Compensation for Missing Voice Coding Frames in Packet Transmission Systems

By

Mary Antoinette Kohler

Bachelor of Science
Clemson University
Clemson, South Carolina
1986

Master of Science
Johns Hopkins University
Baltimore, Maryland
1993

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2000

COMPENSATION FOR MISSING VOICE CODING FRAMES IN

PACKET TRANSMISSION SYSTEMS

Thesis Approved:

Rao Yarlagadda
Thesis Adviser

[illegible]

[illegible]

[illegible]

Wayne B. Powell
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Rao Yarlagadda, for his excellent advice, encouragement, friendship, and support. I am grateful for his technical expertise, which he used to solve many of my problems, and his kindness, which made our time in Stillwater memorable. I am also obliged to the members of my committee, Dr. Keith Teague, Dr. Scott Shepard, and Dr. Art Pentz for their time and for their willingness to advise me when I asked.

I especially wish to express thanks to my husband, Bill Kohler. He encouraged, supported, loved, prayed for, and sacrificed for me, and he helped me solve problems by putting them into a physicist's perspective. I relied on the encouragement and prayers of him, my parents, my sister, my friends, and my family.

This work was partially funded by the Army Research Office, contract #DAAH04-95-1-0463.

TABLE OF CONTENTS

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

NOMENCLATURE

ACB	Adaptive Code Book
ADPCM	Adaptive Differential Pulse Code Modulation
ATM	Asynchronous Transfer Mode
CCITT	International Telegraph and Telephone Consultative Committee
CELP	Code Excited Linear Prediction
dB	Decibels
EDAC	Error Detection And Correction
F_S	Sampling Frequency
FS	Federal Standard
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HDI	Hot Deck Imputation
INPT	Inverse Naturalness Preserving Transform
IPv6	Internet Protocol Version Six
ITU	International Telecommunication Union
LAR	Log Area Ratio
LP	Linear Prediction
LSF	Line Spectral Frequency
LSP	Line Spectral Parameters
LTP	Long Term Prediction

MCP	Markov Chain Prediction
MCRP	Markov Chain Random Process
MELP	Mixed Excitation Linear Prediction
MHz	Mega-Hertz
MSB	Most Significant Bit
μlaw	mu-law
μs	micro-second
ms	millisecond
MTD	Mixture Transition Distribution
NPT	Naturalness Preserving Transform
PCM	Pulse Coded Modulation
RPE	Regular Pulse Excitation
RPE-LTP	Regular Pulse Excitation-Long Term Prediction
RPT	Frame Repetition
SCB	Stochastic Code Book
SNR	Signal to Noise Ratio
TCP/IP	Transmission Control Protocol/Internet Protocol
vocoder	Voice Coding Algorithm
WAN	Wide Area Network

List of Symbols

α	NPT scale parameter
b	Number of Bits in the MTD Lookup Parameter
C_g	MTD Cross-table
d	Number of Vocoder Parameters in a State
φ	MTD transition weights
$H(z)$	Frequency response of filter
$\mathbf{H}_n$	nth order Hadamard matrix
$\mathbf{H}'_n$	nth order normalized Hadamard matrix
$\mathbf{I}_n$	$2^n \times 2^n$ identity matrix
L_Q	Average packet server queue length
$\log(L)$	Log Likelihood
m	Markov Chain Order
λ	Packet arrival rate
μ	Packet service rate
N	Number of mth order Markov Chain transitions
Π_n	$2^n \times 2^n$ NPT transformation matrix
$\mathbf{P}$	Generic Markov Chain Transition Matrix
$\mathbf{P}_{init}$	Initial NPT information

$\mathbf{P}_n$	$2^n \times 2^n$ data matrix
$\mathbf{Q}$	MTD transition matrix
ρ	Packet server system utilization parameter
R	Number of Bits
$\Psi_n(\alpha)$	$2^n \times 2^n$ NPT stochastic operator matrix
S	Number of States in a Markov Chain
T	Average packet delay
u	Theil's Association Measure

CHAPTER ONE

INTRODUCTION

"Mr. Watson, come here, I want you." Since Alexander Graham Bell transmitted that first message in 1876, researchers have been seeking ever more efficient methods for transmitting voice long distances. One of the major innovations along the way was the conversion of voice from an analog signal to a digital one. Periodic regenerators in the transmission path could now re-amplify degraded binary signals, thereby preserving their fidelity. Transmitting voice digitally not only improved its quality, but also facilitated multiplexing, signaling, switching, operability in harsh channel conditions, and services that were impossible with the analog system. And it significantly increased the prospects for further reductions in bandwidth.

In the 1930's, Homer Dudley realized some of those potential gains by developing the world's first voice coder, or vocoder, a parametric algorithm for modeling the physiology of human voice production [23]. Vocoders achieve low transmission rates by sending, not the digitized sound waves themselves, but parameters of the voice model. Vocoders sometimes sacrifice quality, however, to achieve the low rates. In harsh communications environments, where higher bandwidth may not be available, this sacrifice is considered acceptable.

Other factors have influenced the evolution of long distance voice communication. Early systems, the telephone included, delivered voice across wireline or wireless channels that were designed solely for transmitting voice. These circuit-switched networks introduced very little communication delay, and provided channels dedicated to each talker for the duration of the transmission. Researchers knew the transmission characteristics, such as signal-to-noise ratio and error probabilities, of these channels from extensive studies and developed vocoders to operate within these environments.

In 1971, the Advanced Research Projects Agency (ARPA) created what is now known as the Internet to connect heterogeneous machines at universities and military installations. It was designed to transmit bursty computer data that had no real-time delivery requirements. This first long-distance, packet-switched network made more efficient use of network resources for data transmission than the venerable circuit-switched network of the telephone system. Transmission channels were not dedicated to a single user, but were shared among all users. The network transmitter partitioned files into small segments and placed each segment into separate packets. Packet headers contained all the information required to direct the packets through the network to the correct destination and, once there, to reassemble the file. Packet headers also provided a way for the receiver to determine when packets were corrupted. Re-transmitting packets that were lost or received with errors achieved error-free transmission with increased delay. This successful experiment that created the Internet fueled the development of several packet-switched protocols for both wireline and wireless long-distance data networks. Frame relay and asynchronous transfer mode (ATM) are both examples of higher speed packet-switched protocols that rely on links with low errors. Digital cellular communication networks also use packets for integrated voice and data communications.

The massive connectivity available through the Internet motivated the development of diverse applications, including voice transmission. The International Telecommunication Union (ITU) began investigating network voice transmission in the middle 1980's. Their goal was to provide a uniform basis for speech packetization. In 1990, the ITU study group completed G.764 ([38]), the packet voice protocol. In 1995, a packetization guide was released as an appendix to the protocol [39]. G.764 defines a protocol for speech packetization in very high quality circuit applications. The protocol defines formats and procedures for the transport of digitized voice and channel signaling over a wideband packet network. The protocol is designed to exploit the bursty nature of packet-switched networks by transmitting speech in spurts. G.764 also has procedures for handling congestion in the network gracefully, and the G.764 appendix provides suggestions for recovering when voice packets are discarded due to errors, delays, or excessive congestion. Using this protocol, however, will not guarantee high quality transmission of long distance voice. The standard does not address performance issues, speech coding methods, services, implementation techniques, equipment, establishment of the (permanent virtual) circuit, or recovery of missing packet information. The appendix provides guidelines for some of these areas, but many of them are left for further research. Regardless of these challenges, G.764 meets the goal of its designers—providing a uniform basis for speech packetization.

Modern computers come equipped with the hardware necessary to communicate with the Internet and to record and play voice. Some consumers are now choosing voice transmission over the Internet (and other high-speed networks) as a lower-cost alternative to the long distance telephone network. The U.S. military is considering the use of existing packet-switched networks for voice communications rather than building specialized networks for their growing communications requirements. Packet-switched voice transmission will not achieve widespread acceptance, though, until high quality voice can be transmitted across low quality networks. Low quality networks are not robust and can corrupt, delay, or remove packets. Any of these effects renders packets unusable, and consequently the data is unavailable when the voice signal is regenerated. Improper handling of missing speech packets subsequently deteriorates the voice signal. Clearly, a remedy for this problem is desired ([74]).

This thesis describes two methods for coping with missing speech packets. The first method, Naturalness Preserving Transform Reconstruction, transforms the data prior to transmission and recreates the missing samples at the receiver via properties of the transform. The second method, Markov Chain Prediction, forecasts the missing speech parameters based on information in previous packets. These two methods are compared with conventional frame repetition. Performance data for Internet packet switching is more readily available than for other, newer protocols; therefore, Internet statistics influence the discussions. All research, however, has assumed a generic packet-switched network with no special capabilities.

Chapter Two contains brief descriptions of the waveform and parametric voice coding algorithms used in this work. Chapter Three characterizes the challenges of transmitting voice across packet-switched networks, and Chapter Four describes the network simulator used in this study. Chapter Five provides an overview of current techniques from literature. Proposed solutions to the problem are detailed in Chapter Six, which describes Naturalness Preserving Transform Reconstruction, and Chapter Seven, which describes Markov Chain Prediction. Chapter Eight discusses future research for these solutions; Chapter Nine lists other applications of this research; and Chapter Ten contains conclusions.

Chapter Two

Voice Coding

Internet telephony first requires conversion of a talker's voice into digital samples that can be packetized and transmitted. The initial conversion from the analog signal emanating from the talker's voice to a digital one has a well-defined solution. However, this initial digital signal is usually compressed to require fewer network resources. The experiments in this thesis used several different voice compression techniques that are described in the following sections.

Everyone who uses digital voice communications desires high quality and is usually only fully satisfied with sampling rates of 8,000 or 10,000 samples/second and 16 bits/sample. But this demands a substantial transmission rate (128,000-160,000 bits/second). Voice encoding can reduce this, often prohibitive, bandwidth requirement, however. There are three categories of voice-coding algorithms (vocoders): waveform coders, parametric coders, and hybrid coders.

Waveform coders capture the shape of the sound wave and are designed to transmit sounds from any source including music, modem tones, or voice. They generally produce higher quality re-synthesis than parametric coders but demand considerably higher bit rates (16 - 64 kilobits/second). The μlaw coder and adaptive differential pulse coded modulation (ADPCM) coder are examples of common waveform coders, both of which are used by the telephone system.

Parametric coders, conversely, entail a mathematical model of the source of the sound wave, the parameters of which are encoded. These coders are only designed to encode sound from the human vocal production system. They divide the speech signal into frames of 20-30 milliseconds (ms) and analyze each frame of speech to compute the parameters of the model. A synthesizer on the receiving end reconstructs the speech signal using these transmitted parameters. Parametric coders can operate at very low rates (600 – 8,000 bits/second), but degrade rapidly when transmitting sounds other than human voice. At very low bit rates, parametric coders cannot currently retain individual talker characteristics, and the identity of the speaker is difficult, or impossible, to recognize. Multi-band excitation (MBE) and mixed excitation linear prediction (MELP) are two popular examples of parametric coding.

Hybrid coders combine the principles of both waveform and parametric coding– they often use parametric coding for filter parameters and waveform coding for the excitation signal. Code excited linear prediction (CELP), which has many variations, is a commonly encountered hybrid coder. All three–parametric, waveform, and hybrid coders were used for this project.

The selected parametric and hybrid coders use linear prediction (LP) to model speech. Linear prediction is based on mean squared error and exploits the relative short-time stationarity of human voice signals (10-20 ms) to approximate the present speech sample as a linear combination of past speech samples,

$$\tilde{s}(i) = \sum_{k=1}^{p} a_k s(i-k)\,. \tag{2.1}$$

Vocoder designers choose the prediction order, p, large enough to represent the spectrum accurately, but small enough so that the quantizer requires few bits to represent the p prediction coefficients. The prediction order generally varies from 8 to 20.

The prediction residual is the error between the signal created by (2.1) and the original signal,

$$e(i) = s(i) - \tilde{s}(i)\,. \tag{2.2}$$

The quantized LP residual signal is used as the excitation for the synthesis filter. The time-varying all-pole synthesis filter shown in (2.3) represents the composite spectrum effects of sound radiation at the lips and nose, vocal tract shape, and glottal excitation.

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{2.3}$$

In its simplest form, the residual excitation for this filter is chosen from two sources: a periodic signal (e.g., an impulse train) for voiced speech, and random noise (usually Gaussian) for unvoiced speech. The analyzer determines the p prediction coefficients (a_k), gain, voiced/unvoiced switch and, for voiced speech, the fundamental frequency (loosely referred to as pitch) of the impulse train that are transmitted to the receiver.

Prior to quantization, the prediction coefficients are often transformed to other domains with better quantization or interpolation properties, such as line spectral frequencies. Line spectral frequencies (LSFs) or line spectral pairs (LSPs) are the frequencies of the zeros of two $(p+1)$-order polynomials that lie on the unit circle. The two polynomials are calculated by decomposing the z-domain representation of the inverse filter. One polynomial corresponds to a lossless model of the vocal tract with glottis closed and the other polynomial corresponds to the lossless open glottis model.

For those readers familiar with speech coding, a brief description of the voice coders used in this analysis is given below. References to more thorough descriptions are provided in each section. Six coders were used: three waveform coders, two hybrid coders, and a parametric coder.

Pulse Coded Modulation (PCM)

Pulse coded modulation, developed at Bell Laboratories, was the first digital representation of speech. It is a high quality, low complexity voice coding method, but because no compression is performed, it is also the least efficient. Samples are converted every 125 μs (8000 samples/second), uniformly quantized to 2^R amplitude levels, and stored digitally. The number of bits, R, used to represent each sample is usually 16, and so 128 kilobits/second are required for storage. All digital voice coders employ PCM as the first step in the encoding process. More information about PCM can be found in [19].

μlaw

The sensitivity of the human ear is logarithmic in nature, a feature which the uniform quantization in PCM does not make use. μlaw coders exploit this logarithmic sensitivity by using more finely spaced quantization levels for small signal amplitudes and coarser spaced quantization levels for large signal amplitudes. Currently, North American telecommunication companies use a μlaw coder for digital transmission with the non-linear quantizer

$$|y| = X_{\max} \frac{\log(1+(\mu|x|/X_{\max}))}{\log(1+\mu)} \operatorname{sgn}(x) \tag{2.4}$$

When the input, x, is near its mean, the quantization of the output, y, has finer resolution. As x approaches its maximum or minimum values, the quantization grows coarser. Coding with μlaw requires only 8 bits/sample to represent speech with the same fidelity as 16-bit PCM. Phone companies in the U.S. use 8 bits/sample (64 kilobits/second) and μ=256. Figure 2.1 compares the quantization used by PCM and μlaw. More information about μlaw coding (also called companding) can be found in [70].

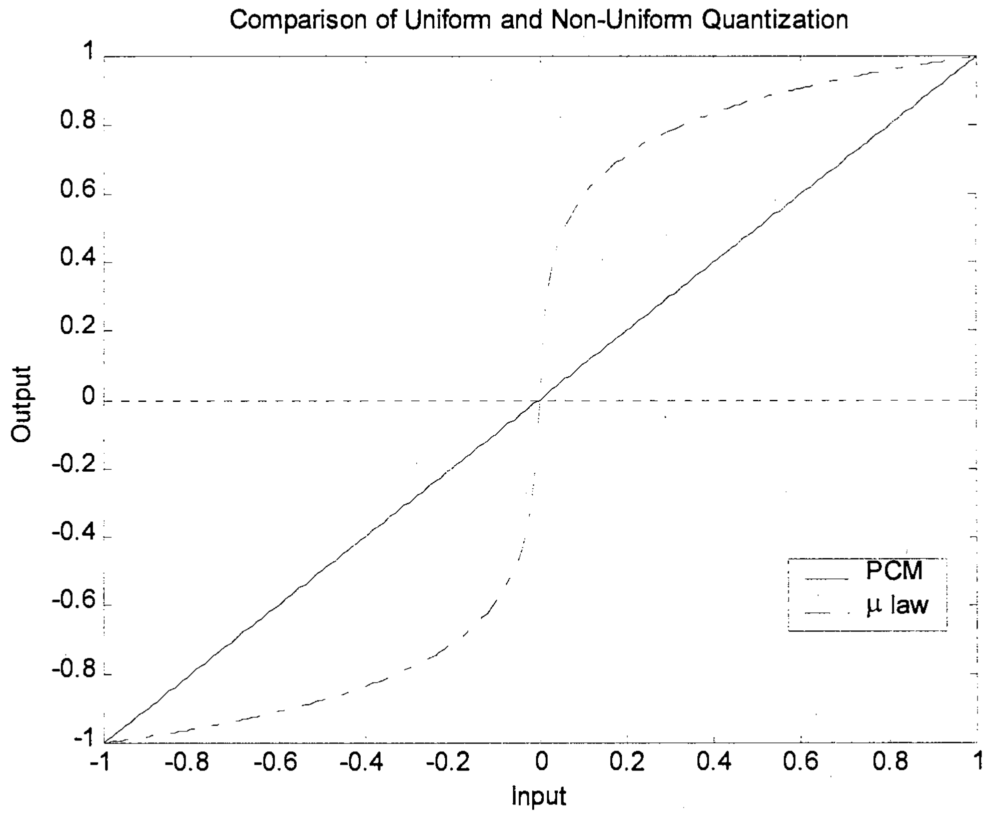


Figure 2.1 Comparison of Uniform and Non-Uniform Quantization

Differential Pulse Coded Modulation (DPCM)

Differential pulse coded modulation (DPCM) exploits the correlation that exists between successive speech samples. It linearly quantizes the differences between successive samples rather than quantizing the individual samples. Because differences between sequential samples are generally smaller than sampled speech amplitudes, fewer bits are required. Most DPCM coders refine the general DPCM approach by using a fixed, low-order linear predictor. With this technique, the difference between the predicted sample and the input sample is uniformly quantized. The same predictor is available at the receiver to calculate the speech samples. Figure 2.2 shows a block diagram of a DPCM encoder, and Figure 2.3 shows a block diagram of a DPCM decoder.

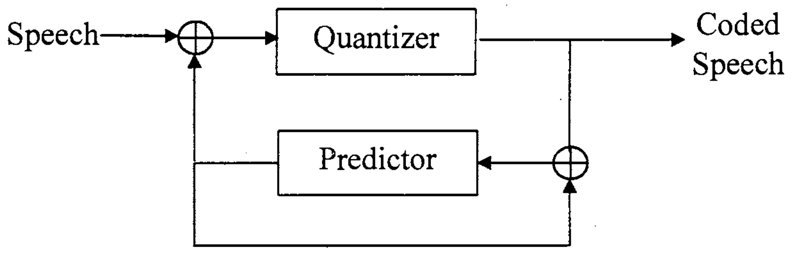


Figure 2.2 DPCM Encoder

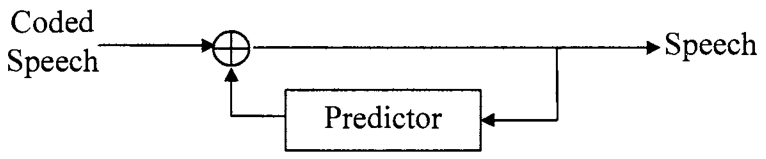


Figure 2.3 DPCM Decoder

In the experiments of this thesis, 7 bits were used to linearly quantize the differences from first-order prediction (56 kilobits/second). More information on differential quantization can be found in [76].

Code Excited Linear Prediction (CELP)

This vocoder quantizes the excitation signal for the linear prediction filter using two codebooks, one adaptive, and one stationary. The codebook indices are determined using analysis–by–synthesis search procedures with a perceptually weighted distortion measure. The Federal Standard 1016 (FS1016) CELP analyzer operates on 30 ms frames of speech sampled at 8000 samples per second (16 bits per sample, 240 samples per frame). The short–term spectrum of the speech signal is modeled using a 10th order LP filter. The residual signal is calculated on four subframes. Long term periodicity, or pitch, is modeled using an adaptive codebook (ACB) vector quantizer, which contains the most recent excitation samples. The residual from exciting the LP filter with the scaled adaptive codebook vector is modeled using a stochastic codebook (SCB) vector quantizer. The search procedures for both codebooks choose the optimal scaled excitation vectors by minimizing a time-varying perceptually weighted distortion measure. The optimal excitation is calculated by adding the scaled adaptive codebook vector to the scaled stochastic codebook vector. The adaptive codebook is updated each subframe with the optimal excitation vector from the previous frame. The ternary valued (0, ±1) stochastic codebook is fixed.

The FS1016 CELP analyzer produces 144 bits from each 30-ms analysis frame. These bits represent the ten line spectral frequencies (LSFs) representing the LP filter coefficients, four adaptive codebook vector indices and gains, four stochastic codebook indices and gains, and forward error correction and synchronization bits. The CELP synthesizer combines the scaled vectors from the adaptive and stochastic codebooks to form the excitation for the LP filter. The output from the LP filter is postfiltered to reduce the effects of quantization noise. Campbell, Tremain, and Welch provide more detailed information on the CELP algorithm in [10]. Figure 2.4 provides a block diagram of the CELP synthesizer.

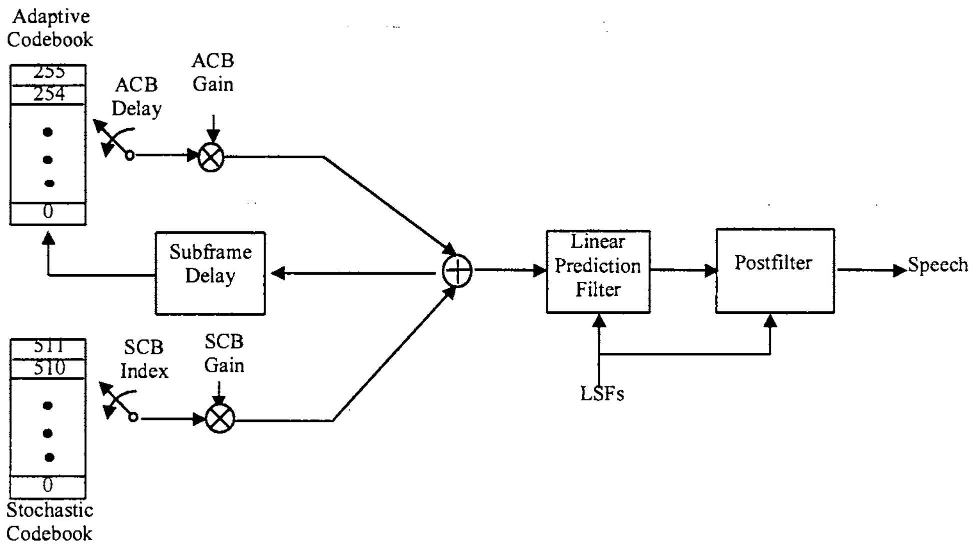


Figure 2.4 CELP Synthesizer

Global System for Mobile Communication (GSM 6.10)

The GSM full rate speech coder uses a regular pulse excitation-long term prediction (RPE-LTP) linear predictive algorithm. It is based on linear prediction of the signal and subframe analysis of the residual. The GSM 6.10 coder operates on 20 ms frames of speech sampled at 8,000 samples per second (13 bits per sample, 160 samples per frame). The short-term spectrum of the signal is modeled using an 8th order LP filter. The residual signal is calculated on four subframes. The long term prediction (LTP) parameters (lag and gain) are estimated in an open-loop method using the residual from the current subframe and three previous subframes. The LTP parameters and the short-term residual are combined to select the pulses for the excitation signal. The normalized quantized excitation signal is used to estimate the residual for the next frame.

The GSM 6.10 coder produces 260 bits from the 20 ms input frame. These bits represent eight log area ratios (LARs) representing LP filter coefficients, four LTP lags, four LTP gains, and four sets of RPE parameters (position of selected subsequence, maximum amplitude, and pulses). The GSM synthesizer interpolates the decoded, de-normalized RPE parameters and adds the LTP contribution to create the excitation for the LP synthesis filter. More information on the GSM coder can be found in [25]. Figure 2.5 provides a block diagram of the GSM synthesizer.

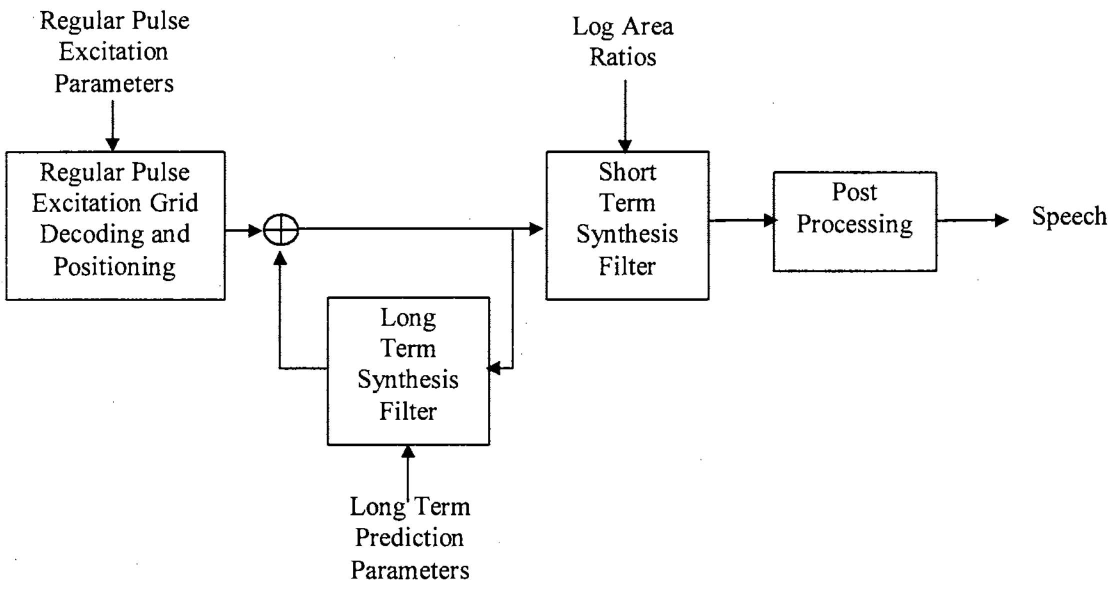

Regular Pulse Excitation Parameters
Regular Pulse Excitation Grid Decoding and Positioning
Log Area Ratios
Short Term Synthesis Filter
Post Processing
Speech
Long Term Synthesis Filter
Long Term Prediction Parameters

Figure 2.5 GSM Synthesizer (taken from [25])

Mixed Excitation Linear Prediction (MELP)

MELP is based on the traditional LP vocoder with either a periodic impulse train (for voiced speech) or white noise (for unvoiced speech) exciting an all-pole filter. MELP has four additional features over classic linear prediction: mixed pulse and noise excitation, periodic or aperiodic pulses, adaptive spectral enhancement, and pulse dispersion filtering. The MIL-STD-3005 MELP algorithm analyzer operates on 22.5 ms frames of speech sampled at 8,000 samples per second (16 bits per sample, 180 samples per frame). The short-term spectrum of the signal is modeled using a 10th order LP filter. The pitch is calculated in an open loop method using the LP residual. The magnitude of the residual spectrum is measured, and these Fourier magnitudes are used to enhance the periodic excitation. A voicing decision is produced over each of five frequency bands to determine the excitation mixture.

The MELP analyzer produces 54 bits from the 22.5 ms input frame. These bits represent the ten line spectral frequencies (LSFs) representing the LP filter coefficients, bandpass voicing, pitch, Fourier magnitudes, aperiodic flag, forward error correction, and synchronization. The MELP synthesizer uses the pitch and Fourier magnitudes to produce the voiced component and a random noise generator to produce the unvoiced component. The bandpass voicing bits determine how to mix these two signals and form the LP excitation. More information on the MELP algorithm can be found in [64] and [65]. Figure 2.6 provides a block diagram of the MELP synthesizer.

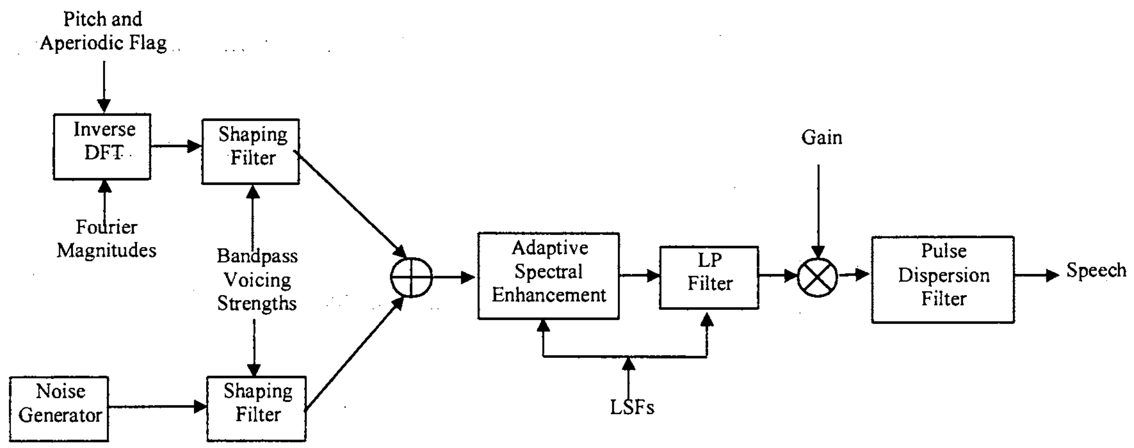


Figure 2.6 MELP Synthesizer

Frame Loss and Quality

Frame-repetition, a simple and convenient technique, is used as the baseline for evaluating other missing frame compensation approaches. It is easy to implement for any voice-coding algorithm–any missing information is simply replaced using the most recently received data. This simplicity is why it is widely used as a compensation method ([29], [35]). Knowing the limits at which frame-repetition will still render intelligible speech gives a baseline against which to judge other frame compensation schemes. These limits were determined for both CELP and MELP using several speech files with varying backgrounds, microphones, and talkers.

Frame-repetition with MELP showed that when more than 15-20% of the frames are lost, speech quality is noticeably degraded. When more than 10% of the frames are missing, with missing frames occurring in pairs, frame repetition degrades noticeably. This is to be expected since speech is slowly time varying. The CELP algorithm is not as robust—speech quality noticeably degrades with a missing frame rate of 2.5%. If two contiguous frames are lost, quality was noticeably degraded when more than 2% of the frames are missing. It therefore seems likely that improvements can be made in this area.

Chapter Three

Transmitting Voice on Packet-Switched Systems

The wireline telephone system is the dominant voice communication system of the world. Because of our extensive experience with it, we compare all our long-distance communication against this standard. It was designed and optimized to facilitate real-time voice conversations and generally provides high quality. When a user places a call, the telephone system identifies a two-wire circuit (talking and listening) for each participant and essentially guarantees quality connections for the duration of the call. If a circuit cannot be established, the call is not placed, and the user receives a busy signal. However, the telephone network is not efficient. Even if a dominant talker prevents participants from taking part, the unused circuits remain connected. Although the connection is high quality, circuit switching offers no protection against system failure while the connection is active—if any link in the circuit fails, then the call is lost.

Packet-switched networks, such as the Internet, were designed and optimized for transferring text and other non-real-time data between computers. Data is divided into packets, and each packet is sent individually. Packet headers contain the information necessary for the packet to reach its destination. Packet headers can be quite lengthy, but most of them contain source and destination addresses, error checking, and sequencing information. When a packet arrives at a router located between the source and destination nodes, it is first stored in memory, then, using the information in the packet header and any available network congestion information, the router selects a suitable output link and forwards the packet toward its destination. Some protocols also analyze the error checking bits at the router. Depending on the switching protocol, the packets of a data file from the same transmitter may follow different routes to the receiver. When a packet reaches its destination, the receiver analyzes the information in the packet header to determine which application it belongs to, its location or ordering in the application whether errors are present, and whether any packets between it and the previous one are missing.

Packet switched networks are robust; node failures do not severely impact the network due to the absence of fixed routing between the transmitter and the receiver. Because the bursty nature of data makes it uneconomical to use continuously connected circuits, all users share the available bandwidth. Faulty data is a more severe problem than timeliness, so most packet protocols are designed to guarantee integrity, regardless of the time required to transmit the information. Data integrity is maintained by retransmitting unacceptable or missing packets.

Figure 3.1 illustrates different ways in which voice traffic enters a wide area packet switched network.

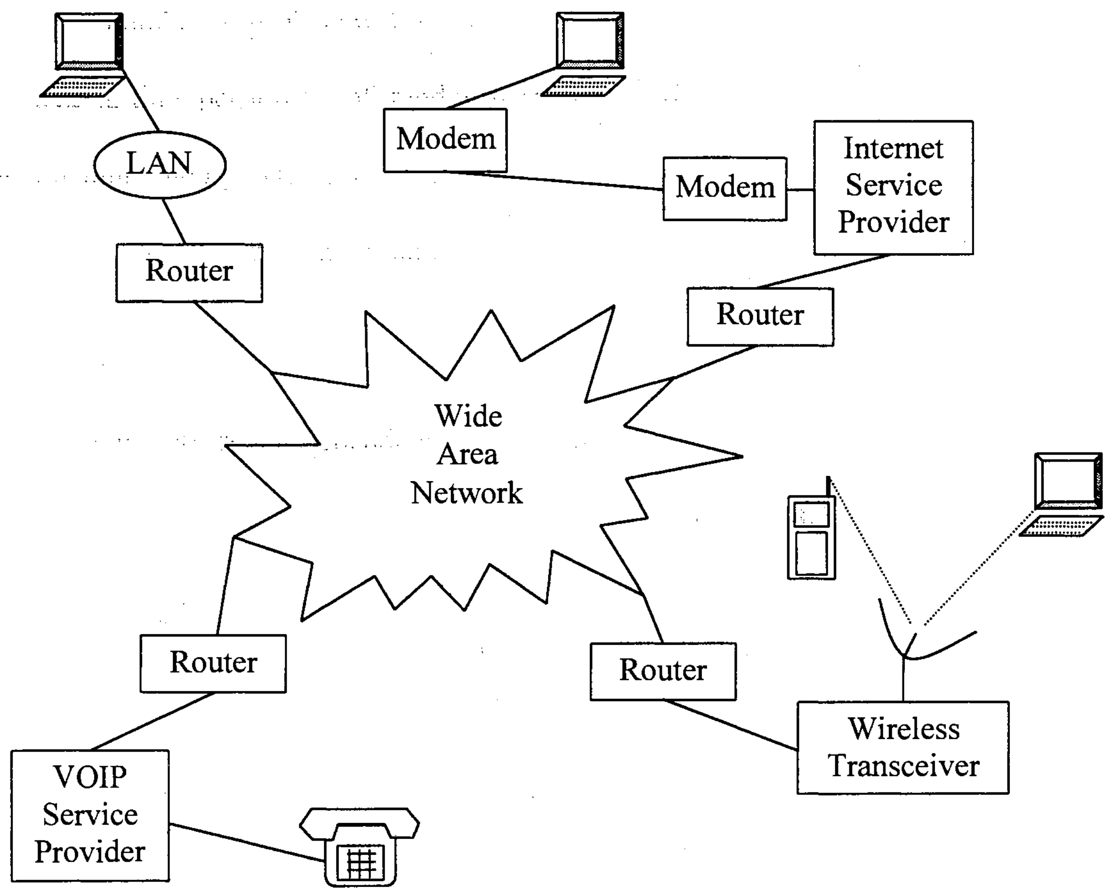


Figure 3.1 Voice Connection to Wide Area Network

Originally, voice communication required a computer at both terminals. Each computer has a speaker and a microphone, and the digitized voice is transmitted between the computer and the common wide area network (WAN) (usually the Internet) through an Internet service provider (ISP) or a local area network (LAN). Modern voice over Internet protocol (VOIP) providers offer voice communication through the Internet without requiring a computer at either end. The user calls the VOIP service using the telephone and the VOIP provider supplies the connection through the Internet to another telephone or computer. Wireless communications using WANs is growing in popularity, but it is not yet common. The wireless transceiver provides the WAN connection for a computer with a wireless connection or a wireless phone. Any one of these connection arrangements can communicate with any other.

Why is transmission of voice across packet-switched networks a challenging problem? Packet-switched networks usually provide a simple, single-class, best-effort service. Two computers can always be connected (i.e., no busy signal), but the path may be unreliable. Unreliable paths result in very high packet delay, which is treated the same as packet loss for real-time applications. Standard designs of high-speed networks show a significant variability in performance, and real-time guarantees cannot be given to applications ([30]). Unreliable connections are unavoidable and can be caused by several events: 1) A congested network node will drop packets that cannot be queued; 2) Excessive channel bit error rates may render packets unacceptable to the receiver; and 3) Heavy loading of the workstations may lead to scheduling difficulties in multi-tasking operating systems. Packets discarded or delayed by network nodes (e.g., routers and gateways) due to congestion are anticipated to become more prominent in future integrated packet networks (IPNs). DaSilva et al. [18] predict that the speed and capacity of future IPNs is likely to be limited by switching in general, and nodal packet processing in particular.

Queuing Theory

Network analyzers evaluate congestion and delay using mathematical models. The most common, simplified models of computer networks use queuing systems based on statistics to predict these parameters. In queuing models, each server contains a transmitter, a receiver, and a buffer. The server receives packets and stores them in the buffer until they are transmitted.

Queuing models contain six elements: arrival process, service process, number of servers, dispatching discipline, queue limiting size, and item population. The arrival process is randomly distributed and represents how jobs arrive in the system from the outside world. The service process is also randomly distributed and models the length of time a server will be occupied by a job. The number of servers available to process packets is a constant value. The dispatching discipline contains the rules for determining priority for servicing jobs. The queue limiting size is the maximum number of waiting jobs that can be accommodated. The item population is the number of potential customers.

One type of very simple queuing model that describes the activity in a single server disregards packet size and varies only the arrival process, service process, and number of servers. These models consider the dispatching discipline as first-in-first-out (FIFO), and the queue limiting size and item populations are treated as infinite.

The inter-arrival times of packets at the server are represented by (3.1).

$$T_1, T_2 - T_1, T_3 - T_2, \ldots \text{ where } 0 < T_1 < T_2 < T_3 < \ldots \tag{3.1}$$

The simplest queuing model is Markovian (or memoryless) for both the arrival process and the service process. In a memoryless model, random packet arrival times are modeled as a Poisson process with rate λ. The inter-arrival times of packets at a node are independently and identically distributed by the exponential probability shown in (3.2).

$$P\{T_{n+1} - T_n \geq t\} = \lambda e^{-\lambda t}, \quad t \geq 0 \tag{3.2}$$

In (3.2), λ represents the mean number of arrivals of jobs per second, i.e., the average arrival rate.

Markovian models also represent service times exponentially. That is, the probability that a server will be occupied by a job for more than t seconds is

$$P\{S_n \geq t\} = \mu e^{-\mu t}, \quad t \geq 0. \tag{3.3}$$

The mean service rate is represented by μ, and S_n represents the transmission time. When both the arrival and service processes of a single server are Markovian, the queue is referred to as M/M/1 (arrival process/service process/number of servers). Other service distributions considered for queuing are deterministic (D) and general (G). An M/G/1 queue has a single server with a Markovian arrival rate but the service times have a distribution that is not necessarily exponential (i.e., general). An M/D/1 queue has a single server with a Markovian arrival rate and a constant service time.

A number of studies in recent years have demonstrated that for some environments, the traffic pattern is correlated ([57], [79], [87]). Bursty traffic patterns tend to exhibit certain degrees of correlation between arrivals, and show long-range dependence on time. This characteristic has been labeled *self-similar traffic*. Self-similarity is a distinctive property of fractals (objects whose appearances are unchanged regardless of the scale at which they are viewed). Poisson traffic models exhibit short-term burstiness, but long-term smoothness. In contrast, self-similar traffic exhibits burstiness at all time scales. The Hurst, or self-similarity parameter H ($0.5 \leq H \leq 1.0$), is a measure of the length of the long-range dependence of a stochastic process. Hurst parameters near 0.5 show few self-similar characteristics; parameters near 1.0 show much self-similarity. A stochastic process, $\mathbf{x}(t)$, is defined as statistically self-similar if, for any real $a > 0$, each of the three conditions listed in (3.4)-(3.6) for the mean, variance, and autocorrelation exist.

$$E[\mathbf{x}(t)] = \frac{E[\mathbf{x}(at)]}{a^H}, \tag{3.4}$$

$$\text{Var}[\mathbf{x}(t)] = \frac{\text{Var}[\mathbf{x}(at)]}{a^{2H}}, \tag{3.5}$$

$$R_x(t,s) = \frac{R_x(at,as)}{a^{2H}}. \tag{3.6}$$

where H is the Hurst self-similarity parameter. Leland et al ([57]) performed analysis that showed that, if the input is self-similar, increased delays and increased buffer size requirements will be experienced in any multiplexing of self-similar streams.

Regardless of the models used to calculate traffic parameters, it is the mean values upon which most designers rely. Using λ to represent mean packet arrival rate and μ to represent mean packet service rate, a system utilization parameter, ρ, can be defined by

$$\rho = \frac{\lambda}{\mu}. \tag{3.7}$$

If the mean packet service rate exceeds the mean packet arrival rate, $\mu > \lambda$, then the system utilization is less than unity, $\rho < 1$, and the server is likely to be in an uncongested state (if variance about the means is not substantial). In an uncongested server, all packets are serviced before the buffer reaches its capacity. However, if the mean packet service rate is equal to or below the mean packet arrival rate, $\mu \leq \lambda$, then system utilization is greater than unity, $\rho \geq 1$, and the server is considered congested. Congested servers drop packets when the buffers reach capacity, resulting in missing voice segments for packet-speech applications.

In an uncongested M/M/1 model ($\mu > \lambda$), the average queue length, L_Q, is given by

$$L_Q = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}. \tag{3.8}$$

The average delay, *T*, is related to the average queue length by Little's result ([97]), shown in (3.9).

$$L_Q = \lambda T \tag{3.9}$$

Using Little's result and the queue length, *L*, the average delay, *T*, can be computed by

$$T = \frac{1}{\mu - \lambda}. \tag{3.10}$$

If $\mu >> \lambda$, the delay through the server is small. As $\mu \rightarrow \lambda$, the delay increases. Excessive service delays at intermediate nodes subsequently delay packet arrival times at the destination. The destination server may have to wait for the individual packets that comprise a file. If different nodes with varying delays service sequential packets, packets can arrive at the destination out of order. If the delay is not constant, packet-speech play-out exhibits an unpleasant jitter. If the delay is excessive, the packets must be considered lost, which results in speech gaps.

The number of packets waiting in the buffer, the waiting time of a packet, and the total queuing time per job are all random variables whose mean values can be computed using queuing theory. However, the average delay, T, and system utilization, ρ, are considered the most important variables.

Channel Errors

Channel errors are dependent on the transmission media and equipment. Optical communication channels have low bit error rates, typically on the order of 10^{-9}. Error rates for analog channels vary from 10^{-3} to 10^{-7} and are typically modeled as 10^{-5}. Bit error rates for purely digital facilities and newer analog facilities are usually closer to 10^{-7}. Errors also tend to occur in bursts on both analog and digital channels, so the conditional probability of additional errors in a block of bits increases significantly after the first error occurs. Wireless channels are especially troublesome due to the multipath signals. In narrowband systems this results in Rayleigh fading which is characterized by deep fades of signal with resulting in large bursts of errors ([78]). The Rayleigh distribution has the probability density function given by (3.11).

$$p(r) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} (0 \leq r \leq \infty) \tag{3.11}$$

The root mean squared (RMS) value of the received signal, r, is σ.

Data packets with errors are unusable. The detected errors are typically not corrected. At network nodes, packets with errors are simply discarded. At the destination, re-transmission is requested when packets with errors are received. Real-time packet-speech applications do not have time to request replacement packets, so gaps in the speech signal will occur when errors cause the packet to be discarded.

Many vocoders add error detection and correction (EDAC) bits to each frame of compressed speech. The EDAC bits protect parameters that most affect the quality of the synthesized speech. The receiver uses the EDAC algorithm to determine whether the channel has corrupted any protected bits and restores them. Error detection and correction is effective for inaccuracies within the frame, but it is ineffective if the entire frame is lost.

Packet Loss Measurements

The previous sections provided a mathematical description of packet loss, packet delay, and bit errors. However, measured statistics do not strictly follow these mathematical descriptions. Several authors discuss measured loss statistics on the Internet ([1], [6], [8], [35], [45], [49], [62], [63], [81], [86]). Internet loss rates are difficult to quantify because measuring methods vary among authors and because the characteristics of the network depend on the unknown behavior of other connections throughout the network. It is not unusual, according to Khansari ([45]) and Sanghi et al. ([81]), to lose about 10% of the transmitted packets, and rates can reach as high as 40%. Delay is difficult to quantify since busy periods vary across circuits. There is general agreement that packet loss is highly correlated, i.e., given that packet *n* is lost, the probability that packet *n+1* is also lost increases. Studies of packet switched networks have shown that packet losses are most likely to occur in pairs or triplets.

As discussed previously, packet loss translates to frame loss in a voice transmission system, and packets arriving too late to be synthesized are unusable and will be considered missing. Missing voice frames not only decrease the quality of the voice transmission, they also decrease the intelligibility. Low-rate voice coding algorithms achieve their high compression by removing much of the redundancy in speech. The frames of low rate algorithms therefore have a high entropy rate, hence missing frames have an even greater effect on the quality and intelligibility of highly compressed speech. Missing-frame compensation algorithms are therefore needed to achieve acceptable voice quality when packet-switched networks are used for transmission of voice.

CHAPTER FOUR

LITERATURE SEARCH

Although researchers have been trying to compensate for missing voice samples in packet-switched speech since 1981, the problem is not yet solved. Five categories of missing-frame compensation techniques have been discussed in the literature: redundant information, loss reduction, network intervention, frame reconstruction, and proprietary. Proprietary techniques are rarely elaborated on, and as such, are not discussed herein.

Redundant Information Methods

One of the newest approaches is to include information about previous voice frames in the current packet, i.e., redundant information. If packet *n* is lost, the redundant information in packet *n+1* is used to construct the missing frame. For example, the Internet FreePhone product, described in [29], creates packets containing both the current compressed voice information and a more compressed version of voice from previous packets. The present voice signal is compressed to 40 kbits/second. The past voice signal is compressed to 4.8-24 kbits/second. This method has been applied to other vocoders as well ([6] and [36]). Erdol, Castelluccia, and Zilouchian describe a method in [24] that appends short-time parameters describing PCM speech segments in future packets. Chin, Hui, and Choo ([13]) describe a system that varies the amount and type of redundant information depending on channel conditions. Kostas, et al, ([49]) and Moffat, Davis and O'Neill ([66]) also discuss redundant methods, but in less detail.

Loss Reduction Techniques

Loss reduction techniques attempt to reduce the impact of the expected packet loss on voice quality. This technique divides the voice frame into separate packets and gives each packet a priority rating based on it's importance in providing high quality voice. The network then discards the less important packets before the more important ones. When only low priority packets are lost, the effect is not as deleterious.

Wong and Goodman, and Lara-Barron and Lockhart ([99], [52]) propose and discuss selective discarding techniques in which the speech segments are classified according to whether they can be reconstructed with adequate fidelity if they are lost. Pitch repetition (discussed in the frame reconstruction section of this chapter) is used for voiced frame reconstruction. Unvoiced frames are reconstructed by repeating the previous frame. The reconstructed speech is compared with the original speech in the transmitter. If the speech segment can be accurately reconstructed, it is marked as low priority.

In [61], Lockhart and Lara-Barron keep the transmitter and receiver synchronized (e.g., filter memories) by assuming that the lower priority packets will not be received. When a packet is lost, the receiver uses the same reconstruction procedure to recover the missing speech samples as the transmitter. When low priority packets are not lost as expected, the receiver uses them as supplemental information to improve the speech. This method produces lower quality speech in loss-free environments than other approaches. For certain applications, this degradation is considered acceptable given the improved performance in adverse channels. Suzuki ([91]) proposes marking frames of ADPCM speech by priority for ATM networks but does not provide much detail on how the priority is determined.

Yin, Li, and Stern ([104]) discuss embedded coding and introduce a detection threshold method for speech energy. Their method compares the energy in the speech segment to two thresholds. Speech segments whose energy is below both thresholds are considered silence, and no packet is transmitted. Segments with energy between both thresholds are considered semi-silence, and the packets are considered low priority. Segments with energy exceeding both thresholds are considered talk-spurts, and are classified as high priority.

Kitawaki, et al., Sriram, and Suzuki and Taka ([47], [86], and [91]) propose embedded coding techniques for ADPCM which place the least significant bits (LSBs) and most significant bits (MSBs) in separate Internet packets or ATM cells. The LSB cells are given lower priority, and are not used for speech synthesis if they are lost.

Yong ([105]) discusses three different network CELP reconstruction techniques that prioritize speech information. The first method classifies based on the energy in the frame and the stationarity of the speech. Frames with low energy or little change from the previous frame are given low priority. The second method divides the quantized parameter bits into low and high priority. Yong suggests the spectral and pitch parameter bits should be classified as high priority. The third method relies on an embedded coder. Speech is coded in two stages with the first stage coding the speech and the second stage coding the residual. The second stage is classified as low priority.

Jayant and Christensen, Jayant, Chen and Chen, and Zagursky and Ginters ([41], [42], [12], [107]) propose variations of loss reduction which place sequential speech samples in separate packets regardless of the importance of the samples. In [41], Jayant and Christensen divide bits from pulse coded modulation (PCM) between packets (i.e., odd samples in one packet and even samples in another packet). At the receiver, simple interpolation between received samples is used to reconstruct missing packets. This method fails if the network loses sequential packets containing adjacent samples. Chen and Chen's method is an *L*th packet interleaving procedure. They employ Wiener and Kalman-based sample interpolation schemes to recover the missing samples. Zagursky and Ginters block code PCM samples (voice samples are written to the columns of a matrix and read from the rows of the matrix into the packets). They describe first- and second-order interpolation algorithms to replace missing samples.

Yuk, Rym, Lee, and Cho, [106] divide voice data into two classes based on the required communication quality. When congestion occurs, the transmitter reduces the bit rate of the conversations that do not require high quality. If this decrease is not sufficient, the transmitter decreases the bit rate of the conversations that require high quality.

Network Intervention

New protocols are being developed to provide guaranteed quality of service (QoS) to users willing to pay for it. Network intervention techniques take advantage of these new protocols to give voice packets a higher priority than data packets or to establish a real-time connection between two communicators.

Goodman and Wei's 1991 paper ([33]) describes a packet reservation multiple access (PRMA) system in which a terminal that generates a sequence of packets obtains a reservation of uncontested use of subsequent time slots. At the end of a talk-spurt, the terminal stops transmitting and the base station informs all terminals that the slot is available for contention in the next frame.

Shah, Atungsiri, Kondoz, and Evans investigate multiplexing low bit rate speech in thin-route telephony ([84]). They state that in multiplexer applications, the encoder is quite close to the multiplexer subsystem that makes the decision to drop speech frames. The encoder can therefore be immediately informed if a frame is dropped and adjust its filter memories so the encoding of the following frames take that frame loss into account. Bolot and Vega-Garcia ([6]) propose controlling the rate at which packets are sent over a connection to match the send rate to the capacity of the connection to minimize packet loss. Internet Protocol version 6 (IPv6), the next generation Internet protocol, will provide the ability to guarantee network resources. Resource Reservation Protocol (RSVP), which is in use now, allows reservation of appropriate amounts of bandwidth for each call.

Frame Reconstruction

Frame reconstruction techniques use the data in the successfully received packets to calculate the data in the missing packets. They require no knowledge of network protocols, and all the information from a voice frame is kept together.

The easiest and least complex frame reconstruction technique is the venerable silence substitution method, where silence covers missing voice frames. Repeating the previous packet is an only slightly more complex alternative that is more acceptable to listeners than silence substitution. Noise filling algorithms that substitute scaled Gaussian noise for missing speech segments are other slightly more complex options. Pitch replication techniques ([51] and [32]) calculate the pitch from T samples of the speech prior to the lost frame. The missing packet is reconstructed by repeating the last T samples received. Yong ([105]) builds on this idea by using the CELP pitch parameter from the previously received frame to create the excitation for the missing frame.

Sanneck ([82]) proposes a variant of frame repetition in which the transmitter divides speech into chunks based on the pitch period. Each chunk contains one pitch period if the speech is voiced, or a larger number of samples (larger than the maximum pitch period) if the speech is unvoiced. Chunks containing transitional speech are split at the transition. Each packet contains two chunks. If the receiver detects a missing packet, adjacent chunks of the previous and following packets are reused.

A more complex method using pattern matching is proposed by Goodman, et al. ([32]). A template created from the M speech samples that arrived just before the missing packet is compared with a history (typically 1000 samples) of speech saved from past packets. The samples following the best match are used for the missing packet. Wasem, et al. ([98]) discuss this one-sided template matching and also propose a two-sided method using search windows in both the past and future.

DaSilva, Petr, and Frost ([18]) propose coding and reconstruction techniques based on the classification of the speech. The transmitter classifies the speech segments and places this information in the packet header. Background noise is encoded using 2-bit

ADPCM and reconstructed by randomly choosing a block from the sample history. Voiced speech and fricatives are encoded using 4-bit ADPCM. Voiced speech is reconstructed using pitch repetition and weighted Gaussian noise is generated to replace the fricatives. A fourth class, called "other", is encoded using 8-bit μlaw and reconstructed using packet repetition.

Husain and Cuperman also classify information for frame reconstruction using CELP coders ([37]). Their algorithm uses four classes of speech: silence, unvoiced, voiced, and transition. The class information is assumed to be available at the receiver regardless of channel characteristics, and the excitation signal is reconstructed based on it. The receiver creates voiced excitation by refining the pitch information received from the previous frame, and unvoiced excitation by using a Gaussian generated signal. The spectral coefficients are extrapolated based on the least-squares fading memory polynomial filter, which makes use of the discrete Laguerre polynomials.

The International Consultative Committee on Telegraphy and Telephony's (CCITT) G.729 coder, as described by Salami, et al in [80], and G.723.1 ([49]) reconstruct the current frame by replacing the missing excitation signal with one of similar characteristics.

Performance

The quality of reconstructed speech from these methods varies greatly. Generally, the more expensive methods (i.e., those requiring greater amounts of bandwidth, memory, delay, computation, cash, etc.) yield better results.

Redundant methods can substantially improve the quality of voice in a packet-loss environment. However, adding redundant information increases the size of packets and therefore the traffic on the network (potentially elevating the congestion and hence further increasing the packet loss and delay problem). Because reconstruction cannot begin until the next packet is received, there is an unavoidable delay built into this process. Excessive delay can render two-way communication impossible.

Loss reduction techniques mitigate the effects of missing voice frames. However, the protocols of the packet-switched network must be known to use these procedures. This dependence eliminates portability to other packet-switched networks. Since the receiver must wait for all the packets containing the information in a voice frame to arrive, the participants will experience additional delay.

Network intervention methods have the potential to eliminate packet loss and delay. But new protocols must first complete lengthy standardization procedures. Integration of the new protocols into existing networks will require time, and some networks may never be upgraded. As with loss reduction techniques, specific protocol dependency will eliminate portability to other networks.

Frame reconstruction techniques are generally the most portable. Many do not add any delay, and most don't excessively increase network traffic. Unfortunately, these techniques presently do not offer very high quality results.

It is difficult to compare the test results of these systems because several different methods were used for testing. Some researchers used formal mean opinion score (MOS) testing and others performed these tests informally. Signal-to-noise ratios were used for much testing since the majority of the techniques were used for waveform coders. All researchers agree, however, that silence substitution is the least acceptable method for compensating for missing frames. Silence substitution works at packet loss rates up to only 1-2%. Packet repetition is considered only slightly better than silence substitution. Some intelligibility can still be maintained up to loss rates of 40%. All other procedures out perform these two baselines.

There is currently no high-quality, portable, low-delay missing frame compensation scheme available that works for any waveform or parametric voice-coding algorithm. The techniques described in the literature are either specific to a particular voice-coding algorithm or reliant on a specific packet-switching protocol.

CHAPTER FIVE

NETWORK SIMULATOR

Testing the missing-speech compensation routines required either access to the protocol structures of a working packet-switched network or a network simulator. A network simulator provides more flexibility, so this choice is preferred. Software packages for network simulators are available, but they are complex, expensive, and difficult to modify. Since flexibility was desired and strict adherence to particular protocols was not required, the missing-speech compensation routines were evaluated on a network simulator written in Matlab.

Figure 5.1 shows a block diagram of this network simulator.

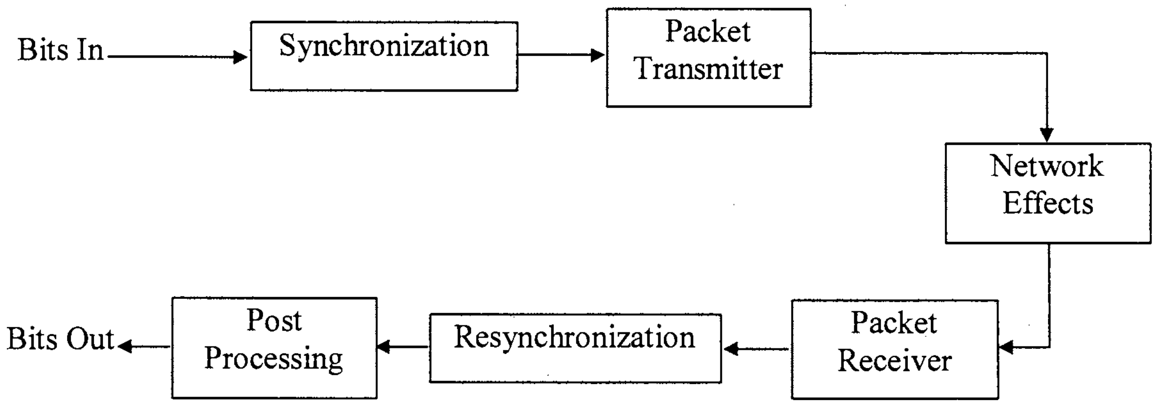


Figure 5.1. Block Diagram of Network Simulator

The input and output data files contain one bit per word. The synchronization and resynchronization blocks are designed for alignment and were not needed for the algorithms described in this paper.

In the Matlab simulation, the user can choose the desired protocol for the "Packet Transmitter" and "Packet Receiver" blocks in addition to the size of the data portion of the packet. The most prevalent wide area network (WAN) protocol, Transmission Control Protocol/Internet Protocol (TCP/IP) is available and can be chosen singly (TCP or IP alone) or as a pair. Both TCP and IP are aligned on octet boundaries, which restricted the flexibility with regard to this project (frames of bits from some vocoders, e.g. MELP, must be split between packets). A generic packet protocol is available to offer more flexibility with various voice-coding algorithms. The generic protocol can also be used to represent multiple protocol situations (e.g., TCP/IP→ATM→TCP/IP) that occur frequently on wide area networks. The generic protocol has all the features of a network protocol: a packet header containing source and destination addresses, sequence information, data size, etc., and a packet data area. The generic protocol does not have the error protection capability of TCP/IP, but this information is not necessary to evaluate packet loss. A final choice that simulates a wire (i.e., no packet structure) is also available.

Packets are corrupted in the "Network Effects" block. Four types of errors were used: Random Bit Error, Random Block Error, Pattern, and Random Packet Loss. The user also selects the percentage of errors that affect the packets or the bits in the packets. The "Random Bit Error" option inserts uniformly distributed errors across all packets. "Random Block Errors" are similar to random bit errors, but the bits are first grouped into blocks. The user specifies the number of bits there are per block. Errors are distributed uniformly across all blocks. When a block is in error, 50% of the bits within the block are corrupted (with a uniform distribution); hence the data is irretrievable. If a non-uniform distribution of errors is required, the "Pattern" option will load a user-created Matlab variable file (MAT-file). The file format is similar to the input and output bit files, i.e., one bit per word. The bits to be corrupted are represented with a "1" and all other bits are represented with a "0" in the pattern file. The pattern bits are reshaped and replicated, if necessary, to match the size of the input data bitstream. The "Random Packet Loss" option selects random packets for obliteration using a uniform distribution with a user-selected probability.

The missing-speech compensation algorithms comprise the "Post Processing" block. It uses sequence numbers in the packet header to locate missing frames. The following frame compensation algorithms are available for user selection:

- Nothing
- Silence Insertion
- Frame Repetition
- Frame Interpolation
- Naturalness Preserving Transform
- Markov Chain Prediction

If no frame compensation is performed, i.e., the "Nothing" option, the received samples are played sequentially regardless of timing. The "Silence Insertion" option generates the appropriate bits needed to create silence in the output speech in place of the missing packets. "Frame Repetition", the most common solution to the missing speech frame problem, repeats the last correctly received frame of speech until a new frame is received. "Frame Interpolation", which adds a one-frame delay to the process, interpolates the speech parameters for the missing frame using the past or future frames. The last two options ("Naturalness Preserving Transform" and "Markov Chain Prediction") are described in Chapters Five and Six. Some of the options require knowledge of the voice-coding algorithm (e.g., "Silence Insertion", "Markov Chain Prediction", and "Frame Interpolation"). For these choices, the output bit stream contains markers to indicate the location of the missing bits that the voice coding software uses for the missing-frame compensation.

The Matlab graphical user interface (GUI) editor provided an easy method to improve the user interface of the network simulator and to offer user-selectable voice coding before and after the network simulation. A picture of the network simulator GUI is shown in Figure 5.2.

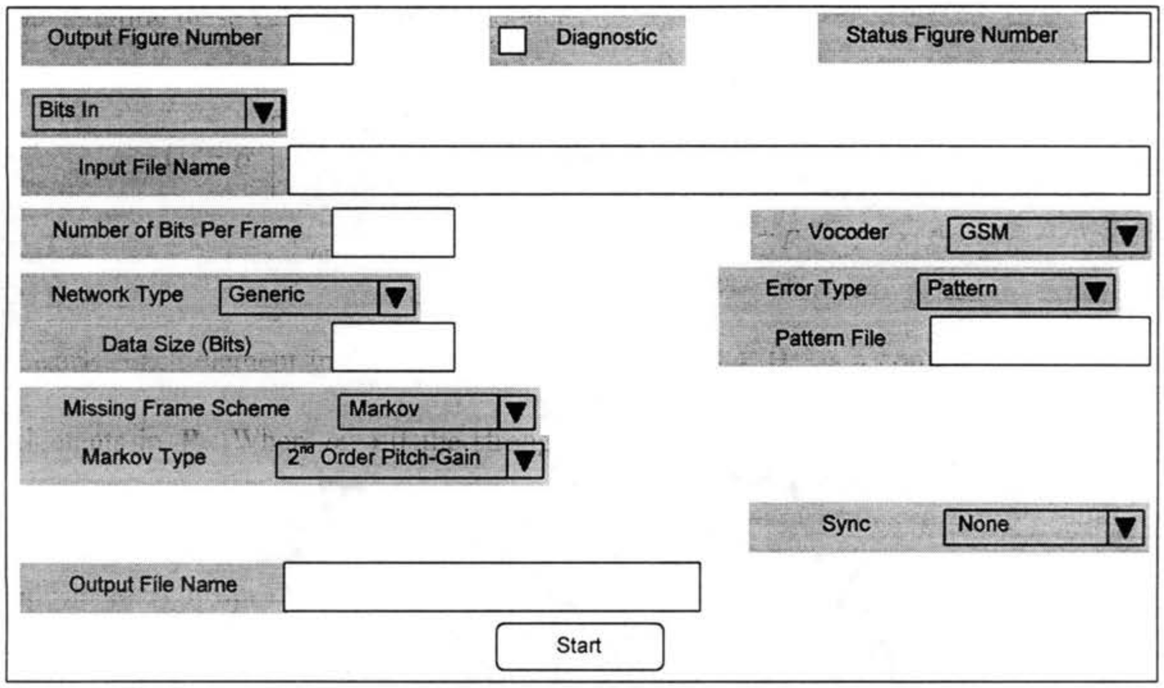


Figure 5.2. Network Simulator Graphical User Interface

Using the GUI, the user can determine the presence of diagnostic data and informational graphs, select input and output file locations, choose voice coding algorithms, and specify parameters for the network simulator. Once the correct inputs are entered, the "Start" button activates the program.

The flexibility to add or change code and the ability to easily change parameters made this program a valuable tool for analyzing the effectiveness of missing frame compensation algorithms.

CHAPTER SIX

NATURALNESS PRESERVING TRANSFORM

The Naturalness Preserving Transform (NPT) is a unitary transform based on a stochastic operator that uses the Hadamard matrix. This transform was first described in [101] and later in [102]. The NPT's success for reconstructing missing portions of images ([101], [102], [27]) made it a viable consideration for reconstruction of missing speech.

Naturalness Preserving Transform Description

The original NPT designed by Yarlagadda and Hershey operates in two dimensions, but Osinubi and King later modified the transform to operate in one dimension ([70]). Both the one- and two-dimensional transforms are described in the following sections.

Both transforms use a stochastic operator, Ψ, defined as

$$\Psi_n(\alpha) = \alpha\mathbf{I}_n + (1-\alpha)\mathbf{H}'_n. \tag{6.1}$$

The weighting factor, α, has restrictions $0 \le \alpha \le 1$, $\alpha \ne 0.5$. $\mathbf{H}'_n$ is the normalized Hadamard matrix, $\mathbf{H}_n$, ([102]) of order n, defined as $2^{-\frac{n}{2}}\mathbf{H}_n$. The size of the stochastic operator matrix, and therefore all other matrices relating to the transform, is dependent on the size restrictions of the Hadamard matrix, i.e., square matrices with 2^n rows and columns.

Two-Dimenensional NPT

The two-dimensional NPT is defined by

$$\Pi_n = \Psi_n(\alpha)\mathbf{P}_n\Psi_n(\alpha), \tag{6.2}$$

where $\mathbf{P}_n$ is a $2^n \times 2^n$ matrix containing the data to be transformed. The inverse transform is given by

$$\mathbf{P}_n = \Psi_n\left(\frac{\alpha}{2\alpha-1}\right)\Pi_n\Psi_n\left(\frac{\alpha}{2\alpha-1}\right), \ \alpha \neq \frac{1}{2}. \tag{6.3}$$

The beauty of the transform is that every sample point in the NPT-coded matrix, Π_n, contains information about the entire $2^n \times 2^n$ original sample matrix, $\mathbf{P}_n$. Using this feature, Yarlagadda and Hershey describe an iterative process for reconstructing portions of images that have been lost due to jamming or errors on the communication channel.

The reconstruction technique, described in [101], performs the iterative computation of (6.4) and (6.5) at the *i*th stage.

$$\mathbf{P}_n^{(i)} = \Psi_n\left(\frac{\alpha}{2\alpha-1}\right)\hat{\Pi}_n^{(i-1)}\Psi_n\left(\frac{\alpha}{2\alpha-1}\right) \tag{6.4}$$

$$\Pi_n^{(i)} = \Psi_n(\alpha)\hat{\mathbf{P}}_n^{(i)}\Psi_n(\alpha) \tag{6.5}$$

The non-excised values in Π_n are substituted at each stage in $\Pi_n^{(i)}$ to obtain $\hat{\Pi}_n^{(i)}$. The posited known values in $\mathbf{P}_n$ are substituted into $\mathbf{P}_n^{(i)}$ to obtain $\hat{\mathbf{P}}_n^{(i)}$. The convergence of this algorithm is detailed in [102].

The speech reconstruction experiments described in this chapter use the two-dimensional NPT.

One-Dimensional NPT

Osinubi, King, et al. have written several papers ([70]-[72]) describing their use of the one-dimensional NPT for reconstructing signals. The one-dimensional NPT of the data vector, $\boldsymbol{S}^q$, is given by

$$\mathbf{R}^q = \Psi(\alpha)\mathbf{S}^q \tag{6.6}$$

The inverse one-dimensional NPT is defined by

$$\begin{gathered}\mathbf{S}^q = \Psi^{-1}(\alpha)\mathbf{R}^q = \Psi(\beta)\mathbf{R}^q \\ \beta = \frac{\alpha}{2\alpha - 1} \\ \alpha \neq \frac{1}{2}\end{gathered} \tag{6.7}$$

They have also developed an iterative method for reconstructing lost segments from a one-dimensional signal. Their method is very similar to Yarlagadda and Hershey's, except it is in one dimension.

A Design for Naturalness Preserving Transform Speech Reconstruction

Reconstruction of missing voice frames is similar to reconstruction of missing image information. Since voice is a one-dimensional signal and the NPT operates on two-dimensional signals, the speech samples must be transformed from $1 \times P$ vectors to $2^n \times 2^n$ matrices. Previous experiments revealed that NPT reconstruction produces optimal results when adjacent samples remain together, either in the rows or columns of the matrix, so the transform can capitalize on the correlation between them. Data compression algorithms usually exploit the correlation between adjacent samples to reduce the bit rate, but for NPT reconstruction, the inter-sample correlation allows for fast convergence. The experiments described in this chapter placed sequential speech samples into the matrix rows.

The matrix is transformed and transmitted using several packets. Each packet contains a header that includes sequence numbers, and the receiver reads the sequence numbers in the packet header at the destination. The receiver uses the sequence number to insure that all the packets have arrived and that they have arrived in the correct order. If the receiver detects a missing packet, the voice data is reconstructed using a modified version of the image reconstruction process described previously.

As shown in (6.4) and (6.5), successful image reconstruction at the receiver requires known values of $\mathbf{P}_n$ (the original image) and Π_n (the transformed image). The latter information is available from the received packet(s). The known values of $\mathbf{P}_n$, $\mathbf{P}_{init}$, are not as easily obtained. In [101] and [102], the authors use stationary image information e.g., a cloudless sky, since the image samples are nearly equal and trivial to reproduce. Equivalent stationary or near stationary regions in speech occur on a short-time basis between words when only the background noise is present. Background noise samples are generally low magnitude and accuracy is unimportant. During the call setup procedure, the transmitter provides background noise samples (which could simply be silence) to the receiver so that $\mathbf{P}_{init}$ is known on both ends of the conversation. If the speech utterance cannot be divided so that background samples are present in a $2^n \times 2^n$ segment, the transmitter inserts $\mathbf{P}_{init}$ into a predetermined location of the matrix as overhead. The artificially inserted background noise is removed at the receiver prior to playing the speech. Some speech compression methods remove silence portions at the transmitter and later recover them at the receiver. NPT reconstuction performs the opposite operation, i.e., it adds silence at the transmitter and removes it at the receiver.

Successful reconstruction is dependent on the weighting parameter. This is seen by evaluating (6.2). Substituting (6.1) into (6.2) produces

$$\Pi_n = (\alpha \mathbf{I}_n + (1-\alpha)\mathbf{H}'_n)\mathbf{P}_n(\alpha \mathbf{I}_n + (1-\alpha)\mathbf{H}'_n) \tag{6.8}$$

and rearranging and combining terms results in

$$\Pi_n = \alpha^2 \mathbf{P}_n + \alpha(1-\alpha)[\mathbf{P}_n \mathbf{H}'_n + \mathbf{H}'_n \mathbf{P}_n] + (1-\alpha)^2 \mathbf{H}'_n \mathbf{P}_n \mathbf{H}'_n . \tag{6.9}$$

When $\alpha \to 1$ the resulting transformed matrix closely resembles the original data matrix because $\mathbf{P}_n$ is weighted more heavily than $\mathbf{H}'_n \mathbf{P}_n \mathbf{H}'_n$ in (6.9). (This is why the transform was named "naturalness preserving.") When $\alpha \to 0$, the resulting transformed matrix has a strong Hadamard component due to the heavier weighting of $\mathbf{H}'_n \mathbf{P}_n \mathbf{H}'_n$ in (6.9). Although the transformed samples are closer to the correct values when $\alpha \to 1$, the individual matrix elements contain less information about the input data values and contribute less to the reconstruction process since each element in the matrix formed by $\mathbf{H}'_n \mathbf{P}_n \mathbf{H}'_n$ is a combination of all of the elements of $\mathbf{P}$. This can be seen in the following 2×2 example with

$$\mathbf{P}_1 = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

and

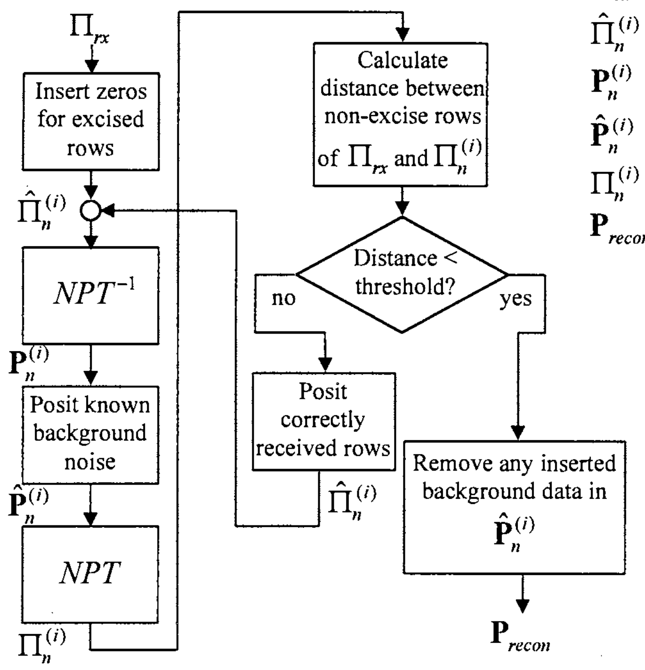
$$\mathbf{H}'_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Substituting these values into (6.9) produces

$$\Pi_1 = \alpha^2 \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \alpha(1-\alpha)\frac{1}{\sqrt{2}} \begin{bmatrix} 2p_{11} + 2p_{12} & p_{11} + p_{22} \\ p_{11} + p_{22} & p_{21} - 2p_{22} + p_{12} \end{bmatrix}$$
$$+ \frac{1}{2}(1-\alpha)^2 \begin{bmatrix} p_{11} + p_{21} + p_{12} + p_{22} & p_{11} + p_{21} - p_{12} - p_{22} \\ p_{11} - p_{21} + p_{12} - p_{22} & p_{11} - p_{21} - p_{12} + p_{22} \end{bmatrix}.$$

Clearly, each element in the matrix produced from $\mathbf{H}'_n \mathbf{P}_n \mathbf{H}'_n$ is a combination of all the elements in $\mathbf{P}_1$. When $\alpha \to 0$ the stronger Hadamard component causes the individual elements of the transform matrix to contribute more toward reconstruction, resulting in considerably fewer iterations for reconstruction. The effect of varying α is illustrated in a later section.

If analysis of the sequence numbers in the packet headers indicates that one or more packets are missing, the process described by (6.4) and (6.5) reconstructs the missing speech samples. In (6.4), the correctly received rows of Π_n are used to form $\hat{\Pi}_n^{(i)}$. The reconstruction algorithm uses the $\mathbf{P}_{init}$ to form $\hat{\mathbf{P}}_n^{(i)}$. Intuitively, the quality of the reconstruction improves as the size of $\mathbf{P}_{init}$ increases. However, since these known samples must intermittantly be transmitted as overhead when the matrix of speech doesn't already contain background noise, efficiency requires as few of these samples as possible to sufficiently reconstruct the speech. The reconstruction process uses the transformed signal, $\Pi_n^{(i)}$, to determine when to cease iteration. The non-excised rows of this signal become progressively more like the non-excised rows of the received transformed signal, Π_n with successive iterations. A simple Euclidean distance is sufficient to measure the error between $\Pi_n^{(i)}$ and Π_n. Figure 6.1 illustrates the reconstruction process in more detail.



Π_{rx}
Insert zeros for excised rows
$\hat{\Pi}_n^{(i)}$
NPT^{-1}
$\mathbf{P}_n^{(i)}$
Posit known background noise
$\hat{\mathbf{P}}_n^{(i)}$
NPT
$\Pi_n^{(i)}$
Calculate distance between non-excise rows of Π_{rx} and $\Pi_n^{(i)}$
Distance < threshold?
no
yes
Posit correctly received rows
$\hat{\Pi}_n^{(i)}$
Remove any inserted background data in $\hat{\mathbf{P}}_n^{(i)}$
$\mathbf{P}_{recon}$

Π_{rx}	received matrix of NPT samples
$\hat{\Pi}_n^{(i)}$	*i*th estimate of Π_n
$\mathbf{P}_n^{(i)}$	*i*th inverse transform of Π_n estimate
$\hat{\mathbf{P}}_n^{(i)}$	*i*th estimate of $\mathbf{P}_n$
$\Pi_n^{(i)}$	*i*th transform of $\mathbf{P}_n$ estimate
$\mathbf{P}_{recon}$	reconstructed speech samples

Figure 6.1. NPT Reconstruction Process

The success of the system described in this figure is described in the following sections.

Naturalness Preserving Transform Speech Reconstruction Parameters

The matrix size, $2^n \times 2^n$, number of samples in $\mathbf{P}_{init}$, and value of the weighting factor, α, all contribute to the success of the algorithm. This section describes some experiments with these parameters.

Matrix Size and Number of Known Samples

Intuitively, as more information is available for reconstruction, the quality of the reconstructed signal should improve and the number of iterations required for convergence should decrease. This can be seen in the plot of Figure 6.2. This plot illustrates the effect of increasing the size of $\mathbf{P}_{init}$ on a 32×32 matrix of speech. This comparison excised the last eight rows of the matrix, reconstructed it, and measured the reconstruction SNR.

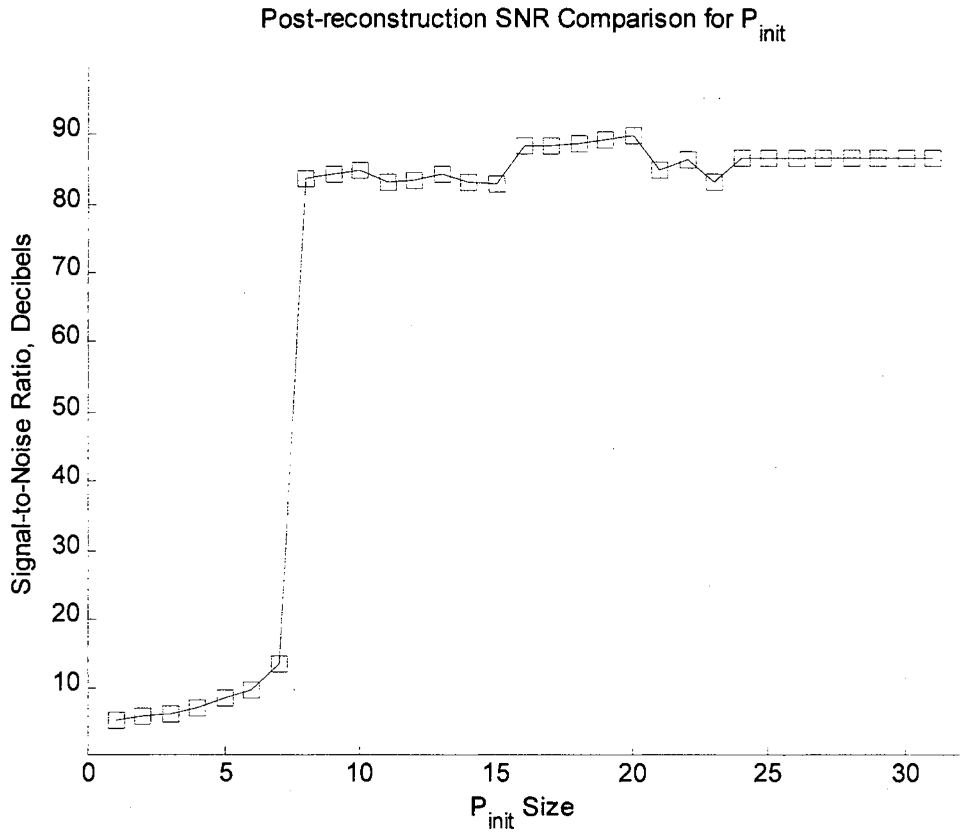


Figure 6.2. Post-reconstruction SNR Comparison for $\mathbf{P}_{init}$

The SNR greatly increased above seven rows of known information. Studies of $\mathbf{P}_{init}$ conducted during this research show that this change occurs when the amount of known information equals the amount of excised information. However, acceptable reconstruction is still achieved when the amount of known data is less than the amount of lost data.

Using a 16×16 matrix ($n = 4$), and only 16 samples (6.25% of the matrix) of known information for $\mathbf{P}_{init}$, the first iteration of the reconstruction process immediately calculates a signal that is very similar to the original signal when 50% of the transformed data is removed. Successive iterations continually refine the signal. Figure 6.3 shows an example of NPT reconstruction of a voiced segment and Figure 6.4 shows an example of unvoiced reconstruction. Each plot in these figures represents one matrix (32 ms, 256 samples) of speech. All five plots in each figure have the same scaling for both horizontal and vertical axes. The top plot in each figure shows the original speech segment. A dashed line in the top plots mark the ends of the known background noise (zeros in this case). The second plot in each figure shows the transmitted transformed data, and the third plot shows the received transformed data with missing packets (replaced by all zeros). The fourth plot in each figure shows the result following a single iteration of the NPT reconstruction process. The basic shape of the original waveform can clearly be seen in the fourth plot. The fifth plot in each figure shows the final reconstructed signal after many iterations. Details of the original signal that are not present after the first iteration are clearly seen in the final signal.

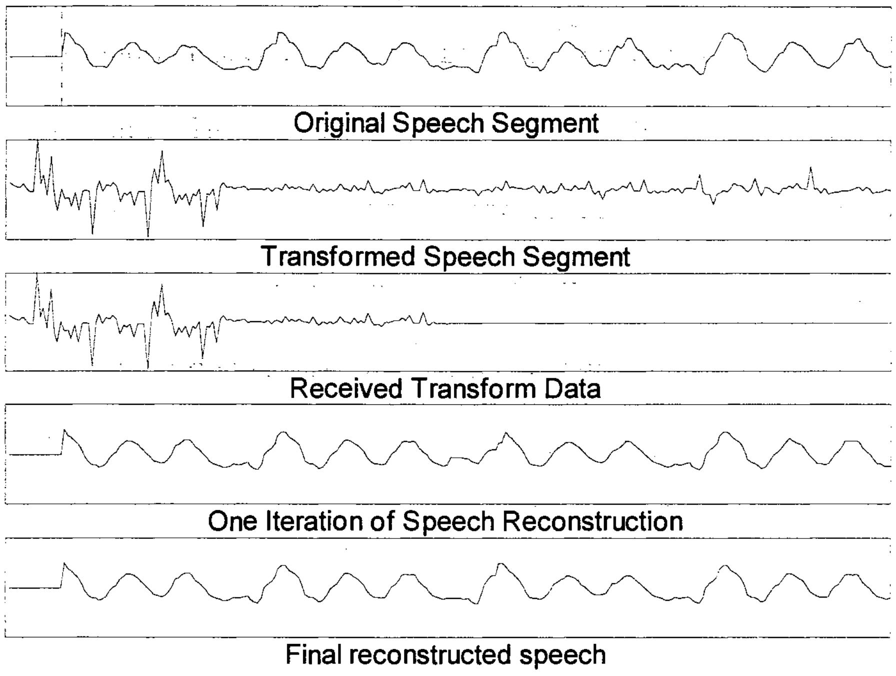


Figure 6.3. NPT Reconstruction of Vowel /I/ (“i” of “fish”)

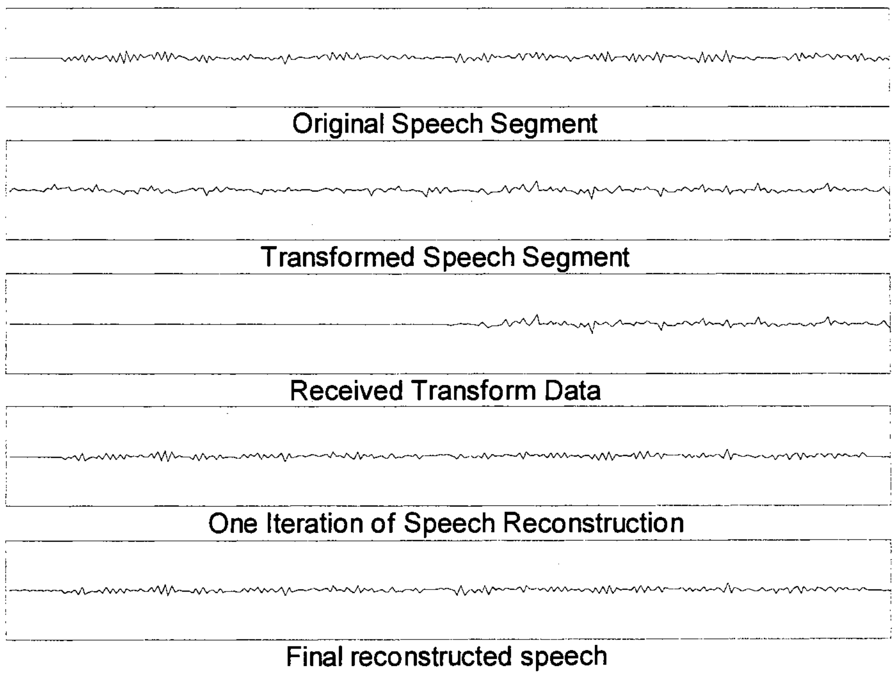


Figure 6.4. NPT Reconstruction of Fricative /s/ (“s” of “bus”)

Weighting Factor

Both the number of iterations and the success of reconstruction are dependent on choosing α correctly. The two experiments described below removed the last 64 samples in every 16×16 matrix of a one-sentence speech segment. The reconstruction algorithm used four rows for $\mathbf{P}_{init}$ to recover the missing samples for the matrix.

Figure 6.5 illustrates how the number of iterations required to successfully reconstruct the missing data varies as α increases from zero to one in steps of 0.05, $\alpha \neq 0.5$. The top line (marked with circles) shows the maximum number of iterations that occurred in the file. The middle line (marked with squares) shows the mean number of iterations that occurred in the file, and the lower line (marked with X's) shows the minimum number of reconstruction iterations that were required in this file.

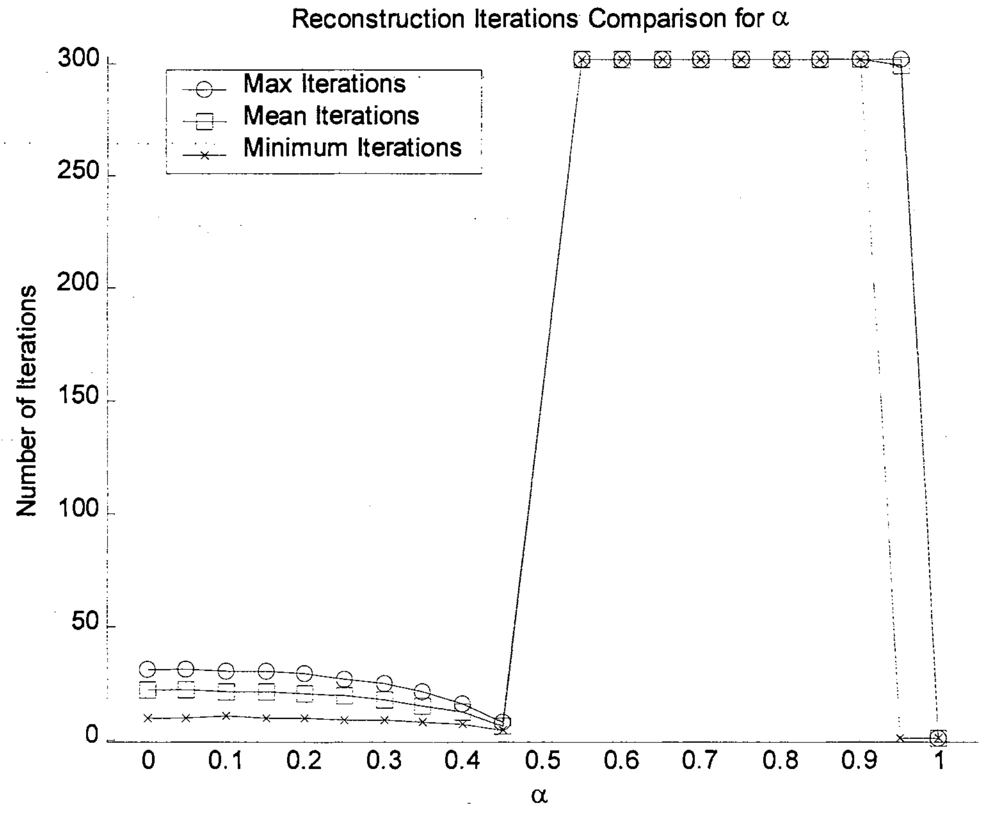


Figure 6.5. Reconstruction Iterations Comparison for Weighting Factor

Measurements were taken only at the values of α marked with circles, squares, or X's, i.e. although the line connecting the symbols passes through $\alpha = 0.5$, this weighting factor was not used. This plot shows less than 35 iterations for reconstruction when $\alpha < 0.5$, with a slight decrease as $\alpha \to 0.5$. However, the number of iterations increases drastically to over 300 when $\alpha > 0.5$, except for $\alpha = 1.0$. When $\alpha = 1.0$, the transformed signal is no different than the original signal. No reconstruction can be performed since the non-excised samples of the transformed signal are the same as the input signal and contain no information about the missing samples.

Figure 6.6 shows how the signal-to-noise ratio varies as α increases from zero to one in steps of 0.05, $\alpha \neq 0.5$. The top line (marked with circles) shows the maximum

SNR in the file. The middle line (marked with squares) shows the mean SNR in the file, and the lower line (marked with X's) shows the minimum reconstruction SNR in the file.

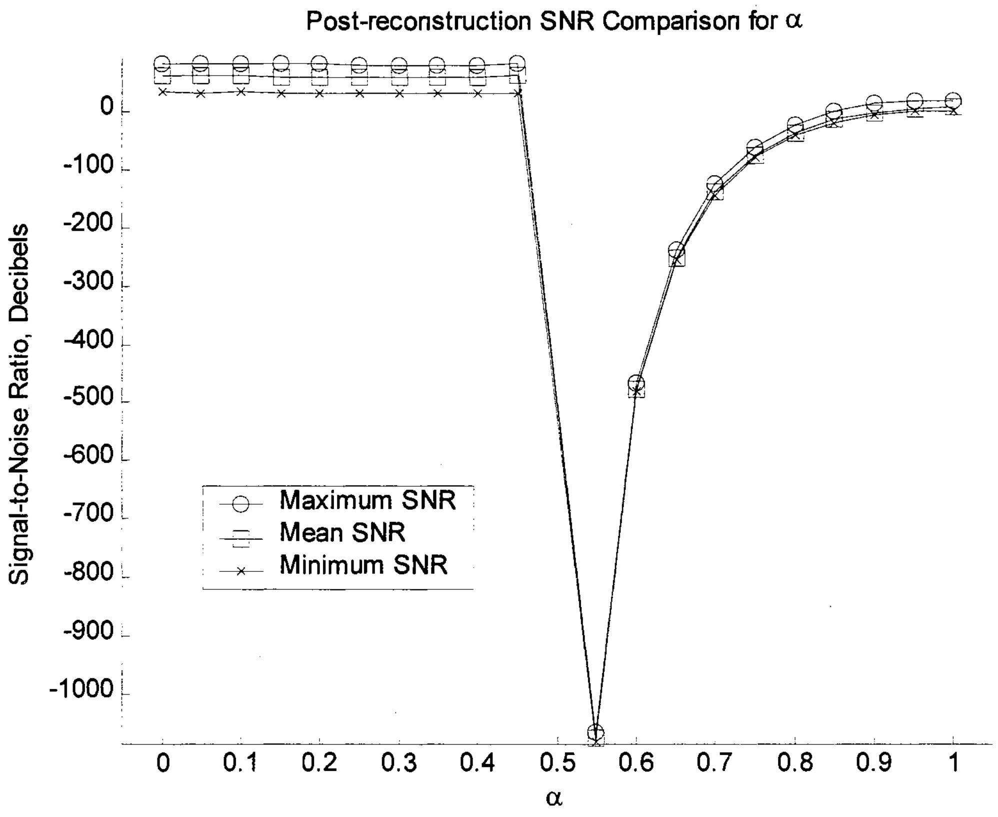


Figure 6.6. Post-reconstruction SNR Comparison for Weighting Factor

Measurements were taken only at the values of α marked with circles, squares, or X's, i.e. although the line connecting the symbols passes through $\alpha = 0.5$, this weighting factor was not used. The SNR decreases slightly (2-3 dB) as $\alpha \to 0.5$. However, there is a tremendous decrease in SNR to nearly -1100 dB when $\alpha = 0.55$. The SNR improves as $\alpha \to 1.0$ above 0.5, but SNR for $\alpha < 0.5$ is always greater than SNR for $\alpha > 0.5$. When $\alpha < 0.5$ the reconstructed speech cannot be differentiated from the uncorrupted speech. However, when $\alpha > 0.5$, the reconstructed speech is quite noisy, though still intelligible.

Although the previous discussion indicates the weighting factor must be low, not all successful NPT reconstruction is reliant on low values of α. In [101], the authors describe an image application using α near 1.0. This application involves a "man-in-the-loop" to reconstruct missing regions in jammed images. With a higher value of α, the transformed image is similar to the original image. The operator then has a coarse version of the image to start with, and can use this to calculate $\mathbf{P}_{init}$ and guide the reconstruction interactively.

Naturalness Preserving Transform Implementation

Delay, complexity and performance are important in implementing any real-time speech processing technique. Delay is a measurement of the time accumulated between a person talking and the listener hearing the voice. Complexity consists of the number of operations (multiplication, addition, etc) and the amount of memory required to complete the reconstruction process. Performance includes the quality and intelligibility of the reconstructed signal. Not only should the listener understand what the talker is saying, the sound of the reconstructed speech should be pleasant. An implementation of the NPT for speech reconstruction is also described in [48].

Delay

Delay is directly related to the size of the NPT matrix. The size restrictions of the Hadamard matrix control the NPT matrix size. An nth order Hadamard matrix is square with 2^n rows and columns (several packets). The transmitter must accumulate a buffer of speech prior to transforming it. The transmitter must then buffer the transformed samples until they can be sent to the receiver. After the receiver reconstructs the samples it must buffer the speech samples until the computer can play them. The entire process accumulates a delay of three buffers. Speech is usually sampled at 8,000 or 10,000 samples per second. At 8,000 Hz, this equates to 125 μs for each sample and $125 \cdot 2^{2n}$ μs for each buffer. The accumulated delay then totals $375 \cdot 2^{2n}$ μs.

The delay for frame repetition is less than for NPT reconstruction. Frame repetition requires only the delay necessary to buffer one packet of speech data. It is still necessary to buffer speech at the transmitter and receiver, but the number of samples in the frame repetition buffer is considerably smaller (one packet versus several).

Complexity

The speed of the processor and the amount of memory required are important implementation issues. Estimates of both complexity measurements for NPT reconstruction are discussed separately followed by a comparison of NPT complexity with frame repetition.

Number of Operations

Calculations for the NPT and its inverse dominate the complexity of the reconstruction process. The number of operations for these calculations can be reduced considerably by postponing the normalization of the Hadamard matrix until after the matrix operations. The expanded transformation of equation (6.9) then becomes

$$\Pi_n = \alpha^2 \mathbf{P}_n + 2^{-n/2} \alpha(1-\alpha)[\mathbf{P}_n \mathbf{H}_n + \mathbf{H}_n \mathbf{P}_n] + 2^{-n}(1-\alpha)^2 \mathbf{H}_n \mathbf{P}_n \mathbf{H}_n \qquad (6.10)$$

The Hadamard matrix elements are all ± 1, so matrix multiplication reduces to 2^{4n} addition and subtraction operations.

Further simplification follows from the nth root ($\mathbf{A}$) of $\mathbf{H}_n$ described in [102]. Since $\mathbf{H}_n = \mathbf{A}^n$, it follows that $\mathbf{H}_n\mathbf{P}_n = \mathbf{A}^n\mathbf{P}_n$. Each row and each column of $\mathbf{A}$ contains only two non-zero values. The non-zero elements in $\mathbf{A}$ are always ± 1, so multiplication by $\mathbf{A}$ is also reduced to computing addition and subtraction only. Specifically, 2^{2n} addition operations are required to multiply a matrix by $\mathbf{A}$. Multiplication by $\mathbf{A}^n$ then requires $n \cdot 2^n$ addition operations. The multiplicative terms containing α and powers of 2 can be pre-calculated and stored, so the only operations required per iteration are the matrix multiplication, addition, and scaling. The NPT transform calculation requires $3 \cdot 2^{2n}$ multiplication operations for scaling ($scalar \times matrix$) and $(4n+3)2^{2n}$ addition operations. The expanded inverse transform has the same form as equation (6.10) and thus requires the same number of multiplication and addition operations per iteration. The appendix contains more details on the calculation of the number of operations required for the NPT transformation.

Memory

Because the number of bytes in a word of memory is processor dependent, memory usage will be represented by words. The NPT and its inverse both require the same amount of memory. The scalars that weight the matrices can be pre-calculated and stored in 3 words. The temporary storage required for the matrix operations can be shared, so the NPT and its inverse each require sufficient space for 6 matrices ($6 \cdot 2^{2n}$ words). The elements of the Hadamard matrix, $\mathbf{H}_n$, and it's nth root $\mathbf{A}$, have only three possible values, 0 or ± 1. Efficient storage schemes will enable these low-resolution matrices to occupy less than one word per matrix element. The appendix contains more details on the calculation of the memory requirements for the NPT transformation.

The NPT and its inverse can be implemented on special purpose hardware since they are achievable with a small amount of memory and few operations. Special purpose hardware will be attractive as the Hadamard transform can be implemented by a section and it can be used several times. The applications for the proposed method in the speech coding area are abundant.

Frame Repetition

Frame repetition is less complex than NPT reconstruction. The transmitter performs only the voice coding and packet construction—there is no additional processing. The receiver only needs enough memory to store the previous packet of received data, and the only operations required for reconstruction are for memory transfers (i.e., copying the stored speech into the output buffer) in the case of a missing packet.

Performance

Performance of voice coding algorithms can be measured using either objective or subjective tests. Objective tests mathematically compare the original speech signal to the speech signal that emerges from the vocoder. Signal-to-noise ratio is a popular objective test. However, mathematical analysis cannot yet accurately measure the human hearing process. Subjective tests using human listeners are the most accurate procedure for measuring the response of human ears. The following sections contain results for both methods.

Signal-to-Noise Ratio

The SNR of the NPT iterative reconstruction process was measured for three different segments of a speech data file: purely voiced, fricative, and transition regions. These segments represented three different classes of speech. The voice system produces voiced speech by oscillating the vocal cords. This creates a signal that is short-term semi-periodic. Forcing air through a constriction at some point in the vocal tract creates fricative or unvoiced sounds. Fricative sounds have no periodic component and resemble low magnitude random noise. A transition is a change from one of these classes to another. The characteristics of the vocal tract are changing, and the characteristics of the waveform correspondingly change.

Figure 6.7 displays the four speech segments that were used to measure SNR. The left plot contains the transition segment, the second plot contains the unvoiced segment, the third plot contains the voiced segment, and the right plot contains a segment of background noise from a modern office.

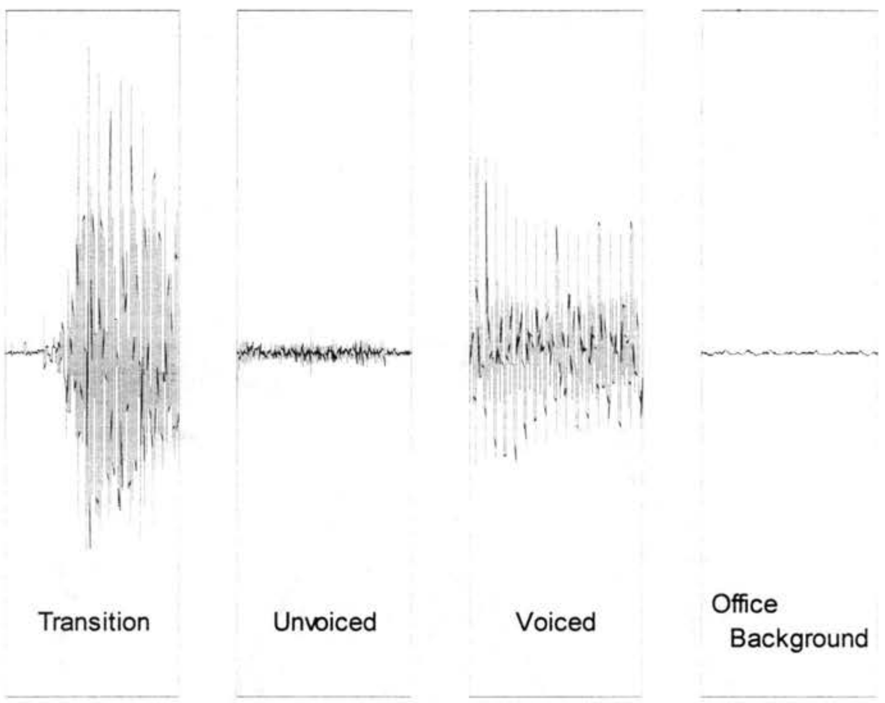


Figure 6.7. Transitional, Unvoiced, Voiced, and Background Speech

From each segment the routine selected one representative 16*x*16 (256 samples) matrix, simulated a random 25% loss, reconstructed the missing data using frame repetition and NPT reconstruction, and calculated the SNR. The latter method used four rows of known information. The result of these calculations is shown in Table 6.1.

	Transition	**Voiced**	**Unvoiced**	**Background**
NPT Reconstruction	88.39	85.58	65.81	46.73
Frame Repetition	-0.93	2.676	2.16	2.76

Table 6.1 NPT SNR Comparison (in decibels)

As this table shows, frame repetition results in very low SNR and is a far inferior method to NPT reconstruction, especially in the difficult transition class. As packet loss increases, the SNR for both methods decreases, but NPT is always considerably higher. In the transition region, frame repetition does not have the correct information for the missing data. The voiced portion of the segment is completely different from the silence portion. The voiced case should do better than the unvoiced or background noise case in frame repetition because there is approximately periodic information in the voiced case whereas no such information exists in the other two cases. The SNR measurements are only on a segment of speech. These examples are isolated cases, but SNR measurements over other segments of speech validate the superiority of NPT over existing methods for speech applications.

Listening

Subjective tests with eleven listeners compared NPT reconstruction against frame repetition. The NPT configuration was the same for the listening test as for the SNR measurements described previously. The program placed one quarter of each transformed matrix into separate packets prior to simulating random packet loss.

The test files contained male and female talkers, and they were processed with PCM, μlaw, and differential pulse coded modulation (DPCM). These three waveform coders were tested to evaluate the performance of NPT on compressed data.

The transformed coded-speech data was subjected to simulated random individual packet loss of 5%, 10%, 15%, 20%, and 25% then reconstructed using NPT reconstruction and frame repetition prior to waveform decoding.

Listeners compared four sentences (two male and two female talkers) in each coder/error combination. All sentences were processed with both NPT reconstruction and frame repetition and the errors occurred in the same samples for both reconstruction methods. The test randomized the sentence order and listeners chose which sentence sounded the best. Table 6.2 shows the preference for NPT reconstruction for all sentences and all listeners.

	5%	**10%**	**15%**	**20%**	**25%**
PCM	95%	100%	95%	98%	98%
μlaw	84%	98%	98%	98%	100%
DPCM	86%	98%	91%	95%	82%

Table 6.2 Preference for NPT Reconstruction

As this table shows, listeners overwhelmingly preferred the NPT reconstruction to the frame repetition, which should not be a surprise as the available NPT data has information about every sample in the original data.

The 25% overhead for this testing is somewhat high. The reconstructed speech quality deteriorates as the overhead decreases, but a smaller informal listening test showed that even at only 6% overhead, NPT reconstruction is still the preferred method–even in error rates as high as 50%. Reconstruction using NPT can also be considered as a scalable lossy error correction method–as the channel deteriorates the transmitter can add more redundancy to the matrices. Conventional error correction methods cannot be scaled so easily. Post processing of the reconstructed speech can further improve the quality and allow a decrease in the necessary overhead.

Because of the naturalness in the transform of the NPT, the speech sounds are much smoother compared to other techniques, such as frame repetition or silence insertion. These techniques require further processing to smooth the transition between existing and reconstructed frames. After NPT reconstruction one could use filtering, also, to further improve the quality of the speech. Since the NPT uses additions and subtractions, due to the structure of the Hadamard matrix, any effects of white noise errors will be significantly reduced. In other words, NPT has automatic low pass filtering effects. Furthermore, since the recovery uses the redundant information in the data, $\mathbf{P}_{init}$, the smoothing effects are minimal.

Replication techniques depend on signal stationarity to mask the replacement signal. Even though the SNR is quite low, it may not be correspondingly audible in stationary regions. In non-stationary regions, signal replication is quite audible. Signal stationarity is not guaranteed, especially in transitional regions. The NPT performs well in any class of speech because it depends on the inherent information in the surviving transformed packets to reproduce the missing samples.

NPT Quantization

The tests described in the previous section used unquantized NPT coefficients. The experiments in this section show that adequate quantization will not reduce the ability to perform high quality reconstruction of missing speech samples. These experiments use both a uniform quantizer and an optimal quantizer for the NPT-transformed samples. The last four rows in each of 144 16×16 matrices (4.6 seconds) of a one-sentence speech segment were removed. The reconstruction algorithm used 64 samples for $\mathbf{P}_{init}$ to recover the missing samples for the 144 matrices in the file.

The uniform quantization implementation divided the signal into 2^R equally spaced levels, where R is the number of bits available for quantization. It then chose the level closest to the NPT coefficient as the quantized value. The reconstruction process used the quantized NPT coefficients to recreate the missing speech samples.

The effect of quantization on the individual NPT coefficients is considered first. Figure 6.8 shows the change in quantization noise as the number of bits per NPT coefficient increases. Quantization noise is measured using SNR on the ordinate. The number of bits per NPT coefficient R is shown on the abscissa. The top line, marked with circles, represents the maximum SNR due to quantization from the file. The middle line, marked with squares, represents the mean SNR due to quantization from the file, and the lower line, marked with X's, represents the minimum SNR due to quantization noise from the file.

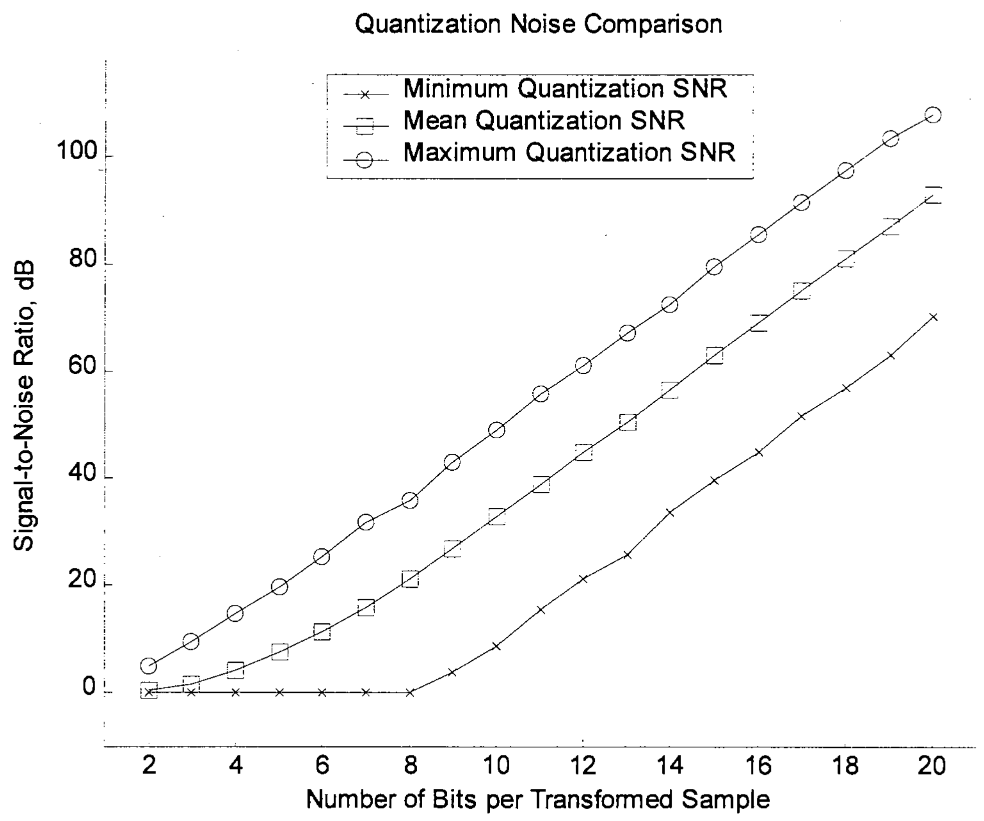


Figure 6.8. Uniform Quantization Noise Comparison

Signal-to-noise ratio increases nearly linearly as the number of bits increases, and the ~6dB/bit rule of thumb for uniform quantization is evident in this plot. The uncorrupted quantized NPT coefficients produce intelligible speech for this file when the rate is greater than 5 bits/coefficient, and the speech is indistinguishable from the original when the rate is greater than 13 bits/coefficient.

Listening evinces that below 6 bits the corrupted, reconstructed file is unintelligible and above 13 bits the corrupted reconstructed quantized speech and the corrupted reconstructed unquantized speech sound the same. Figure 6.9 and Figure 6.10 present noise measurements for NPT reconstruction using quantized coefficients. Figure 6.9 illustrates the change in the number of iterations required for reconstruction as the number of quantization levels increases. The ordinate contains the mean number of iterations and the abscissa shows the number of bits per NPT coefficient. The dotted line near the top of the plot shows the mean number of iterations (21.618) required for NPT reconstruction using unquantized coefficients.

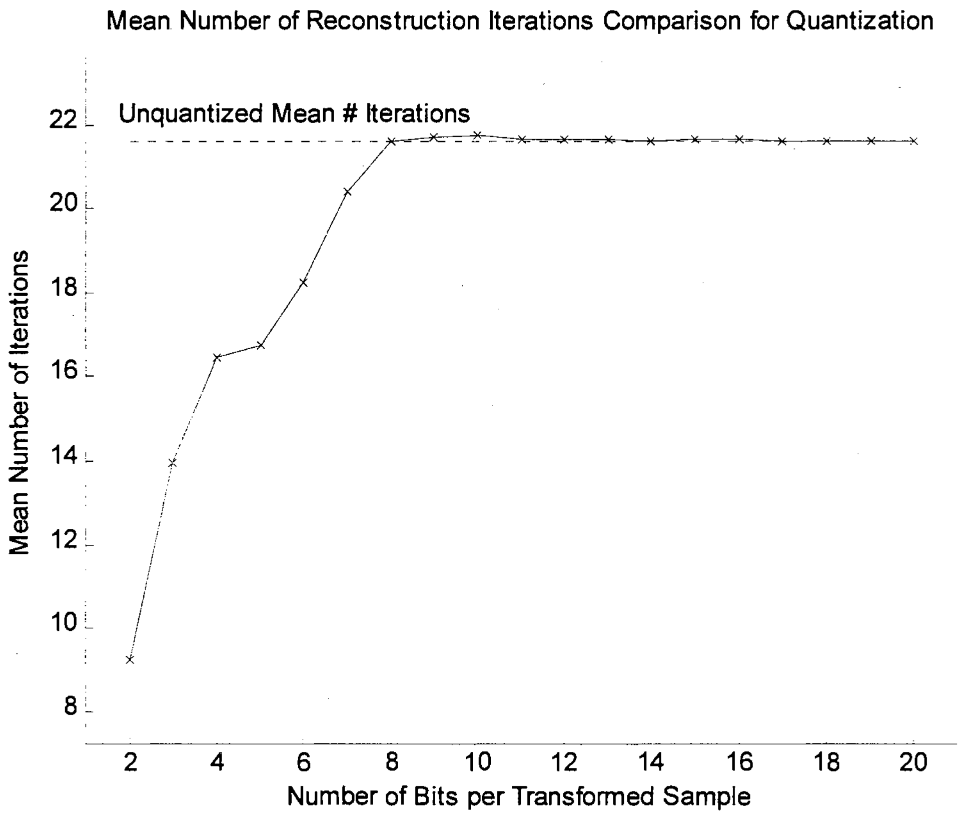


Figure 6.9. Mean Number of Reconstruction Iterations Comparison for Uniform Quantization

The number of iterations increases quickly until it reaches the unquantized measurement. Above seven bits the mean number of reconstruction iterations matches the number required by unquantized NPT. The coarsely quantized transformation matrices (less than 8 bits) provide fewer possibilities in the search space, and are therefore reconstructed more quickly.

Figure 6.10 compares the SNR between the original speech and the corrupted reconstructed speech as the number of quantization levels increases. The ordinate shows the SNR and the abscissa shows the number of bits per NPT coefficient. The top line, marked with circles, shows the maximum SNR for the file as the number of bits increases. The middle line, marked with squares, shows the mean SNR and the lower line, marked with X's, shows the minimum SNR for the file as the number of bits increases. For comparison, the plot also shows the maximum (81.207 dB), mean (60.092 dB), and minimum (32.690 dB) SNR for unquantized reconstruction.

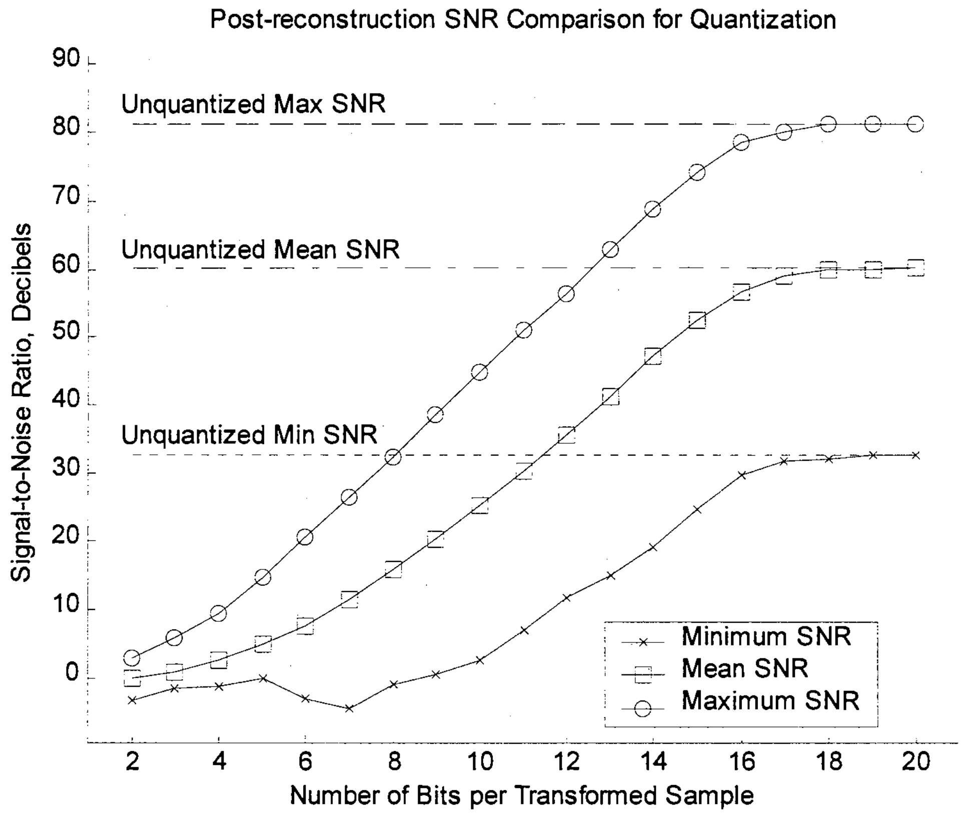


Figure 6.10. Post-reconstruction SNR Comparison for Uniform Quantization

Signal-to-noise ratio increases nearly linearly until it reaches the unquantized levels. The linear portion of the maximum SNR and minimum SNR curves exhibit the ~6dB/bit improvement expected from uniform quantization.

It is well known that uniform quantization is an inefficient technique for many data distributions. Experiments show that NPT coefficients for speech have a distribution similar to the generalized Gaussian distribution described in [92]. Analysis of Lloyd-Max quantizers ([19], [43]) with the generalized Gaussian distribution show that equal or better SNR is achieved with at least two fewer bits than uniform quantization. Listening to the output from the optimized quantizer reveals that only three bits per sample are required for intelligible corrupted, reconstructed speech. Above 11 bits the corrupted, reconstructed quantized speech and the corrupted, reconstructed unquantized speech sound the same.

NPT Reconstruction of Parametrically Coded Speech

The methods described previously work well for waveform coded speech because each sample of input speech produces one sample of coded speech. However, as discussed in Chapter 2, parametrically coded speech produces a frame of bits to represent a frame of speech samples–there is no one-to-one correspondence between speech samples and coded samples. It is possible, however, to take advantage of the low rate transmission capabilities of the parametric speech coders and the advantages of NPT reconstruction. Two different approaches were tested using parametric coders: parametric coding of the transformed speech and transformation of the parametrically coded speech.

Parametric coding of transformed speech

This experiment used the voice coding algorithm to quantize the transformed signal to a very low bit rate. Figure 6.11 illustrates the system used for this analysis.

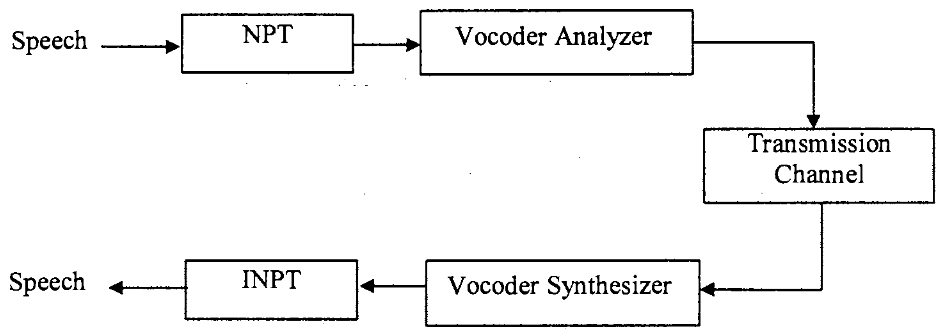


Figure 6.11. Parametric Coding of Transformed Speech

This system has some buffering issues since parametric coders operate on fixed size frames of speech (150-250 samples), and the NPT must use matrices with different fixed sizes ($2^n \times 2^n$) because of the restrictions on the Hadamard matrix. A circular buffer between the NPT/INPT process and the vocoder analyzer/synthesizer process will solve this problem. For this experiment the output from the NPT algorithm and the vocoder were saved onto the hard drive between steps so buffering was not an issue.

This system was tested using both the CELP and MELP vocoders described in Chapter 2. For both coders, the quality of speech with no channel errors was intelligible but noisy with α close to 1.0. However, as $\alpha \to 0$, the resulting speech grew unintelligible. Since reconstruction has been shown previously to work optimally when $\alpha \to 0$, this problem must be remedied before the system can be used. Further investigation showed that as α decreases and the Hadamard component grows stronger, the resulting transformed data contains signal magnitudes that exceed the magnitude of the input signal. The parametric vocoder analyzer is designed for data with defined magnitude bounds. When these bounds are exceeded, the analyzer cannot correctly code the signal.

Scaling the input signal may solve this problem. The scaling factor will depend on the bounds of the NPT as a function of α. Further study is required to define the NPT boundaries.

Transformation of Parametrically Coded Speech

The second experiment quantized the input signal with a low rate voice coding algorithm, and transformed the quantized vocoder parameters. Figure 6.12 shows the system used for this analysis.

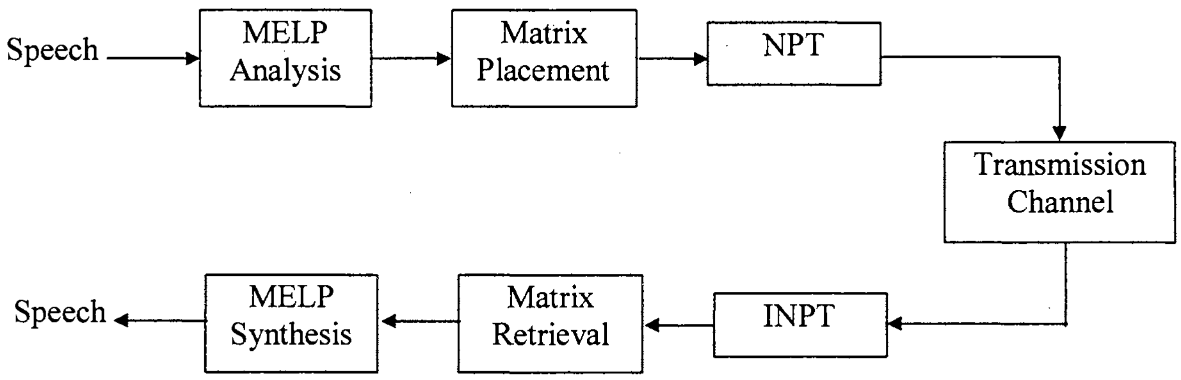


Figure 6.12. Transformation of Parametrically Coded Speech

Each 16×16 matrix contained two frames of quantized MELP parameters. The resulting matrices contained a large number of zeros for use in the reconstruction process. Since the transformation process is a floating-point operation, the matrices were quantized using a simple rounding procedure prior to transmission. Reconstruction worked very well with up to seven missing rows. The reconstructed speech sounded identical to the errorless speech. However, the speech became totally unintelligible when more than seven rows were lost. The reconstruction process requires considerably more iterations using this method.

Discussion

Experiments with several waveform coders indicate that NPT reconstruction is a high quality method for recreating missing speech samples using a modest amount of overhead. NPT reconstruction is not limited to waveform coders; it works well with parametric coders also. Since the amount of overhead is proportional to the rate of convergence and the quality of the resulting signal, NPT reconstruction is a scaleable error protection technique, and adequate quantization will not limit the quality of the reconstructed speech signal.

CHAPTER SEVEN

MARKOV CHAIN PREDICTION

Statistical analysis also has challenges with missing data. When researchers collect information for later statistical analysis, they store the results in matrices where the rows of the matrix represent different cases or respondents from the test, and the columns represent the individual responses or measurements. When statisticians wish to analyze a database of information collected from surveying a sample of the population, there may be missing data in some categories. This missing data can result from non-response to survey questions, equipment failure, or inability of respondents to express preferences. Analysis of all available data will produce the most accurate results, but the statistician must correctly deal with the cases that have incomplete information. Several references discuss solutions to this problem ([59], [22], [96], [46], [67], [15]), and they provide many different methods for performing statistical analysis with incomplete data. Imputation, or replacement, is a common approach for dealing with missing data in statistical analysis.

Statistical Imputation

Imputation-based procedures "fill in" the missing values. The resultant completed database is then analyzed by standard statistical methods as if there were no missing values. Statistical literature describes several different imputation procedures, including mean imputation, cold deck imputation, and hot deck imputation. Mean imputation replaces the missing values with the mean from the responding units. This is analogous to simple smoothing for speech parameters and would not produce speech with very high quality. Cold deck imputation replaces a missing value of an item by a constant value from an external source. This method is similar to the low-quality silence substitution procedure discussed previously. Many other imputation methods exist, but only hot deck imputation (HDI) is feasible for replacing missing speech data.

Hot deck imputation for statistical analysis replaces missing data with individual values drawn from similar responding units. It can involve very elaborate schemes for selecting units for imputation, i.e., determination of the units that are similar. If the parameters of a missing speech frame can be imputed, or replaced, with parameters that have occurred in a similar situation, a listener may not be able to distinguish the compensated synthesized speech from the original synthesized speech. For HDI to succeed on speech, a technique must be defined to determine what constitutes a similar situation. Parameters from past speech frames are the easiest data to acquire at a receiver. Information from the past can be used to determine future voice activity if speech can be described as a Markov chain.

Markov Chains

A discrete stochastic process is said to be a first-order Markov process if

$$\begin{gathered}\Pr\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\} = \\ \Pr\{X_{n+1} = x_{n+1} | X_n = x_n\}\end{gathered} \tag{7.1}$$

is true for all states $x_1, x_2, \ldots, x_n, x_{n+1} \in \aleph$, where X_{n+1} is the current state. That is, the current value of a Markov process can be determined probabilistically by evaluating the previous value. The first order Markov chain described by (7.1) is often represented with a transition matrix **P** that contains the probability of moving from one state at time *n* to another state at time *n+1*. The transition matrix elements, P_{ij}, represent the conditional probability that at time $n+1$ the system is in state *j* given at time *n* the system was in state *i*:

$$P_{ij} = \Pr\{X_{n+1} = j | X_n = i\} \quad i, j = 0,1,2,\ldots \tag{7.2}$$

where $P_{ij} \geq 0$ and $\sum_{j=0}^{\infty} P_{ij} = 1$ for $i = 0,1,2,\ldots$

A stochastic process on $\aleph$ is a Markov chain of order *m* if

$$\begin{aligned}\Pr\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\} = \\ \Pr\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_{n-m-1} = x_{n-m-1}\}\end{aligned} \tag{7.3}$$

for all *n>m* and all $x_1, x_2, \ldots, x_n, x_{n+1} \in \aleph$. That is, the current value of a Markov process of order *m* can be determined probabilistically by evaluating the previous *m* values.

Properties of Markov Chains

Several useful properties of Markov chains can be defined using the transition matrix defined by (7.2). If the P_{ij} depend only on the states and not on the time ($P_{ij} = p(j|i) \quad i, j = 0,1,2,\ldots$) then the conditional probabilities are *constant* or *stationary*. If the conditional probabilities are stationary and the P_{ij} are known for all states, then the process is completely defined by the transition matrix and the initial probability, *q*,

$$q(x) = \Pr\{X_0 = x\} \quad x = 0,1,\ldots. \tag{7.4}$$

Texts devoted to Markov processes ([4] and [56]) describe other properties of Markov chains:

- A state i is called *essential* if the existence of a positive integer v and a state j ($\neq i$) such that $P_{ij}^{(v)} > 0$ implies the existence of a positive integer w such that $P_{ji}^{(w)} > 0$. A state that is not essential is called *inessential.*

- Let i and j be two states of a Markov chain; then i and j are said to *communicate* if there exist a $v \geq 1$ and a $w \geq 1$ such that $P_{ij}^{(v)} > 0$ and $P_{ji}^{(w)} > 0$. If i and j communicate, it is written $i \rightarrow j$.

- A set of states $T \in \Im$ is called *closed* if no one-step transition is possible from any state in T to any state in $\Im - T$, the complement of the set T ($P_{ij} = 0$ for $i \in T, j \in \Im - T$). If T contains only one state, the state is called an *absorbing state.* If the state space contains two or more closed sets, the chain is called *decomposable* or *reducible.*

- If $n_{ij}(t)$ denotes the number of times $x_{t-1} = i$ and $x_t = j$, and $n_{ij} = \sum_t n_{ij}(t)$ then the *micro maximum likelihood estimator* is $P_{ij} = n_{ij} \Big/ \sum_j n_{ij} \geq 0$. This estimate is consistent but it is not generally unbiased. However, as the sample size increases, the bias tends to zero.

Not all these properties are necessary for missing frame compensation using Markov chain prediction.

Chapman-Kolmogorov Equation for Step Transitions

A prediction of the future state at time X_{k+m} given only the current value at time X_k is contained in the m-step transition probabilities given by

$$P_{ij}^{(m,m+k)} = \Pr\{X_{m+k} = j | X_k = i\}. \tag{7.5}$$

The step probability $P_{ij}^{(m,m+k)}$ is often represented by $P_{ij}^{(m)}$. All possible paths (sequences of states) between X_k and X_{m+k} must be considered to determine the step probabilities. A recursive procedure for calculating the m-step transition probabilities is given by the Chapman-Kolmogorov equation. The discrete form of this equation is

$$P_{ij}^{(m,m+k)} = \sum_{k \in S} P_{ir}^{(m,r)} P_{rj}^{(r,m+k)}, \; m < r < m+n. \tag{7.6}$$

Letting $\mathbf{P}^{(m)}$ represent the m-step transition matrix, the Chapman-Kolmogorov equation for the process's transition matrix can be written by

$$\mathbf{P}^{(m)} = \mathbf{P}^{m}. \tag{7.7}$$

That is, the m-step state transition matrix is simply the one-step state transition matrix raised to the mth power.

Speech Parameters and Markov Chains

If speech is considered a Markov process of order m, then a set of hot deck similar responding units can be found by evaluating probabilities of sequential vocoder parameters, i.e., the Markov chain probabilities. This method is similar to the waveform replication techniques discussed in Chapter Five. However, waveform replication uses a history buffer containing less than a half-second of speech to determine the best substitution data. A Markov chain method can use statistics of voice coding parameters collected off-line on large amounts of speech. Statistics collected from ample data will provide a much better estimate of the behavior of speech parameters, and this behavior can be used to determine the most likely missing parameters. The receiver will have the parameter statistics available so when a speech frame is lost, it can use the collected statistical information to determine the most likely parameters.

For this notion to succeed, the conditional probabilities, P_{ij}, must be stationary and the initial state, $q(x)$, defined by (7.4), must be known. It is facile to assume that all speech begins in silence, so the initial state consists of vocoder parameters representing silence. It is also reasonable to assume that the transition probabilities are not dependent on time since a speech signal is considered statistically stationary on a short time basis. If there are any closed sets in the state space, these may be employed to more efficiently predict missing parameters.

Prior experiments with voice coders for this research showed that the quality of reconstructed speech is more degraded by inaccurate pitch and gain than by inaccuracies in the other parameters. If accurate values for pitch and gain can be determined, the other parameters can be calculated using simple methods, such as smoothing or replication. Increased prediction accuracy may be obtained by including additional vocoder parameters (e.g., spectrum) at the expense of increased complexity.

Examination of approximately 21 seconds of speech processed through MELP (952 frames) indicated that the individual transition probabilities for pitch and gain resembled a second order Markov process. Figure 7.1 illustrates the second order pitch transitions for MELP pitch. The current pitch value is shown on the abscissa and the percentage of transitions to that value is shown on the ordinate. In this figure, both of the previous pitch values were quantized to the unvoiced state (represented with pitch=0). Over half of the frames with this transition characteristic remained in the unvoiced state; however, there were several other transitions to other pitch values.

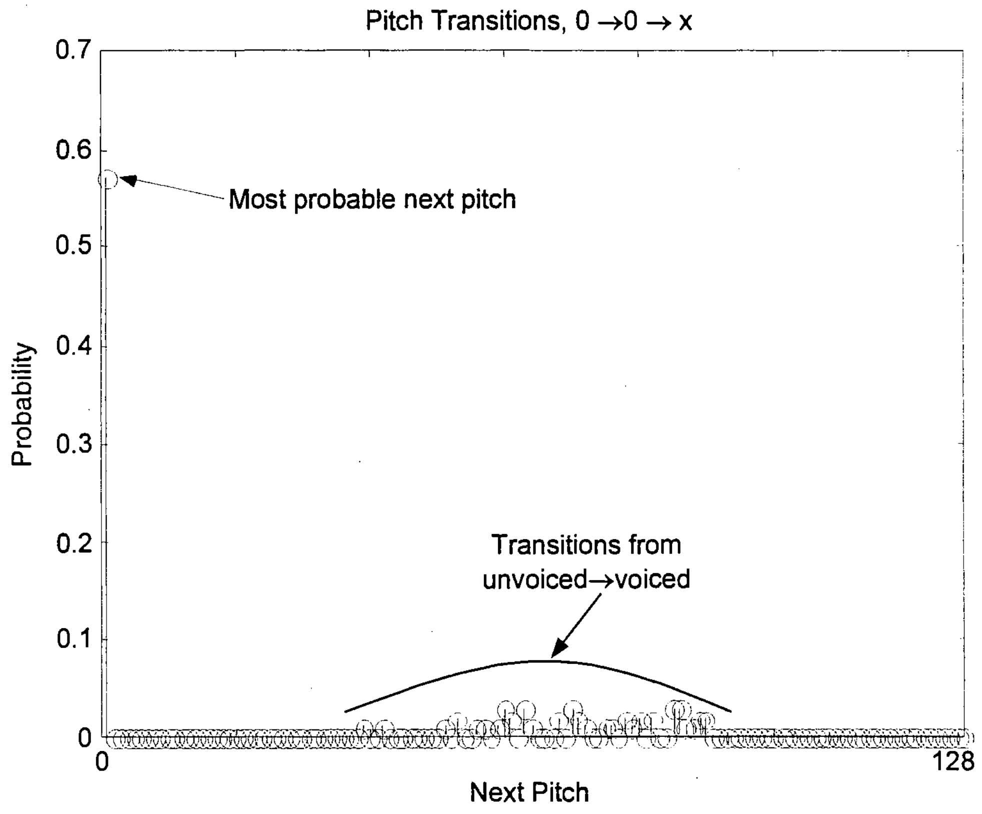


Figure 7.1 Probability of 2nd order pitch transitions

The statistics from the speech data were used to create a lookup table based on the highest probability transitions for pitch and gain (e.g., $0 \rightarrow 0 \rightarrow 0$ for the pitch data in Figure 7.1). Markov Chain Prediction (MCP) was tested on a short speech segment by determining pitch and gain values in missing frames via table lookup. All other parameters were simply repeated. The resulting speech from this method was not acceptable. Evaluation of the reconstructed speech indicated that errors occurred because the pitch parameter also contains voicing information. Several of the most probable speech transitions were voiced→unvoiced or unvoiced→unvoiced; however, some transitions did not follow the majority and transitioned to voiced (an example of this can be seen between 50 and 90 on the abscissa in Figure 7.1). These incorrect pitch/voice transitions were the source of the quality degradation. This indicates that either pitch may need to be paired with another parameter (such as gain) to create a state, or that pitch should be modeled as a higher order Markov chain.

A 21-second speech file was used to count the number of different pitch-gain pairs. This short file contained 428 different pitch-gain states, and over half of these states (239) only occurred once. Analysis of the first-order transition probabilities of the states showed that most of the states had equiprobable transitions. This data indicated that higher-order Markov chains should represent the pitch-gain state transitions.

Higher Order Markov Chains and the Mixture Transition Distribution

The calculation of the transition probabilities using the micro maximum likelihood estimator for an *m*th order Markov chain with *S* states requires $S^{m+1} + S^m$ memory locations (S^{m+1} locations to store the individual *m*th-order state-to-state transition counts and S^m locations to store the number of times each state is visited). Once the transition probabilities have been calculated, S^m memory locations are required to store the state transition table for lookup.

The MELP encoder logarithmically quantizes pitch to seven bits. Of the 128 possible values for pitch, only 100 represent valid pitch values. (Quantized pitch values with Hamming weights of one or two are reserved for error protection.) Representing pitch as a second-order Markov chain requires approximately 10^6 memory locations to calculate the transition probabilities. The MELP encoder uniformly quantizes $\log(gain)$ to five bits. Representing the pitch-gain state as a second-order Markov chain requires over 78×10^6 memory locations. Calculating the transition probabilities for these parameters as 3rd order Markov chains requires over 10^8 memory locations for pitch alone and over 33×10^9 memory locations for the pitch-gain state. The storage requirements for a second-order chain are formidable, but the storage requirements for a 3rd order chain are excessive (and prohibitive for many computers). Prior to evaluation of higher order Markov chains for states of one or more vocoder parameters, a low-memory method of creating the lookup table is needed.

King uses Markov chain random processes (MCRP) based on step probabilities ([46]) to model speech. He creates an MCRP with the same power spectrum as the data. The resulting transition matrix is used to generate an MCRP time series that approximates the speech. The advantage of using MCRP's to predict the speech is that step probability transition matrices do not require large amounts of memory and can be calculated using the Chapman-Kolmogorov equations. However, the nature of speech indicates that important information may be contained in the path between states, not just the end points.

In [76] and [77], Raftery describes a low-memory method for determining the transition probabilities for high-order Markov chains using the mixture transition distribution (MTD) shown in (7.8).

$$\begin{aligned}P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_{n-m-1} = x_{n-m-1}) &= p(i_0 | i_1, \ldots, i_m) \\ &= \sum_{j=1}^{m} \varphi_j q(i_0 | i_j)\end{aligned} \tag{7.8}$$

where φ_j is the *j*th of *m* lag weights, and $q(i_0|i_j)$ is the transition probability from state i_j, the *j*th state, to state i_0, the current state, in the Markov chain. Raftery's papers do not specify approaches for calculating $\{\varphi\}$, the lag weights, or $Q = \{q(i|j)\}$, the transition matrix. He does define the properties for the parameters, however. The transition matrix is a column stochastic matrix satisfying

$$q(i|j) \geq 0 \text{ and } \sum_{r=1}^{S} q(r|j) = 1, j = 1, \ldots, S. \tag{7.9}$$

A row stochastic matrix is permissible if appropriate indexing is used. The conditional probability $p(i_0|i_1, \ldots, i_m)$ must be ≥ 0, and the lag weights must sum to 1 (there is no restriction that $0 \leq \varphi_j \leq 1$, although this considerably simplifies the procedure for finding the optimal **Q** and φ's). The MTD reduces the number of independent parameters to $S(S-1) + m - 1$.

Raftery suggests that maximum likelihood estimation be used to find **Q** and $\{\varphi\}$ using

$$\log L = \sum_{n(i_0, i_1, \ldots, i_m)} n(i_0, i_1, \ldots, i_m) \log p(i_0 | i_1, \ldots, i_m). \tag{7.10}$$

The number of times that the sequence $i_m \rightarrow i_{m-1} \rightarrow \ldots \rightarrow i_0$ occurs is represented by $n(i_0, i_1, \ldots, i_m)$. The sum is over all $i_0, i_1, \ldots, i_m$ with $n(i_0, i_1, \ldots, i_m) > 0$. For models with high values of *m* or *S*, the number of potential patterns, $(i_0, i_1, \ldots, i_m)$, can be extremely large. Raftery suggests labeling patterns as

$$i = 1 + \sum_{j=0}^{m} (i_j - 1) S^{m+1-j} . \qquad (7.11)$$

The MTD method may not provide mathematically accurate transition probabilities; however, the location of most probable next states is more important than accuracy of specific probabilities for this application.

Estimation of Mixture Transition Distribution Parameters

Because there is no algebraic solution to the maximization of the log-likelihood of the MTD model, Berchtold ([5]) has developed a procedure to calculate the lag weights and transition matrix that produce the highest log-likelihood. His process requires an initialization step and an optimization step.

Initialization of Mixture Transition Distribution Parameters

Selection of the proper initial parameters insures that the optimization process will converge to the global maximum. Incorrect initial parameters may cause the optimization process to converge to local maxima.

The transition matrix, **Q**, of the MTD model simultaneously represents all step probabilities of the data, i.e., each of the following situations is represented:

$$\begin{gathered} P(X_t = j | X_{t-1} = i) \\ \vdots \\ P(X_t = j | X_{t-m} = i) \end{gathered} . \tag{7.12}$$

The MTD lag weight parameters, $\varphi_1,\ldots,\varphi_m$, scale the transition probability, q_{ij}, according to the lag.

Berchtold assumes that the initial lag weights and transition matrix are related to the strength of the relations or association between each lagged period and the present. To determine the initial values, he first forms a cross-table containing the number of events of the form $X_{t-g} = i_g$ and $X_t = i_0$, where g is the lag between states. The cross-table takes the form

$$C_g = \begin{pmatrix} C_g(1,1) & \cdots & C_g(1,S) \\ \vdots & \ddots & \vdots \\ C_g(S,1) & \cdots & C_g(S,S) \end{pmatrix}, \tag{7.13}$$

where $C_g(p,q)$ is the number of times $X_{t-g} = state\#p$ and $X_t = state\#q$. There are m separate cross-tables, one for each lag.

Berchtold uses Theil's association measure ([93]), u, to compute the strength of the relation between each lag (i.e. m delays) and the present. Theil's association measure is based on Shannon's entropy and takes a value on the space [0,1]. A value of 0 indicates independence between the rows and columns of the cross-table and a value of 1 indicates perfect association. Theil's u is calculated by

$$u_g = \frac{\sum_{i=1}^{S}\sum_{j=1}^{S} C_g(i,j)\log_2\left(\frac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)}{\sum_{j=1}^{S} C_g(\cdot,j)\log_2\left(\frac{C_g(\cdot,j)}{TC_g}\right)}, \tag{7.14}$$

where

$$C_g(i,\cdot) = \sum_{j=1}^{S} C_g(i,j) \tag{7.15}$$

is a row sum,

$$C_g(\cdot,j) = \sum_{i=1}^{S} C_g(i,j) \tag{7.16}$$

is a column sum, and

$$TC_g = \sum_{i=1}^{S}\sum_{j=1}^{S} C_g(i,j) \tag{7.17}$$

is a matrix sum.

The initial lag weight vector is calculated by normalizing the association measure,

$$\varphi_g = \frac{u_g}{\sum_{k=1}^{m} u_k}, \tag{7.18}$$

and the initial transition matrix is the *k*th order transition matrix,

$$Q_k \text{ where } k = \arg\max_{g=1,\ldots,m} u_g \tag{7.19}$$

$$Q_g = [P(X_t = j | X_{t-g} = i_g)]$$
$$= \begin{bmatrix} \frac{C_g(1,1)}{C_g(1,\cdot)} & \cdots & \frac{C_g(1,S)}{C_g(1,\cdot)} \\ \vdots & \ddots & \vdots \\ \frac{C_g(S,1)}{C_g(S,\cdot)} & \cdots & \frac{C_g(S,S)}{C_g(S,\cdot)} \end{bmatrix}. \tag{7.20}$$

These initial parameters are guaranteed to cause the optimization process to converge to a global maximum.

Optimization of Mixture Transition Distribution Parameters

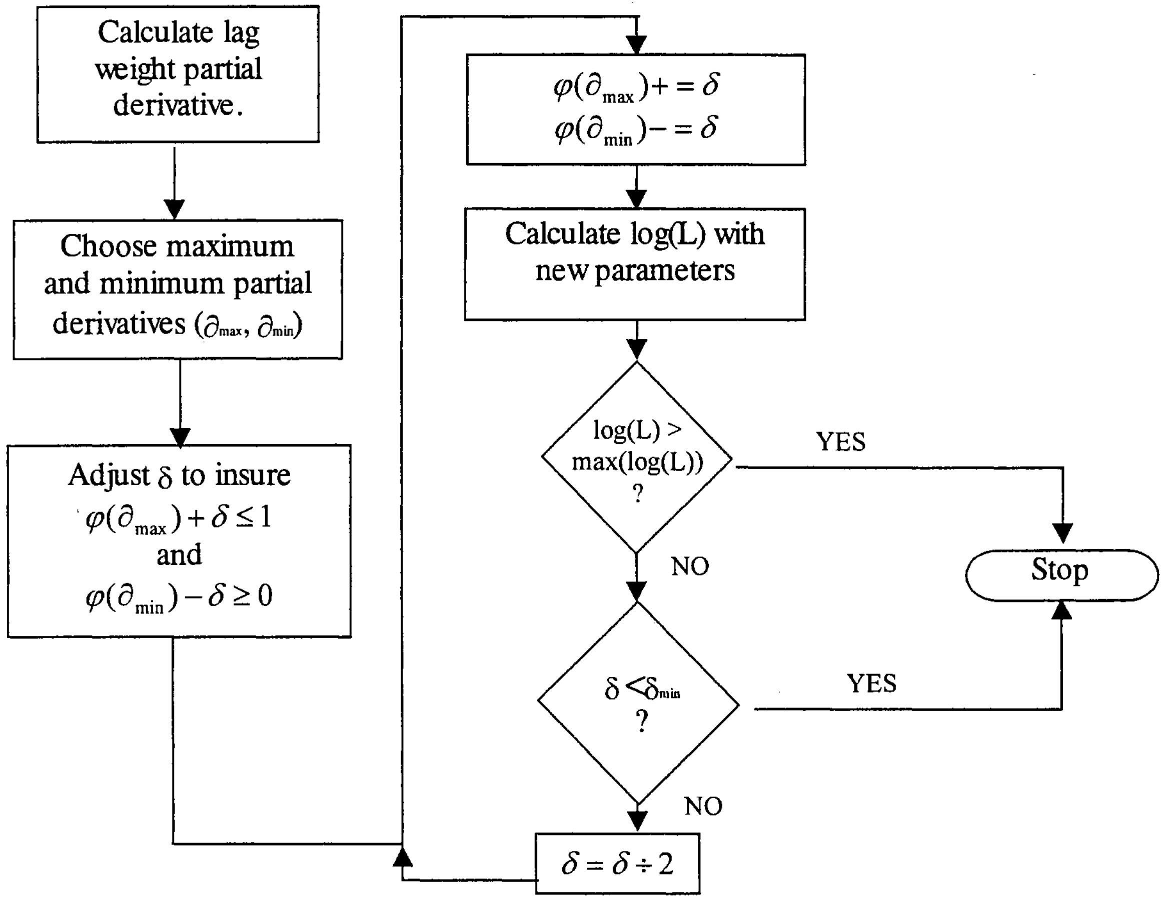
Berchtold's optimization process iteratively modifies the lag weights and transition matrix until the increase in log-likelihood falls below a threshold. The set of MTD parameters is divided into *S+1* subsets: *S* rows of the transition matrix and the vector of *m* lag weights. Each of these subsets sum to one. The optimization process analyzes the partial derivatives for each of these subsets to determine which elements within the subsets to modify. The partial derivative with respect to the *k*th lag parameter is calculated by

$$\frac{\partial \log(L)}{\partial \varphi_k} = \sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m) \frac{q_{i_k i_0}}{p(i_0|i_1,\ldots,i_m)}, \tag{7.21}$$

and the partial derivatives with respect to the rows of the transition matrix are calculated by

$$\frac{\partial \log(L)}{\partial q_{i_k i_0}} = \sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m) \frac{\varphi_k}{p(i_0|i_1,\ldots,i_m)}. \tag{7.22}$$

The partial derivative gives a measure of the local impact produced by the change of one parameter upon the log-likelihood. An increase or decrease in one parameter requires a corresponding decrease or increase of another parameter to maintain the constant sum. The log-likelihood is improved by increasing the parameter corresponding to the largest derivative and decreasing the parameter corresponding to the smallest derivative. The algorithm changes both parameters by the same amount, δ. Modification of the lag weights follows the iterative procedure shown in Figure 7.2.



Calculate lag weight partial derivative.
Choose maximum and minimum partial derivatives ($\partial_{\max}$, $\partial_{\min}$)
Adjust δ to insure $\varphi(\partial_{\max}) + \delta \leq 1$ and $\varphi(\partial_{\min}) - \delta \geq 0$
$\varphi(\partial_{\max}) += \delta$
$\varphi(\partial_{\min}) -= \delta$
Calculate log(L) with new parameters
log(L) > max(log(L)) ?
YES
NO
Stop
$\delta < \delta_{\min}$?
YES
NO
$\delta = \delta \div 2$

Figure 7.2 Calculation of Optimal Lag Weights

The optimization process improves the transition matrix by following the procedure illustrated in Figure 7.2 on each individual row. During each iteration of the optimization process, all $S+1$ MTD sets are individually evaluated and improved, if improvement is possible. After all sets are evaluated, the optimization process calculates the improved log-likelihood. When the increase in log-likelihood falls below a threshold, the optimization process ends. The overall optimization process is shown in Figure 7.3.

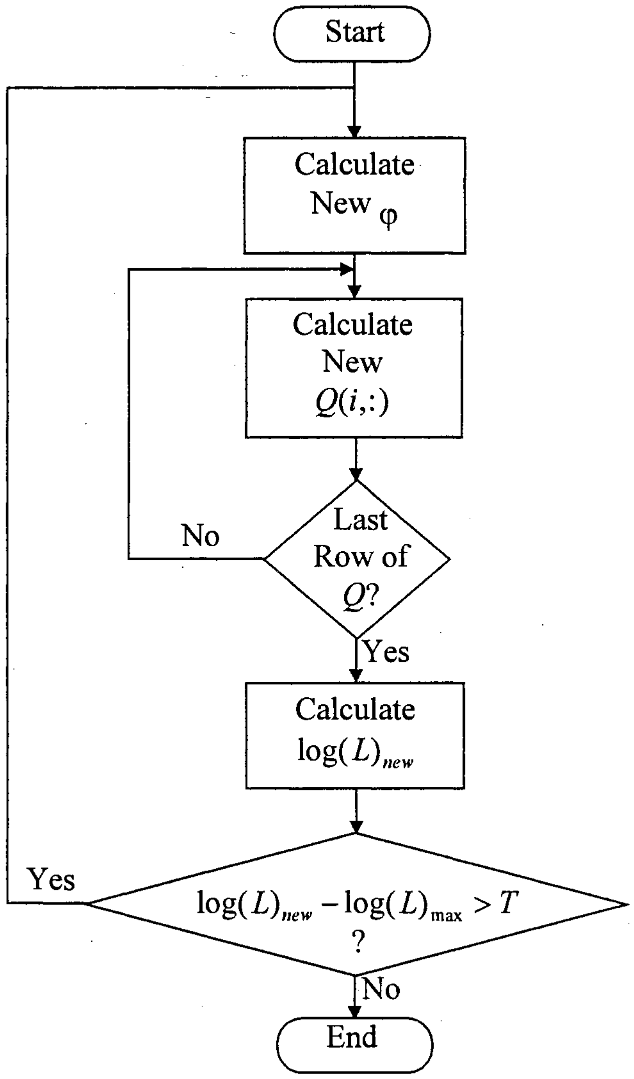


Figure 7.3 Overall Optimization Process

Figure 7.3 does not detail all optimization processing. However, the operational flow is clearly evident.

Mixture Transition Distribution Implementation

Markov chain prediction with the MTD was tested with two vocoders: MIL-STD-3005 MELP and GSM 6.10. Since previous experiments in this research demonstrated that independent prediction of vocoder parameters performed badly, the states of the Markov chain contained combinations of vocoder parameters, e.g., all triads of pitch, gain, and spectrum parameters that occur in the training data. Selecting the vocoder parameters to include in the state is a challenging problem. If all of the parameters are included, the amount of memory required to calculate the transition table and lag weights is prohibitive for most computers (approximately 9×10^{15} combinations for the MELP algorithm). Even limiting the number of parameters to three or four can result in too many combinations for a desktop computer to manage, depending on the number of quantized values for each parameter.

It is likely that not all combinations of vocoder parameters will occur. However, as the number of possible parameter combinations increases, the amount of training data required to find even the more probable combinations also increases. If the training routines for calculating the optimal transition matrix and lag weights do not have sufficient data, then the operational data will contain parameter combinations that didn't occur in the training data thereby disabling prediction and decreasing the quality of the reconstructed speech.

MCP with Mixed Excitation Linear Prediction

The first test of this system used the MELP 2400 bit/second MIL-STD-3005 algorithm described in Chapter Two. The parameters comprising the state were chosen from the 24 most important parameters published in the standard ([65]). These bits are shown in Table 7.1.

Parameter	Bits	Voicing
LSF1	1-7	Voiced and Unvoiced
LSF2	3-6	Voiced and Unvoiced
Gain2	2-5	Voice and Unvoiced
Pitch/Voicing	2-7	Voiced and Unvoiced
Bandpass Voicing	4	Voiced
Aperiodic Flag	1	Voiced
Sync	1	Voiced and Unvoiced

Table 7.1 Twenty-four Most Important MELP Bits

The first column lists the MELP parameters, and the second column provides the bits from those parameters that are considered among the most important. The third column tells which voicing state requires the parameter. If all these parameters except sync are used to define a state, the number of possible combinations is 8,388,608–requiring over 1000 terabytes of memory to train and over 200 terabytes of memory to store the transition matrix once it is calculated. Informal listening indicated that if all other parameters from missing MELP frames are repeated, having the correct seven-bit pitch/voicing and five-bit second-gain parameters (i.e. pitch/voicing, gain2) had the largest effect on quality, so the states represent this parameter pair.

MELP Mixture Transition Distribution Parameters

The state look-up matrix, which matches pitch-gain pairs to state numbers, is ordered on quantized pitch values first, then on quantized gain values i.e., $(P=0,G=0),(P=0,G=1),(P=0,G=2),\ldots,(P=127,G=31)$. The training data for calculating the optimal transition matrix and lag weights consisted of 25 approximately equal sized speech files from 25 different talkers (12 females, 13 males) totaling 19.6 minutes (52,293 MELP frames). The training data contained 2,209 of the possible 4096 pitch-gain states. The ordered pitch-gain pairs associated with each state are shown in Figure 7.4.

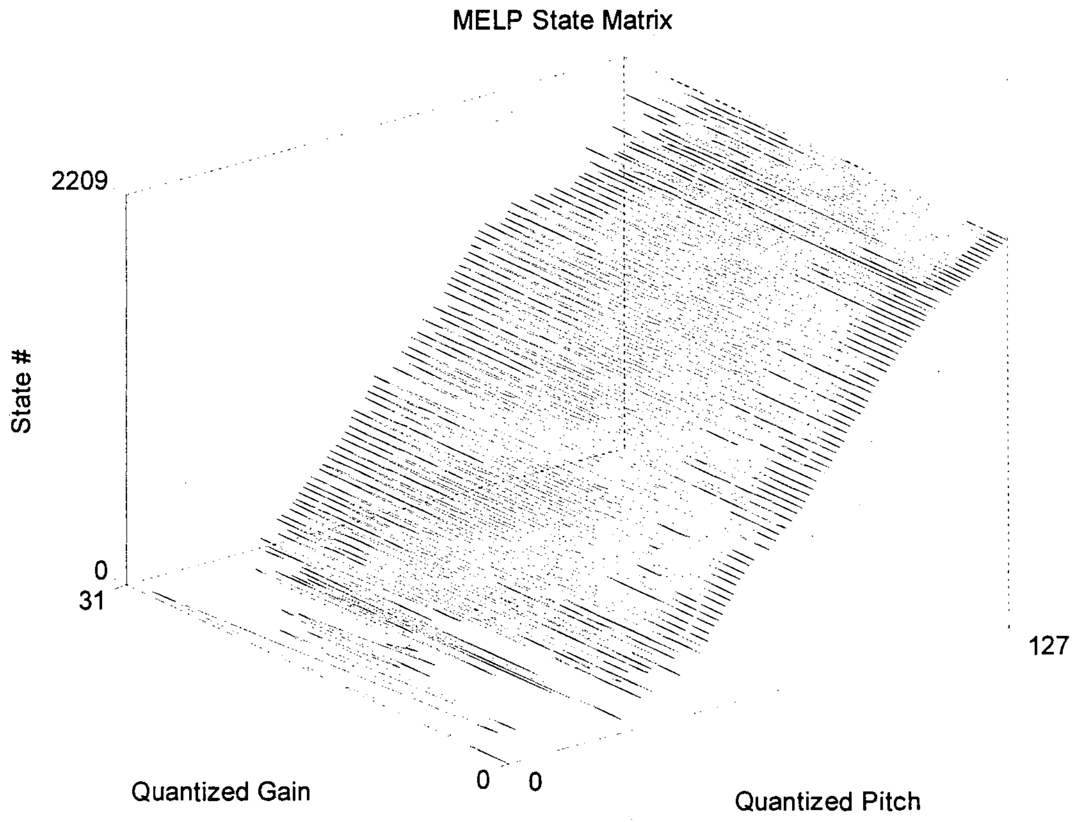


Figure 7.4 MELP State Matrix

The quantized pitch-gain pair values are shown on the abscissa and ordinate, with the state number indicated by the height. Gaps in the plot show quantized pitch values with Hamming weights of one or two bits that MELP does not use. It is evident in this plot that the lower, voiced pitches did not occur with all possible gains.

Berchtold's initialization and optimization procedures were performed for a second-order Markov chain. The training data contained 34,591 second-order transitions of the pitch-gain states with a slightly higher association for first-order transitions ($\varphi_{init} = [0.52, 0.48]$). The optimization procedure analyzed the lag weights and 2209 rows of the transition matrix 28 times to produce the optimal MTD parameters. Figure 7.5 shows the changes in log-likelihood and lag weights throughout the optimization process.

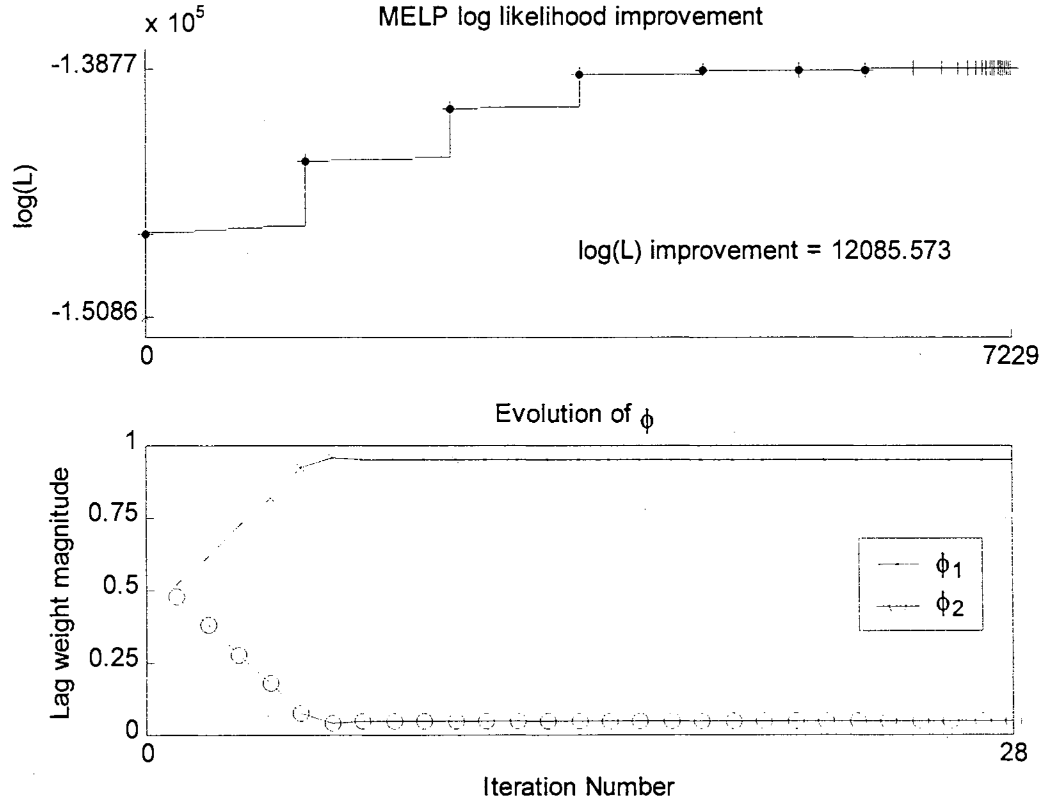


Figure 7.5 MELP MTD Optimization

The top plot shows the improvement in log-likelihood. The initial log-likelihood is denoted as an X, the log-likelihood measurements due to changes in lag weights are marked with dots, and the locations where the first row of the transition matrix is modified are shown with plus-marks. The bottom plot illustrates the changes in lag weights to improve the log-likelihood. The values of the first lag weight, φ_1, are denoted by X's and the values of the second lag weight, φ_2, are marked with O's. Over 7200 changes were made to the MTD parameters in 28 iterations to provide an increase of 12,085 in log-likelihood.

The sharp increases prior to the first four dots on the top plot illustrate how the initial lag weight alterations yield substantial improvements in log-likelihood. As the lag weights converge to the optimal values, the magnitude of the log-likelihood improvement decreases. The decreasing slope of the line between lag weight changes indicates that the transition matrix exhibits the same trend of diminishing improvement as it converges to the optimal value. The decreasing distance between successive plusses on the log-likelihood line depicts the abating number of rows that are changed each time the optimization process evaluates the transition matrix as the individual rows approach their optimal values.

The bottom plot shows that $\varphi_1 \to 1$ and $\varphi_2 \to 0$ to reach their optimal values. The first four changes are more substantial than the later ones, and by the seventh iteration, the lag weights have reached their optimal values. The lag weights remain unchanged through the final 75% of the iterations.

The optimization procedure produced second-order lag weights, [0.953 0.047]. The lag weights indicate a nearly first-order Markov chain. Figure 7.6 shows the transition matrix produced with the optimization process..

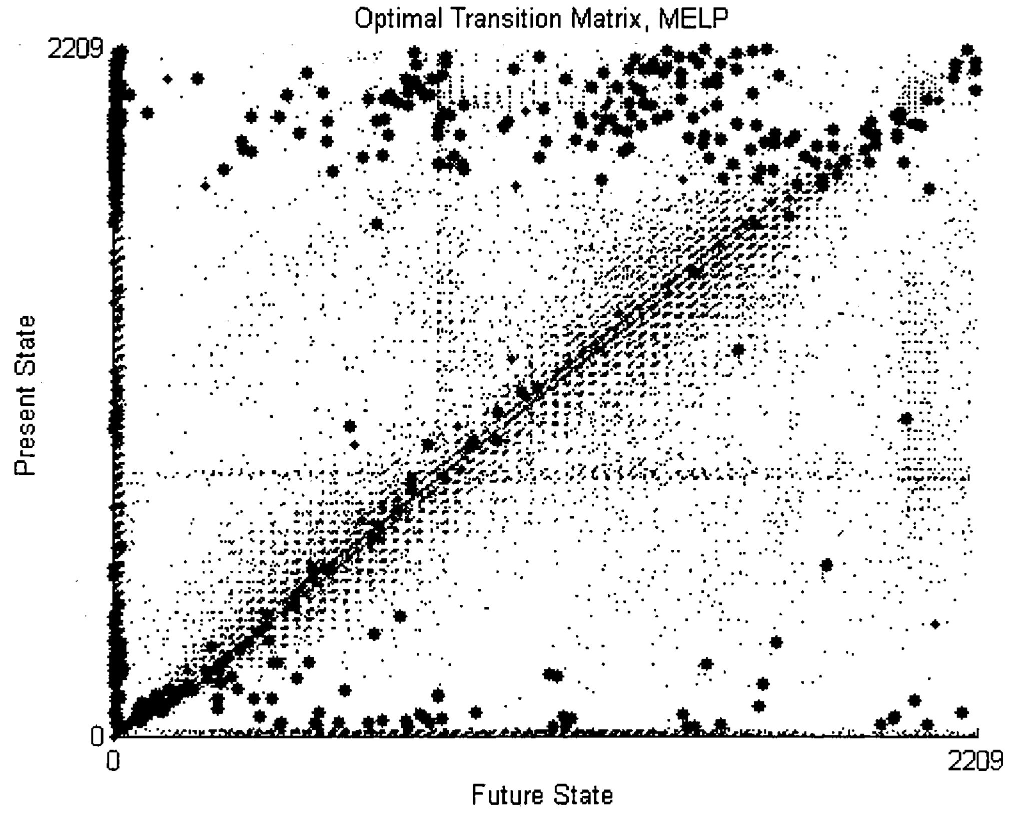


Figure 7.6 MELP Optimal Transition Matrix

This figure uses three different sized dots to represent the magnitude of the probabilities in the transition matrix. The smallest dots mark the 20,803 locations where $0 < Q \leq 0.5$. The medium dots represent the 112 locations where $0.5 < Q < 1.0$, and the largest dots represent the 409 locations where $Q = 1.0$.

Only 0.44% of the transition matrix is non-zero. This indicates the potential for stronger prediction than if the probabilities were spread among more matrix elements. A linear concentration of non-zero probabilities is evident along the diagonal, left edge, and bottom edge of the transition matrix. The heavy weighting of the first lag weight, indicating a strong association for first order transitions, must be considered when analyzing the structure in the transition matrix. The diagonal concentration indicates the tendency of the final state to repeat, i.e., if the transition is $s_a \rightarrow s_b$, the tendency is for the following state to remain s_b.

Knowledge of the pitch and gain values comprising the individual states is required to analyze the probability concentrations along the left and bottom edges. The pitch value of the first 29 states is zero, which synthesizes unvoiced speech in the receiver. The gain value of the first 29 states increases from zero to 29, which synthesizes logarithmically increasing energy in the receiver. The first state, therefore, represents unvoiced speech with zero energy, i.e., silence. Correspondingly, the first column represents the probabilities when the next state is silence, and the first row represents the probabilities when the current state is silence. The concentrations along the left and bottom edge of the plot show that many states transition to and from silence.

The optimal transition matrix has a large number rows (18.5%) with only one probability ($=1.0$). These rows in the initial transition matrix contained one probability, also. The training data showed a larger association for first order transitions, so the initial transition matrix is the transition matrix of the first order Markov chain, according to (7.19) and (7.20). This suggests two possibilities: 1) these state transitions occurred only once, or 2) these transitions always ended in the same state. If the former possibility is true, then this transition is rare if sufficient data was used to train the initial parameters. If the latter possibility is true, then this transition can be predicted with high confidence. The first-order cross-table (used to calculate Q_{init}) for this implementation contained 404 states with only one transition, indicating that most of the rows with only one probability were the consequence of unique state transitions.

MELP Mixture Transition Distribution Performance

When the receiver detected a missing frame, it calculated the probability of transitioning to each of the 2209 states from the previous two states using (7.8). If the highest MTD conditional probability exceeded a minimum threshold, the algorithm chose the state with this probability. It then used the pitch and gain associated with the selected state to synthesize the missing frame, and all other parameters were repeated. If the highest calculated MTD probability did not exceed the threshold, then the algorithm repeated all vocoder parameters from the previous frame, since frame repetition sounds better than poorly predicted speech. If the prediction algorithm encountered a pitch-gain pair that was not represented in the state matrix, it used frame repetition. When several sequential frames were missing, the receiver used the same procedure for each missing frame i.e., it used the parameters of the previous predicted frames to predict the parameters for future frames. Most of the predicted states exceeded the minimum threshold, but occasionally a pitch-gain pair occurred in the test data that wasn't in the state matrix, and therefore could not be predicted.

MELP missing frame compensation using MTD was compared to frame repetition using both objective and subjective methods. The difference between missing speech compensated with the two techniques was quantified objectively with spectral and energy distance measures and evaluated subjectively by discussions with volunteer listeners. The evaluation process insured that all MCP test segments contained pitch-gain pairs that were predictable, i.e. all pairs in the test data also occurred in the training data, and the highest calculated MTD probability exceeded the threshold.

Spectrum and Energy Distance Measures

Although no simple objective distance measure adequately models human perception, the L_p norms have been shown to have some correlation with human hearing for measuring spectral differences ([34]). The L_p norms for the difference between two spectral models, $V(\theta)$, with sampling frequency F_S are defined by d_p where

$$(d_p)^p = \int_{-FS/2}^{FS/2} |V(\theta)|^p \frac{d\theta}{F_S} \tag{7.23}$$

Speech researchers commonly use the L_2 norm, or Euclidean distance measure, to measure spectral distance for speech. The discrete form of the L_2 norm for the difference between two sampled spectrums, $V(k)$, is

$$(d_2)^2 = \sum_{n=1}^{k} V^2(k) \tag{7.24}$$

This distance measure provides an assessment of the effects on spectrum, however, a time-domain measure is also beneficial for evaluating the two frame compensation techniques. A simple but perceptual time domain measurement is the energy in the speech segment. The speech energy is calculated using

$$E^2 = Y \times Y^T \tag{7.25}$$

where Y is a row vector containing the samples in the segment. The energy measurement is directly related to the gain parameter in the MCP states.

Voiced, unvoiced, background, and transitional segments of speech were randomly chosen for the distance measurements. The network simulator described in Chapter Four removed four frames of speech from the segments and replaced them using MCP and frame repetition. Figure 7.7 shows the speech segments used for the objective analysis.

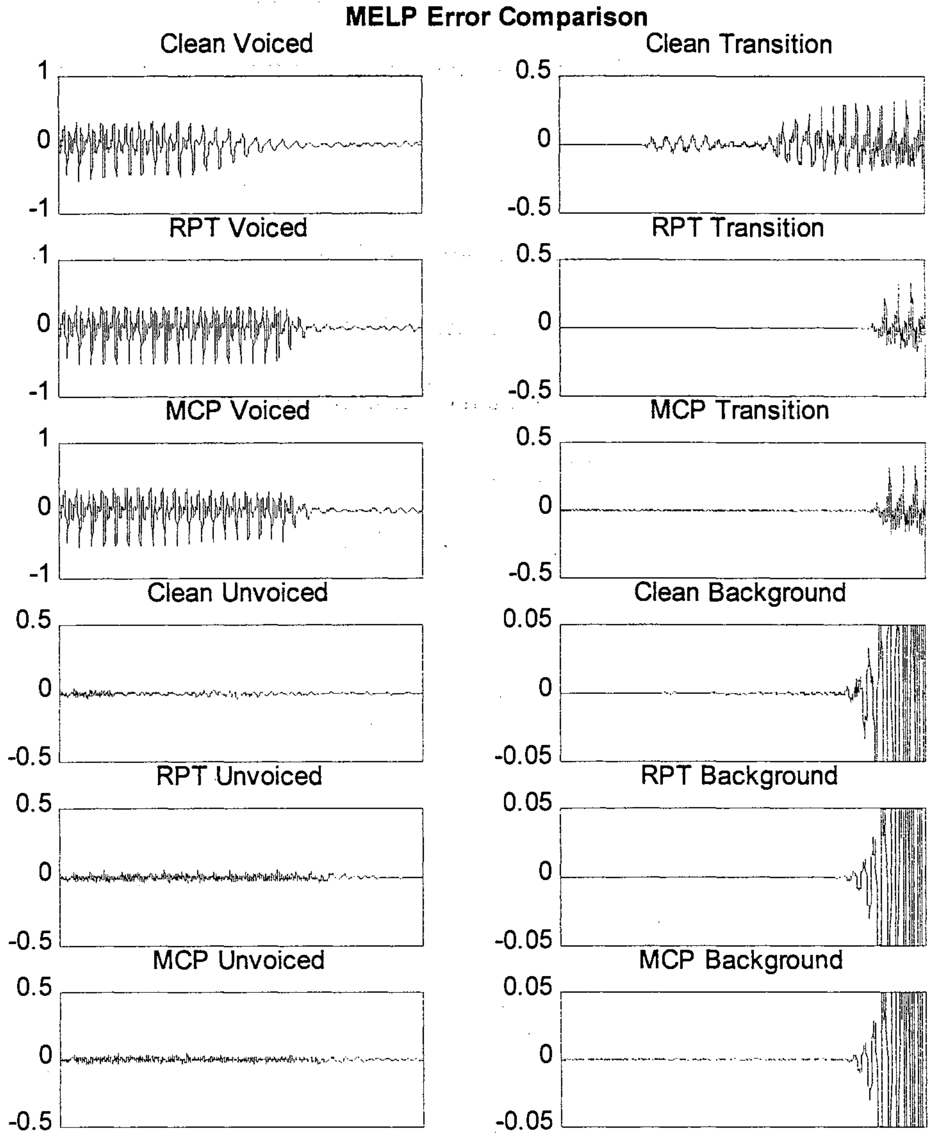


Figure 7.7 Speech used for MELP Objective Error Calculations

The labels of the twelve plots in Figure 7.7 denote their contents. The plots labeled "Clean" show MELP-synthesized speech with no missing speech. The plots labeled "RPT" show speech received with four missing frames and compensated with frame repetition. The plots labeled "MCP" show speech received with four missing frames and compensated with Markov Chain Prediction. Each plot shows a few frames of uncorrupted speech on each side of the compensated frames. There is not a substantial difference between the voiced speech plots for MCP and frame repetition, but the samples compensated using frame repetition have a constant frequency and constant magnitude. The effect of this trait is examined further in the discussion on the listening test. The unvoiced and transitional speech looks similar for both compensation techniques. Neither method is accurate for the transitional speech segment. Markov chain prediction increases the signal magnitude for the low-energy portion of the transitional speech and the background speech. Fortunately, the low magnitude of the signal keeps the perceptual effects of this error negligible.

The objective measurements calculated the distance between the clean speech and each of the compensation techniques for the speech segments shown in Figure 7.7. Table 7.2 shows the distance measures for these segments.

	MCP	RPT
L_2 on LP Spectrum		
Voice	31.231 dB	31.648 dB
Unvoiced	48.828 dB	48.601 dB
Transition	48.309 dB	58.698 dB
Background	56.620 dB	56.620 dB
Energy Error		
Voice	0.482 dB	0.678 dB
Unvoiced	1.138 dB	1.826 dB
Transition	3.021 dB	2.929 dB
Background	0.055 dB	0.055 dB

Table 7.2 Objective Measurements for MCP and Frame Repetition with MELP

Both measurements show that MCP performs better (i.e. the distance is smaller) when the missing segments occurred in voiced segments. When the segment is unvoiced, MCP has a lower error in energy, but frame repetition has a lower error in the spectrum. The spectral match is considerably better for MCP in the transitional speech segment, but the energy match is better for frame repetition in this segment. Both techniques perform similarly in the background segment for both measures.

PERCEPTUAL OPINION OF MELP MCP

Initial listening to speech with packet loss indicated that when less than four frames were lost, the parameter smoothing in the MELP algorithm caused lost speech compensated using MCP to be perceptibly indistinguishable from lost speech compensated using frame repetition. The network simulator removed four to eight sequential frames of MELP data (90-180 ms of speech) from a single word in a speech sentence for test files containing both male and female talkers. Careful listening to the affected words indicated that Markov chain prediction improved the quality of the corrupted speech in many of the missing speech segments. In most of the remaining segments, the difference in quality was indistinguishable. Figure 7.8 shows speech with frame loss reconstructed using both frame repetition and Markov chain prediction.

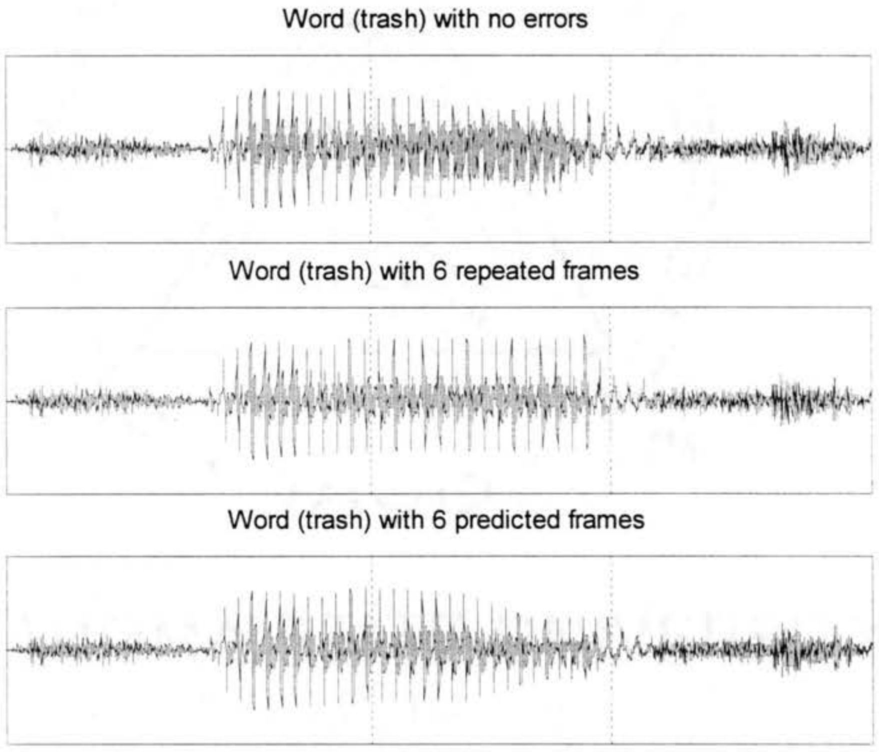


Figure 7.8 Comparison of Reconstructed Speech

The first plot of this figure shows the word "trash" when the MELP bits were transmitted across a network with no packet loss. The second plot shows the same speech segment when six frames are lost and reconstructed using frame repetition, and the third plot shows the speech segment when six frames are lost and reconstructed using Markov chain probabilities to predict the six missing MELP frames. The section of the plot containing the missing speech is marked with dashed lines in all three plots for comparison. It is clear that each frame in the second plot has constant energy until the voiced→unvoiced transition ("a" →"sh") is reached at the end of the vowel. MELP contains some smoothing that alters the magnitude at this point. The imputed speech frames in the third plot clearly do not maintain constant energy. Although it is not evident visually, the repeated speech in the second plot had a constant frequency that paired with the constant magnitude to produce an unpleasant buzz. The predicted speech in the third plot varies in frequency as well as magnitude producing a more acceptable sound.

Several volunteer listeners compared the two methods for many files of speech. Most of the listeners stated that the low-rate voice-coding algorithm introduced irregularities that masked the effects of the missing frames, making it difficult to choose a preference. When the individual words with the compensated frames were isolated for comparison, the listeners preferred MCP to frame repetition.

MCP with Global System for Mobile Communications

The GSM algorithm described in Chapter Two provided a higher quality alternative for testing Markov chain prediction. Based on the results from the MELP algorithm, pitch and gain were chosen as candidates for the state information. Pitch is represented by long term prediction (LTP) lag. Two gains are calculated for this vocoder, a block gain, and an LTP gain. The GSM 6.10 standard ([25]) provides a ranking of the importance of speech parameters. Table 7.3 provides these parameters.

Class	Parameters	Bits in Class
1	MSB: LAR1, Block Amplitude	5
2	MSB: LAR2, LAR3 MSB-1: LAR1	3
3	MSB: LAR4, LAR5, LAR6 MSB-1: LAR2, LAR3, Block Amplitude MSB-2: LAR1, LAR2 MSBs1-5: LTP lag	31
4	MSB: LAR7, LTP gain, grid position MSB-1: LAR4, LAR5, LAR6 MSB-2: Block Amplitude MSB-3: LAR1 MSBs5-6: LTP lag	25
5,6	Rest of bits	196

Table 7.3 GSM Bits in Order of Importance

The first column lists the class numbers in which the GSM designers designated the bits. The second column lists the parameters and the bits, and the third column specifies the number of bits in each class. The second column uses abbreviations to decrease space. "MSB" represents the most significant bit. "MSB-1" represents the second most important bit, and "MSB-2" represents the third most important bit. "MSBs1-5" represents the first five most significant bits, and the fifth and sixth most important bits are "MSBs5-6".

As the table shows, the block amplitude's importance is ranked higher than LTP gain. Combining the six bits of the quantized block gain with the seven-bit quantized LTP lag results in 8,192 possible pitch-gain pairs–too many to represent on a desktop computer, even if only half of the pairs occur in speech. The LTP gain, however, is quantized with only four bits. Pairing the LTP gain with pitch creates only 2048 possible combinations. Even if all the combinations occur naturally, the memory on a desktop computer will not be exceeded. As described in Chapter Two, the pitch and lag gain are calculated each sub-frame–four times per GSM frame.

GSM Mixture Transition Distribution Parameters

The training data for calculating the optimal lag weights and transition matrix consisted of 26 approximately equal sized speech files from 26 different talkers (13 males, 13 females) totaling 20.4 minutes. This represented 61,227 GSM 6.10 frames and 244,908 sub-frames. The training data contained 324 of the possible 2048 pitch-gain states. The pitch-gain pairs in the state matrix were again ordered first on pitch and then on gain. Berchtold's initialization and optimization procedure was performed for a second-order Markov chain. The training data contained 219,354 second-order transitions of the pitch-gain states with a slightly higher association for second-order transitions ($\varphi_{init} = [0.49 \;\; 0.51]$). Figure 7.9 illustrates the change log-likelihood and lag weights throughout the optimization process.

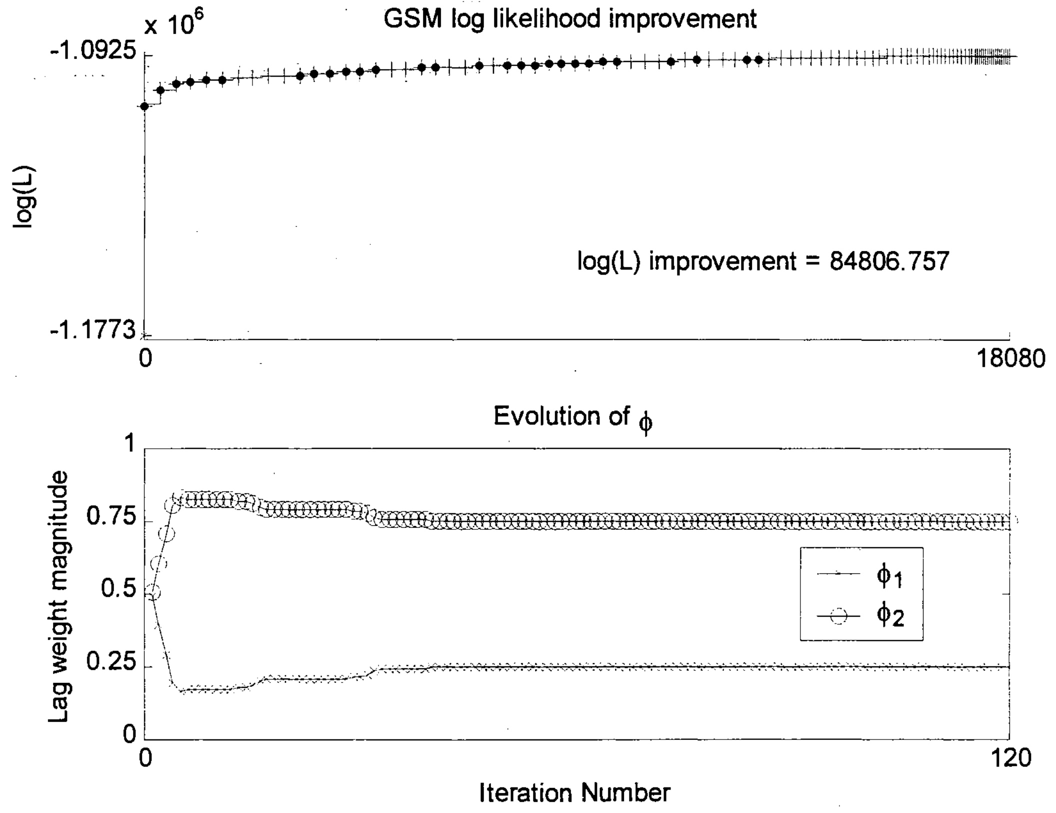


Figure 7.9 GSM MTD Optimization

The top plot shows the improvement in log-likelihood. The initial log-likelihood is denoted as an X, the log-likelihood measurements due to changes in lag weights are marked with dots, and the locations where the first row of the transition matrix is modified are shown with plusses. The bottom plot illustrates the changes in lag weights. The values of the first lag weight are denoted by X's and the values of the second lag weight are marked with O's. Over 18,000 changes were made to the MTD parameters in 120 iterations to provide an increase of 84,806 in log-likelihood.

The top plot shows that the first change in lag weights produces over 82% of the entire log-likelihood improvement. The next two lag weight changes produce noticeable improvements, but the remaining lag weight changes improve the log-likelihood little more than the individual transition matrix row improvements. The flattening of the slope between plusses shows how the improvements decrease as the transition matrix converges to its optimal value. The decreasing space between plusses shows that fewer transition rows contribute to improvement during each iteration, also. The lag weights converge to their optimum in 28 changes during the first 43 iterations.

The bottom plot shows that $\varphi_1 \to 0$ and $\varphi_2 \to 1$ as they converge to the optimum values. The first three changes are more substantial than the remaining 25, and by the 44th iteration, the lag weights have reached their optimum value. The lag weights remain unchanged during the final 63% of the iterations.

The optimization procedure produced second-order lag weights, [0.27 0.73]. These lag weights indicate the possibility that this could be a third-order system, but third-order initialization and optimization produced lag weights [0.211 0.774 0.015] showing that this is a second-order system. Figure 7.10 shows the optimal transition matrix.

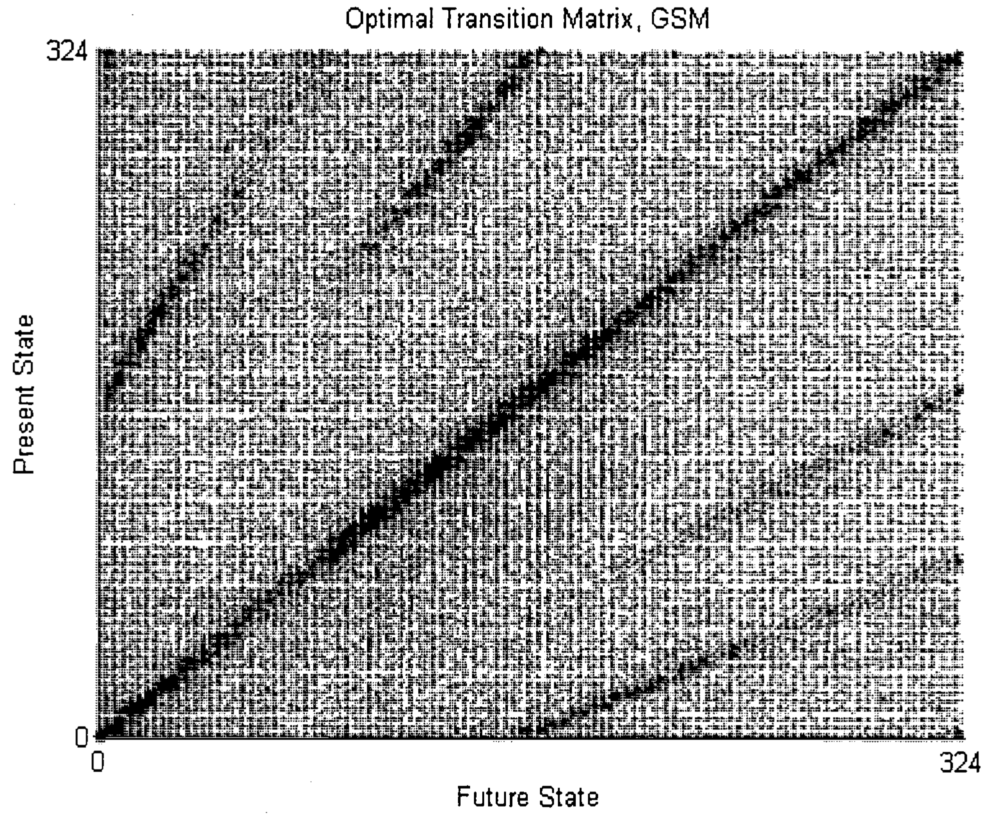


Figure 7.10 Second Order GSM Transition Matrix

This figure uses three different sized dots to represent the magnitude of the probabilities in the transition matrix. The smallest dots mark the 47,184 locations where $0 < Q \leq 0.05$. The medium dots represent the 539 locations where $0.05 < Q \leq 0.02$, and the largest dots represent the 11 locations where $Q > 0.2$.

The GSM transition matrix has a completely different character than the MELP transition matrix. The only common attribute is the prominent main diagonal, which contains the largest probabilities for the GSM transition matrix. Over 45% of the matrix is non-zero, and this accounts for the lower priorities overall. The highest probability in this matrix is 0.6, and only two probabilities are greater than 0.3. These lower probabilities forewarn the possibility of weak prediction. None of the rows contain only a single probability, and all of the second-order transitions occurred more than once.

The second-order cross-table exhibited a higher association measure; therefore the initial transition matrix contained the second-order step probabilities. The prominent diagonal for this matrix indicates the propensity for second-order repetition, i.e., if the second-order state transition is $s_a \rightarrow s_b$, the tendency is for the following state to be s_a. Four other diagonal concentrations are evident in this plot, two above and two below the diagonal. Since the state matrix is ordered first on pitch, then on gain, these diagonals illustrate the tendency of the third state to have a larger gain, but the same pitch as the first state.

GSM MCP Implementation and Performance

Markov chain prediction for GSM followed the same algorithm as for MELP. However, because of the sub-frame design of the GSM coder, each 20 ms frame required four predictions–one for each 5 ms sub-frame. The prediction algorithm used the third and fourth sub-frames from the last correctly received frame to predict the first sub-frame of the missing frame, and shifted forward in sub-frames to predict the subsequent sub-frames, as shown in Figure 7.11.

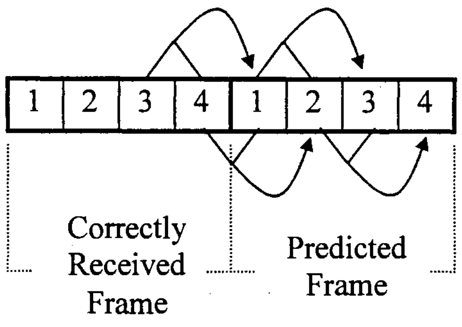


Figure 7.11 GSM Sub-frame Prediction

If the probability of the predicted replacement state did not exceed the minimum threshold, the previous sub-frame was repeated. All pitch-gain pairs found in the test data corresponded to a state identified in the training data, but many of the predictions were below the minimum threshold.

GSM missing frame compensation using MTD was compared to frame repetition using both objective and subjective methods. The difference between missing speech compensated with the two techniques was quantified objectively with spectral and energy distance measures and evaluated subjectively by a listening test with volunteer listeners. The evaluation process insured that all MCP test segments contained pitch-gain pairs that were predictable, i.e. all pairs in the test data also occurred in the training data, and the highest calculated MTD probability exceeded the threshold.

SPECTRUM AND ENERGY DIFFERENCE

The GSM objective evaluation used the same spectral and energy distance as the MELP objective evaluation (equations (7.24) and (7.25)). Voiced, unvoiced, background, and transitional segments of speech were once again randomly chosen for the distance measurements. Four frames of speech were removed from the segments and replaced using MCP and frame repetition. Figure 7.12 shows the speech segments.

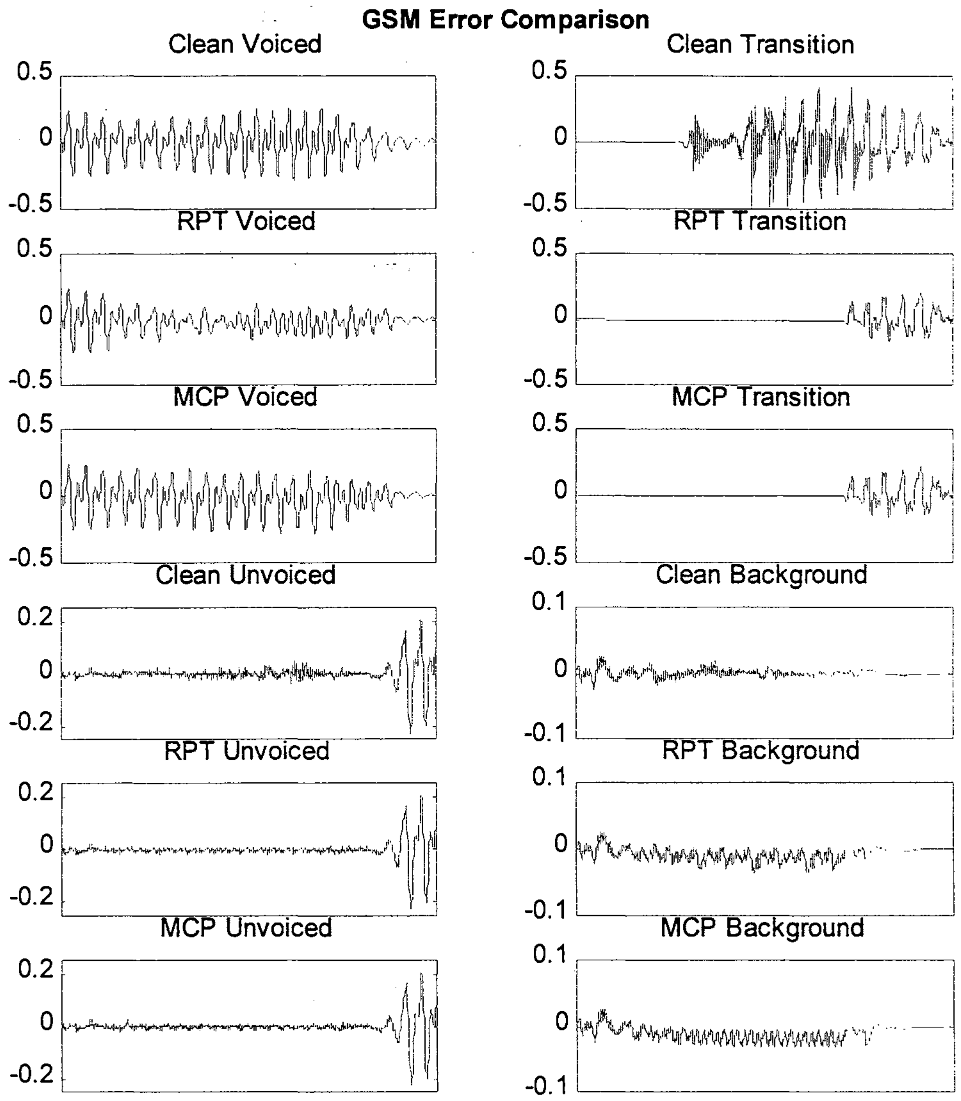


Figure 7.12 Speech Used for GSM Objective Error Calculations

The labels of the twelve plots in Figure 7.12 denote their contents. The plots labeled "Clean" show GSM-synthesized speech with no errors. The plots labeled "RPT" show speech with four missing frames compensated with frame repetition, and the plots labeled

"MCP" show speech with four missing frames compensated with Markov Chain Prediction. Each plot shows a few frames of speech on each side of the compensated frames. A visual analysis of these plots yields results similar to that of the MELP plots. The voiced speech segment shows a marked improvement in the speech reconstructed using MCP over that of frame repetition. The compensated unvoiced and transitional speech appears similar for both methods. The background speech appears to have the same energy for each compensation technique, but the signals look very different. With the low magnitude of the signal, however, the differences are imperceptible.

The objective measurements calculated the distance between the clean speech and each of the compensation techniques for the speech segments shown in Figure 7.12. Table 7.4 lists the objective measures for these speech segments.

	MCP	**RPT**
L_2 on LP Spectrum		
Voice	18.051 dB	19.896 dB
Unvoiced	9.686 dB	10.662 dB
Transition	52.302 dB	52.303 dB
Background	48.438 dB	34.887 dB
Energy Differences		
Voice	0.124 dB	1.772 dB
Unvoiced	0.147 dB	0.162 dB
Transition	4.085 dB	4.085 dB
Background	4.149 dB	3.214 dB

Table 7.4 Objective Measures for MCP and Frame Repetition with GSM

Both objective measures show that the MCP error is smaller, and is therefore superior to frame repetition in the voiced and unvoiced segments. Both compensation approaches perform similarly for the transitional segment, and frame repetition is better for the background segment.

Perceptual Opinion of GSM MCP

Initial listening to speech with packet loss indicated that when less than three frames were lost, lost speech compensated using MCP was perceptibly indistinguishable from lost speech compensated using frame repetition. The network simulation program removed three to six sequential frames of GSM data (60-120 ms of speech) from a single word in one-sentence test files containing both male and female talkers. Sixteen listeners compared missing speech reconstructed with Markov chain prediction to missing speech reconstructed using frame repetition. Replication of the last correctly received sub-frame sounded better than replication of the four sub-frames of the last correctly received frame, so the listening test compared sub-frame repetition to MCP. Listeners chose the file that sounded better. The listening test provided a "No Difference" choice for listeners who considered the quality of both files equal. The results shown in Table 7.5 discount the "No Difference" responses.

	3 Missing Frames	**4 Missing Frames**	**5 Missing Frames**	**6 Missing Frames**
Markov Chain Prediction	59%	83%	62%	75%
Frame Repetition	41%	17%	38%	25%

Table 7.5 Results of Listening Test for GSM 6.10

As Table 7.5 shows, listeners preferred speech reconstructed using MCP to speech reconstructed using frame repetition.

Markov Chain Prediction Using Future Frame

Using information on both sides of the missing frame, i.e., both past and future frames, has the potential to further improve the accuracy of the prediction and therefore improve the quality of the reconstructed speech. In the previous discussion, the prediction algorithm used only past frames to predict the missing speech, and the variable argument from (7.8) was i_0. However, if a future frame is used to aid prediction, the variable argument from (7.8) is i_1, i.e., the receiver calculates the probability of transitioning to i_0 from $i_m \rightarrow i_{m-1} \rightarrow \cdots \rightarrow i_2$ and each of the S possible states for i_1. The receiver selects the i_1 producing the highest probability to reconstruct the missing speech segment. This procedure is statistically valid if the data is stationary.

Markov chain prediction using future frames was tested using the MELP algorithm. Speech quality is noticeably improved for one missing frame, especially in transition regions. However, a frame from the future must be available to predict the missing parameters. Since most packet losses occur in clusters, this procedure is only applicable for the last missing frame.

Markov Chain Prediction Complexity

Complexity of Markov chain prediction is dependent on the number of states and the order of the Markov chain. As the size of these factors grows, the number of operations and memory required for both training (creating the optimal lag weights and transition matrix) and utilization (determining parameters for missing frames) increases. Training requires far more resources than utilization. The number of operations for utilization can be decreased at the expense of memory by pre-calculating all possible transition probabilities and storing them in a look-up table.

Memory

The amount of storage required for MTD training and utilization is dependent on many factors: the number of states, S; the Markov chain order, m; the number of vocoder parameters in a state, d; the number of bits in the parameter used as a lookup variable, b; and the number of mth-order state transitions, N, in the training data (i.e., N is the number of unique mth-order state transitions in $n(i_0,\ldots,i_m)$). The voice-coding algorithm that creates the bit-files used for training also requires storage space, but it will not be considered for this analysis. This analysis assumes that every variable and constant will occupy one word of memory. Since the number of bytes per word changes for different processors, (i.e., a digital signal processing chip may have only four bytes per word, and a desktop computer can have four to 64 bytes per word), the estimates reported are for memory words, not bytes.

The training process consists of an initialization process and an optimization process to create the lag weights and transition matrix. The amount of memory required for these processes is shown in Table 7.6.

	Words of Memory
Initialization	$S^2(m+1)+S(2+d)+2(m+N)+3\cdot 2^b+9$
Optimization	$2(S^2+m+N)$
Total	$S^2(m+3)+S(2+d)+4(m+N)+3\cdot 2^b+9$

Table 7.6 Memory Words Needed for MTD Training

The appendix contains more information on the calculation of estimates for the amount of memory occupied by training.

The MTD calculates conditional probability each time a missing frame is detected. The amount of memory required for utilization is far less than the amount of memory required for training. Only the transition matrix and lag weights are needed, resulting in S^2+m+1 words of memory. The appendix contains more information on the calculation of estimates for the amount of memory occupied by utilization.

Number of Operations

The number of operations required for MTD training and utilization is dependent on four factors: the number of states, *S*; the Markov chain order, *m*; the amount of training data; and the number of *m*th-order state transitions, *N*, in the training data.

Initialization and optimization of the lag weights and transition vector comprise the training process. Increasing the amount of training data increases the number of operations required by the voice-coding algorithm. Vocoder complexity is a substantial part of the processing requirements, but it is not quantified for this analysis. Increasing the amount of training data also increases the possibility of finding new states and new *m*th-order state transitions. Table 7.7 shows the number of operations (adds, multiplies, and divides) for initialization and optimization training processes.

	Initialization	**Optimization**
$+$	$m(605S^2+604S+1)+N(603+m)$	$N(m(S+2)+S+604)+2$
$\times$	$m(1804S^2+1802S)+N(1802+m)$	$N(m(S+2)+S+1803)$
$\div$	$m(3S(S+1)+1)+S^2+2N$	$N(S+3)$

Table 7.7 Number of MTD Operations for Training by Type

The appendix contains details on the calculation of estimates for the number of operations consumed by training. The total number of operations (initialization + optimization) for training is shown in Table 7.8.

	Total
$+$	$N(1207+3m+S(m+1))+m(1+S(604+605S))+2$
$\times$	$N(1802+m(S+3))+S(2m(902S+901)+1)+1803$
$\div$	$N(S+5)+S(S(1+3m)+3m)+m$

Table 7.8 Total Number of MTD Operations

The number of transitions dominates the other factors ($m << S << N$). Training will take fewer operations if less training data is used, but the resulting predicted speech would be low quality.

Using the MTD to calculate conditional probabilities consumes far fewer processor operations than training. Calculation of all S possible replacement states to determine which state will best replace the missing data requires Sm adds and Sm

multiplies. The appendix contains details on the calculation of estimates for the number of operations consumed by utilization. Creating a lookup table can decrease the number of operations required for utilization at the expense of additional memory.

Markov Chain Prediction Delay

The amount of delay accrued for Markov chain prediction is dependent on the placement of the missing frame in the conditional probability, $p(i_0|i_1,\ldots,i_m)$, and the frame size for the voice-coding algorithm. If the missing frame is i_0, then only past frames are required to calculate the probabilities. Markov chain prediction using state information from only past frames adds no delay to the communication process. If the missing frame is considered any of $i_1,\ldots,i_m$, then future frames are required to calculate the probabilities. If future frames are used, then one frame of delay for each future state desired is added to the communication process.

Discussion

Experiments with both MIL_STD-3005 and GSM 6.10 indicated that Markov chain prediction works best in non-transitional areas of speech. In voiced speech the difference is more noticeable than in unvoiced speech, due to the higher magnitude in the voiced signal. When the number of missing frames is small, there is no noticeable difference in the quality of frame repetition and MCP. When the number of sequential missing frames is larger, MCP doesn't always reproduce the correct sound, but it does synthesize speech that's more appealing than frame repetition. If the voice coding algorithm degrades the quality of speech appreciably, the effects of missing frames are difficult to detect, and frame repetition performs as well as MCP.

Frame repetition is less complex than MCP, but there is a memory/computation tradeoff for MCP. If memory is available for lookup tables, MCP requires very little computation. If memory is precious, direct calculation of the Markov chain probabilities for each state requires little memory. Chapter Nine provides a comparison of MCP and other missing frame compensation techniques.

CHAPTER EIGHT

OTHER APPLICATIONS AND SUGGESTIONS FOR FUTURE STUDY

As with all research, the algorithms described in this thesis have application beyond the problem that motivated their discovery and their analysis is not exhausted. This chapter lists other applications that could benefit from these techniques and describes research that may yield improvements.

Other Applications

Although this work is directed toward frame loss in packet-switched networks, its usefulness is not restricted to this environment. Successful frame-loss compensation techniques can be used to overcome other communication challenges.

Wireless voice communications is a rapidly growing field for personal communication. It uses low rate voice coding algorithms to provide more efficient use of the wireless spectrum. A wireless transmitter sends individual vocoder frames sequentially to the receiver. Deep fades, a common malady in wireless communications, essentially remove entire portions of the signal. Frame compensation techniques can restore the frames affected by deep fading. Commercial wireless communication, such as the aviation industry where speech conversations sometimes contain missing portions, can similarly benefit.

Communications through harsh channels may not result in the loss of entire frames, but the received data may be too corrupted for the error detection and correction routines. Receivers can use frame compensation techniques to provide high quality communication even in the presence of bursty errors.

Hostile parties often use jamming to prevent communication. Spread spectrum techniques can reduce the effects of jamming by spreading the signal across many frequencies, but some signal portions may still pass through jammed frequencies and these portions will accordingly suffer signal loss and therefore frame loss. Frame compensation techniques can restore the missing frames that are corrupted by jamming.

Suggestions for Future Study

The two solutions for coping with missing speech described in this thesis, Naturalness Preserving Transform reconstruction and Markov chain prediction, are superior to common techniques. However, further research may yield improvements to the efficiency of the algorithms and to the resulting speech quality. This chapter describes several possible research areas for both methods.

Naturalness Preserving Transform Reconstruction

Adding a second loop to the NPT may improve the ability to reconstruct missing frames. The first loop computes the NPT transform of speech segments as discussed previously. The second loop computes an NPT transform on a combination of transformed segments. The transmitter sends the doubly transformed segment across the network.

This paper discusses scalar quantization for the transformed speech signal, but more efficient quantization techniques, such as vector quantization, have not yet been addressed. Effective vector-quantization designs will substantially decrease the bit rate.

The effects of random bit errors on NPT reconstruction have not been investigated. Reconstruction using NPT relies on knowing the location of correct transform data. Random errors in the received transformed data will effect the accuracy of reconstruction, and the level of speech quality deterioration as a function of bit errors should be determined. If the reduction in speech quality is deleterious, adequate recovery techniques will be required.

It is desirable to further reduce the overhead required for NPT reconstruction without severely sacrificing quality. A low-quality signal that has been reconstructed from a minimal amount of overhead information may be improved through common post-processing techniques.

The analysis of NPT reconstruction with parametric coders was not extensive. A study of the amount of overhead required for vocoder parameters will improve the efficiency for this application. Also, other waveform and parametric voice coding algorithms should be tested.

Special purpose hardware development of the NPT algorithm is another area of research, as the speech applications in the commercial telephony are abundant.

Markov Chain Prediction

Currently, sequential missing frames are replaced using the same procedure, therefore a prediction error in an early frame propagates to ensuing frames. Trellis-type decoding combined with the Markov chain prediction may generate more accurate prediction of consecutive missing speech frames. It may also be advantageous to calculate the accumulated probabilities of sequential missing frames for threshold comparison rather than individual probabilities.

Memory limitations restrict the number of parameters comprising a state. It is possible that a cascading transition matrix structure would allow consideration of more parameters without substantially increasing memory requirements. The first transition matrix, $\mathbf{Q}_1$, contains the transition probabilities for combinations of two or three parameters. The second transition matrix, $\mathbf{Q}_2$, contains the transition probabilities for one to two additional parameter combinations, and subsequent transition matrices represent other parameters. The inter-dependence of the parameters is not exploited with this procedure, but the parameters can be predicted rather than simply repeated, thereby increasing quality without substantial memory increases.

Transition regions are challenging for MCP. An extensive study of vocoder parameters may reveal combinations or sequences of combinations that accurately predict voice transitions. When a possible transition is detected, the prediction algorithm could use alternate parameter combinations and transition matrices for prediction. This will improve both the quality and the intelligibility of the reconstructed speech.

The MCP implementation described in this paper is a generic algorithm and does not incorporate any vocoder-specific enhancements. If MCP employs characteristics of individual vocoders to improve prediction or enhance the predicted parameters the resulting speech will sound better.

The structure of the MCP transition matrix was not thoroughly analyzed in this study. It is possible that a thorough study of the structure of the transition matrix will reveal techniques for more efficient storage, or dependencies for improved prediction.

Repetition is employed for missing parameters that cannot be predicted, i.e., if the parameter is not included in the state table, if the transition probability is below the threshold, or if a parameter combination is found in the input data that does not exist in the state table. This simple algorithm certainly contributes to quality degradation in the resulting speech. Higher quality solutions to these situations, such as smoothing, should be identified and incorporated.

CHAPTER NINE

CONCLUSIONS

This thesis describes two original and innovative techniques for improving the quality of voice communication across lossy packet-switched networks. The two techniques, Naturalness Preserving Transform reconstruction and Markov Chain Prediction, improve speech quality by recreating the missing speech data that was contained in lost packets. Both techniques are portable across networks and across voice coding algorithms. Based on both objective and subjective measurements, the resulting speech quality from both algorithms is far superior to baseline frame repetition.

The Naturalness Preserving Transform is based on a stochastic operator that uses a normalized Hadamard matrix. This unitary transform has properties that enable an iterative reconstruction process that can achieve very high quality speech on networks with high packet loss rates. The reconstruction process relies on knowing a portion of the original speech matrix, which it inserts as overhead if necessary. Naturalness Preserving Transform reconstruction requires conversion of data at both the transmitter and the receiver, and will increase the bit rate of the original voice-coding algorithm to some extent. The magnitude of the bit rate increase depends on the amount of overhead and the quantization efficiency.

The Markov Chain Prediction technique models voice coding parameters or combinations of parameters as states in a Markov chain. It relies on state transition probabilities that were previously calculated from training data and predicts missing parameters using data from previous packets and the from state transition probabilities.

Since high order Markov chains with many states can require prohibitive amounts of memory to store the transition probabilities, the algorithm employs a Mixture Transition Distribution to represent the Markov chain. A Mixture Transition Distribution calculates high-order state transition probabilities using a two-dimensional state transition matrix that represents all step transitions, and a lag weight vector that scales the step transitions. The lag weight vector and transition matrix are calculated from large amounts of training data.

A direct comparison of the attributes of Naturalness Preserving Transform reconstruction, Markov Chain Prediction, and baseline frame repetition is difficult because of the diversity of the techniques. For instance, the memory requirement of Naturalness Preserving Transform reconstruction is dependent on the size of the Hadamard matrix, while the memory requirement of Markov Chain Prediction is dependent on the number of states, and neither of the dependencies can be accurately scaled for direct comparison. This is not to say that no comparisons can be made, however. Several important features of the methods can indeed be ranked, as shown in Table 9.1.

In the table, ranking ranges from one to three where one is the best and three is the worst. "Memory" refers to the amount of memory required for operation of the procedure, "Number of Operations" refers to the quantity of processor operations required by the procedure, and "Delay" refers to additional time between the talker and the listener in addition to what system already requires. "Speech Quality" refers to the quality of the resulting speech when the channel experiences high packet loss rates.

"Vocoder Portability" refers to dependence on particular features of specific voice coding algorithms, and "Network Portability" refers to dependence particular features of specific network protocols. "Overhead" refers to additional bits in addition to what the voice coding algorithm already produces, and "Ease of Upgrade" refers to the complexity of introducing the algorithm to an existing communication system. An ideal system requires little memory, requires few processor operations, accrues no delay, produces high quality speech, is portable across vocoders and networks, adds no overhead, and is easy to insert into existing network communications systems.

	NPT Reconstruction	**Markov Chain Prediction**	**Frame Repetition**
Memory	2	3	1
Number of Operations	3	2	1
Delay	2	1	1
Speech Quality	1	2	3
Vocoder Portability	1	1	1
Network Portability	1	1	1
Overhead	2	1	1
Ease of Upgrade	2	1	1

Table 9.1 Comparison of Missing Frame Compensation Algorithms

Note that for many attributes, two or three methods are equally ranked. Frame repetition is a portable, low complexity, no overhead, and no delay method, but the resulting speech is very low quality. The transition table of Markov chain prediction requires more memory than the other algorithms, but it consumes fewer operations, it requires less delay, and is more easily implemented than Naturalness Preserving Transform reconstruction. However, Naturalness Preserving Transform reconstruction produces the highest quality speech of the three techniques. A unique feature of Naturalness Preserving Transform reconstruction is its overhead/quality scalability—reconstructed speech quality improves with additional overhead data. It is more difficult to implement than the other two methods since it requires a transform at both the transmitter and the receiver, while Markov chain prediction and frame repetition operate only in the receiver.

Both of the new procedures discussed in this thesis are superior in different applications. If a new system is being designed, and delay and overhead are not constraints, then Naturalness Preserving Transform reconstruction is the optimal technique. If an existing system is to be upgraded without introducing incompatibility between legacy and new equipment, Markov chain prediction is the optimal technique. Frame repetition is optimal if expected packet loss will be low, and if memory, processor, and delay constraints are very tight.

Packet loss is inevitable in all packet-switched networks. Transmission of voice using voice-coding algorithms across these lossy packet-switched networks will result in missing voice frames. High quality voice can still be maintained in this environment with the missing frame compensation schemes described in this thesis.

CHAPTER TEN

REFERENCES

[1] Alvarez-Cuevas, F., Bertran, M., Oller, F., Selga, J. M., "Voice Synchronization in Packet Switching Networks", IEEE Network, Volume 7, Issue 5, pp 20-25, September 1993.

[2] Atungsiri, S., Soheili, R., Kondoz, A., Evans, B, "Effective Lost Speech Frame Reconstruction for CELP Coders," Proceedings of EUROSPEECH, Volume 2, pp 599-602, Geneva, Italy, September 1991.

[3] Bellamy, J., *Digital Telephony*, John Wiley and Sons. 1991.

[4] Bharucha-Reid, A.T., *Elements of the Theory of Markov Processes and Their Applications*, McGraw-Hill Book Company, Inc., New York, 1960.

[5] Berchtold, Andre, "Estimation of the Mixture Transition Distribution Model," Technical Report no. 352, Department of Statistics, University of Washington, Seattle WA, April 1999.

[6] Bolot, J.C., Vega-Garcia, A., "Control mechanisms for packet audio in the Internet", Proceedings IEEE INVOCOM, Volume 1, pp 232-239, March 1996.

[7] Bolot, J.C., Fosse-Parisis, S., "Adding Voice to Distributed Games on the Internet," Proceedings IEEE Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Volume 2, pp 480-487, April 1998.

[8] Borella, M.S., Swider, D., Uladag, S., Brewster, G.B., "Internet packet loss; measurement and implications for end-to-end QoS," Proceedings of the 1998 ICCP Workshops on Architectural and OS Support for Multimedia Applications/Flexible Communication Systems/Wireless Networks and Mobile Computing, pp 3-12, August 1998.

[9] Boules, R., "Class Notes from 605.417.31, Principles of Data Communication Networks," Johns Hopkins University, Fall 1997.

[10] Campbell, J. P. Jr., Tremain, T. E., Welch, V. C., "The Federal Standard 1016 4800 bps CELP Voice Coder," Digital Signal Processing 1, pp 145-155 1991.

[11] Chemical Rubber Company, *Handbook of Tables for Probability and Statistics*, Chemical Rubber Company, 1966.

[12] Chen, Y., Chen, B., "Model-based multirate representation of speech signals and its application to recover of missing speech packets," IEEE Transactions on Speech and Audio Processing, Volume 5, Issue 3, pp 220-231, May 1997.

[13] Chin, K.V., Hui, S.C., Foo, S, "Enhancing the quality of Internet voice communication for Internet telephony systems, " Journal of Network and Computer Applications, Volume 21, Issue 3, pp 203-218, July 1998.

[14] Choi, Q.W., Constantinides, A.G., "Effects of Packet Loss on 3 Toll Quality Speech Coders", Second IEE National Conference on Telecommunications, pp 380-385, 1989.

[15] Claerbout, J.F., "Model fitting by least squares", sepwww.stanford.edu/public/docs/sep65/toc_html/index.html.

[16] Cover, T., M, Thomas, J. A., *Elements of Information Theory*, John Wiley and Sons, New York, 1991.

[17] Cox, R. V., Kroon, P., "Low Bit-Rate Speech Coders for Multimedia Communication", IEEE Communications Magazine, December 1996.

[18] DaSilva, L.A., Petr, D. W., Frost, V. S., "A Class Oriented Replacement Technique for Lost Speech Packets," Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies, Volume 3, pp 1098-1105, April 1989.

[19] Deller, J. R., Proakis, J. G., Hansen, J. H. L., *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company, New York, 1993.

[20] Dettmer, R., "Packet Phone," IEE Review, Volume 44, Issue 2, pp 58-61, March 1998.

[21] Difu, S., Srivastava, J., Jey-Hsin, Y., "Investigating factors influencing QoS of Internet phone," Proceedings of the 1999 IEEE International Conference on Multimedia Computing and Systems, volume 1, pp 308-313, June 1999.

[22] Dodge, Y., *Analysis of Experiments with Missing Data*, John Wiley and Sons, New York, 1985.

[23] Dudley, H., "Remaking Speech," *Joint Acoustical Society of America*, Volume 11, pp 169-177, 1939.

[24] Erdol, N., Castelluccia, C., Zilouchian, A., "Recovery of missing speech packets using the short-time energy and zero-crossing measurements," IEEE Transactions on Speech and Audio Processing, Volume 1, Issue 3, pp 295-303, July 1993.

[25] ETSI/TC SMG Recommendation GSM 06.10, "GSM Full Rate Speech Transcoding," Release 92, Phase 1, February 1992.

[26] Fantacci, R., Zoppi, L., "A CDMA Wireless Packet Network for Voice-Data Transmissions", IEEE Transactions on Communications, Volume 45, Number 10, pp 1162-1166, October 1997.

[27] Fore C.R., Yarlagadda, R., "Convex Projection Restoration of Hadamard Naturalness Preserving Transform Coded Images," 1998 IEEE Southwest Symposium on Image Analysis and Interpretation," Tucson, AZ.

[28] Fore C.R., Yarlagadda, R, "Coding and Restoration of Hadamard Naturalness Preserving Transform," Workshop on Data Compression Processing Techniques for Missile Guidance Datalines, Redstone Arsenal, Alabama, 1998.

[29] FreePhone URL http://www.inria.fr/rodeo/fphone

[30] Froehlich, K., *The Froehlich/Kent Encyclopedia of Telecommunications*, Volume 15, Marcel Dekker, 1998.

[31] Gillespie, D.T., *Markov Processes, An Introduction for Physical Scientists*, Academic Press, Inc. Boston, MA.

[32] Goodman, D. J., Lockhart, G. B., Wasem, O.J., Wong, W., "Waveform Substitution Techniques for Recovering Missing Speech Segments in Packet Voice Communications", IEEE Transactions on Acoustics Speech and Signal Processing, Volume ASSP-34, Number 6, pp 1440 - 1448, December 1986.

[33] Goodman, D. J., Wei, S.X., "Efficiency of Packet Reservation Multiple Access", IEEE Trans on Vehicular Technology, Volume 40, Number 1, pp 170-176, February 1991.

[34] Gray, A.H., Markel, J. D., "Distance Measures for Speech Processing", IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume ASSP-24, Number 5, October 1976.

[35] Hardman, S., Handley, W., "Reliable Audio for Use over the Internet", http://inet.nttam.com, 1995.

[36] Himowitz, M. J., "Phoning on the Internet may be next revolution", The Baltimore Sun, February 27, 1995.

[37] Husain, A., Cuperman, V., "Reconstruction of missing packets for CELP-based speech coders," 1995 International Conference on Acoustics, Speech, and Signal Processing, pp 245-248, Volume 1, May 1995.

[38] International Telecommunication Union – Telecommunication Standardization Sector Recommendation G.764, "General Aspects of Digital Transmission Systems; Terminal Equipments, Voice Packetization – Packetized Voice Protocols," Geneva, 1990.

[39] International Telecommunication Union – Appendix I to Telecommunication Standardization Sector Recommendation G.764, "General Aspects of Digital Transmission Systems, Packetization Guide," November 1995.

[40] Iosifescu, M., *Finite Markov Processes and Their Applications*, John Wiley and Sons, New York.

[41] Jayant, N., Christensen, S., "Effects of Packet Losses in Waveform Coded Speech and Improvements due to an odd-even sample-interpolation Procedure," IEEE Transactions on Communications, Volume com-29, Number 2, February 1981.

[42] Jayant, N., "Signal Compression: Technology Targets and Research Directions", IEEE JSAC, Volume 10, Number 5, pp 796-818, June 1992.

[43] Kabal, P., "Quantizers for the Gamma Distribution and Other Symmetrical Distributions," IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume ASSP-32, Number 4, August 1994.

[44] Keepence, B., "Quality of service for voice over IP," IEE Colloquium on Services over the Internet, What does Quality Cost?, pp 4/1-4/4, June 1999.

[45] Khansari, M, Bhaskaran, V, "A low-complexity error-resilient H.263 coder," IEEE International Conference on Acoustics, Speech, and Signal Processing, pp 2737-2740, 1997.

[46] King, S, "The Spectral Properties of Markov Chain Random Processes with Application to Speech Modeling and Spread Spectrum Communications," Ph.D. Dissertation, Oklahoma State University, December 1991.

[47] Kitawaki, N., Nagabuchi, H., Taka, M., Takahashi, K., "Speech Coding Technology for ATM Networks", IEEE Communication Magazine, pp 21-27, January 1990.

[48] Kohler, M. A., Yarlagadda, R, "Naturalness Preserving Transform for Missing Frame Compensation," Proceedings of the IEEE International Symposium on Circuits and Systems, Volume 4, pp 118-122, June 1999.

[49] Kostas, T.J., Borella, M. S., Sidhu, I., Schuster, G. M., Grabiec, J, Mahler, J, "Real-time voice over packet-switched networks", IEEE Network, Volume 12, Issue 1, pp 18-27, January-February 1998.

[50] Kotz, J., *Encyclopedia of Statistical Sciences*, John Wiley and Sons, Volume 5, 1985.

[51] Lara-Barron, M. M., Lockhart, G. B., "Correlation Techniques for Reconstructing Lost Speech Packets", Proceedings IERE Conference on Digital Processing of Signals in Communications, Loughborough University, pp 197-202, 1988.

[52] Lara-Barron, M. M., and Lockhart, G. B., "Selective Discarding Procedure for Improved Tolerance to Missing Voice Packets", Electronic Letters, 19, pp 1269-1271, 1989.

[53] Lara-Barron, M. M., Lockhart, G. B., "Missing packet recovery of low bit rate encoded speech using a novel packet-based embedded coder", Proceedings EUSIPCO-90, Barcelona, Spain, 1990.

[54] Lara-Barron, M. M., Lockhart, G. B., "Speech Encoding and Reconstruction for Packet-Based Networks", IEE Colloquium on Coding for Packet Video and Speech Transmission, London, 1992.

[55] Lara-Barron, M. M., Lockhart, G. B., "Packet-based embedded encoding for transmission of low-bit-rate-encoded speech in packet networks," IEE Proceedings I, Communications, Speech, and Vision, Volume 139, Issue 5, pp 482-487, October 1992.

[56] Lee, T.C., Judge, G. G., Zellner, A., *Estimating the Parameters of the Markov Probability Model from Aggregate Time Series Data*, North-Holland Publishing Company, London, England, 1970.

[57] Leland, W., Taqqu, M., Willinger, W., Wilson, D., "On the Self-similar Nature of Ethernet Traffic (Extended Version)," IEEE/ACM Transactions on Networking, February 1994.

[58] Lines, B. M., "The integration of real-time interactive speech with data on local area network (LANs)-a step towards 'multi-media' in a commercial industrial environment", IEE Colloquium on Time Critical Data Communications, pp 5/1-5/4, 1994.

[59] Little, R. J. A., Rubin, D. B., *Statistical Analysis with Missing Data*, John Wiley and Sons, New York, 1987.

[60] Lockhart, G. B., "Speech Processing in Packet Communication Systems", IEE Colloquium on Digitized Speech Communication via Mobile Radio, pp 11/1 - 11/3, 1988.

[61] Lockhart, G. B., Lara-Barron, M. M., "Implementation of Packet-Based Encoding Schemes for Speech Transmission", Sixth International Conference on Digital Processing of Signals in Communications, pp 326-330, 1991.

[62] Matsumoto, K., "Real-Time High Accurate Cell Loss Recovery Technique for Speech Over ATM Networks", IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 1, pp 248-250, 1996.

[63] Maxemchuk, N. F., Lo, S., "Measurement and Interpretation of Voice Traffic on the Internet", IEEE International Conference on Communications, Volume 1, pp 500-507, 1997.

[64] McCree, A., Truong, K., George, E.B., Barnwell, T.P., Viswanathan, V., "A 2.4 Kbit/s MELP Coder Candidate for the new U.S. Federal Standard," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 1, pp 200-203, 1996.

[65] MIL-STD-3005, "Analog to Digital Conversion of Voice by 2,400 Bit/Second Mixed Excitation Linear Prediction," December 1999.

[66] Moffat, I.G., Davis, A.G., O'Neill, A.W., "The Internet telephone-a new paradigm", IEE Colloquium on Advances in Interactive Voice Technologies for Telecommunication Services, pp 10/1 – 10/6, June 1997.

[67] Nichols, D., "Estimation of missing data by least squares," Stanford Exploration Project, Report 65, pp 269-292, January 1998.

[68] Null, C., "No more hang-ups over voice", LAN Times (from web).

[69] O Ruanaidh, J. J. K., Fitzgerald, W. J., "Interpolation of missing samples for audio restoration", IEEE Electronics Letters, Volume 30, Number 8, pp 622-623, April 1994.

[70] Osinubi, A. O., King, R.A., Barbosa, A.M., "Predictive naturalness preserving transform coding of monochrome images", Proceedings of the Electrotechnical Conference, pp 292-295, April 1989.

[71] Osinubi, A. O., King, R.A., "Naturalness-preserving transform (NPT) reconstruction of signals degraded by non-stationary noise processes", Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Volume 2, pp 1199-1202, May 1989.

[72] Osinubi, A.O., King, R.A., "One-dimensional Hadamard naturalness-preserving transform reconstruction of signals degraded by nonstationary noise processes", IEEE Transactions in Signal Processing, Volume 40, Issue 3, pp 645-659, March 1992.

[73] Perkins, M. E., Dvorak, C. A., Lerich, B. H., Zebarth, J. A., "Speech transmission performance planning in hybrid IP/SCN networks," IEEE Communications Magazine, Volume 37, Issue 7, July 1999.

[74] Petr, D. W., DaSilva, L. A. Jr., Frost, V. S., "Priority discarding of speech in integrated packet networks", IEEE Joint Select Areas Communication, pp 644-656, June 1989.

[75] Rabiner, L. R., Schafer, R. W., *Digital Processing of Speech Signals*, Prentice-Hall, New Jersey, 1978.

[76] Raftery, T., "A Model for High-order Markov Chains," Joint Royal Statistical Society B, Volume 47, Number 3, pp 528-539, 1985.

[77] Raftery, T., "Estimation and Modelling Repeated Patterns in High Order Markov Chains with the Mixture Transition Distribution Model," Applied Statistics, Volume 43, Number 1, pp 179-199, 1994.

[78] Rappaport. T.S., *Wireless Communications, Principles and Practice*, Prentice-Hall, 1996.

[79] Sahinoglu, Z, Tekinay, S, "On Multimedia Networks: Self-Similar Traffic and Network Performance," IEEE Communications Magazine, Volume 37, Number 1, pp 48-52 January 1999.

[80] Salami, R., Laflamme, C., Adoul, J., Kataoka, A., Hayashi, S., Moriya, T., Lamblin, C., Massaloux, D., Proust, S., Kroon, P., Shoham, Y., "Design and Description of CS-ACELP: A Tool quality 8 kb/s Speech Coder", IEEE Transactions on Speech and Audio Processing, Volume 6, Number 2, March 1998.

[81] Sanghi, D., Agrawala, A. K., Gudmundsson, O., Jain, B.N., "Experimental assessment of end-to-end behavior on Internet," Proceedings of INFOCOM 1993, Volume 2, pp 867-874, March 1993.

[82] Sanneck, H., "Concealment of lost speech packets using adaptive packetization," Proceedings IEEE International Conference on Multimedia Computing and Systems, pp 140-149, 28 June-1 July 1998.

[83] Scott, D. W., *Multivariate Density Estimation*, Wiley, 1992.

[84] Shah, A, Atungsiri, S, Kondoz, A, Evans B, "Lossy multiplexing of low bit rate speech in thin route telephony", Electronics Letters, pp 95-97, January 1996, Volume 32, Issue 2.

[85] Spragins, J. D., Hammond, J. L., Pawlikowski, K., *Telecommunications, Protocols and Design*, Addison Wesley, 1991.

[86] Sriram, K., McKinney, R. S., Sherif, M. H., "Voice Packetization and Compression in Broadband ATM Networks", IEEE Journal on Selected Areas in Communications, Volume 9, Issue 3, pp 294-304, April 1991.

[87] Stallings, W, *ISDN and Broadband ISDN with Frame Relay and ATM*, Prentice Hall, 1995.

[88] Steel, R. G. D., Torrie, J. H., Dickey, D. A., *Principles and Procedures of Statistics: A Biometrical Approach*, WCB/McGraw-Hill, 1997.

[89] Sundstrom, K., Rued, A., McLeod, R. D., "Internet telephony compression algorithms," Conference Proceedings IEEE Communications, Power and Computing, pp 13-18, 1997.

[90] Suzuki, J., and Taka, M., "Missing Packet Recovery Techniques for Low Bit Rate Coded Speech", IEEE Joint Selected Areas Communications, SAC-7, pp 707-717, June 1989.

[91] Suzuki, T., Noguchi, O., Yokota, K, Shoji, Y., "A New Speech Processing Scheme for ATM Switching Systems", IEEE International Conference on Communications, Volume 3, pp 1515-1519, 1989.

[92] Tanabe, N., Farvardin, N., "Subband image coding using entropy-coded quantization over noisy channels," IEEE Journal on Selected Areas in Communications, Volume 10, Issue 5, Pp 926-943, June 1992.

[93] Theil, H., "On the Estimation of Relationships Involving Qualitative Variables," American Journal of Sociology, Volume 76, Issue 1, pp 103-154, July 1970.

[94] Turner, L., Aoyam, T., Pearson, D., Anastassiou, D., Minami, T., "Guest Editorial Packet Speech and Video," IEEE Journal on Selected Areas in Communications, Volume 7, Number 5, June 1989.

[95] Valenzuela, R.A., Animalu, C.N., "A New Voice-Packet Reconstruction Technique", Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing, pp 1334-1336, 1989.

[96] van Buuren, S., van Mulligen, E.M., Brand, J.P.L., "Routine Multiple Imputation in Statistical Databases," International Working Conference on Scientific and Statistical Database Management, pp 74-78, September 1994.

[97] Walrand, J., *Communications Networks: A First Course*, Aksen Associates, Boston, MA, 1991.

[98] Wasem, O. J., Goodman, D. J., Dvorak, C. A., Page, H. G., "The Effect of Waveform Substitution on the Quality of PCM Packet Communications", IEEE Transactions on Acoustics Speech and Signal Processing, Volume ASSP-36, Number 3, March 1988.

[99] Wong, W.C., Goodman, D. J., "Blank and Burst Transmission of Data Over Active Speech Channels", IEEE Electronics Letters, Volume 24, Number 11, May 1988.

[100] Yajnik, M., Kurose, J., Towsley, D., "Packet loss correlation in the Mbone multicast network," Proceedings of Global Telecommunications Conference, 1996, pp 94-99, November 1999.

[101] Yarlagadda, R., Hershey, J., "A Naturalness-Preserving Transform for Image Coding and Reconstruction", IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume ASSP-33, Number 4, August 1985.

[102] Yarlagadda, R., and Hershey, J., *Hadamard Matrix Analysis and Synthesis*, Kluwer Academic Publishers, 1997.

[103] Yates, R. D., Goodman, D. J., *Probability and Stochastic Processes*, John Wiley and Sons, New York, 1999.

[104] Yin, N., Li, S.Q., Stern, T., "Congestion control for packet voice by selective packet discarding," IEE Trans Communications, Volume COM-38, pp 674-683, May 1990.

[105] Yong, M., "Study of voice packet reconstruction methods applied to CELP speech coding," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 2, pp 125-128, March 1992.

[106] Yuk, S., Ryu, J., Lee, K,, Cho, D., "Multiclass rate control mechanism for prioritised voice service in packet networks, " Electronics Letters, Volume 35, Issue 18, pp 1519-1520, September 1999.

[107] Zagursky, V., Ginters, E,, "Speech signal recovery in packet-switched communication networks," Proceedings of the 1999 IEEE Systems Readiness Technology Conference, pp 179-182, September 1999.

APPENDIX

Calculation of Naturalness Preserving Transform Complexity, Number of Operations

It is necessary to expand the NPT into its component matrices before analyzing the number of operations.

$$\begin{aligned}\Pi_n &= \Psi_n(\alpha)\mathbf{P}_n\Psi_n(\alpha) \\ &= [\alpha\mathbf{I}_n + (1-\alpha)\mathbf{H}'_n]\mathbf{P}_n[\alpha\mathbf{I}_n + (1-\alpha)\mathbf{H}'_n] \\ &= \alpha\mathbf{I}_n\mathbf{P}_n\alpha\mathbf{I}_n + \alpha\mathbf{I}_n\mathbf{P}_n(1-\alpha)\mathbf{H}'_n + (1-\alpha)\mathbf{H}'_n\mathbf{P}_n\alpha\mathbf{I}_n + (1-\alpha)\mathbf{H}'_n\mathbf{P}_n(1-\alpha)\mathbf{H}'_n \\ &= \alpha^2\mathbf{P}_n + \alpha(1-\alpha)\mathbf{P}_n\mathbf{H}'_n + \alpha(1-\alpha)\mathbf{H}'_n\mathbf{P}_n + (1-\alpha)^2\mathbf{H}'_n\mathbf{P}_n\mathbf{H}'_n\end{aligned} \quad (A.1)$$

The transform algorithm can save operations by normalizing the Hadamard matrix, $\mathbf{H}'_n$, after the matrix is combined with $\mathbf{P}_n$. The transform then becomes

$$\Pi_n = \alpha^2\mathbf{P}_n + 2^{-n/2}\alpha(1-\alpha)(\mathbf{P}_n\mathbf{H}_n + \mathbf{H}_n\mathbf{P}_n) + 2^{-n}(1-\alpha)^2\mathbf{H}_n\mathbf{P}_n\mathbf{H}_n\,. \quad (A.2)$$

Since the entries of $\mathbf{H}_n$ are all ±1, matrix multiplication is reduced to only addition and subtraction. The algorithm can save further operations by pre-calculating the following variables and storing them, since they remain constant with regard to changes in P_n:

$$\alpha^2,$$

$$2^{-n/2}\alpha(1-\alpha),$$

and $$2^{-n}(1-\alpha)^2.$$

Table A.1 shows the calculations needed for each iteration when the above modifications are applied to the transform. Note that the matrices $\mathbf{H}_n$, $\mathbf{P}_n$ are both $N\times N$ matrices where $N = 2^n$.

Calculation	$\times$	$+$
$\alpha^2\mathbf{P}_n$	N^2	
$\mathbf{P}_n\mathbf{H}_n$		$(N-1)N^2$
$\mathbf{H}_n\mathbf{P}_n$		$(N-1)N^2$
$\mathbf{P}_n\mathbf{H}_n+\mathbf{H}_n\mathbf{P}_n$		N^2
$scalar\cdot(\mathbf{P}_n\mathbf{H}_n+\mathbf{H}_n\mathbf{P}_n)$	N^2	
$\mathbf{H}_n\mathbf{P}_n\mathbf{H}_n$		$2(N-1)N^2$
$scalar\cdot\mathbf{H}_n\mathbf{P}_n\mathbf{H}_n$	N^2	
Add 3 matrices together		$2N^2$
Total	$3N^2$	$(4(N-1)+3)N^2$

Table A.1 Number of Operations for Transformation

Further reduction can be achieved by using the root, $\mathbf{A}$, of $\mathbf{H}_n$. $\mathbf{A}$ is a matrix of the same size as $\mathbf{H}_n$ with 2^n-2 zeros and two ± 1's in each row and column. When $\mathbf{A}$ is multiplied by an $N\times N$ (= $2^n\times 2^n$) matrix, 2^{2n} additions result. Since $\mathbf{H}_n=\mathbf{A}^n$, it follows that $\mathbf{H}_n\mathbf{P}_n=\mathbf{A}^n\mathbf{P}_n$. This operation requires $n\cdot 2^{2n}$ additions. Table A.2 shows the number of calculations required for each iteration using the root of $\mathbf{H}_n$. For simplicity,

$$\xi=2^{\frac{-n}{2}}\alpha(1-\alpha) \tag{A.3}$$

and

$$\delta=2^{-n}(1-\alpha)^2 \tag{A.4}$$

in this table.

Calculation	$\times$	$+$
$\alpha^2 P_n$	2^{2n}	0
$\xi(\mathbf{P}_n\mathbf{H}_n + \mathbf{H}_n\mathbf{P}_n)$	2^{2n}	$2n \cdot 2^{2n} \cdot 2^{2n}$
$\delta\mathbf{H}_n\mathbf{P}_n\mathbf{H}_n$	2^{2n}	$2n \cdot 2^{2n}$
Total	$3 \cdot 2^{2n}$	$(4n+3)2^{2n}$

Table A.2 Number of Operations for Transformation, Reduced

The number of operations for the inverse transformation is equal to the number of operations for the transformation.

Calculation of Naturalness Preserving Transform Complexity, Memory

The calculation to determine the amount of memory needed to perform the transform will also use the expanded form of the NPT from (A.2). This equation is divided into three parts:

$$\alpha^2 \mathbf{P}_n , \tag{A.5}$$

$$\xi(\mathbf{P}_n \mathbf{H}_n + \mathbf{H}_n \mathbf{P}_n) , \tag{A.6}$$

and

$$\delta \mathbf{H}_n \mathbf{P}_n \mathbf{H}_n . \tag{A.7}$$

Where ξ and δ have the same value as defined in (A.3) and (A.4). Since different processors have varying numbers of bytes per word of memory, all calculations will be based on the number of memory words, assuming that every variable or matrix element is stored in one word of memory. Each of the matrices $(\mathbf{H}_n, \mathbf{P}_n)$ and their products and sums require $N^2 = 2^{2n}$ words and each of the scalar calculations require one word. Table A.3 presents the amount of memory occupied by the NPT.

Term	Memory Words	
$\alpha^2 \mathbf{P}_n$	N^2	
α^2	1	Reusable
$\mathbf{P}_n$	N^2	
$\xi(\mathbf{P}_n\mathbf{H}_n + \mathbf{H}_n\mathbf{P}_n)$	N^2	Reusable after adding to $\alpha^2 P_n$
ξ	0	Use reusable memory
$\mathbf{P}_n$	0	Already Stored
$\mathbf{H}_n$	N^2	
$\mathbf{P}_n\mathbf{H}_n$	N^2	Reusable
$\mathbf{H}_n\mathbf{P}_n$	N^2	Reusable
$\delta\mathbf{H}_n\mathbf{P}_n\mathbf{H}_n$	0	Use reusable memory
δ	0	Use reusable memory
$\mathbf{H}_n$	0	Use reusable memory
$\mathbf{P}_n$	0	Use reusable memory
$\mathbf{H}_n\mathbf{P}_n$	0	Use reusable memory
$\mathbf{H}_n\mathbf{P}_n\mathbf{H}_n$	0	Use reusable memory
Total	$6N^2 + 1$	

Table A.3 NPT Memory Requirements

Calculation of Mixture Transition Distribution Complexity, Number of Operations

Training the MTD and using the MTD require different numbers of operations. Training is performed once to compute the lag weights and transition matrix needed for calculating the conditional probabilities. MTD is used operationally when the conditional probabilities are calculated for a frame of missing speech. The number of operations required for training is dependent on the amount of training data. The number of operations required for operational use is dependent on the number of states and the Markov chain order. Training requires far more resources than utilization.

Training

Training involves two separate steps: initialization and optimization. If the correct initial values are used, the optimization process will converge to the parameters with the global maximum likelihood. The following analysis calculates the number of operations separately for each step, and provides measurements from training for two vocoders.

Initialization

The initialization process is as follows: 1) Find all states in the training data; 2) Calculate the initial lag weights; 3) Calculate the initial transition matrix; 4) Calculate the initial log-likelihood. Figure A.1 illustrates these steps.

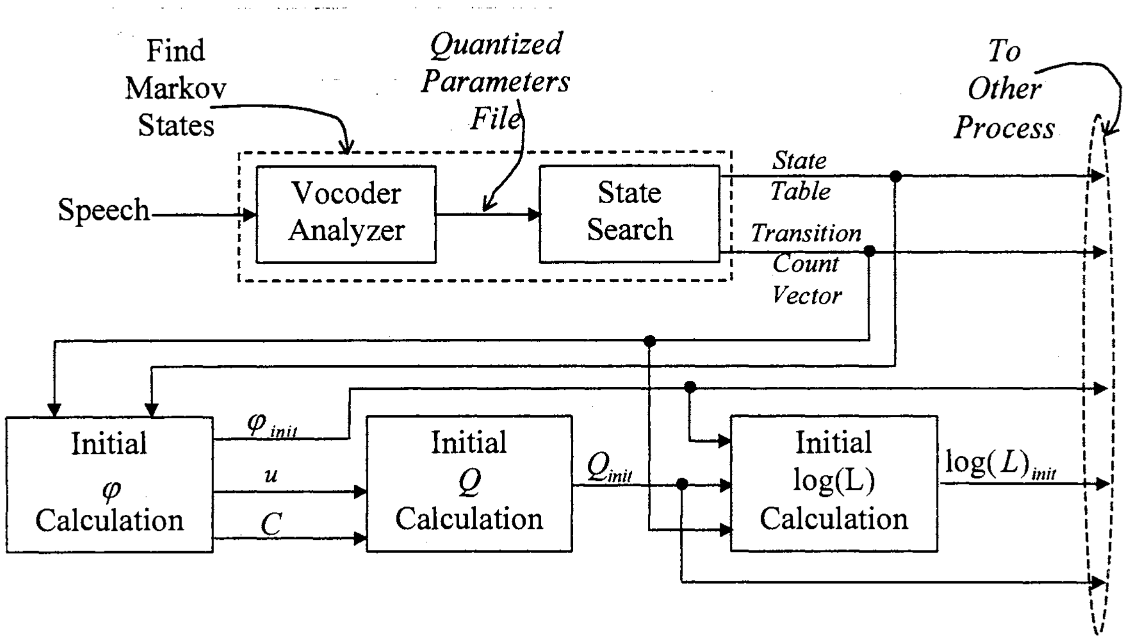


Figure A.1 MTD Initialization Procedure

Each of the initialization activities shown in this figure is analyzed separately in the following sections.

DETERMINATION OF MARKOV STATES

The initialization algorithm finds the states by first running the analyzer of the voice coding algorithm on all the training data. The vocoder analyzer creates a bit-file containing a frame of quantized parameters for each frame of speech. The initialization procedure then looks at each frame of parameters in the bit-file. For each frame, it extracts the bits corresponding to the parameters that comprise the state. If the frame contains a parameter combination that has not yet occurred, the combination is added to the state table.

The number of operations required for this procedure is dominated by the number of operations required for the vocoder analyzer and the amount of training data. Once the vocoder creates the bit files, the parameter extraction procedure uses a table to determine which bits to extract from each frame of data. Each parameter combination is compared to the existing state table to determine if it is a new state. As the state table grows, the comparison requires more operations. When a new state is found in the training data, the state number and parameters are added to the matrix. The operations needed to perform these steps are mainly programming constructs (e.g., looping, indexing).

Calculation of Initial Lag Weights

Both the initial lag weight and the initial transition matrix calculations require Theil's association measure. The association measure calculation uses the $m \times S \times S$ cross-table. The number of operations performed to create the cross-table is dependent on the amount of training data. For each frame of training data, the parameter combination must be extracted using a table lookup; the state number corresponding to this parameter must be extracted, also using a table lookup; and m transitions must be tallied in the cross-table (one addition per transition per order).

Once the cross-table is created, the initialization algorithm calculates the m association measures using

$$u_g = \frac{\sum_{i=1}^{S}\sum_{j=1}^{S} C_g(i,j)\log_2\left(\frac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)}{\sum_{j=1}^{S} C_g(\cdot,j)\log_2\left(\frac{C_g(\cdot,j)}{TC_g}\right)} . \tag{A.8}$$

The number of operations for this equation is dependent on the number of states. The equation contains two logarithmic calculations. Unless special purpose processors are used, logarithms are calculated from series expansion. The logarithmic series

$$\ln(x) = 2\left[\frac{x-1}{x+1} + \frac{1}{3}\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}\left(\frac{x-1}{x+1}\right)^5 + \cdots\right] \tag{A.9}$$

can be used to calculate the natural logarithm. It requires two additions, one multiply, and a division for the first term, and for each additional term, one addition and three multiplies (assuming the fraction $(x-1)/(x+1)$ and its powers are calculated once and saved for the next term). Another division is needed to get the base-two logarithm of the argument (assuming $\ln(2)$ is stored as a constant) bringing the total number of operations for n terms to $2+n$ additions, $1+3n$ multiplies, and two divisions. The number of terms required in the series expansion is dependent on the magnitude of the argument and the error threshold. As $x \to 0$ and $x \to \infty$, the number of terms necessary to maintain a given error increases. Large logarithmic arguments are not a concern for the association measure calculation, but the argument may be small. Six hundred terms is sufficient to compute $\ln(0.003)$ to within 0.0001 of the true value, so $n = 600$ will be used for this analysis.

The number of operations required to compute each of the m association measures is listed in Table A.4.

Operation	+	×	÷	
Numerator: $C_g(i,\cdot)$	S^2			S additions for each $\sum_{i=1}^{S}$
$C_g(\cdot,j)$	S^2			S additions for each $\sum_{j=1}^{S}$
$C_g(i,\cdot)C_g(\cdot,j)$		S^2		1 multiply for each $\sum_{i=1}^{S}\sum_{j=1}^{S}$
TC_g	S			Can calculate using $\sum_{i=1}^{S}C_g(i,\cdot)$ or $\sum_{j=1}^{S}C_g(\cdot,j)$, calculate once and reuse.
$C_g(i,j)TC_g$		S^2		1 multiply for each $\sum_{i=1}^{S}\sum_{j=1}^{S}$
$\dfrac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}$			S^2	1 divide for each $\sum_{i=1}^{S}\sum_{j=1}^{S}$
$\log 2\left(\dfrac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)$	$602S^2$	$1801S^2$	$2S^2$	Calculate for each $\sum_{i=1}^{S}\sum_{j=1}^{S}$
$C_g(i,j)\log_2\left(\dfrac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)$		S^2		1 multiply for each $\sum_{i=1}^{S}\sum_{j=1}^{S}$
$\sum_{i=1}^{S}\sum_{j=1}^{S}C_g(i,j)\log_2\left(\dfrac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)$	S^2			
Denominator: $C_g(\cdot,j)$				Use numerator calculation
TC_g				Use numerator calculation
$\dfrac{C_g(\cdot,j)}{TC_g}$			S	1 divide for each $\sum_{j=1}^{S}$
$\log 2\left(\dfrac{C_g(\cdot,j)}{TC_g}\right)$	$602S$	$1801S$	$2\,S$	Calculate for each $\sum_{j=1}^{S}$

Operation	+	×	÷	
$C_g(\cdot,j)\log 2\left(\frac{C_g(\cdot,j)}{TC_g}\right)$		S		1 multiply for each $\sum_{j=1}^{S}$
$\sum_{j=1}^{S} C_g(\cdot,j)\log_2\left(\frac{C_g(\cdot,j)}{TC_g}\right)$	S			
Total	$605S^2$ $+604S$	$1804S^2$ $+1802S$	$3S^2$ $+3S$	

Table A.4 Number of Operations for Computation of each MTD Association Measure

More efficient methods of computing logarithms will have the greatest impact on this operation.

The initial lag weights are calculated from the association measures using

$$\varphi_g = \frac{u_g}{\sum_{k=1}^{m} u_k}. \tag{A.10}$$

This calculation is dependent on the Markov chain order, m. The denominator requires m additions, but the resulting sum can be saved and reused for the remainder of the lag weight calculations. One divide is also needed for each lag weight. The total number of calculations needed to calculate all the initial lag weights is m adds and m divides.

Calculation of Initial Transition Matrix

Calculation of the initial transition matrix depends on parameters calculated for the initial lag weights, i.e., the cross-tables and association measures. The association measure determines which cross-table to use in

$$Q_g = \begin{bmatrix} \frac{C_g(1,1)}{C_g(1,\cdot)} & \cdots & \frac{C_g(1,S)}{C_g(1,\cdot)} \\ \vdots & \ddots & \vdots \\ \frac{C_g(S,1)}{C_g(S,\cdot)} & \cdots & \frac{C_g(S,S)}{C_g(S,\cdot)} \end{bmatrix} . \tag{A.11}$$

This equation is dependent on the number of states. Some computation can be saved by re-using row sums from the association measure calculation. If the row sums are not recalculated, the computation requires S^2 divides.

Calculation of Initial Log-Likelihood

The initial log-likelihood is needed as a baseline for the optimization procedure. It is calculated using

$$\log L = \sum_{n(i_0,i_1,\ldots,i_m)} n(i_0,i_1,\ldots,i_m) \log p(i_0|i_1,\ldots,i_m) . \tag{A.12}$$

The sum is over all *m*th order transitions that occur at least once, so the calculation is dependent on the number of *m*th order transitions found in the training data, *N*. The conditional probability in this equation is computed using

$$p(i_0|i_1\ldots i_m) = \sum_{j=1}^{m} \varphi_j q(i_0|i_j) . \tag{A.13}$$

One multiply is needed for each of the *m* sums. The number of operations needed to calculated the conditional probability is *m* additions and *m* multiplies. Table A.5 shows the number of operations required to calculate log-likelihood.

Operation	+	×	÷	
$p(i_0 \| i_1 \ldots i_m)$	Nm	Nm		m additions and m multiplies for each $\sum\limits_{n()>0}$
$\log p(i_0 \| i_1 \ldots i_m)$	$602N$	$1801N$	$2N$	Calculate for each $\sum\limits_{n()>0}$
$n(i_0, i_1, \ldots, i_m) \log p(i_0 \| i_1 \ldots i_m)$		N		Calculate for each $\sum\limits_{n()>0}$
$\sum\limits_{n(i_0, i_1, \ldots, i_m)>0} n(i_0, i_1, \ldots, i_m) \log p(i_0 \| i_1 \ldots i_m)$	N			
Total	$603N + Nm$	$1802N + Nm$	$2N$	

Table A.5 Number of Operations for Log-likelihood Calculation

A more efficient method for computing logarithms will decrease processor requirements.

TOTAL NUMBER OF OPERATIONS FOR INITIALIZATION

Once the initialization procedure calculates the Markov states from the vocoder bitstream and creates the cross-table, the initialization procedure requires $m(605S^2 + 604S + 1) + N(603 + m)$ additions, $m(1804S^2 + 1802S) + N(1802 + m)$ multiplies, and $m(3S(S+1)+1) + S^2 + 2N$ divides, where S is the number of states, m is the Markov chain order, and N is the number of mth order transitions existing in the training data. This number excludes manipulation of data, error checking, and programming constructs.

Implementation Measurements

A better understanding of the number of operations required for initialization can be obtained by observing actual implementations. The MELP second-order MTD implementation contained 34,591 transitions ($m = 2, N = 34{,}591$) from 52,293 frames of training data. The initialization algorithm required 60 minutes on a 400 MHz Pentium II processor to calculate the initial lag weights and transition matrix for 2209 states ($S = 2209$) using Matlab routines. The GSM second-order MTD implementation contained 219,354 transitions ($m = 2, N = 219{,}354$) from 244,908 sub-frames of training data. The initialization algorithm required 297 minutes (4.96 hours) on a 400 MHz Pentium II processor to calculate the initial lag weights and transition matrix for 324 states ($S = 324$) using Matlab routines. In addition to processing the data, the initialization routines for both voice-coding algorithms performed error checking and wrote status information to the display and to digital files. It's apparent from these measurements that the number of *m*th order transitions has a greater effect on the number of operations than the number of states.

Optimization

The number of operations required for the optimization algorithm depends on the number of iterations required to meet the user-defined log-likelihood improvement threshold. For each iteration, the optimization algorithm performs the following procedure: 1) Calculate partial derivatives for the lag weights; 2) Apply and evaluate changes in lag weights; 3) Calculate partial derivatives for each row of the transition matrix; 4) Apply and evaluate changes in the transition matrix rows.

The partial derivative for the lag weights is calculated using

$$\frac{\partial \log(L)}{\partial \varphi_k} = \sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m)\frac{q_{i_k i_0}}{p(i_0|i_1,\ldots,i_m)}. \tag{A.14}$$

The number of operations needed for this computation is dependent on the Markov chain order, *m*, and on the number of *m*th-order state transitions, *N*, in the training data. Table A.6 shows the number of operations for this calculation.

Operation	$+$	$\times$	$\div$	
$p(i_0\|i_1,\ldots,i_m)$	Nm	Nm		*m* additions, *m* multiplies for each $\sum_{n(i_0,\ldots,i_m)}$
$\frac{q_{i_k i_0}}{p(i_0\|i_1,\ldots,i_m)}$			N	1 divide for each $\sum_{n(i_0,\ldots,i_m)}$
$n(i_0,\ldots,i_m)\frac{q_{i_k i_0}}{p(i_0\|i_1,\ldots,i_m)}$		N		1 multiply for each $\sum_{n(i_0,\ldots,i_m)}$
$\sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m)\frac{q_{i_k i_0}}{p(i_0\|i_1,\ldots,i_m)}$	N			1 addition for each $\sum_{n(i_0,\ldots,i_m)}$
Total	$N(m+1)$	$N(m+1)$	N	

Table A.6 Number of Operations for Lag Weight Partial Derivative

The partial derivative for each row of the transition matrix is calculated using

$$\frac{\partial \log(L)}{\partial q_{i_k i_0}} = \sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m)\frac{\varphi_k}{p(i_0|i_1,\ldots,i_m)}. \tag{A.15}$$

This calculation is the same form as the partial derivative for the lag weights and requires the same number of operations, $N(m+1)$ additions, $N(m+1)$ multiplies, and N divides per row. The number of operations to calculate the partial derivative for the entire transition matrix is therefore $SN(m+1)$ additions, $SN(m+1)$ multiplies, and SN divides.

The modification for both the lag weights and the rows of the transition matrix follows the same procedure: the element corresponding to the maximum partial derivative is increased by δ; the element corresponding to the minimum partial derivative is decreased by δ; the log-likelihood is calculated with the modified elements; if the log-likelihood is improved, the next set is modified; if the log-likelihood is not improved, the procedure is repeated with $\frac{1}{2}\delta$. The element modification requires two additions. The log-likelihood calculation, calculated previously, is dependent on the Markov chain order, *m*, and the number of *m*th order transitions in the training data, *N*. It requires $N(603+m)$ additions, $N(1802+m)$ multiplies, and $2N$ divides and obviously dominates the element modification process.

Total Number of Operations for Optimization

Each iteration of the optimization process requires $N(m(S+2)+S+604)+2$ additions, $N(m(S+2)+S+1803)$ multiplies, and $N(S+3)$ divides where m is the Markov chain order, N is the number of *m*th order transitions in the training data, and S is the number of states. This estimate does not include data manipulation, error checking, or programming constructs.

Implementation Measurements

A better understanding of the number of operations required for optimization can be obtained by observing actual implementations. The MELP second-order MTD implementation contained 34,591 transitions ($m=2, N=34{,}591$) from 52,293 frames of training data. The optimization algorithm ran for 91.5 hours on a 400 MHz Pentium II processor using a DOS executable written in C-code to reach a log-likelihood change threshold of 0.1. The GSM second-order MTD implementation contained 219,354 transitions $(m = 2, N = 219{,}354)$ from 244,908 sub-frames of training data. The optimization algorithm ran for 5 hours on a 400 MHz Pentium II processor using a DOS executable written in C-code to reach a log-likelihood change threshold of 0.1. In addition to processing the data, the optimization routines for both voice coding algorithms performed error checking and wrote status information to the display and to digital files. It's apparent from these measurements that the number of states has a greater effect on the number of operations than the number of mth order transitions. However, convergence is dependent on the nature of the data and is probably the most dominant factor.

Utilization

Once the initialization algorithm calculates the optimal lag weights and transition matrix, the implementation is comparatively simple. When the receiver detects a missing frame, the probability for each of the possible next states is calculated using (A.13). The number of operations for the conditional probability was calculated earlier to be m adds and m multiplies, so calculation of all S possible replacement states requires Sm adds and Sm multiplies to replace each missing frame. Creating a lookup table that contains the state with the highest probability for each mth order transition will decrease the number of operations.

Calculation of Mixture Transition Distribution Complexity, Memory

Both MTD training and utilization require data storage. Training is performed once to create the lag weights and transition matrix needed to calculate the conditional probabilities. MTD is used operationally when the conditional probabilities are calculated for a frame of missing speech. Training requires far more resources than utilization. Since different processors have varying number of bytes per word of memory, all calculations will be based on the number of memory words, assuming that every variable or matrix element is stored in one word of memory.

Training

Training involves two separate steps: initialization and optimization. If the correct initial values are used, the optimization process will converge to the parameters with the global maximum likelihood. The following analysis calculates the memory separately for each step.

Initialization

The initialization process performs the following procedure: 1) Find all states in the training data; 2) Calculate the initial lag weights; 3) Calculate the initial transition matrix; 4) Calculate the initial log-likelihood. This procedure is illustrated in Figure A.1. Each of the initialization activities are analyzed separately.

DETERMINATION OF MARKOV STATES

The initialization algorithm finds the states by first running the voice coding algorithm's analyzer on all the training data. The vocoder analyzer creates a bit-file containing a frame of quantized parameters for each frame of speech. The initialization procedure then looks at each frame of parameters in the bit-file. For each frame, it extracts the bits corresponding to the parameters that comprise the state. If the frame contains a parameter combination that has not yet occurred, the combination is added to the state table.

The amount of memory required for this procedure is driven by the amount of memory required for the vocoder analyzer, the number of parameters, d, that comprise a state, and the amount of training data. As the number of parameters increases, the number of possible parameter combinations (and therefore the number of states) increases. As the amount of training data increases, the number of bit-files increases and the number of states can also increase. If the initialization process uses a sufficient number of training data, the number of states will reach the maximum number of realistic combinations. The state table is an $S \times d$ matrix requiring Sd memory words. The Markov state table building process also creates a state lookup table that aids in locating the state number associated with each parameter combination. One parameter is chosen to be the lookup variable in the state lookup table. The state lookup table contains every value of the lookup variable and the range of states that include it, i.e., each row contains the lookup variable value, the first state that contains it, and the last state that contains it. The amount of memory required to store the lookup table is dependent on the number of possible values for the parameter chosen as the lookup variable. If the lookup variable has b bits, the state lookup table is a $2^b \times 3$ matrix requiring $3 \cdot 2^b$ words.

Calculation of Initial Lag Weights

Both the initial lag weight and the initial transition matrix calculations require Theil's association measure. The association measure calculation requires the $m \times S \times S$ cross-table, which occupies mS^2 words of memory.

Once the cross-table is created, the initialization algorithm calculates the m association measures using (A.8). The series expansion of the logarithm (A.9) requires five words of memory to calculate, if temporary storage is reused. The amount of memory required for computing the m association measures is listed in Table A.7.

Term	Memory Words	
Numerator: $C_g(i,\cdot)$	S	
$C_g(\cdot,j)$	S	
$C_g(i,\cdot)C_g(\cdot,j)$	1	Reusable
TC_g	1	
$C_g(i,j)TC_g$	1	Reusable
$\frac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}$	1	Reusable
$\log 2\left(\frac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)$	3	Use 2 words of reusable memory, reusable
$C_g(i,j)\log_2\left(\frac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)$	0	Use reusable memory
$\sum_{i=1}^{S}\sum_{j=1}^{S}C_g(i,j)\log_2\left(\frac{C_g(i,\cdot)C_g(\cdot,j)}{C_g(i,j)TC_g}\right)$	1	
Denominator: $C_g(\cdot,j)$	0	Previously calculated
TC_g	0	Previously calculated
$\frac{C_g(\cdot,j)}{TC_g}$	0	Use reusable memory
$\log 2\left(\frac{C_g(\cdot,j)}{TC_g}\right)$	0	Use reusable memory
$C_g(\cdot,j)\log 2\left(\frac{C_g(\cdot,j)}{TC_g}\right)$	0	Use reusable memory
$\sum_{j=1}^{S}C_g(\cdot,j)\log_2\left(\frac{C_g(\cdot,j)}{TC_g}\right)$	1	
u_g	m	
Total	$S(mS+2)+m+9$	

Table A.7 Number of Memory Words for Computation of MTD Association Measure

Once the m association measures are calculated, all memory except that used to store the association measure and row sums is available for other uses.

The initial lag weights are calculated from the association measures using (A.10). The m association measures are already residing in m memory words. The denominator sum will use one word, and the lag weights will need m words. The initial lag weight calculation requires $2m+1$ words. One word will be free after this calculation is complete.

Calculation of Initial Transition Matrix

Calculation of the initial transition matrix depends on parameters calculated for the initial lag weights, i.e., the cross-tables and association measures. The association measure determines which cross-table to use in (A.11). The row sums and the cross-table elements are already stored, so the initial $S \times S$ transition matrix calculation requires the S^2 words required to store the matrix.

Calculation of Initial Log-Likelihood

The initial log-likelihood is needed as a baseline for the optimization procedure. It is calculated using (A.12). The sum is over all mth order transitions that occur at least once. The number of mth order transitions in the training data, N, can be extremely large, so Raftery suggests a labeling pattern for the state transitions by the number

$$i = 1 + \sum_{j=0}^{m}(i_j - 1)S^{m+1-j} \tag{A.16}$$

Using this pattern, the mth order transitions can be stored in an $N \times 2$ matrix. One column contains the pattern label; one column contains the number of times the transition occurred. The conditional probability in the log-likelihood calculation is computed using (A.13). This calculation requires the $S \times S$ transition matrix, and the $m \times 1$ lag weight vector. Table A.8 shows the amount of memory required to calculate the log-likelihood.

Term	Memory Words	
$p(i_0 \vert i_1 \ldots i_m)$	$S^2 + m + 1$	
$\log p(i_0 \vert i_1 \ldots i_m)$	5	Reusable
$n(i_0, i_1, \ldots, i_m)$	$2N$	
$n(i_0, i_1, \ldots, i_m) \log p(i_0 \vert i_1 \ldots i_m)$	0	Use reusable memory
$\sum_{n(i_0, i_1, \ldots, i_m) > 0} n(i_0, i_1, \ldots, i_m) \log p(i_0 \vert i_1 \ldots i_m)$	1	
Total	$S^2 + 2N + m + 7$	

Table A.8 Amount of Memory for Log-Likelihood Calculation

Seven words of memory are free after this calculation.

TOTAL AMOUNT OF MEMORY FOR INITIALIZATION

Much of the memory required for initialization is used to store the transition matrix (Q), cross-tables (C_g), and transition counts ($n(i_0, \ldots, i_m)$). These matrices are calculated once and reused throughout the initialization process. Memory used for temporary storage is freed after each calculation and can be used by subsequent calculations. So the total amount of memory is less than the sum of the number of words for the individual parts. The total amount of memory required for initialization is $S^2(m+1) + S(2+d) + 2(m+N) + 3 \cdot 2^b + 9$ words, where S is the number of states, m is the Markov chain order, d is the number of parameters in the state, N is the number of *m*th order transitions, and *b* is the number of bits in the lookup parameter. This number excludes manipulation of data, error checking, and programming constructs.

Optimization

The optimization process takes the initial lag weights, transition matrix, and matrix of *m*th order transitions and creates new lag weights and transition matrix which have the maximum log-likelihood. For each iteration, the optimization algorithm performs the following steps: 1) Calculate partial derivatives for the lag weights; 2) Apply and evaluate changes in lag weights; 3) Calculate partial derivatives for each row of the transition matrix; 4) Apply and evaluate changes in the transition matrix rows.

The partial derivative for the lag weights is calculated using (A.14). This calculation uses the $S \times S$ transition matrix and the $N \times 2$ transition count matrix. Table A.9 shows the amount of memory required for this computation.

Terms	**Memory Words**	
$p(i_0 \mid i_1,\ldots,i_m)$	S^2+m+1	1 word is reusable
$\dfrac{q_{i_k i_0}}{p(i_0 \mid i_1,\ldots,i_m)}$	1	Reusable
$n(i_0,\ldots,i_m)\dfrac{q_{i_k i_0}}{p(i_0 \mid i_1,\ldots,i_m)}$	$2N$	Reuse 1 word
$\sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m)\dfrac{q_{i_k i_0}}{p(i_0 \mid i_1,\ldots,i_m)}$	m	
Total	$S^2+2(N+m+1)$	

Table A.9 Amount of Memory for Lag Weight Partial Derivative

All memory except the transition matrix and lag weights is free after this calculation.

The partial derivative for each row of the transition matrix is calculated using (A.15). The calculation uses the m lag weights and the $N \times 2$ transition count matrix. Table A.10 shows the amount of memory required for this computation.

Terms	Memory Words	
$p(i_0 \mid i_1,\ldots,i_m)$	S^2+m+1	1 word is reusable
$\dfrac{\varphi_k}{p(i_0 \mid i_1,\ldots,i_m)}$	1	Reusable
$n(i_0,\ldots,i_m)\dfrac{\varphi_k}{p(i_0 \mid i_1,\ldots,i_m)}$	$2N$	Reuse 1 word
$\sum_{n(i_0,\ldots,i_m)} n(i_0,\ldots,i_m)\dfrac{\varphi_k}{p(i_0 \mid i_1,\ldots,i_m)}$	S	
Total	$+2(N+1)+S(1+S)$	

Table A.10 Amount of Memory for Transition Row Partial Derivative

After this calculation is complete, $S+2$ words of memory are free.

The optimization algorithm analyzes the partial derivative vectors to determine which elements of the lag weight vector and transition matrix to change. The memory required for the analysis is largely taken with programming constructs. It uses the same transition matrix, lag weights, and transition count vector (S^2+m+2N words of shared memory) that were used for both partial derivatives.

Much of the memory used for the optimization process is occupied by the lag weights, transition matrix, and transition count matrix. These matrices are calculated once and reused throughout the optimization process. Memory used for temporary storage is freed after each calculation and can be used by subsequent calculations. So the total amount of memory required to optimize the lag weights and transition matrix is $2(S^2+m+N)$.

Utilization

Once the optimal lag weights and transition matrix are calculated, the implementation is comparatively simple. When the receiver detects a missing frame, the probability for each of the possible next states is calculated using (A.13). This calculation has already been shown to require $S^2 + m + 1$ words of memory. The lookup table suggested in the discussion on number of operations will occupy S^m memory words, but will eliminate the transition table and lag weight storage.

VITA

Mary A. Kohler

Candidate for the Degree of

Doctor of Philosophy

Thesis: COMPENSATION FOR MISSING VOICE CODING FRAMES IN PACKET TRANSMISSION SYSTEMS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Clermont, Florida, On April 7, 1964, the daughter of William J. and Marilyn P. Priebe.

Education: Graduated from Clermont High School, Clermont, Florida in May 1982; received Bachelor of Science degree in computer engineering from Clemson University, Clemson, South Carolina in May 1986; received Master of Science degree in electrical engineering from Johns Hopkins University, Baltimore, Maryland in May 1994. Completed requirements for Doctor of Philosophy degree with a major in Electrical Engineering at Oklahoma State University in May 2000.

Experience: Raised on a farm in Clermont, Florida; employed as computer programmer for Premore Groves during school breaks; employed by McDonald's as a clerk; employed by Florida Power Corporation as a summer intern; employed by United States Department of Defense as a senior electrical engineer.

Professional Memberships: Institution of Electrical and Electronic Engineers, Tau Beta Pi, Sigma Xi