A Very low bit-rate speech recognition system

Matthew Baecher
A Very Low Bit-Rate Speech Recognition System

By

Matthew B. Baecher

A Thesis Submitted

in

Partial Fulfillment

of the

Requirements for the Degree of

MASTER OF SCIENCE

in

Electrical Engineering

Approved by:

PROF. Vincent Amuso

Thesis Advisor)

PROF.

PROF.

PROF.

(Department Head)

DEPARTMENT OF ELECTRICAL ENGINEERING

COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

OCTOBER, 2001
Reproduction Permission

I, Matthew B. Baecher, would like to be notified before any reproduction is made in reference to the intended purpose of the reproduction.

matthew_baecher@hotmail.com
© Copyright 2001
Matthew B. Baecher
All Rights Reserved
ACKNOWLEDGMENTS

I would like to thank my family for being a constant source of love and support throughout my entire life.

Thank you Dr. Amuso, for your help and guidance in this whole thesis process.

Thanks to my best friends, Phil and Chris, for showing me what it means to set goals and never give up.

Finally, I’d like to specially thank my younger sister, Carolyn. You’ve really had more of an effect on helping me grow up than you know.
ABSTRACT

When using extracted speech feature coefficients for speech synthesis, quantization is considered a lossy compression scheme. The data being compressed cannot be recovered or reconstructed exactly. However, in a speech recognition system for command and control purposes, a certain amount of quantization can be allowed, with comparable results. In some cases, quantization even serves to “close the gaps” between the coefficients of the incoming speech signal and those of the templates. Since the coefficients are not being used to reconstruct the signal, a very coarse quantization can be used, enabling a very low bit-rate transmission with very good recognition results.

To reduce the bandwidth further, a binary coding procedure, such as Huffman or Arithmetic Coding, can be applied to the quantized coefficients. Upon receipt of the transmission, the quantized coefficients are decoded and used to perform speech recognition. The sets of coefficients are compared to the templates for each of the commands in the vocabulary. Speech, however, is dynamic in nature and a dynamic recognition procedure is needed to allow for different vocal inflections and durations. A procedure called Dynamic Time Warping is used to “warp” the time axis of the templates to more closely fit the information coming in.

By combining all these techniques, a very accurate, very low bit-rate recognizer has been developed and is discussed in this paper.
TABLE OF CONTENTS

Acknowledgments ........................................................................................................ ii
Abstract ..................................................................................................................... iii
Table of Contents ....................................................................................................... iv
List of Tables ............................................................................................................... vi
List of Figures ............................................................................................................ vii

CHAPTER 1: Introduction ............................................................................................ 2

CHAPTER 2: Literature Review .................................................................................. 4
Section 2.1: Low Bit-Rate Speech Transmission ......................................................... 4
Section 2.2: Quantization .......................................................................................... 5
Section 2.3: Arithmetic Coding ................................................................................ 5
Section 2.4: Dynamic Time Warping ........................................................................ 6

CHAPTER 3: Pattern Recognition ................................................................................ 8

CHAPTER 4: Signal Feature Measurement Techniques ............................................. 10
Section 4.1: Linear Predictive Coding ...................................................................... 10
  Section 4.1.1: LPC Theory ..................................................................................... 10
  Section 4.1.2: Linear Predictive Coefficient Coding ............................................. 14
Section 4.2: Frequency Domain Techniques ............................................................. 15
  Section 4.2.1: Discrete Fourier Transform ............................................................. 15
  Section 4.2.2: Power Spectral Density .................................................................. 15
  Section 4.2.3: Cepstral Coefficients .................................................................... 16
Section 4.3: Recognition ........................................................................................... 17
CHAPTER 5: Recognizer .................................................................  18
  Section 5.1: Thresholding .......................................................  18
  Section 5.2: Training .............................................................  19

CHAPTER 6: Compression Techniques ...........................................  20
  Section 6.1: Quantization .....................................................  20
  Section 6.2: Coding Techniques ...........................................  20
    Section 6.2.1: Arithmetic Coding .......................................  20
    Section 6.2.2: Arithmetic Decoding .....................................  22
    Section 6.2.3: Adaptive Huffman Coding ...............................  24

CHAPTER 7: Dynamic Time Warping .............................................  27

CHAPTER 8: Results .................................................................  36
LIST OF TABLES

Table 6-1: Huffman Codes ................................................................. 25
Table 8-1: ASR Matrix for No Quantization ................................. 40
Table 8-2: ASR Matrix for 64 Quantization Levels ......................... 40
Table 8-3: ASR Matrix for 32 Quantization Levels ......................... 40
Table 8-4: ASR Matrix for 16 Quantization Levels ......................... 41
Table 8-5: ASR Matrix for 8 Quantization Levels ........................ 41
Table 8-6: ASR Matrix for 64 Quantization Levels ......................... 41
LIST OF FIGURES

Figure 5-1: Thresholding for Endpoint Detection .................................................. 18
Figure 6-1: Huffman Coding Path ................................................................. 25
Figure 7-1: Dynamic Time Warping Path ......................................................... 28
Figure 7-2: Dynamic Time Warping Path Choices ........................................... 28
Figure 7-3: Local Distances in First Step of Algorithm ..................................... 31
Figure 7-4: Global Distances after First Step of Algorithm .............................. 31
Figure 7-5: Local Distances in Second Step of Algorithm ............................... 32
Figure 7-6: Global Distances after Second Step of Algorithm ......................... 32
Figure 7-7: Global Distances of First Two Columns ....................................... 33
Figure 7-8: Local Distances of All Cells ......................................................... 34
Figure 7-9: Global Distances of All Cells ......................................................... 34
Figure 7-10: Best Time Alignment Path of Frames ......................................... 35
Figure 8-1: Results For Each Digit in Linear Predictive Coefficient System .... 37
Figure 8-2: Results For Each Digit in Cepstral Coefficient System .................. 37
Figure 8-3: Achievable Bit-Rates for Each Coding Technique ......................... 39
Figure 8-4: Overall ASR Rates ........................................................................ 42
Figure 8-5: Results for Spoken Digit “Zero” ...................................................... 43
Figure 8-6: Results for Spoken Digit “One” ...................................................... 43
Figure 8-7: Results for Spoken Digit “Two” ...................................................... 44
Figure 8-8: Results for Spoken Digit “Three” .................................................... 44
Figure 8-9: Results for Spoken Digit “Four” ...................................................... 45
Figure 8-10: Results for Spoken Digit “Five” ......................... 45
Figure 8-11: Results for Spoken Digit “Six” .......................... 46
Figure 8-12: Results for Spoken Digit “Seven” ...................... 46
Figure 8-13: Results for Spoken Digit “Eight” ....................... 47
Figure 8-14: Results for Spoken Digit “Nine” ....................... 47
Missing Page
Chapter 1
Introduction

As today's technology results in smaller and smaller devices, the available space for buttons and other interactive mechanisms gets smaller as well, which makes speech recognition extremely useful for handheld technology. However, the amount of space saved through having fewer buttons would be negated by the space needed for the circuitry to perform complex speech recognition calculations and to store templates for a large vocabulary. For this reason, a system in which the feature parameters of the speech signal are transmitted before recognition would allow the benefits of speech recognition, without the need for the extra circuitry. This type of recognizer could be useful for voice dialing for cell phones, X10 control of home appliances through a PDA, and even for remote control of electronics within the household.

For the system to be useful however, it must have a low enough bit-rate to allow it to function within the limits of the bandwidth available to today's second-generation cellular technology.

The problem of lowering bit-rate has been studied extensively for speech transmission, but not for the purpose of speech recognition. In the realm of speech transmission, far less distortion is allowed to still provide useful results. Due to this fact, the bit-rate achievable for speech recognition purposes will be far lower than the bit-rate achievable for speech transmission.

Before any transmission occurs, the best speech features to extract for recognition purposes are determined. To lower the bit-rate of the recognition scheme, the extracted speech features are quantized and coded. The two major methods of binary coding are
Huffman Coding and Arithmetic Coding. These two methods are examined to determine which one can provide the lowest achievable bit-rate. To provide good results at the receiver, a pattern recognition technique known as Dynamic Time Warping is implemented. The goal of this paper is to determine the best speech features to implement, the coarsest allowable quantization, and the lowest achievable bit-rate that will provide useful results.
Section 2.1: Low Bit-Rate Speech Transmission

Richard Cox [1-5] has done an extensive amount of work on the topic of low bit-rate speech coders. His work includes a very low bit-rate speech coder that offers average bit-rates of about 800 b/s during conversation spurs. This coder implements Mel-Frequency Cepstral Coefficients (MFCCs), and uses the Euclidean distance between neighboring frames to find the best unit from a database. The unit is transmitted along with the gain, in order to reconstruct the signal at the decoder. This method provided very low bit-rate for a speech coder that could be analyzed for recognition at the decoder, but also required a large unit feature database, an undesirable consequence.

Another of Cox’s works discusses scalar quantization for mobile speech transmission. In this work the advantages of a scalar quantization technique over a vector quantization technique are discussed. Vector quantization usually becomes overly complex for the encoder, especially in mobile applications, while scalar quantization is much more efficient. Cox also has done a lot of work in forecasting how speech recognition will affect technology in the coming years.

In the work by Seiji Hayashi [6], a low bit-rate coder is proposed with a bit-rate in the range of 4 kb/s. This method uses a backward prediction algorithm, which requires a 13.5 ms coding delay. A delay of any type in a speech system is an unwanted effect, and while it is not as noticeable in recognition applications, it is still undesirable. To transmit speech using this method before recognition would be an option for a low bit-rate recognizer but would provide a greater delay at a higher bit-rate than achievable.
Mei Yong [7] uses a linear predictive coding technique to reduce the complexity of the codebook for code-linear prediction speech coders. In this work, Yong interpolates the impulse response of an LPC filter, but the coder discussed is still only of the order of 8 kb/s, a higher bit-rate than desired.

Others have done work [8-9] with regard to lowering bit-rates for speech transmission, but little has been done with regard to low bit-rate speech recognition. Speech recognition does not need the precision in transmission of the features of the speech signal, since it only need to be useful to the recognizer, not to the human ear.

Section 2.2: Quantization

In terms of LPC quantization for speech transmission, the major contribution has been separate classification of voiced and unvoiced speech for transmission by Hagen [10]. In this work, the speech frames are classified phonetically for distinctive bit allocation. Using this scheme, unvoiced speech required at least 9 b/frame and voiced speech required at least 25 b/frame. These rates are the minimum, however, and actual implementation rates are higher. Lawrence Rabiner [11-12] has also done work in the area of vector quantization of LPCs, combining energy and the LPC parameters to more efficiently transmit speech.

Section 2.3: Arithmetic Coding

Arithmetic coding is a very popular and efficient lossless compression technique, but can be complex in implementation. For this reason, work has been done to create less
complex methods of arithmetic coding. Moffat [13] describes an improved method for arithmetic coding that requires fewer multiplications and uses low-precision arithmetic, allowing for fast shift/add operations. Jer Min Jou and Pei Yin Chen [14] also describe a faster, more efficient way to arithmetically code. Their method uses a table-lookup incorporating fuzzy logic to create a fast arithmetic coding algorithm suitable for VLSI implementation.

Van der Vleuten [15] discusses two methods of arithmetic coding that remove multiplication entirely. The first method incorporates rounding in place of truncation, but even though it is multiplication-free, it actually is more complex than methods using multiplication. The second method builds off the first and uses a method called “partial rounding”. In this method, an OR statement of the two least significant bits is used to reduce the number of additions needed in an effort to reduce the complexity as compared to full rounding. Both of Van der Vleuten’s methods, while eliminating multiplication, are still quite complex.

Section 2.4: Dynamic Time Warping

Dynamic Time Warping is an extremely useful tool in speech recognition. It is a time-scale modification technique that more closely aligns two utterances than linear time analysis. In Wong et al [16], the focus is on reducing the computational load associated with the technique. In this method, time-scale compression is applied to both the template and the incoming speech signal before feature extraction occurs. The experimental data shows that complexity was reduced by up to 75%, and noise in the
utterances was even suppressed, making it more effective in noisy situations than other Dynamic Time Warping techniques.
Chapter 3
Pattern Recognition

Pattern recognition consists of four basic steps – feature measurement, pattern training, pattern classification, and decision logic.

In the feature measurement step, a sequence of measurements is made on the input signal to obtain a test pattern. The measurements used in this step can be the output of a spectral analysis technique such as filter bank analysis, linear predictive coding analysis, or a discrete Fourier transform analysis. For the purposes of this recognizer, a linear predictive coding analysis and a cepstral discrete analysis were implemented and compared.

In pattern training, one or more test patterns are used to classify the features of speech signals of the same class. The reference pattern can be a template derived from an averaging technique that characterizes the features of the class of speech signals.

Pattern classification occurs when an incoming speech signal is compared to the trained templates and a measure of similarity between the spectral vectors of each of the templates and the spectral vectors of the incoming signal is computed. In order to compare the sequences of spectral vectors, a local distance measure and a global distance measure are required. The local distance measures the spectral distance between two vectors, and the global distance measure helps compensate for different rates of speaking in the two patterns when used in a global time alignment procedure. This is discussed in greater detail in the section on Dynamic Time Warping.

Finally, decision logic uses the reference pattern similarity scores to determine which reference pattern best matches the unknown incoming pattern.
As in many signal-processing techniques, pattern recognition has positive and negative aspects that must be considered to determine if it is a worthwhile method for the specific application.

The first problem is that the reference patterns are sensitive to the speaking environment. Factors such as background noise and the transmission characteristics of the medium in which the speech is created affect the speech spectral characteristics. Another problem is that the performance of the system is sensitive to the amount of training; generally, the more the training, the higher the correct ASR (Automated Speech Recognition) rate. A third pitfall is that for a large number of sound classes, the computations required for pattern training and pattern recognition often become prohibitive.

There are problems with the pattern-recognition technique, but there are also many benefits. The first benefit is that no speech-specific knowledge is used in the system, making it insensitive to the choice of vocabulary words. Since the system is insensitive to sound class, a wide range of speech sounds can be used including phrases, whole words, and subword units. It is also relatively simple to incorporate syntactic constraints directly into the pattern recognition structure, improving accuracy and reducing the computation load.
Chapter 4  
Signal Feature Measurement Techniques

Section 4.1: Linear Predictive Coding

Section 4.1.1: LPC Theory

Before looking into the theory behind Linear Predictive Coding, it is useful to look into why it is such a widely used method for speech recognition. The first reason it is so widely used is because it provides a good model of the speech signal for voiced regions of speech. In voiced speech, the all-pole model of LPC provides a good approximation of the vocal tract. The second reason is because the LPC analysis of speech signals leads to reasonable source-vocal tract separation, which allows for accurate representation of the vocal tract characteristics. This is important because the vocal tract characteristics are highly linked to the speech signal being produced. The third reason is because LPC is mathematically precise, yet still simple to implement in software, as well as in hardware. The final reason LPC is so widely used is because past experience has proven it to perform extremely well in speech recognition applications.

Now that the reasons for using LPC have been explored, the actual LPC model will be examined. The basic idea behind the LPC model is that a given speech sample at time \( n \) can be approximated as a linear combination of the past \( p \) speech samples, where \( p \) is the order of the LPC model.

\[
s(n) = a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p) \tag{4-1}
\]

Equation 4-1 can be re-written as an equality to provide an approximation of the speech signal.
\[
\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k)
\]  

(4-2)

The prediction error can then be found.

\[
e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k)
\]  

(4-3)

The goal of LPC analysis is to determine the predictor coefficients that properly describe the signal of interest. A signal is characterized by the predictor coefficients that minimize the mean-squared prediction error over a short segment of the speech waveform. To find the equations needed to determine the predictor coefficients, the short-term speech and error segments at time \( n \) are defined.

\[
s_n(m) = s(n + m)
\]

\[
e_n(m) = e(n + m)
\]  

(4-4)(a-b)

The mean-squared error at time \( n \) is what we want to minimize.

\[
E_n = \sum_{m} e_n^2(m)
\]  

(4-5)

Equation 4-5 can be re-written using Equation 4-3 for \( e(n) \) derived above.

\[
E_n = \sum_{m} \left[ s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2
\]  

(4-6)

To find where the mean-squared error is minimized, the derivative of Equation 4-6 with respect to each coefficient is found and set to zero.
$$\frac{\partial E_n}{\partial a_k} = 0 \quad \text{for } k = 1,2,\ldots, p$$  \hspace{1cm} (4-7)$$

This differentiation results in Equation 4-8.

$$\sum_{m} s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_{m} s_n(m-i)s_n(m-k)$$  \hspace{1cm} (4-8)$$

Equation 4-8 can then be re-written in terms of the short-term covariance.

$$\phi_n(i,0) = \sum_{k=1}^{p} \hat{a}_k \phi_n(i,k)$$  \hspace{1cm} (4-9)$$

To solve for the predictor coefficients, Equation 4-9 is solved for $1 \leq i \leq p$ and $0 \leq k \leq p$, resulting in a set of $p$ simultaneous equations. These equations are then solved using the autocorrelation method.

The autocorrelation method uses the speech sample in a window of $N$ samples, allowing the covariance to be defined over the window.

$$\phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k) \quad 1 \leq i \leq p, \ 0 \leq k \leq p$$  \hspace{1cm} (4-10)$$

Equation 4-10 can be re-written to make the function dependent only on $(i-k)$. This allows the covariance function to reduce to the simpler autocorrelation function.

$$r_n(i-k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k)$$  \hspace{1cm} (4-11)$$

Using the symmetry of the autocorrelation function, the LPC equations can be expressed in a solvable form.
\[ \sum_{k=1}^{p} r_{n}(i-k) \hat{a}_k = r_{n}(i) \quad 1 \leq i \leq p \]  \hspace{1cm} (4-12)

Equation 4-12 in matrix form provides a better view into how the coefficients are solved for:

\[
\begin{bmatrix}
    r_{n}(0) & r_{n}(1) & r_{n}(2) & \ldots & r_{n}(p-1) \\
    r_{n}(1) & r_{n}(0) & r_{n}(1) & \ldots & r_{n}(p-2) \\
    r_{n}(2) & r_{n}(1) & r_{n}(0) & \ldots & r_{n}(p-3) \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    r_{n}(p-1) & r_{n}(p-2) & r_{n}(p-3) & \ldots & r_{n}(0)
\end{bmatrix}
\begin{bmatrix}
    \hat{a}_1 \\
    \hat{a}_2 \\
    \hat{a}_3 \\
    \vdots \\
    \hat{a}_p
\end{bmatrix}
= 
\begin{bmatrix}
    r_{n}(1) \\
    r_{n}(2) \\
    r_{n}(3) \\
    \vdots \\
    r_{n}(p)
\end{bmatrix}
\]

The \( pxp \) matrix of autocorrelation values is a Toeplitz matrix, which can be solved using a variety of matrix methods. The method used to solve this system of equations was the Levinson-Durbin recursive algorithm.

\[
k_m = \frac{r(m) - \sum_{k=1}^{m-1} a_{m-1}(k)r(m-k)}{E_{m-1}}
\]

\[
a_{m}(m) = k_m
\]

\[
a_{m}(k) = a_{m-1}(k) - k_m a_{m-1}(m-k) \quad \text{for } 1 \leq k \leq m-1
\]

\[
E_m = \left(1 - k_m^2\right)E_{m-1}
\]

This provides the set of \( L \) linear predictive coefficients, where \( L \) is the order of the linear analysis.
Section 4.1.2: Linear Predictive Coefficient Coding

The first step in LPC processing is pre-emphasis. The digitized speech signal is received by the LPC processor, and is passed through a first-order high-pass filter to flatten the signal’s spectrum. The flattening of the spectrum makes it less susceptible to finite precision effects that accompany the processing of the signal. The most widely used pre-emphasis network is the high-pass first-order fixed filter given by Equation 4-14.

\[ H(z) = 1 - \alpha z^{-1} \]  

(4-14)

A value of 0.95 is commonly used for the variable \( \alpha \), and is the value used in the pre-emphasis network. The output of the pre-emphasis filter, \( \tilde{s} \), is related to the input, \( s \), by Equation 4-15.

\[ \tilde{s}(n) = s(n) - \alpha s(n-1) \]  

(4-15)

After pre-emphasis, the next phase of the LPC processing is frame blocking. Here, the pre-emphasized signal is blocked into overlapping frames. For this recognizer, the signals were broken into frames of 240 samples with 80 samples between frames. At a sampling rate of 8 kHz, this works out to 30 ms frames separated by 10 ms. These frames are then windowed to minimize the signal discontinuities at the edges of each frame. This is accomplished by tapering the signal to zero at the beginning and end of the frame. The typical window for LPC models using the autocorrelation method is the Hamming window, given by Equation 4-16.

\[ w(n) = 0.54 - 0.46 \cos \left( \frac{2\pi n}{N-1} \right) \]  

(4-16)

where \( N = \) number of samples per frame
After the pre-emphasized signal is blocked into frames and multiplied by the Hamming window, an autocorrelation vector is obtained for each frame. A benefit of autocorrelation is that the zero lag value indicates the energy of the frame, an important parameter in speech detection.

Section 4.2: Frequency Domain Techniques

Section 4.2.1: Discrete Fourier Transform

A representation of a signal in the frequency domain can be obtained using the Discrete Fourier Transform (DFT). The DFT of a signal \( s(n) \) is given by Equation 4-17.

\[
X(\omega) = \frac{1}{2\pi N} \sum_{n=0}^{N-1} s(n)e^{-j\omega n} \tag{4-17}
\]

The Discrete Fourier Transform results in a set of sinusoids representing the frequency components of the signal, also known as the spectrum of the signal. This transform is the foundation of many other short-time frequency domain processes, such as Power Spectral Density.

Section 4.2.2: Power Spectral Density

The Discrete Fourier Transform of a signal can be modified slightly to obtain the power spectrum from the signal samples. The power spectrum of a signal is given by Equation 4-18.

\[
S(\omega) = \frac{1}{2\pi N} \left| \sum_{n=0}^{N-1} s(n)e^{-j\omega n} \right|^2 \tag{4-18}
\]
The power spectrum of a signal can be divided into multiple bands and the energy of each band can be summed to obtain the Filter Bank Coefficients (FBCs). These FBCs can then be used in the same manner as LPCs to perform the task of speech recognition. In implementation, all these calculations are not performed. Instead, a bank of bandpass filters, each centered at a different frequency, is created and the RMS strength of the signal at the output of each filter is used as the FBC for that band. This method has long been used in analog techniques but has only recently been performed digitally using the Fast Fourier Transform (FFT).

Section 4.2.3: Cepstral Coefficients

Taking the Inverse Fourier Transform of the logarithmic amplitude spectrum produces the signal cepstrum. This process results in the cepstral coefficients \( c(\tau) \) given by Equation 4-19.

\[
c(\tau) = F^{-1} \log|X(\omega)|
\]

Using the DFT to calculate the cepstral coefficients, Equation 4-19 is rewritten in discrete form in Equation 4-20.

\[
c_n = \frac{1}{N} \sum_{k=0}^{N-1} \log|X(k)| e^{\left(\frac{j2\pi kn}{N}\right)} \quad n = 0, 1, ..., N - 1
\]

The cepstral coefficients can also be produced using the LPCs of a signal. This is a less complex recursive method, given by Equations 4-21.
\[ c_0 = \ln \sigma^2 \]
\[ c_m = a_m + \sum_{k=1}^{m-1} \left( \frac{k}{m} \right) c_k a_{m-k} \quad 1 \leq m \leq p \]  \hspace{1cm} (4-21)(a-c)
\[ c_m = \sum_{k=1}^{m-1} \left( \frac{k}{m} \right) c_k a_{m-k} \quad m > p \]

where \( \sigma^2 \) is the gain term of the LPC model and \( p \) is the order of the LPC analysis.

Generally, a cepstral representation with \( Q \) coefficients is used where \( Q \equiv \left( \frac{3}{2} \right)^p \).

**Section 4.3: Recognition**

Speech recognition can be achieved by comparing two sets of extracted speech features. Similarity is determined by calculating how closely the two sets match. The most common distance metric is the Euclidean distance between two vectors because it is a quick and effective measure with low computational impact. For a 1xL dimensional vector, where \( L \) is the order of the linear or cepstral analysis, Equation 4-22 provides the Euclidean distance.

\[ \text{distance} = \sum_{i=1}^{L} (a_{i1} - a_{i2})^2 \]  \hspace{1cm} (4-22)

Based on the theory discussed above, a suitable speech coding algorithm can be realized.
Section 5.1: Thresholding

As mentioned in the previous chapter, the zero lag value of the frame is the energy of the frame. In this recognizer, simple thresholding is used to determine if a word is being spoken, using four different energy thresholds, as seen in Figure 5-1. The first threshold is used to determine if the energy of the incoming signal is above background noise. To guard against breath noise, a second, slightly higher threshold is used. If the second threshold is crossed before the energy goes back below the first threshold, the beginning of the word is considered the time when the first threshold was crossed. The end of the word is found when the energy goes back below the third threshold. If the word is too short or too weak it is rejected. A word is determined to be too weak if the energy of the word never gets above the fourth threshold.

Figure 5-1: Thresholding for Endpoint Detection
The training and the recognition use the same thresholding, because correct estimation of the start and end of speech is extremely important in template-based speech recognition. Incorrect endpoints cause poor alignment for comparison to the templates, and in one experiment [17], the speech recognition error rate was 7% if the endpoints were correct, the error rate was 10% if there were endpoint errors of ±60 ms, and the error rate was 30% if the first 130 ms were missed.

Section 5.2: Training

To train the recognizer, each of the digits 0-9 were repeated three times, and the LPCs and cepstral coefficients of each word were found, for use as the templates in each of the methods. Since the same thresholds are used for both training and recognition, the same part of the word is analyzed for training and recognition. Having three templates for each of the digits allows for variation in the enunciation of each word, but still is not enough training to create a good recognizer. To improve the rate of correct recognition, a method known as Dynamic Time Warping is used to better align the incoming speech signal with the templates. Dynamic Time Warping is discussed in further detail in Chapter 7.
Chapter 6
Compression Techniques

Section 6.1: Quantization

To arithmetically code the signal's feature coefficients, the values first needed to be quantized. A uniform quantization was used for 64, 32, 16, 8, and 4 quantization levels. Since the values of the coefficients for the samples used were all in the range of (-3,3), the quantization was performed using Equation 6-1.

\[ Q = \left\lfloor a_i \frac{\# \text{levels}}{6} + \frac{\# \text{levels}}{2} \right\rfloor \]  (6-1)

To illustrate that Equation 6-1 properly quantizes over the range, consider an example with a quantization of 16 levels. If \( a_i = 0.7376 \),

\[ Q = \left\lfloor 0.7376 \left( \frac{16}{6} \right) + \frac{16}{2} \right\rfloor \]
\[ Q = \left\lfloor 0.7376 \times 2.6667 + 8 \right\rfloor \]
\[ Q = \left\lfloor 9.9669 \right\rfloor \]
\[ Q = 10 \]

This means that the coefficient lies in the tenth quantization level. If this process is repeated for all the coefficients, the alphabet to be coded only consists of the 16-symbol alphabet of quantized values.

Section 6.2: Coding Techniques

Section 6.2.1: Arithmetic Coding

The quantized values of the coefficients were then used to arithmetically code the information. The total count (TC) is simply the order of the analysis, while the number
of coefficients that lie in each quantization level determine the cumulative counts (CC). The total and cumulative counts are transmitted as side information to the arithmetic code.

To arithmetically code the quantized information, the following algorithm was implemented. The word length was determined using Equation 6-2.

\[ W = \lceil \log_2 TC \rceil + 2 \]  

(6-2)

The lower limit, \( l \), is set to a binary string of zeros with length \( W \), the upper limit, \( u \), is set to a binary string of ones of the same length, and \( s_3 \) is set to zero. The symbol to be encoded, \( a_r \), is received, and the limits are updated.

\[
l' \leftarrow l + \left\lfloor \frac{(u - l + 1) \times CC(r - 1)}{TC} \right\rfloor
\]

\[
u \leftarrow l + \left\lfloor \frac{(u - l + 1) \times CC(r)}{TC} \right\rfloor - 1
\]

(6-3)(a-c)

\[ l \leftarrow l' \]

This step requires conversion of the binary strings to decimal, and then requires conversion of decimal back to the binary strings.

With the new limits in binary form, they are examined to see if they fit any of the rescaling conditions. If the MSB of the upper limit is equal to the MSB of the lower limit, the strings meet the \( E_1 \) or \( E_2 \) rescaling conditions. If the 2\(^{nd}\) MSB of the upper limit is zero and the 2\(^{nd}\) MSB of the lower limit is one, the strings meet the \( E_3 \) rescaling condition. If the \( E_1 \) or \( E_2 \) rescaling conditions are met,

\[ b = \text{MSB}(l) = \text{MSB}(u) \]
The value of $b$ is then transmitted, and the limits are updated. The binary string corresponding to $l$ is shifted to the left 1, and a 0 LSB is added. The binary string corresponding to $u$ is also shifted to the left 1, but a 1 LSB is added. The complement of the value of $b$ is then transmitted $s_3$ times, and $s_3$ is reset to zero.

If the $E_3$ rescaling condition is met, $s_3$ is incremented by one, and the limits are updated as described in the $E_1$ or $E_2$ rescaling procedure. The new MSB of the limits are then complimented. After either of the rescaling procedures has been performed, the new limits are checked to see if they still meet the rescaling conditions. If they still meet any of the conditions, the corresponding rescaling procedures are performed until no rescaling conditions are met.

If there are more symbols to encode, the whole process is repeated for the next symbol. If there are no more symbols to encode and $s_3$ is zero, the $W$ bits of $l$ are transmitted. If there are no more symbols to encode and $s_3$ is not zero,

$$b = \text{MSB}(l)$$

The first bit of $l$ is then transmitted, followed by $s_3$ repetitions of the compliment of $b$, followed by the remaining bits of $l$, and the transmission is complete.

**Section 6.2.2: Arithmetic Decoding**

To perform the arithmetic decoding, the above procedure was closely followed in reverse. The value of $W$ was found using Equation 6-2. The lower limit, $l$, was set to a binary string of zeros of length $W$, while the upper limit, $u$, was set to a binary string of
ones of the same length, and the first W bits of the transmitted bit stream were read into
the tag, t. The new tag, $t^*$, was found using Equation 6-4.

$$t^* \leftarrow \frac{(t - l + 1) \times TC - 1}{(u - l + 1)}$$  \hspace{1cm} (6-4)$$

With the new value of $t^*$, $r$ was incremented by one until $t^* \geq CC(r)$. The decoded
quantization level is then equal to $r$. Once the symbol has been received, if there are
more to decode, the process is continued, if not, the process is ended. The limits are then
updated according to Equations 6-5.

$$l' \leftarrow l + \left\lfloor \frac{(u - l + 1) \times CC(r - 1)}{TC} \right\rfloor$$
$$u \leftarrow l + \left\lfloor \frac{(u - l + 1) \times CC(r)}{TC} \right\rfloor - 1$$  \hspace{1cm} (6-5)(a-c)$$
$$l \leftarrow l'$$

If the updated limits meet the $E_1$ or $E_2$ rescaling conditions, the lower limit is
shifted to the left by 1 and receives a 0 as the LSB, the upper limit is shifted to the left by
1 and receives a 1 as the LSB, and the tag is shifted to the left by 1 and receives the next
bit of the transmitted bit stream as the LSB.

If the updated limits meet the $E_3$ scaling condition, the limits and tag are shifted in
the same way and receive the same LSBs as in the $E_1$ or $E_2$ rescaling procedure, but the
MSB of the new limits and tag are complimented.

The new limits are then used to find the new value of $t^*$, according to Equation 6-4,
and the process continues until all the symbols have been decoded.
The arithmetic decoding provides the quantization levels of the coefficients being transmitted, but not the actual coefficients. To obtain the coefficients from the transmitted quantization levels, Equation 6-6 was used.

\[
\hat{C} = 6 \times \frac{Q}{\# \text{levels}} - 3 \tag{6-6}
\]

In the quantization example earlier in this section, a quantization value of 10 was determined for a coefficient value of 0.7376. When the value of 10 is received at the decoder, it is transformed back to the estimated coefficient value as follows.

\[
\hat{C} = 6 \times \frac{10}{16} - 3
\]

\[
\hat{C} = 3.75 - 3
\]

\[
\hat{C} = 0.75
\]

So the original coefficient value of 0.7376 is received as 0.75 after transmission.

Section 6.2.3: Adaptive Huffman Coding

The Huffman coding algorithm is based on two observations about prefix codes. The first is that in optimum code, symbols that occur more frequently will have shorter codeword lengths than more uncommon symbols. The second observation is that the two least frequent symbols will have the same codeword length. The procedure is then obtained by adding one further restriction that the two least frequent codewords only differ in the last bit. Huffman coding is more easily described using an example.

For an example, consider the five-symbol alphabet in Figure 6-1. Initially, the symbols are ordered according to probability from most to least probable. The two least
frequent symbols are assigned either a one or a zero, their probabilities are combined, and the list of symbols is rearranged accordingly. This is done until only two symbols are remaining. The codeword for each symbol is the binary string when tracing backwards from the final set to the root of the symbol. The codewords for this example can be seen in Table 6-1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_2 )</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>( \alpha_3 )</td>
<td>0.2</td>
<td>01</td>
</tr>
<tr>
<td>( \alpha_5 )</td>
<td>0.2</td>
<td>000</td>
</tr>
<tr>
<td>( \alpha_1 )</td>
<td>0.1</td>
<td>0010</td>
</tr>
<tr>
<td>( \alpha_4 )</td>
<td>0.1</td>
<td>0011</td>
</tr>
</tbody>
</table>

Table 6-1: Huffman Codes

Figure 6-1: Huffman Coding Path
In the adaptive Huffman coding procedure, neither the transmitter nor the receiver has any knowledge of the statistics of the source. The Huffman tree consists of a single node that has a weight of zero. As more symbols are received, the Huffman tree is updated to accommodate for the new probabilities.
Chapter 7
Dynamic Time Warping

Two samples of the same word spoken by the same speaker will not be orated in exactly the same manner. There will be fluctuations in the duration of each of the phonemes, as well as in the duration of the actual word. To allow for such variations, a time alignment procedure must be implemented. Time alignment is exemplified in the time-time matrix in Figure 7-1. As the input is received, it is compared against the template for the word “seven”. The best possible path is taken and a sum of the distances is calculated. This alignment is performed for each of the templates, and the template corresponding to the lowest distance sum is determined to be the incoming word.

To examine every possible path from the lower left box to the upper right would entail an extraneous amount of calculations, since the number of paths is exponential with respect to the number of input frames. To avoid this, some constraints on the matching process must be considered and used to create an efficient algorithm. The constraints on matching are:

- Matching paths cannot go backwards in time.
- Every frame in the input and template must be used in a matching path.
- Local distance scores are combined to give a global distance.

The second constraint means that at a point \((i,j)\) in the time-time matrix, where \(i\) is the index of the input pattern, and \(j\) is the index of the template pattern, the previous point
Time-Time Matrix of Spoken Digit "Seven"

Figure 7-1: Dynamic Time Warping Path

Figure 7-2: Dynamic Time Warping Path Choices
must have been \((i-1,j-1), (i-1,j),\) or \((i,j-1)\), as in Figure 7-2. The important concept behind dynamic programming is that at \((i,j)\), the lowest distance path from \((i-1,j-1), (i-1,j),\) or \((i,j-1)\) gets used in the global distance.

Dynamic Time Warping finds the lowest distance path through the matrix with the least amount of computation. The Dynamic Time Warping algorithm operates in a time-synchronous manner. This means that each frame of the input is processed sequentially, or each column of the time-time matrix is considered in succession. So, for a template of length \(N\), the maximum number of paths being considered at any time is \(N\).

If \(D(i,j)\) is the global distance up to the point \((i,j)\) and the local distance at \((i,j)\) is given by \(d(i,j)\)

\[
D(i, j) = \min[D(i-1, j-1), D(i-1, j), D(i, j-1)] + d(i, j) \tag{7-1}
\]

Given that \(D(1,1) = d(1,1)\), \(D(i,j)\) can be found recursively using Equation 7-1. The final global distance \(D(n,N)\) is the overall matching score of the template with the incoming word. The incoming word is recognized as the word corresponding to the template with the lowest matching score.

Dynamic Time Warping has a small memory requirement; the only storage required by the search is an array that holds a single column of the time-time matrix.

The algorithm to find the global distance of a template compared to an incoming signal is:

1. The local distance between the first frame of the template and the first frame of the input is calculated. This is also the global distance to the first cell of the time-
time matrix. The global distance for each of the above cells is the local distance of that cell plus the global distance to the cell below it.

2. The global distance to the bottom most cell of the next column is the local distance to the cell plus the global distance to the bottom most cell of the previous column.

3. The global distances for the rest of the cells in the column are calculated by finding the local distance at the point \((i,j)\) and adding to it the minimum global distance at \((i-1,j-1)\), \((i-1,j)\), or \((i,j-1)\).

4. The global distances in the column for the current frame are then considered the distances of the last frame, and step 2 is repeated until all the columns have been calculated.

5. The overall global distance is the value in the upper right cell after all the columns have been calculated.

To demonstrate how this algorithm works, an example using six incoming and six template frames is used.

In the first step, the local distances between the first frame of the incoming signal and the frames of the template are calculated.
From these values, the global distances of the first column are calculated by going up the column and adding the local distance of each cell to the global distance of the cell below it.
The local distances between the next incoming frame and the template frames are then calculated.

![Local Distances in Second Step of Algorithm](image1.png)

Figure 7-5: Local Distances in Second Step of Algorithm

The second step in the algorithm adds the global distance of the bottom cell of the first column to the local distance of the bottom cell of the next column to find the global distance of that cell.

![Global Distances after Second Step of Algorithm](image2.png)

Figure 7-6: Global Distances after Second Step of Algorithm
According to the third step of the algorithm, the global distance of the next cell up is the local distance of that cell, in this case 1.5, added to the minimum of the cells to the left, below, and to the lower left, in this case 0.8, making the global distance 2.3. This process is repeated up the entire column.

<table>
<thead>
<tr>
<th>13.4</th>
<th>11.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>8.5</td>
</tr>
<tr>
<td>7.0</td>
<td>6.1</td>
</tr>
<tr>
<td>4.3</td>
<td>3.8</td>
</tr>
<tr>
<td>2.0</td>
<td>2.3</td>
</tr>
<tr>
<td>0.8</td>
<td>2.1</td>
</tr>
</tbody>
</table>

Incoming Frames

This is repeated for all the columns corresponding to each incoming frame until all the frames have been used. The local distances of the entire block are seen in Figure 7-8, and the global distances are seen in Figure 7-9.
<table>
<thead>
<tr>
<th>Template Frames</th>
<th>3.3</th>
<th>3.2</th>
<th>3.4</th>
<th>2.2</th>
<th>2.3</th>
<th>2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>2.4</td>
<td>3.3</td>
<td>1.7</td>
<td>1.8</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>2.3</td>
<td>2.4</td>
<td>1.4</td>
<td>2.2</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>1.8</td>
<td>2.2</td>
<td>2.1</td>
<td>1.6</td>
<td>3.1</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>1.5</td>
<td>1.4</td>
<td>1.3</td>
<td>2.7</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>1.3</td>
<td>3.0</td>
<td>3.2</td>
<td>3.3</td>
<td>3.5</td>
<td></td>
</tr>
</tbody>
</table>

**Incoming Frames**

Figure 7-8: Local Distances of All Cells

<table>
<thead>
<tr>
<th>Template Frames</th>
<th>13.4</th>
<th>11.7</th>
<th>11.9</th>
<th>9.8</th>
<th>9.9</th>
<th>9.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>8.5</td>
<td>9.4</td>
<td>7.6</td>
<td>7.7</td>
<td>9.0</td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>6.1</td>
<td>6.2</td>
<td>5.9</td>
<td>7.8</td>
<td>8.9</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>3.8</td>
<td>4.5</td>
<td>5.6</td>
<td>6.4</td>
<td>9.5</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>2.3</td>
<td>3.5</td>
<td>4.8</td>
<td>7.5</td>
<td>10.7</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>2.1</td>
<td>5.1</td>
<td>8.3</td>
<td>11.6</td>
<td>15.1</td>
<td></td>
</tr>
</tbody>
</table>

**Incoming Frames**

Figure 7-9: Global Distances of All Cells
For this example, the best path is not the linear time path from the bottom left to the top right, but is the path seen in Figure 7-10.

According to the fifth step of the algorithm, the overall global distance of the path is 9.8, as seen in the top right cell. This is the indicator of how well the template and the incoming signal match. After all the scores have been found for each template, the one with the lowest score is the best match.
Chapter 8

Results

To test the recognizer, one hundred utterances of each digit were recorded for use as the control data. The digits were recorded rather than re-spoken for each of the quantization schemes to make the only experimental variable the quantization. As previously mentioned, no two utterances of the same word will be exactly the same. Accordingly, no sequence of one hundred utterances will be the same as the previous one hundred. So, to remove any variability in the incoming test speech, the same set of recorded digits was used for each feature analysis, coding, and quantization scheme.

Before any quantization or coding was performed on the system, the results using linear predictive coefficients and cepstral coefficients were compared. The cepstral coefficient system, with an overall recognition rate of 95.3%, greatly outperformed the linear predictive coefficient system, with a correct recognition rate of 91.0%. Therefore, cepstral coefficients were used in the system to be coded and quantized. The results for each digit in the two systems are seen in Figures 8-1 and 8-2.
Figure 8-1: Results For Each Digit in Linear Predictive Coefficient System

Figure 8-2: Results For Each Digit in Cepstral Coefficient System
Since both Huffman Coding and Arithmetic Coding are lossless methods, the effects of quantization will be the same for both methods. So, the only consideration when selecting the coding scheme is the achievable bit rates of each technique.

To actually transmit the speech signal sampled at 8 kHz, with an 8-bit resolution would require 64 kb/s. Even with some of the most complex compression techniques in use today, a bit-rate of 2.4 kb/s is at the lower limit for acceptable speech transmission.

The Arithmetic Coding procedure provided very low bit-rates. The 64-level quantization scheme required 3.395 kb/s, the 32-level scheme required 3.023 kb/s, the 16-level quantization required 2.504 kb/s, the 8-level scheme only required 1.778 kb/s, and the 4-level scheme was reduced to only 1.110 kb/s.

However, the Adaptive Huffman Coding procedure outperformed the Arithmetic Coding while remaining much less complex. The 64-level scheme required a bit-rate of 2.983 kb/s, the 32-level scheme required 2.439 kb/s, the 16-level scheme required 1.859 kb/s, the 8-level scheme required only 1.354 kb/s, and the 4-level scheme was reduced to 1.091 kb/s. The achievable bit-rates can be seen in Figure 8-3.
Tables 8-1 through 8-6 show the recognition matrices for each of the quantization schemes. The recognition rates stayed very steady for the 64-level, the 32-level, the 16-level, and even for the 8-level quantization schemes. This is seen in the overall recognition of the system, shown in Figure 8-4. The overall recognition is up around 95% for the four aforementioned quantization schemes before going down to 78.8% for the 4-level quantization scheme. To more clearly see how the individual digits performed in each of the quantization schemes, Figure 8-5 though Figure 8-14 contain a side-by-side comparison of each of the digits’ performance in each of the quantization schemes.
<table>
<thead>
<tr>
<th>No Quantization</th>
<th>Spoken Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized Digit</td>
<td></td>
<td>0</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>100</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>3</td>
<td>85</td>
<td></td>
<td>2</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>100</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>93</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>84</td>
</tr>
</tbody>
</table>

Table 8-1: ASR Matrix for No Quantization

<table>
<thead>
<tr>
<th>Q = 64</th>
<th>Spoken Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized Digit</td>
<td></td>
<td>0</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>100</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>3</td>
<td>83</td>
<td></td>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>100</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>91</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td>86</td>
</tr>
</tbody>
</table>

Table 8-2: ASR Matrix for 64 Quantization Levels

<table>
<thead>
<tr>
<th>Q = 32</th>
<th>Spoken Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognized Digit</td>
<td></td>
<td>0</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>100</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>2</td>
<td>84</td>
<td>1</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>99</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>92</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>85</td>
</tr>
</tbody>
</table>

Table 8-3: ASR Matrix for 32 Quantization Levels
<table>
<thead>
<tr>
<th>Q = 16</th>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>96</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>82</td>
<td>4</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>100</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>85</td>
<td></td>
</tr>
</tbody>
</table>

Table 8-4: ASR Matrix for 16 Quantization Levels

<table>
<thead>
<tr>
<th>Q = 8</th>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>98</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>97</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>83</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>98</td>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>83</td>
<td></td>
</tr>
</tbody>
</table>

Table 8-5: ASR Matrix for 8 Quantization Levels

<table>
<thead>
<tr>
<th>Q = 4</th>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>71</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>19</td>
<td></td>
<td></td>
<td>95</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td>4</td>
<td>56</td>
<td>38</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>99</td>
<td>11</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td>2</td>
<td>21</td>
<td></td>
<td>3</td>
<td>1</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>58</td>
<td></td>
</tr>
</tbody>
</table>

Table 8-6: ASR Matrix for 4 Quantization Levels
Figure 8-4: Overall ASR Rates
Figure 8-5: Results for Spoken Digit “Zero”

Figure 8-6: Results for Spoken Digit “One”
Figure 8-7: Results for Spoken Digit “Two”

Figure 8-8: Results for Spoken Digit “Three”
Figure 8-9: Results for Spoken Digit “Four”

Figure 8-10: Results for Spoken Digit “Five”
Figure 8-11: Results for Spoken Digit “Six”

Figure 8-12: Results for Spoken Digit “Seven”
Figure 8-13: Results for Spoken Digit “Eight”

Figure 8-14: Results for Spoken Digit “Nine”
Many of the digits performed extremely well even down to the 4-level scheme. The digits one, three, four, six, and eight all were largely unaffected, performing well regardless of the quantization. The digits zero and two both performed well down to the 8-level scheme, but were greatly affected by the 4-level scheme, with a big drop in correct Automated Speech Recognition (ASR) rate. The digits five, seven, and nine did not perform exceptionally well in any of the schemes, but all also performed far worse in the 4-level scheme. The reason the recognition of these digits was less accurate than the others is evident in the ASR matrices. Five, seven, and nine often falsely triggered each other. The phonemes "i" and "v" in "five" are similar to the phonemes "e" and "v" in "seven", causing them to sometimes be confused. This is the same problem for the words "five" and "nine", which are also phonetically similar.
The second-generation cellular technology available in America has a bandwidth of only about 9.6 kb/s. Such a small bandwidth represents a significant challenge in terms of writing and receiving information. The intention of this work was to create a voice recognition system that operates at the lowest possible bit-rate without affecting its accuracy.

To create the most efficient recognizer, the performance of the cepstral coefficients was compared to that of the linear predictive coefficients. The cepstral coefficients provided a correct recognition rate over 4% higher than the linear predictive coefficients, thus fulfilling the objective of an efficient recognition system.

The next objective was to lower the bit-rate of the system. To do this, a combination of quantization and coding techniques was implemented. Through experimentation, it was determined that Adaptive Huffman Coding provided lower bit-rates than Arithmetic Coding. While all the quantization schemes discussed in this paper would perform within the limited bandwidth using Adaptive Huffman Coding, there is not significant degradation in recognition capability from the 64-level scheme down to the 8-level scheme. Since the 8-level scheme requires less than half the bandwidth of the 64-level scheme, it would appear to represent the optimal scheme for this system. The 8-level quantization scheme requires only 1.354 kb/s bandwidth in its transmission line, while providing a 94.4% overall recognition rate. This bit-rate is low enough that a system using this scheme would require less than 15% of the bandwidth of a second generation cellular communication system.
Bibliography and Citation Index


