction errors.

$$s_{ez}(k) = \sum_{j=1}^6 b_j (k-1) \hat{e}_q(k-j) \quad (3.4)$$

Coefficients are also generated from a simplified adaptive gradient algorithm. The two predictors outputs are summed together to produce the final prediction:

$$\bar{s}(k) = s_e(k) + s_{ez}(k) \quad (3.5)$$

The dual filter configuration was chosen as a tradeoff between having a very long memory (desirable for a high performance predictor) and error propagation in the system decoder due to transmission errors [4]. The pole predictor has only two taps and restricted coefficient values to maintain predictor stability since a single error could potentially affect many reconstructed samples. The control of decoder stability was the reason the two pole configuration was used. The zero predictor was added to increase the predictor memory, and therefore performance, since it does not infinitely propagate errors.

More extensive details about predictors and stability of coefficients can be found in [1] and [5], and a theoretical analysis of the ADPCM algorithm can be found in [6].

D. Quantizer

ADPCM uses a quantizer which is backward adaptive as well, allowing the step size of the quantizer to increase and decrease corresponding to changes in the prediction error, $e(k)$. Since $e(k)$ can vary drastically depending on the input signal $s(k)$, the quantizer is scaled by a quantity $\Delta(k)$, called the step size:

$$\Delta(k) = \Delta^\beta(k-1)M(|e_q(k)|) \quad (3.6)$$

where $M(\)$ is a time invariant multiplier which is dependent on the magnitude of the transmitted symbol, $e_q(k)$. The damping factor β is used to make the quantizer more robust to transmission errors. Commonly, values of β approach, but never exceed, one. This means that old step sizes will be forgotten over time. Fig. 4 shows the workings of an eight level quantizer in graphical form with the corresponding multipliers for each level. An important aspect of these multipliers is the fact they allow the step size to increase at a faster rate than decreases occur. This is due to research done by Jayant [7] which shows that overload noise in the quantizer causes a large drop in signal to noise ratio and is very disconcerting to listeners.

A Gaussian characteristic for quantizer decision and output levels is used, and the step size is adapted based on work done by Jayant, as well as Goodman and Wilkinson [8].

The inverse quantizer is present in the encoder so the encoding system can duplicate the decoders reconstruction and properly adapt the quantizer and predictor. If reconstructed values were not used in the adaption, the decoder would not be

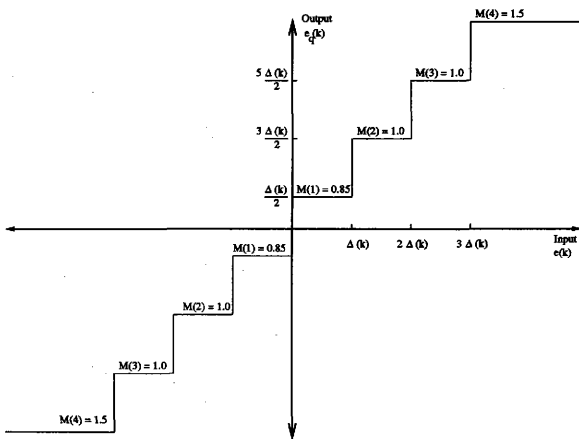


Fig. 4. An example uniform eight level quantizer

able to make the same adaption decisions as the encoder, causing mistracking, and correspondingly, a loss in quality.

As previously alluded to, the ADPCM standard must also allow for the transmission of voiceband data. This is done by slowing the rate of quantizer adaption down to minimally interfere with the symbols that modems use. The predictor coefficients are also reset to zero when entering this "locked" mode. For more details, see the ITU-T G.726 standard [1].

CHAPTER IV

SUBBAND CODING

A. Introduction

Since being introduced in 1976 by Esteban *et al.* [9] and Crochiere *et al.* [10], subband coding has developed a wide following, and much attention has been devoted to it by the speech coding community. Subband coding consists of breaking the original signal into frequency bands, then decimating and coding the individual bands. Although subband coding now has applications to images, this discussion will limit itself to speech coding aspects only.

The ideas behind subband coding are quite straight forward: a signal is separated into frequency bands, effectively translated to baseband, and sampled at its Nyquist rate. Each band is then coded individually with however many bits per band the designer wants, allowing perceptually important bands to be "weighted" with more bits. In other words, this approach places bits in frequency bands where they are needed the most. Separate bands have other advantages as well. They not only allow containment of quantization noise within the band, they also allow each band to be adapted independently, proportional to the root mean square (RMS) average speech level in their bands. Both abilities lead to perceived quality of the speech being improved.

This improvement in quality only occurred, however, if the proposed infinite impulse response (IIR) filters had very sharp transition bands and high stop band attenuation since intraband aliasing and frequency distortion were present. IIR filters have phase distortion problems as well [11]. Such problems suggest many areas for further research. This chapter will give an overview of the major results of this

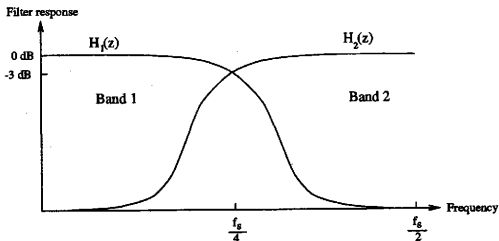


Fig. 5. Quadrature mirror filter pair

research related to subband filters. Additional details and references can be found in [12] and [13].

B. Quadrature Mirror Filters

If the filters used to create these frequency subbands are designed in a certain way, Esteban and Galand discovered [14] a full bandwidth signal can be divided into two overlapping subbands (illustrated in Fig. 5) such that the overlapped regions have special properties. These properties allow two subbands to be sampled at half the original sampling rate and still be transparently reconstructed.

Esteban and Galand named their new subbanding method a quadrature mirror filter; the block diagram of which is presented in Fig. 6. The sampled input speech signal, $x(k)$, is sent through two filters, $H_1(z)$ and $H_2(z)$, which have the property $H_2(z) = H_1(-z)$, or in the time domain,

$$h_2(k) = (-1)^k h_1(k) \quad (4.1)$$

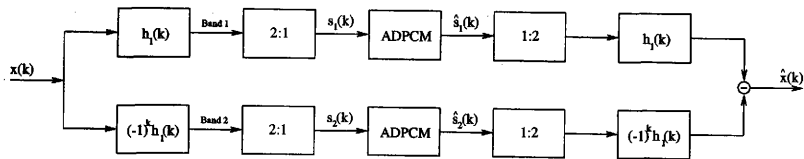


Fig. 6. A complete two band quadrature mirror filterbank with coding

$h_1(k)$ is an even tap, linear phase, low-pass finite impulse response (FIR) filter with its 3 dB point located at $f_s/4$, where f_s is the sampling rate of the input signal, $x(k)$. $h_1(k)$ completely specifies the QMF structure as the remaining analysis and synthesis filters are computed by modulating $h_1(k)$ by $e^{jk\pi}$, which is equivalent to multiplying by $(-1)^k$ as shown in equation 4.1. $s_1(k)$ and $s_2(k)$ are the outputs of the filters, containing the lower and upper frequency bands, respectively. They are formed by convolving the input sequence with the respective filters:

$$s_n(k) = \begin{cases} x(k) * h_n(k) & \text{for } k \text{ even} \\ 0 & \text{for } k \text{ odd} \end{cases} \quad (4.2)$$

where the zero (odd) terms correspond to the 2 : 1 sub-sampling. The sub-sampling introduces aliasing since the sampling rate is now half the original rate and the signal bandwidth of each band exceeds $f_s/4$ (violating the Nyquist criteria). The quadrature mirror filter structure, however, completely cancels aliasing upon reconstruction—assuming no noise is introduced into the system. See Appendix A for a proof of QMF aliasing cancellation. If the sub-sampled sequence is coded, the aliasing cancellation will be limited to the resolution of the coding scheme.

To reconstruct the subbands, each band is up-sampled by inserting a zero after every sample and running the resulting sequence through the reconstruction filter. After the pair of bands is up-sampled, they are combined to form the final output stream, $\hat{x}(k)$, as shown in equation 4.3.

$$\hat{x}(k) = \hat{s}_1(k) * h_1(k) - \hat{s}_2(k) * h_2(k) \quad (4.3)$$

There are several drawbacks of Esteban's QMF's, however. Since they are exactly symmetric, only a division by two can be made each time, meaning a tree structure

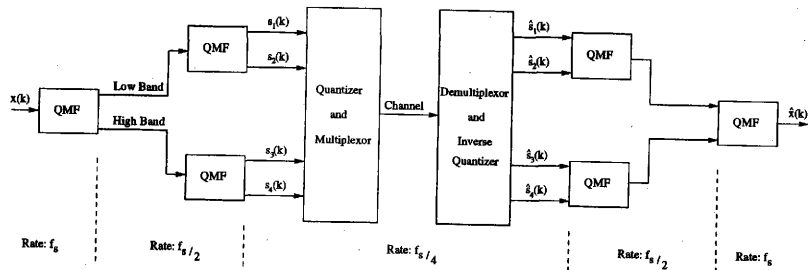


Fig. 7. A four band tree structured QMF with coder

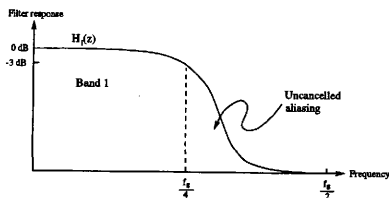


Fig. 8. Uncancelled aliasing in a QMF

will need to be used if more bands are desired. As the four band example in Fig. 7 shows, $x(k)$ can be divided into high and low frequency bands and sub-divided again, producing a total of four bands, $s_1(k)$ through $s_4(k)$, each with a bandwidth of one fourth the original signal. Although this recursive structure is easy to implement, a substantial delay is added for each level to the tree due to the large number of taps that must be used since Estebans QMF's are FIR based.

An additional problem is that if any noise is introduced into a band, the aliasing for both the "noisy" band and its mirror will not be completely cancelled. Noise can be caused not only by bit errors and packet losses, but also by a coarse quantizer. In fact, the worst noise in a quadrature mirror filter is caused by allocating zero bits to a quantizer, thereby causing that band to not be transmitted. This is the worst case for a QMF because the receiver will no longer have a way to cancel the the aliasing caused by its mirror, shown in Fig. 8.

These problems led researchers to continue investigating the quadrature mirror filters for further breakthroughs, which they found in polyphase filterbank structures.

C. Polyphase quadrature mirror filters

First presented by Rothweiler in 1983 [15], polyphase quadrature mirror filters were proposed as an improved subband coding filterbank structure. This structure reduces both complexity and delay since more than two bands can be formed directly from the original input samples. In addition, filters can typically be shorter since further filtering is usually not needed, hence the cascading, uncanceled aliasing is no longer a problem.

Cox pointed out these filters should really be called generalized pseudoquadrature mirror filters rather than polyphase quadrature mirror filters since (1) they are a generalization of the two band quadrature mirror filter concept to multiple bands, (2) the filters do not have to be implemented in a polyphase manner [16], and (3) they are not true quadrature mirror filters as the aliasing in every band is not completely cancelled [17]. Instead, only aliasing from the two nearest neighboring bands is cancelled. These designs assume aliasing from distant bands will be reduced significantly due to the stopband attenuation.

The design methodology for generalized pseudoquadrature mirror filters (abbreviated as GQMF in Cox's paper) begins with the design of a low-pass prototype with a bandwidth of one-half the desired subband bandwidth. The actual subband filters are created by modulating the low-pass filter with pairs of differently phased sinusoids, creating the desired number of subbands. The phases of the sinusoids are chosen so aliasing from adjacent bands will be cancelled and the overall frequency response of the system is as flat as desired. A totally flat frequency response is not only impossible to design for [18], but is also unnecessary since quantization noise normally exceeds the distortion caused by passband ripple on well designed GQMF's. However, before poor GQMF performance is attributed to quantization noise,

it should be noted quantization noise is often blamed for noise actually caused by uncancelled aliasing [19].

D. Infinite impulse response based QMF

More recently, many researchers have returned to IIR filters, this time in GQMF environments and with new design tools. IIR filters are attractive because they provide smaller transition regions and better stop band attenuation than FIR filters of longer length. Many individual IIR filters have been designed to exceed the performance of their FIR counterparts. Unfortunately, there is no unified design method and these filters are still a topic of intense research. This is due to the nonlinearity and complexity of the IIR filter design problem [20].

E. Subband choice

Generalized pseudoquadrature mirror filters are the obvious choice for use in this speech coding project. They provide a method of developing an arbitrary number of subbands from a prototype filter with minimal complexity. However, the design criteria for the prototype filters are quite stringent, making them very difficult to design [16]. Since the QMF filters, which Johnston spent considerable time designing [18], are known to minimize the frequency distortion inherent in the overall system, it makes logical sense to use these filters in a tree structure while minimizing the problems of tree coders outlined above. One major saving factor is the fact delay is not a serious issue since Internet packets have non-deterministic delays. The complexity issue must be addressed, however, due to the desire for real-time operation on generalized CPU's.

CHAPTER V

SYSTEM DESIGN

A. Overview

This chapter provides a detailed component description of the speech coder that was developed. The system is comprised of three main sections: a filterbank, a speech coder, and a Huffman coder. A block diagram of the encoding system is provided in Fig. 9. Speech samples are fed into the filterbank which produces five low rate frequency bands, which are then individually coded, Huffman coded, multiplexed, and transmitted over the channel. To decode the compressed samples, the receiver performs the same operations in the reverse order.

The chapter will proceed as follows: a description of the quadrature mirror filter structure used in this project is presented, followed by details about the predictors and quantizer implemented. The chapter closes with a discussion of the Huffman codes that were developed and a highlight of several packet loss issues.

B. Quadrature Mirror Filter

As was mentioned earlier, a quadrature mirror filter (QMF) can be used to break a signal into bands of specific bandwidth, allowing each to be coded separately. The most appealing aspect of using QMF's is the ability to assign a different numbers of bits to each band, depending on the the desired quality. Although QMF's can be performed in many ways, the ones used in this project consist of two parts: a generic filter unit to produce each subband and a decimation section.

The filterbank of QMF's are arranged in a tree structure to produce the subbands. The tree structure used in this system is the same as shown in Fig. 7 except that the

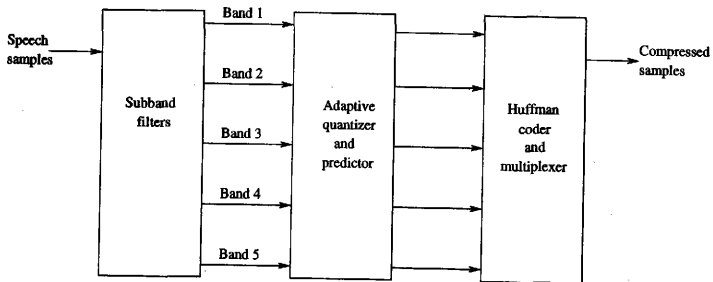


Fig. 9. Block diagram of implemented system

Table II. Filterbank output

Band Number	Frequency Range (Hz)	Bandwidth (Hz)
1	0-500	500
2	500-1000	500
3	1000-2000	1000
4	2000-3000	1000
5	3000-4000	1000

low frequency band, $s_1(k)$, is split one additional time to form a total of 5 subbands, summarized in Table II.

To perform the filtering, a convolution was implemented to produce each subband $s_i(k)$,

$$s_i(k) = \sum_{m=0}^{F-1} h_i(m)s(k-m) \quad (5.1)$$

for band i , where $s(k)$ is the input speech and h_i are the F filter coefficients for band i . Appendix B lists the coefficients used in this implementation. To save computation time, the decimation operation was also included in the convolution, actually yielding the final filter operation

$$s_d(l) = \sum_{m=0}^{(F-1)/2} h_i(2m)s(l-2m) \quad (5.2)$$

so $s_d(l)$ is the decimated version of $s_i(k)$, in other words, $s_d(l) = s_i(2l)$. The reconstruction is performed in a similar manner, using equation 5.1 after zeros are

interleaved into $\hat{s}_i(k)$ to provide up-sampling.

C. Speech coder

This section will cover the core speech coding component of the system, comprising two main sections, an adaptive predictor and an adaptive quantizer. The system is modeled after the ITU-T ADPCM standard with a few simplifications of components which were either unneeded or not worth the additional complexity for the marginal quality improvement at this low bit rate.

1. Predictor

As mentioned in Section C of Chapter III, a predictor's job is to remove redundancies in the speech waveform. This can be done two ways: by taking advantage of short term sample to sample correlation, or by longer term pitch period repetition. Adaptive predictors were chosen since their performance is much better than fixed-tap predictors at low bit rates [21]. Detailed below are the two schemes implemented in this coder.

a. Pole-Zero predictor

The short term predictor, commonly called the pole-zero predictor, is the predictor used in the ADPCM standard. It removes sample to sample correlation using weighted linear combinations of previous reconstructed samples and prediction errors, given by equations 3.3 and 3.4. The coefficients for the pole predictor are adapted by the gradient algorithm, which produces

$$\hat{a}(k+1) = \alpha_a \hat{a}(k) + K(k+1)e_q(k+1) \quad (5.3)$$

with

$$K(k+1) = \frac{g_a V(k)}{100 + V^T(k)V(k)} \quad (5.4)$$

where $\hat{a}^T(k) = [a_1(k) \ a_2(k)]$ is the vector of pole predictor coefficients and $V^T(k) = [\hat{s}(k) \ \hat{s}(k-1)]$ is the vector of previous reconstructed samples. g_a is a parameter to optimize system performance and α_a dampens previous predictions for more robust transmission over noisy channels. The adaption rule for the zero predictor coefficients is identical, replacing ALL occurrences of a with b , and letting $\hat{b}^T(k) = [b_1(k) \ b_2(k) \ \dots \ b_6(k)]$, with $V^T(k) = [e_q(k) \ e_q(k-1) \ \dots \ e_q(k-5)]$.

Since the IIR predictor has only two poles, the stability can be easily checked with the following constraint:

$$-0.75 \leq a_2 \leq 0.75 ; |a_1| \leq \frac{15}{16} - a_2$$

b. Pitch predictor

Pitch prediction, another method for removing redundancy, was implemented in this work. Its job is to remove any correlation that occurs between pitch periods in speech, as displayed in Fig. 10. The pitch period of most speakers lie in a range between 55 and 400 Hz, with males falling in the lower end of the range and females and children tending towards the upper end. That frequency range corresponds to looking back over a range of 20 to 145 previous samples for speech sampled at 8000 Hz.

In order to find the pitch period for a frame of speech, one must search over the possible samples to find the most likely period. One method to do this is to find the maxima of the normalized correlation, given by

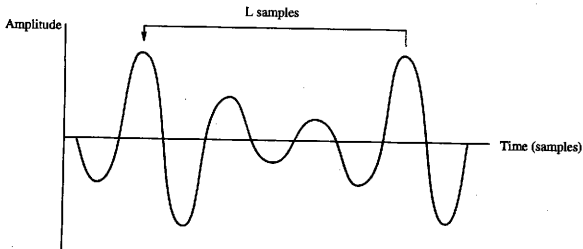


Fig. 10. Example pitch redundancy in speech

$$\rho(L) = \frac{\phi(0, L)}{\sqrt{\phi(0, 0) \cdot \phi(L, L)}} \quad (5.5)$$

where L is searched over the expected lag period range mentioned above and $\phi(i, j)$ is the auto-covariance,

$$\phi(i, j) = \sum_{k=0}^{N-1} s(k-i)s(k-j) \quad (5.6)$$

with N representing the number of samples in a frame and $s(k)$ being the input sample to be predicted [22].

Once the pitch lag L is located, the long-term predictor operates as a three tap lag filter, given by

$$P(z) = \beta_{-1}z^{-(L-1)} + \beta_0z^{-L} + \beta_1z^{-L+1} \quad (5.7)$$

where the set $\{\beta_{-1}, \beta_0, \beta_1\}$ are the long-term predictor coefficients. This predictor is

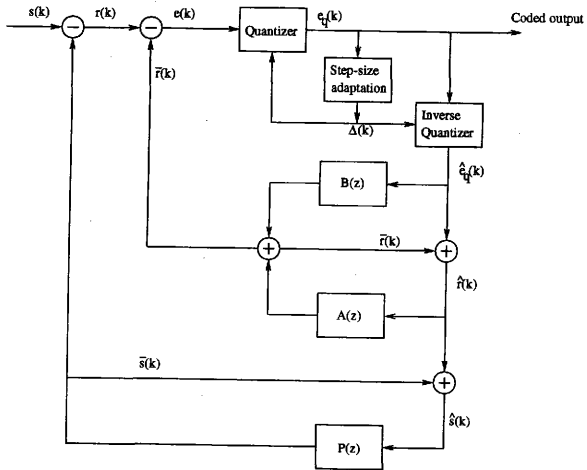


Fig. 11. Typical pitch predictor structure

shown in Fig. 11 along with the short term predictors $A(z)$ and $B(z)$.

It should be quickly noted the pitch of a speaker almost never remains constant, even within a frame of speech. It usually changes slowly, however, and these changes can be tracked. One method of tracking changes is by using an adaptive step gradient algorithm [23]. This algorithm and the adaptation rule for the long-term predictor coefficients β_n are described in detail in [23] and [24].

2. Quantizer

In order to turn the sampled speech into something which is transmittable with a reasonable number of bits, each sample is quantized using either 2, 3, or 4 bits (corresponding to 4, 8, or 16 levels), depending on the quality desired by the user. The operation of the adaptive quantizer was briefly described in Section D of Chapter III, but we will give more detail here on the actual implementation.

An adaptive uniform quantizer was chosen because of its very low complexity and high performance, even when compared to nonuniform quantizers. For each speech sample, a corresponding output from the quantizer is chosen, as given by the mapping from $n \cdot \Delta(k) \leq e(k) < (n + 1) \cdot \Delta(k)$ to $\hat{e}_q(k) = (n + 0.5) \cdot \Delta(k)$, where $\Delta(k)$ is the step size and n is an integer which varies over the range $-L/2$ to $L/2$, L being the number of levels for the quantizer.

The step size is then adapted according to equation 3.6 with $\beta \simeq 1 - 2^{-9}$. Upper and lower limits for the step size are enforced to guarantee the step size does not adapt to a value much larger or smaller than is normally needed to represent the sample. As Jayant pointed out, the adaptive predictor should expand at a faster rate than it decreases [25]. The values of $M(\cdot)$ used in this quantizer are shown in Table III. These values are more aggressive than the ones which Jayant suggested, as explained in the Huffman coding section below.

Table III. Adaptive quantizer multipliers

	L=4	L=8	L=16
M(1)	0.92	0.90	0.90
M(2)	1.60	0.95	0.90
M(3)		1.25	0.95
M(4)		1.75	0.97
M(5)			1.20
M(6)			1.60
M(7)			2.50
M(8)			5.50

D. Huffman coder

In order to lower the bit rate of the system beyond that of simple quantization, a Huffman code was implemented. Huffman codes are probabilistic based entropy codes which represent symbols with predetermined bit patterns. These bit patterns are variable length, based on how likely it is for each symbol to be transmitted. A symbol that is very likely to be transmitted will be represented with only a few bits, while highly unlikely symbols may contain a very high number of bits.

For example, if four characters (A, B, C, and D) on a keyboard were of equal probability to be typed and transmitted across a digital link, it would require two bits per character. However, if the likelihood of someone typing 'A' is much higher than the chances they would type 'D', some savings can be had. Given the probabilities as listed in Table IV for each letter, it is easy to see that one bit will be saved every time the character 'A' is typed. This means that, on average, it only takes 1.75 bits

Table IV. Example Huffman code

Symbol	Likelihood	Bit pattern	Bits Allocated
A	50% = 0.50	0	1
B	25% = 0.25	10	2
C	12.5% = 0.125	110	3
D	12.5% = 0.125	111	3

to transmit each character, as computed below.

$$0.50 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75$$

Of course, if the probability of 'A' were to go up even more, the average bit rate goes down further. The lower limit to the compressibility of a character set is given by

$$H = - \sum_{i=1}^N p_i \cdot \log_2(p_i) \quad (5.8)$$

where H is called the entropy of the source or character set containing N elements, each with a given probability of occurrence, p_i . Further examples and details on the theory of Huffman codes can be found in [26].

The adaptive quantizer for the coder was designed to expand at a slightly more rapid rate and contract at a slower rate than was suggested by Jayant. It was found this had minimal effect on quality while greatly increasing the number of samples which get quantized to inner levels. With the majority of the samples being quantized to two inner levels, a Huffman code based on averaging the quantizer statistics from

several speakers was designed.

As can be seen from Table V, the Huffman code is integrated into the quantizer outputs, so the quantized values do not have to be re-coded with the Huffman code.

E. Packet loss issues

The occurrence of packet loss in speech transmission systems is a major problem. Discontinuity of the speech waveform can cause the decoded speech to be very difficult, if not impossible, to comprehend. It can also cause predictor mistracking, making it even harder to understand what the speaker is saying. Most current Internet speech coders simply play silence during the missing packet, which produces large discontinuities. Previous research has suggested that replacing the missing speech packet with a segment of previously transmitted speech or even possibly random noise results in a much less disruptive waveform [27]. Interleaving speech samples between packets, so interpolation is possible, has also been proposed [28].

We submit a new idea to this field: transmitting the lowest frequency subband band twice, once in the packet it is normally transmitted in, and once in the following packet. The bandwidth required for this second transmission is minimal, approximately 3 kbits/sec, compared to the possible improvement in speech continuity. If packet loss occurs, the missing packet can be partially reconstructed since the low frequency band information (the one with the highest speech content) will be contained in the following packet as well. This concept can obviously be extended to double transmitting any number of bands, depending on the available bandwidth or desired "protection."

Table V. Subband Huffman code

Quantizer Input	Transmitted Output		
	L=4	L=8	L=16
$-8\Delta(k)$			11111111
$-7\Delta(k)$			11111110
$-6\Delta(k)$			11111101
$-5\Delta(k)$			11111100
$-4\Delta(k)$		11111	111100
$-3\Delta(k)$		11110	111010
$-2\Delta(k)$	111	1100	1010
$-1\Delta(k)$	10	100	1100
$1\Delta(k)$	0	0	0
$2\Delta(k)$	110	101	100
$3\Delta(k)$		1101	1101
$4\Delta(k)$		1110	1011
$5\Delta(k)$			11100
$6\Delta(k)$			111011
$7\Delta(k)$			111101
$8\Delta(k)$			111110

CHAPTER VI

RESULTS

A. Overview

This chapter describes the results obtained for the developed coder through the four perspectives of bit rate, complexity, quality, and delay, showing how each one satisfies the desired objectives of a usable real-time system. We will conclude by drawing together results from all of the areas.

B. Bit Rate

The premier objective of this research was to develop a software only speech coder which operates at medium to low bit rates. This is achieved through the use of subband coding so a different number of bits can be allocated to each band, producing a system which successfully scales from under 12.5 kbits/s to over 18 kbits/s. The bit rate selection, the allocation of bits to each band as listed in Table II, totally dictates the resulting quality of the coded speech. This is shown in Table VI where the bit allocations used to achieve various average bit rates are listed, along with their SNR.¹ For example, the table entry '4 4 3 2 0' means that bands one and two are allocated 4 bits for quantization (a 16 level quantizer), band three is allocated 3 bits (8 levels), band four is allocated 2 bits (4 levels), and band five is allocated 0 bits (i.e., no information in that band will be transmitted). In order to achieve bit rates below 13 kbits/s, band five can not be allocated any bits. Although this degrades the crispness, it does not harm comprehension of the speech substantially, nor does it usually impede identification of the speaker.

¹The SNR is calculated as outlined in Appendix C.

Table VI. Selected subband bit allocations

Bits Allocated per band	Average Bit rate (kbits/s)	Signal to Noise Ratio (dB)	Bits Allocated per band	Average Bit rate (kbits/s)	Signal to Noise Ratio (dB)
4 4 4 2 2	20.6	17.2	4 4 4 2 0	17.2	14.9
4 4 3 2 2	19.6	16.8	4 4 3 2 0	16.2	14.6
4 4 2 2 2	17.5	15.8	4 4 2 2 0	14.1	14.0
4 3 2 2 2	17.0	13.3	4 3 2 2 0	13.6	12.2
4 2 2 2 2	16.0	7.8	4 2 2 2 0	12.5	7.5
3 2 2 2 2	14.9	7.5	3 2 2 2 0	11.5	7.2
2 2 2 2 2	13.8	6.5	2 2 2 2 0	10.4	6.3

On lower frequency bands, going from a sixteen to an eight level quantizer penalized the SNR somewhat, but did not substantially lower the bit rate due to the already low sampling rate. This is why most bit allocations listed in Table VI are either 2 or 4 bits per band.

C. Complexity

A major concern throughout the design process was keeping complexity as low as possible. It was initially suspected that a 32 tap subband FIR filters would provide a good tradeoff between computational complexity and reconstruction quality. Not only was this found to be true, but a 24 tap filter by Johnston [18] was found which provided comparable, and sometimes better, quality for lower complexity. The 24 tap filter sometimes exceeded the performance of the 32 tap due to it having a stop band attenuation of over 60 dB while the 32 tap had 51 dB. The fewer number

Table VII. Time spent filtering

	Unified filter		Split filters	
	(msec)	(% of total)	(msec)	(% of total)
Encode (32 tap)	8.0	84%	4.5	76%
Encode (24 tap)	7.2	80%	4.1	71%
Decode (32 tap)	13.7	89%	10.8	86%
Decode (24 tap)	9.6	85%	8.4	81%

of taps is reflected in the wider transition band, which was found to be inaudible.

When the program was first designed, a single filter procedure was used for both encoding and decoding, producing a "modular" structure which made modifications to the filter design easier. However, with the decimation included in the filter procedure (to save needless multiplication operations), the exact behavior of the procedure depended on whether the filter was being used to encode or decode speech. This difference meant the compiler could not optimize the filter procedure properly since it was too generic.

When it was apparent that the speech coder was using more CPU than anticipated, the filter was analyzed and this problem was discovered. To allow the optimizer to do a better job, the filter procedure was split into two filter functions, one for encoding and one for decoding. This resulted in two functions which could be specialized for their task, and the compiler optimized them much better, as shown in Table VII. In the table, "Unified filter" refers to the case where one filter procedure was used for both encoding and decoding and "Split filters" refers to the two specialized filter functions which could be optimized to a higher degree. The average amount of time

required to process a typical speech packet is presented in both milliseconds² and as a percentage of the whole encoding or decoding process. Since the filter, when used for encoding, operates on decimated speech with half the number of samples that the decoding process does, one would expect it to be twice as fast. Table VII shows that the "unified" method does not come close to achieving that expectation. This was the premise for trying the "split filters" which, as the table shows, resulted in the encoding process indeed being more than twice as fast as decoding. Even after this optimization though, it is quite obvious from the table that the vast majority of the system complexity remains in producing the subbands.

D. Quality

Perhaps the most important aspect of the coder, the quality of the developed system is quite scalable and is totally dependent on the desired bit rate. Due to lack of resources and proper testing conditions, obtaining meaningful MOS scores for this system was not feasible. Therefore, SNR numbers are provided in Table VI as a means of comparison between coders. As mentioned before, however, the SNR of a system does not always correspond directly to the perceived speech quality. The developed system exhibited several cases where this was apparent, most notably the '4 3 2 2 2' allocation, which sounds more pleasing than the '4 4 3 2 0' allocation despite the SNR calculations favoring the later.

A surprise came when it was discovered that using a 16 level quantizer in the upper frequency bands was often indistinguishable from using 8 levels. This is attributed to the aggressive nature of the quantizer expansion multipliers, described in Table III. This led to the choice of the '4 2 2 2 0' and '4 3 2 2 0' allocations as the

²As measured on an otherwise idle Sun Sparc 10.

best tradeoffs between quality and bit rate.

Double transmission of the lowest one or two subbands greatly increases the smoothness and understandability of the speech when compared to simply playing silence in place of missing packets. Fig. 12 displays the spectrogram of an original, uncoded speech sequence.³ Darker regions signify higher frequency content in that time interval. When a packet of speech is lost, as is shown in Fig. 13, the double transmission of the low frequency band enables the coder to still recover the most important frequency ranges (Fig. 14). Results vary depending on which portion of a speech segment is lost, but the SNR of the replaced packets is typically within 1.5 dB of the originally coded packet, and is 3 dB better than playing silence.

The pitch predictor was tried but removed because the decimated speech changed pitch periods too quickly for the predictor to take advantage of the redundancy. Due to the coarseness of the decimation, the pitch period can change every two to three samples, which simply isn't enough time for the pitch predictor to adapt. In attempt to counteract this problem, identification of the pitch period before decimation was performed. This resulted in better speech quality since pitch tracking was now correct, but required transmitting the pitch information over the channel, using approximately 6 kbits/s (6 bits per sample at 1000 samples per second, for the lowest subband). This is obviously not an acceptable solution in the context of desiring a low bit rate system.

The six tap, "zero" predictor was also disabled since preliminary results showed that decimation worked against its additional memory, providing little prediction or quality benefits.

³The utterance displayed is "Oak is strong and also gives shade."

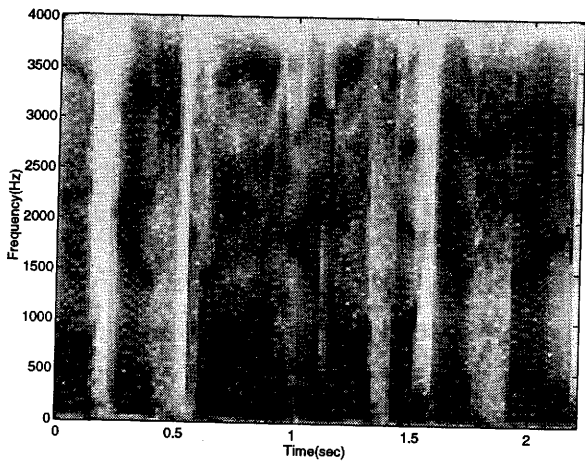


Fig. 12. Uncoded speech sample

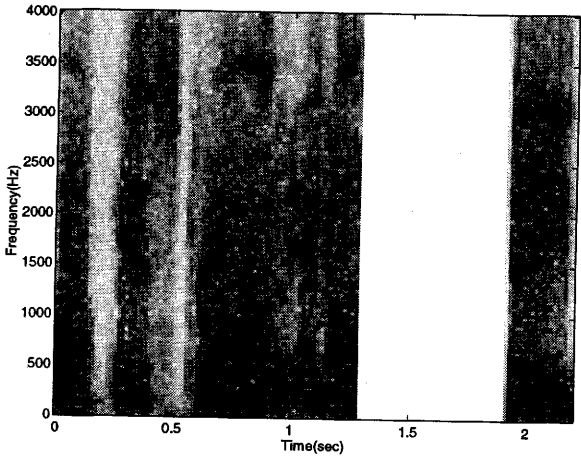


Fig. 13. Speech sample after suffering packet loss

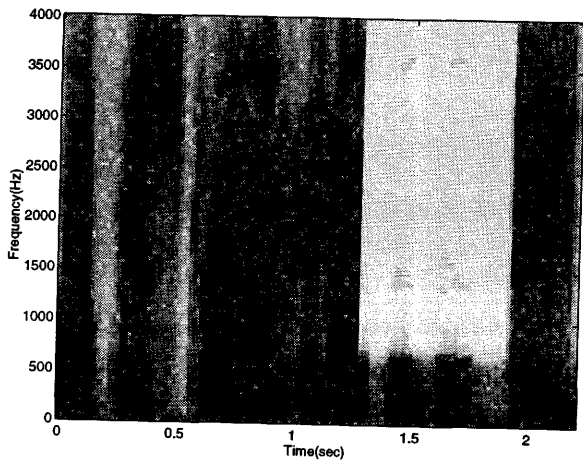


Fig. 14. Speech sample after packet loss with double transmitted low band

Table VIII. Total delay time

	Unified filter (msec)	Split filter (msec)
Encode (32 tap)	40	24
Encode (24 tap)	35	21
Decode (32 tap)	62	50
Decode (24 tap)	47	40

E. Delay

Although delay was not a design criteria for the system since packet delivery times over the Internet are not guaranteed, it is obviously desirable to have the the delay as low as possible. Experimentation summarized in Table VIII shows for typical packets of around 1 kbyte, overall system delay (encoding and decoding combined) is quite low—averaging well under 100 msec. This falls well under 250 msec, the amount of delay which is commonly considered distracting to normal conversations.

F. Summary

A real-time speech coder with several interesting aspects has been presented. The combination of subband coding, adaptive predictors, and Huffman codes was used to achieve a low bit rate with reasonable complexity. Subband coding allows the quality, and therefore the bit rate, to be scalable, based on the demands of the user or channel. All the components (bit rate, quality, and complexity) are intimately connected, driven by the users choice of bit rate.

The use of subband coding makes it reasonable to transmit a portion of the speech

twice, helping maintain continuity of the speech waveform even in the event of packet loss. Transmitting the lowest frequency band a second time adds under 3.5 kbits/s and transmitting the lowest two bands adds approximately 5 kbits/s (using a 16 level quantizer in the lowest band and a 4 level in the next lowest band). This helps counteract the problem of missing segments of speech currently plaguing Internet voice systems.

The work contained herein has led to the identification of many possible future research areas. The system that was developed can be used as a stepping stone toward the pursuit of any of the ideas listed below.

G. Areas for Future Study

Despite all efforts in this work, the resulting speech coder has a complexity level which is still quite high. If the host machine is being used by multiple users, as Sun Sparcs are designed to be, the system which was developed could tax the capabilities of the processor. As pointed out in the results section, the filtering operation constitutes the vast majority of the complexity of this scheme. A possible way to counteract this issue is to replace the floating point filter function with an integer implementation. This should operate with much lower processing cost since floating point operations (both conversions and multiplies) are typically 'expensive.' Following this route could result in a coder which could be run on lower end machines, including those without floating point processors. To cut complexity even further, time could be spent researching and designing the IIR filters needed to replace the FIR filters, while maintaining similar quality.

Although this coder is scalable, whenever a band is left out (allocated zero bits), the aliasing for its mirror remains uncanceled, added noise to the reconstructed

output. If filters with higher stop band attenuation were designed, the added noise due to uncancelled aliasing would be minimal. Since FIR filters already take a large amount of processing time, however, increasing their length even further is not a realistic option. IIR filters appear to be the natural solution for this problem.

Turning attention to the predictors, several more research areas have been identified. As mentioned in the Quality section of this chapter, the pitch predictor was not found to be useful due to the quick changes of the pitch period of the decimated speech. If an algorithm of reasonable complexity could be developed to track pitch changes in highly decimated speech, the quality of the reconstructed output would be much higher.

Another possible speech coding method worth investigating is the use of line spectral pairs (LSP) or frequencies (LSF) to transmit speech parameters to the receiver [29]. The additional low bit rate stream could be used to considerably enhance the reconstructed speech.

Lastly, an area which has received attention in the video conferencing market (where bandwidth is not at such a premium) is wideband audio. Wideband refers to coding 7 kHz of speech rather than the typical "telephone bandwidth" of 3.5 to 4 kHz. Wideband audio sounds much clearer and is much more enjoyable to listen to for long periods of time. The scalability of this coder makes it ideal for such an application, enabling users or routers to prune off the upper band if network congestion or a low bit rate channel is encountered.

REFERENCES

- [1] CCITT Recommendation G.726, 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM), December 1990.
- [2] N. Benvenuto, G. Bertocci, W. R. Daumer, and D. K. Sparrell, "The 32-kb/s ADPCM coding standard," *AT&T Technical Journal*, vol. 65, pp. 12-21, September/October 1986.
- [3] W. R. Daumer, X. Maitre, P. Mermelstein, and I. Tokisawa, "Overview of the ADPCM coding algorithm," in *Session 23, IEEE Global Telecommun. Conf.*, (Atlanta, GA), November 1994.
- [4] J. D. Gibson, "Adaptive prediction for speech encoding," *IEEE ASSP Magazine*, vol. 1, pp. 12-25, July 1984.
- [5] J. D. Gibson, "Adaptive prediction in speech differential encoding systems," *Proceedings of the IEEE*, vol. 68, pp. 488-525, April 1980.
- [6] M. Bonnet, O. Macchi, and M. Jaidane-Saidane, "Theoretical analysis of the ADPCM CCITT algorithm," *IEEE Trans. on Communications*, vol. 38, pp. 847-858, June 1990.
- [7] N. S. Jayant, "Adaptive quantization with a one-word memory," *Bell System Tech. J.*, vol. 52, pp. 1119-1144, September 1973.
- [8] D. J. Goodman and R. M. Wilkinson, "A robust adaptive quantizer," *IEEE Trans. on Communications*, vol. COMM-23, pp. 1362-1365, November 1975.

- [9] A. Croisier, D. Esteban, and G. Galand, "Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques," in *Int. Conf. on Inform. Sci. Systems*, (Patras, Greece), 1976.
- [10] R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital coding of speech in sub-bands," *Bell System Tech. J.*, vol. 55, pp. 1069-1085, October 1976.
- [11] T. P. Barnwell, "Subband coder design incorporating recursive quadrature filter and optimum adpcm coders," *IEEE Trans. on Acoust., Speech, Sig. Proc.*, vol. ASSP-30, pp. 751-765, October 1982.
- [12] O. Alkin and H. Caglar, "Design of efficient M -band coders with linear-phase and perfect-reconstruction properties," *IEEE Trans. on Signal Processing*, vol. 43, pp. 1584-1590, July 1995.
- [13] R. V. Cox, S. L. Gray, S. R. Quackenbush, N. Seshadri, and N. S. Jayant, "New directions in subband coding," *IEEE Journal on selected areas in communications*, vol. 6, pp. 391-409, February 1988.
- [14] D. Esteban and C. Galand, "Application of quadrature mirror filterbanks to splitband voice coding schemes," in *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing*, (Hartford, CT), pp. 191-195, May 1977.
- [15] J. H. Rothweiler, "Polyphase quadrature filters—a new sub-band coding technique," in *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Proc.*, (Boston, MA), pp. 1280-1283, April 1983.
- [16] R. V. Cox, "The design of uniformly and nonuniformly spaced pseudoquadrature mirror filters," *IEEE Trans. on Acoust., Speech, Sig. Proc.*, vol. ASSP-34, pp. 1090-1096, October 1986.

- [17] H. J. Nussbaumer, "Pseudo QMF filterbank," *IBM Tech. Disclosure Bull.*, vol. 24, pp. 3081-3087, November 1981.
- [18] J. D. Johnston, "A filter family designed for use in quadrature mirror filters banks," in *IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, (Denver, CO), pp. 291-294, April 1980.
- [19] F. K. Soong, R. V. Cox, and N. S. Jayant, "Sub-band coding of speech using backward adaptive prediction and bit allocation," *IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, vol. 3, pp. 1672-1675, April 1985.
- [20] A. Alkhairy, "An efficient method for IIR filter design," in *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing*, (Austin, TX), April 1994.
- [21] J. D. Gibson, "Sequentially adaptive backward prediction in ADPCM speech coders," *IEEE Transactions on Communications*, vol. COMM-25, pp. 145-150, January 1978.
- [22] S. McClellan, "Pitch prediction." Dept. of Elec. Eng., Texas A&M Univ., unpublished, September 1994.
- [23] R. Pettigrew and V. Cuperman, "Backward pitch prediction for low-delay speech coding," in *Session 34, IEEE Global Telecommun. Conf.*, (Dallas, TX), November 1989.
- [24] H. C. Woo and J. D. Gibson, "Low delay tree coding of speech at 8 kbit/s," *IEEE Trans. on Acoust., Speech, Sig. Proc.*, vol. 2, pp. 361-370, July 1994.
- [25] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1984.

- [26] K. Sayood, *Introduction to Data Compression*. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.
- [27] D. J. Goodman, G. B. Lockhart, O. J. Wasem, and W.-C. Wong, "Waveform substitution techniques for recovering missing speech segments in packet voice communications," *IEEE Trans. on Acoust., Speech, Sig. Proc.*, vol. ASSP-34, pp. 1440-1447, December 1986.
- [28] N. S. Jayant and S. W. Christensen, "Effects of packet losses in waveform coded speech and improvements due to an odd-even sample interpolation procedure," *IEEE Transactions on Comm.*, vol. COMM-29, pp. 101-109, February 1981.
- [29] A. M. Kondoz, *Digital Speech*. New York City, New York: John Wiley & Sons, Inc., 1994.
- [30] J. W. Woods, *Subband Image Coding*. Norwell, MA: Kluwer Academic Publishers, 1991.

APPENDIX A

PROOF OF QMF ALIAS CANCELLATION

Given a low-pass filter, $H_1(z)$, and high-pass filter, $H_2(z)$, it can be shown that $G_1(z)$ and $G_2(z)$ can be designed with certain characteristics to perform aliasing cancellation [30]. As can be seen from Figure 15, the output of the system is given by $\hat{x}(k)$, can be compactly written in the transform domain as

$$\begin{aligned}\hat{X}(z) &= G_1(z) \cdot \frac{1}{2} [H_1(z)X(z) + H_1(-z)X(-z)] \\ &\quad + G_2(z) \cdot \frac{1}{2} [H_2(z)X(z) + H_2(-z)X(-z)]\end{aligned}\quad (\text{A.1})$$

where $H_n(-z)X(-z)$ are the aliasing terms caused by the decimation. The aliasing terms can be isolated, giving

$$\begin{aligned}\hat{X}(z) &= \frac{1}{2}X(z)[H_1(z)G_1(z) + H_2(z)G_2(z)] \\ &\quad + \frac{1}{2}X(-z)[H_1(-z)G_1(z) + H_2(-z)G_2(z)]\end{aligned}\quad (\text{A.2})$$

where the first term is the desired signal and the second term is the undesired re-

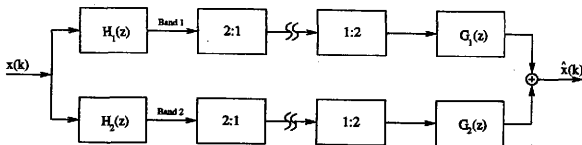


Fig. 15. Two-band QMF without coding

quency aliasing. This aliasing can be made to equal zero by carefully choosing the filter designs such that

$$H_1(-z)G_1(z) + H_2(-z)G_2(z) = 0 \quad (\text{A.3})$$

First, the low and high pass synthesis filters can be chosen to be mirrors of each other,

$$H_2(e^{j\omega}) = H_1(e^{j(\omega+\pi)})$$

or equivalently,

$$H_2(z) = H_1(-z) \quad (\text{A.4})$$

Since $G_1(z)$ must be a lowpass filter, an obvious candidate is

$$G_1(z) = H_1(z) \quad (\text{A.5})$$

When equations A.4 and A.5 are substituted into A.3, it is apparent that

$$G_2(z) = -H_2(-z) \quad (\text{A.6})$$

which when applied to equation A.2, gives

$$\begin{aligned} \hat{X}(z) &= \frac{1}{2}X(z)[H_1(z)G_1(z) + H_2(z)G_2(z)] \\ &= \frac{1}{2}X(z)[H_1^2(z) - H_1^2(-z)] \end{aligned} \quad (\text{A.7})$$

leaving no aliasing in the reconstructed output signal.

APPENDIX B

QMF FILTER COEFFICIENTS

The QMF coefficients used in this system were produced and verified by Johnston [18] to minimize stop band energy and ripple for a given set of constraints, such as filter order and width of the transition band.

Listed below are Johnston's coefficients which were used in the implemented system. They represent a compromise between quality and computational complexity. Although only the first half of the coefficients for the filter, called 24C, are needed to describe it, all coefficients will be listed to demonstrate the symmetry of the filters.

Continuing with the notation from Figure 5 and equation 4.1:

$$h_1(k) = \{$$

0.0003833096, -0.0013929110, -0.0013738610, 0.0064858790,
 0.0014464610, -0.0190199300, 0.0038915220, 0.0442397600,
 -0.0256153300, -0.0982978300, 0.1160355000, 0.4731289000,
 0.4731289000, 0.1160355000, -0.0982978300, -0.0256153300,
 0.0442397600, 0.0038915220, -0.0190199300, 0.0014464610,
 0.0064858790, -0.0013738610, -0.0013929110, 0.0003833096

};

$$h_2(k) = \{$$

0.0003833096, 0.0013929110, -0.0013738610, -0.0064858790,
 0.0014464610, 0.0190199300, 0.0038915220, -0.0442397600,
 -0.0256153300, 0.0982978300, 0.1160355000, -0.4731289000,
 0.4731289000, -0.1160355000, -0.0982978300, 0.0256153300,
 0.0442397600, -0.0038915220, -0.0190199300, -0.0014464610,
 0.0064858790, 0.0013738610, -0.0013929110, -0.0003833096

};

APPENDIX C

SNR CALCULATIONS

The signal to noise ratio (SNR) calculations used in this research are computed as shown below, with $x(k)$ representing the original, uncoded speech samples and $\hat{x}(k)$ representing the final reconstructed output sequence of N samples.

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum_{k=1}^N x^2(k)}{\sum_{k=1}^N (x(k) - \hat{x}(k))^2} \right) \quad (\text{C.1})$$

VITA

Marc Randolph was born and raised in Tulsa, Oklahoma, except for a brief six year stay in Ras Tanura, Saudi Arabia. He graduated Texas A&M University (College Station, TX) in 1994 with a Bachelor of Science degree in Electrical Engineering after participating in the cooperative education program with Electrospace Systems in 1992. Following an internship at Compaq Computers (Houston, TX) in 1994, he returned to Texas A&M to pursue a Masters of Science degree in Electrical Engineering.

Mr. Randolph now works for ADC Telecommunications in Richardson, TX, and can be reached at marc.randolph@pobox.com or 17440 E 84th St N, Owasso, OK, 74055.