THE POWER OF THREE: THE FRONT-END WITH SPECIAL EMPHASIS ON FFTs, LP TRANSFORMATIONS AND FEATURE SELECTION

L. A. Webster

The Speech Processing Group
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, Mississippi 39762
webster@isip.msstate.edu

ABSTRACT

Communication is a key factor in life. In order to be a productive communicator, speech must be generated and comprehended, by fully understanding the speech signal. Information theory states that speech can be represented in terms of its message content. Another way of describing speech is in terms of an acoustic waveform, the signal relaying the message content. A typical speech recognizer is composed of three main elements: the front-end or acoustic model, search, and language modeling. In this project, the first element of a speech recognizer, the front-end model will be discussed with special emphasis given to the fast Fourier transform (FFT) based measurements, linear prediction coefficient (LPC) transformations, and feature selection. The over-all purpose of this project is to design and implement the specific aspects of the acoustic model mentioned above with the final goal of incorporating them into a speech recognition system.

9. INTRODUCTION

“Then you should say what you mean,” the March Hare went on.

“I do,” Alice hastily replied: “At least - at least I mean what I say - that’s the same thing, you know.”

“Not the same thing a bit!” said the Hatter. “Why, you might just as well say that ‘I see what I eat’ is the same thing as ‘I eat what I see’!”

-Lewis Carroll, Alice’s Adventures in Wonderland.[40]

Speech serves as a person’s primary means of communication.[6] Just like the fictitious Alice in Alice’s Adventure’s in Wonderland, people often find themselves saying things that do not express what they truly mean. What we say and what we mean sometimes may be totally unrelated vocalizations. If the human neural system has a difficult time deciphering a fellow human’s vocalizations, imagine the difficulty a speech recognizer must have. Afterall, we humans are the ones developing the recognizer.

Speech processing depends heavily on basic research in the hearing science as well as research in the speech science.[5] Both of these sciences have been studied extensively over the past century and will continue to be studied well into the next century. One must understand basic concepts from those sciences to effectively dissect and model a speech waveform.

In order to physically communicate, a speaker has to produce a vocalization or speech signal. This speech signal travels a path from the speaker’s mouth to the listener’s ear in the form of a sound pressure wave, or more accurately stated, as an acoustic waveform. The acoustic waveform is also generated from the throat, nasal cavity, and cheeks of the speaker. The ordering of these sounds stems from various rules relative to a language. Linguistics, the study of language and the way relative rules re used in the communication process, and phonetics, the study of the traits or characteristics of sound production, comprise a large part of speech recognition. Phonetics will be the main theme of this paper as this paper’s primary focus is on the front-end model of a speech signal.[5]

The conversion of an acoustic waveform into a written equivalent of the information transmitted is termed speech recognition. Speech recognition is highly sensitive to the various constraints placed on a speaker, the speaking environment, and the context of the information being spoken. Future applications for speech recognition systems continue to be unlimited.[5] From directory assistance in telephone communications to voice communication with man-made machines, the need for reliable, robust speech recognition systems is unmistakable.[35]

One of the initial steps in speech recognition is the parameterization of a speech signal. Meaningful parameters which simulate the human auditory and perceptual systems aid algorithms that model the very complicated speech signal. The ultimate goal of these algorithms is to maximize the speech recognizer’s performance. For this reason, the process of signal modeling can best be described as the methods through which sequences of speech signals are adapted to vectors to depict specific events in a probability space.[1]

Four basic components of signal modeling beneficial to speech research are spectral shaping, spectral analysis, parameter transformation, and statistical modeling. Spectral shaping comprises itself of analog-to-digital (A/D) conversion (from a sound pressure wave to a digital signal) and digital filtering (focusing on the key frequency components of the speech signal). Spectral analysis, the principal theme of this paper, is formed form six major components:

- digital filter bank
- the Fourier transform filter bank
- cepstral coefficients
- linear prediction coefficients
- linear prediction-derived filter bank amplitudes
- linear prediction-derived cepstral coefficients.

Today, both the Fourier transform and linear prediction components play highly instrumental roles in various applications of speech processing. The third component of signal modeling, parameter transformation, is processed in two ways: differentiation and concatenation. A parameter vector that contains the minimal estimates of the signal forms the output of this stage. By minimal, we mean that the smallest number of vectors needed to reconstruct the speech signal are the output. The final aspect of signal modeling, the statistical modeling of the signal parameters, involves forcing a model of some sort on the data, training that model, and measuring the quality of the approximation. Because of its fundamental function to a speech recognition system, extremely knowledgeable models are implemented.[1]

10. THE FRONT END

A speech recognition system can be broken up into three main components: the front-end, search, and language modeling. The focus of this paper will be on particular aspects of the front end. Figure 1 displays the three key components of a speech recognizer. All three components are individual players in the game of speech recognition, but they must all work together in some fashion in order to have a fully working speech recognizer.

The special features of the front-end that are of primary concern in this project are: fast Fourier transform (FFT) analysis with cepstral coefficients, linear prediction(LP) transformations, and feature selection. Figure 2 depicts how these measurements can be taken from the speech signal.

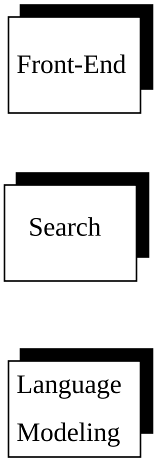

Figure 1. Components of a Speech Recognizer.

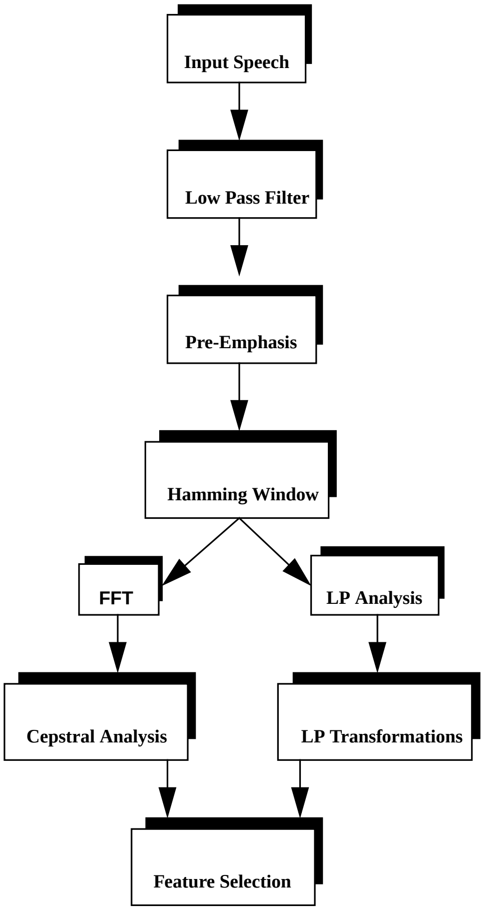

Figure 2. The Front-End Features.

10.1. The Speech Signal

The speech signal or waveform is a time versus amplitude depiction of the vocalizations formed by the speaker. Each speaker’s vocal tract shape is characterized by a set of resonant frequencies. From the system modeling point of view, the articulators (vocal cords, velum, tongue, teeth, and lips) are the key elements which determine the properties of the speech system filter. Because these resonances tend to “form” the overall spectrum, they are often referred to as “formants”. It should also be noted that typical human speech is limited to a bandwidth that typically ranges from 7 kHz to 8 kHz. Figure 3 shows a speech utterance.[5]

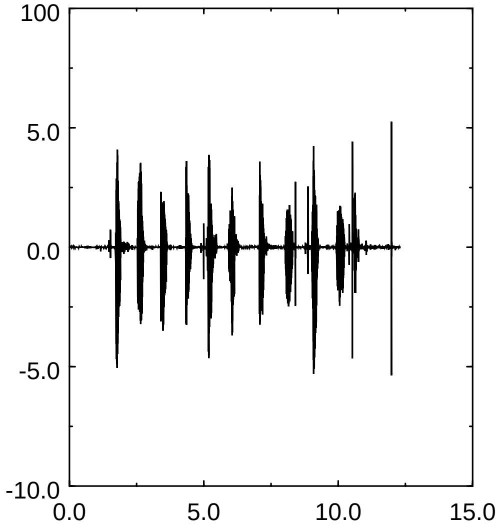

Figure 3. A speech waveform.

10.2. Acoustic Theory of Production

Much of modern signal processing notions related speech stem from extensive engineering research performed on the analog acoustic modeling of speech production. A great deal of this research was performed in the mid-twentieth century. The primary motivation for understanding the means of speech production comes from the fact that speech is a human being’s main means of communication. Because of the developments in acoustic theory, more and more aspects of human voice production are now understood by researchers. The continued quest of basic speech analysis has provided new and more realistic means of performing speech synthesis, coding, and recognition. It should also be noted that the mathematical representations of human speech production provide a sound foundation for applications in the process of speech recognition.[5]

11. SIGNAL MODELING

11.1. Parameter Selection

The parameters that were selected for the testing of the acoustic model are as follows:

- Sample Frequency = 16 kHz
- Pre-Emphasis Factor = 0.95
- Frame Duration = 20 msec
- Window Duration = 30 msec
- A Hamming Window

The pre-emphasis factor was selected to be 0.95 because typical values range between 0.4 and 1.0. The values of 0.95 was finally chosen because it can be implemented in hardware easier than other values, and it amplifies the area of the spectrum above 1 kHz, where hearing has been found to be more sensitive. The pre-emphasis factor also improves the efficiency of the analysis. The frame and window duration were selected as a pair. These values are based on the rate of change of the vocal tract shape. The sample frequency of 16 kHz was chosen because 16 kHz offers better time and frequency resolution than a lower sample frequency would. It should be noted that if working in a telecommunications environment, a lower sample frequency, i.e, 8 kHz, should be chosen. Finally, the Hamming window was chosen to help form a smoothed spectral estimate of the power in the regions where the power changes rapidly. Other windows could have been chosen, but most speech recognition systems today utilize the Hamming window.[1]

11.2. Frame Analysis

The continuous-time Fourier transform is a particularly beneficial tool in the analysis of a speech signal. In its fundamental definition, however, the Fourier transform requires our knowledge of the signal for all time, and whatever feature we wish to discover by use of the Fourier transform (i.e., a spectrum) must remain time-invariant during the duration of the signal. An underlying assumption in most speech processing schemes is that the properties of a speech signal change relatively slowly with time.[2]] Most signals fail to remain “stationary” with respect to the desired measurement for the signal duration. This is very true when you refer to the speech signal. Therefore, we must analyze the signal on a “short-term” basis. One short-term tool for signal analysis is the discrete Fourier transform (DFT). The DFT focuses on “static” analysis in which a single frame of a signal is operated on. We cannot be content however to analyze speech on a short-term basis in a single frame. Therefore, we must analyze a sequence of frames of the speech signal as those frames move through time and attempt to capture the transient features of the signal. A frame of speech can be defined as a product of a shifted window with the speech sequence.

$$f_s(n, m) = s(n)w(m-n)$$

s(n) represents the speech sequence and is a power signal. w(n) is a window function which will be explained later. Practically, a frame is a “segment” of speech which has been tapered by a window. Formally, it is a new sequence on *n* which is zero outside the frame. A frame of speech may not necessarily be a periodic signal.[5] Frames can be repeated periodically as often as desired and compose a sequence of samples.[2]

11.3. Window Analysis

A window in the time domain, is a real, finite length sequence that is used to select a specific frame of the original signal by a simple multiplication process. Some commonly used windows are the rectangular window, the Kaiser window, the Hamming window, the Hanning window, and the Blackman window. It is usually assumed that windows are causal sequences beginning at time n=0 when speaking of w(n) and are linear phase. The duration of a window is normally denoted by N. The most commonly used window in speech recognition is the Hamming window. The Hamming window gives greater attenuation outside the passband, and attenuation is independent of window duration.[2] The following equation shows the formula for the Hamming window and other relevant information.[31]

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

$$n = 0, 1, \ldots, N-1$$

When sampling at this rate, no information is lost, so it is highly possible to exactly fabricate the sequence. The Hamming window operates as a smoother to distort the temporal waveform on the range of N points, but with less abrupt truncations at the boundaries.[2] In other words, the Hamming window aids in aliasing. Aliasing is when a high frequency takes on the identity of a lower frequency. It is avoided if the Fourier transform is bandlimited and the sampling frequency is (1/T) > 2Fn.[3] The window slides along the signal in time selecting portions of the signal in the process. A longer window requires a longer period to cross transitional boundaries in the signal. Also, events from different regions of the signal will probably be blurred together more frequently than if a shorter window had been selected. Therefore, a trade-off in window selection is encountered in the choice of window length. A longer window tends to produce a better spectral picture of the signal, but only while the window lies in a stationary region of the signal. A shorter window tends to resolve events in the signal better in time, often referred to as spectral-temporal resolution trade-off.[5]

11.4. Sampling

To use digital signal processing (DSP) methods on speech, it is necessary to represent the signal as a sequence of numbers. This operation is performed by sampling the analog signal, $X_a(t)$, periodically to produce the sequence.

$$x(n) = X_a(nT)$$

$$-\infty < n < \infty$$

Typically, values for the frequency range from 8 kHz to 16 kHz, where T=1/frequency.[3]

Speech signals are not necessarily bandlimited, although the spectrum does tend to fall off at rapidly high frequencies. To accurately represent all speech sounds, a sampling rate greater than 20 kHz would be required because for voiced sounds, the high frequencies are more than 40 dB below the peak of the spectrum for all frequencies above 4 kHz. For unvoiced sounds, the spectrum has not fallen off noticeably even above 8 kHz. But, in most applications, a 20 kHz sampling rate is not needed. For example, if the speech is filtered by a sharp cut-off analog filter prior to sampling, to force the Nyquist rate to 4 kHz, then a sampling rate of 8 kHz is highly possible.[5]

The conditions under which the sequence of samples is valid can be shown as follows: If a signal, $X_a(t)$, has a bandlimited Fourier transform, $X_a(j\omega)$, such that $X_a(j\omega) = 0$ for $\omega >= \pi Fn$, then $X_a(t)$ can ultimately undergo a unique reconstruction from equally spaced samples $X_a(nT)$ if 1/T=2Fn and Fn = Nyquist frequency. This is known as the Sampling Theorem. When given samples of a bandlimited analog signal, it is possible to reconstruct the original signal if the sampling rate is at least 2Fn by the following equation[3]

$$X_a(t) = \sum_{n=-\infty}^{\infty} X_a(nT) \frac{\sin\left(\pi\frac{(t-nT)}{T}\right)}{\pi\frac{(t-nT)}{T}}$$

Practical D/A (digital-to-analog) strive to approximate the above equations. An important factor which is often overlooked when sampling is that although the signal waveform may have a bandlimited spectrum, the signal may be constructed by wideband random noise prior to A/D conversion. For these cases, the signal plus noise combination should be filtered with an analog low pass filter which cuts off sharply above the Nyquist frequency so that replications of the high frequency noise are not aliased into the baseband.[2]

Sampling is practical in many speech processing algorithms that try to estimate basic parameters of speech production, such as pitch and formant frequencies. In such cases, an analog function is not available to be sampled directly, as in the case of sampling the speech waveform itself. However, such parameters change very slowly with respect to time and can be estimated or sampled at rates on the order of 100 samples/second. When given samples of a speech parameter, a bandlimited analog function for that parameter can be constructed. Sometimes, the term decimation is used when talking about the sampling process. Decimation is a process of sampling rate reduction. One important consideration when implementing decimators is in the choice of the low pass filter (LPF). A noticeable amount of computational efficiency over other alternative filter types can be obtained by using a finite impulse response filter (FIR) in a standard direct form implementation. In the decimation process, only one of each output samples needs to be calculated.[2]

12. LP TRANSFORMATIONS

One of the most powerful tools for speech analysis is the method of linear predictive analysis. This method has become the leading technique for estimating the basic speech parameters, such as pitch, formants, spectrum, and vocal tract area functions, to describe a signal or the system that generated the signal. Linear prediction is also beneficial for representing speech for low bit rate transmission or storage. It provides extremely accurate estimates of the speech parameters and is highly efficient in computational matters.[2]

Linear prediction is widely used in speech processing to model the speech source. A *p*th order all pole filter is found, such that when excited by impulses, its output has a power spectrum that matches the spectrum of a given speech signal. Normally, it is required that the autocorrelation of the output signal be equivalent to the autocorrelation of the original speech signal. If the modeled system is truly a *p*th order all-pole system excited by one impulse, the model derived during the linear prediction process will be the same as the original system. However, voiced speech is represented by a periodic signal exciting the system and interference between successive periods can cause a deviation from the desired response.[18]

The fundamental idea behind linear prediction is that a speech sample can be approximated as a linear combination of past speech samples. By minimizing the sum of the squared differences between the actual speech samples and the linearly predicted ones, a unique, distinct set of predictor coefficients can be determined. The predictor coefficients are the weighting coefficients used in the linear combinations. Linear prediction provides a robust, reliable, and accurate method for estimating the parameters that characterize the linear, time-varying system of speech.[5]

The common set of linear prediction analysis techniques is usually referred to as linear predictive coding or LPC. These techniques and methods of linear prediction have been available in engineering for many decades.[2] In the usual applications of LPC to the analysis of speech, the same parameter is used in both the voiced and unvoiced cases.[25]

When used in speech processing, linear prediction refers to a variety of equivalent formulations of the difficult problem of modeling the speech waveform. The key differences among the various formulations are usually those of the way in which the problem is viewed, or, even in the details of the computations used to obtain the predictor coefficients. The various, sometimes equal, formulations of linear prediction include:

- covariance method
- autocorrelation formulation
- lattice method
- inverse filter formulation
- spectral estimation
- maximum likelihood formulation
- inner product formulation

Only the first three of these methods really need to be studied, because the remainder of the formulations are equivalent in some way to one of the first three.[2] In the conventional LP, the linear prediction coefficients α_k's are found by either the autocorrelation method or the covariance method.[3]

The importance of linear prediction lies in the accuracy with which the basic model applies to the speech waveform.[3]

12.1. Fundamental Principals

In the discrete-time model for producing speech, the complex spectrum effects of radiation, vocal tract, and glottal excitation are represented by a time varying digital filter whose steady state system function can be shown to be in the form of

$$H(z) = \frac{S(z)}{U(z)}$$

$$= \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

This system is excited by an impulse train for voiced speech, or by a random noise sequence for unvoiced speech. Therefore, the parameters of the model are: voiced/unvoiced classification, pitch period for voiced speech, the gain parameter (G), and the coefficients $\{\alpha_k\}$ of the digital filter.[2]

If the order (P) is high enough, the all-pole model provides a good representation for almost all the sounds of speech. The major advantage of this model is that the gain parameter and the filter coefficients can be estimated in a very straight forward and computationally adept fashion by linear prediction.[2]

The speech samples, represented by s(n), are related to the excitation, represented by u(n), by the difference equation

$$s(n) = \sum_{k=1}^{p} A_k s(n-k) + Gu(n)$$

A linear predictor with predictor coefficients α_k is a system that has the following output

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

The system function of a pth order linear predictor is

$$P(z) = \sum_{k=1}^{p} a_k z^{-k}$$

The predictor error, represented by e(n), is

$$e(n) = s(n) - \tilde{s}(n)$$

$$= s(n) - \sum_{k=1}^{p} a_k s(n-k)$$

which shows that the predictor error sequence is the output of a system whose transfer function is

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}$$

If the speech signal obeys the equation for s(n), an if $\alpha_k = A_k$, then e(n) = Gu(n). Thus, the predictor error filter A(z) will be the inverse filter for the system H(z).[2]

The basic problem of linear prediction is to find a set of predictor coefficients α_k directly from the speech signal as to obtain a good estimate of the spectral properties of the speech signal through

$$H(z) = \frac{G}{U(z)}$$

Because of the time-varying nature of the speech signal, the predictor coefficients must be estimated from short segments of the speech signal. The fundamental approach is to discover a set of predictor coefficients that will minimize the mean-squared predictor error over a short segment of speech. The resulting parameters are then assumed to be the parameters of the system function H(z) in the model for speech production.[2]

The short-time average predictor error is defined as

$$E_n = \sum_m (s_n(m) - \tilde{s}_n(m))^2$$

$$= \sum_m \langle s_n \langle m \rangle - \sum_{k=1}^{p} a_k s_n \langle m-k \rangle \rangle^2$$

By setting

$$\frac{\partial E_n}{\partial a_i} = 0 \qquad i = 1, 2, \ldots, p$$

we can find the values of α_k that minimize E_n. This results in

$$\sum_m s_n(m-i)s_n(m)$$
$$= \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k)$$
$$1 \le i \le p$$

where ak are the values of α_k that minimize E_n. If we define

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k)$$

then

$$\sum_{k=1}^{p} a_k \phi_n(i,k) = \phi_n(i,0)$$
$$i = 1, 2, \ldots, p$$

This set of *p* equations in *p* unknowns can be mathematically solved efficiently for the unknown predictor coefficients α_k. These coefficients are the ones that minimize the average squared predictor error for the segment $s_n(m)$. The minimum mean-squared predictor error can then be written as

$$E_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} a_k \sum_m s_n(m)s_n(m-k)$$

In order to solve for the optimum predictor coefficients, we have to first calculate $\phi_n(i,k)$ for *i* on the range of [1,p] and *k* on the range of [0,p]. Once this operation is performed, only the equation

$$\sum_{k=1}^{p} a_k \phi_n(i,k) = \phi_n(i,0)$$
$$i = 1, 2, \ldots, p$$

has to be solved for the α_k's.[2]

12.2. The Autocorrelation Method

One of the methods used to find linear prediction coefficients is to use the autocorrelation method. One way to find the limits on

$$E_n = \sum_m e_n^2(m)$$

is to assume that the part of the waveform, $s_n(m)$, is identically zero outside [0,N-1]. This can be written as

$$s_n(m) = s(m+n)w(m)$$

where w(m) is a finite length window (i.e, a Hamming window) that is identically zero outside [0,N-1].

If $s_n(m)$ is nonzero only for [0,N-1] then the predictor error, represented as $e_n(m)$, for a *p*th order predictor is nonzero on [0,N-1+p]. So,

$$E_n = \sum_{m=0}^{N+p-1} e_n^2(m)$$

In

$$e(n) = s(n) - \tilde{s}(n) \quad ,$$

it can be seen that the predictor error is likely to be very large at the onset of the interval [0,p-1] because we are trying to predict the signal from samples that have been set equal to zero. It can also be seen that the error can be large at the end of the interval [N,N+p-1] because we are trying to predict zero from nonzero samples. Because of this, a window which tapers the segment, $s_n(m)$, to zero is normally used for w(m). But, since $s_n(m)$ is identically zero outside the interval [0,N-1]

$$\phi_n(i,k) = \sum_{m=0}^{N+p-1} s_n(m-i)s_n(m-k)$$

can be represented as

$$\phi_n(i,k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k)$$

Moreover, it can be shown that $\phi_n(i,k)$ is equal to the short-time autocorrelation function $R_n(k)$ which is represented by the following equation

$$\sum_{m=0}^{N-1-k} x(m+n)w'(m)x(n+m+k)w'(m)$$

evaluated for (i-k). That is, $\phi_n(i,k)=R_n(i-k)$ where

$$R_n(k) = \sum_{m=0}^{N-1-k} s_n(m)s_n(m+k)$$

Since $R_n(k)$ is an even function, then it follows that $\phi_n(i,k)$ can also be shown to be equivalent to the following equation

$$\begin{gathered}\phi_n(i,k) = R_n(|i-k|)\\ i = 1, 2, \ldots, p\\ k = 0, 1, \ldots, p\end{gathered}$$

Therefore,

$$\begin{gathered}R_n(i) = \sum_{k=1}^{p} a_k R_n(|i-k|)\\ 1 \le i \le p\end{gathered}$$

and

$$E_n = R_n(0) - \sum_{k=1}^{p} a_k R_n(k)$$

When shown in matrix form, we get

$$\begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & \ldots & R_n(p-1)\\ R_n(1) & R_n(0) & R_n(1) & \ldots & R_n(p-2)\\ R_n(2) & R_n(1) & R_n(0) & \ldots & R_n(p-3)\\ \ldots & \ldots & \ldots & \ldots & \ldots\\ R_n(p-1) & R_n(p-2) & \ldots & \ldots & R_n(0)\end{bmatrix}\begin{bmatrix}\alpha 1\\ \alpha 2\\ \alpha 3\\ \ldots\\ \alpha p\end{bmatrix} = \begin{bmatrix}R_n(1)\\ R_n(2)\\ R_n(3)\\ \ldots\\ R_n(p)\end{bmatrix} \quad [2]$$

12.3. The Covariance Method

This method imposes a fixed interval over which the mean-squared error is computed. In other words,

$$E_n = \sum_{m=0}^{N-1} e_n^2(m)$$

so it follows that

$$\begin{gathered}\phi_n(i,k) = \sum_{m=0}^{N-1} s_n(m-i)s_n(m-k)\\ 1 \le i \le p\\ 0 \le k \le p\end{gathered}$$

If the index of summation is changed

$$\begin{gathered}\phi_n(i,k) = \sum_{m=-i}^{N-i-1} s_n(m-i)s_n(m-k)\\ 1 \le i \le p\\ 0 \le k \le p\end{gathered}$$

To evaluate $\phi_n(i,k)$ over *i* we have to use the values of $s_n(m)$ in the range of [-p,N-1]. It does not make sense to taper the segment of speech to zero at the ends as in the autocorrelation method because the needed values for the covariance method are available outside the interval [0,N-1]. This method also leads to a function which is the cross-correlation between two very similar finite length segments of the speech waveform. Evaluating

$$\sum_{k=1}^{p} a_k \phi_n(i,k) = \phi_n(i,0)$$
$$i = 1, 2, \ldots, p$$

for the covariance, supplies the following matrix

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \phi_n(1,3) & \ldots & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & \phi_n(2,3) & \ldots & \phi_n(2,p) \\ \phi_n(3,1) & \phi_n(3,2) & \phi_n(3,3) & \ldots & \phi_n(3,p) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \phi_n(p,1) & \phi_n(p,2) & \ldots & \ldots & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} \alpha 1 \\ \alpha 2 \\ \alpha 3 \\ \ldots \\ \alpha p \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \phi_n(3,0) \\ \ldots \\ \phi_n(p,0) \end{bmatrix}$$

In the autocorrelation method, the signal is windowed by an N-point window, and $\phi_n(i,k)$ quantities are obtained using short-time autocorrelation function. In the covariance method, the signal is assumed to be known for the set of values on the range of [-p,N-1]. Outside the interval, no assumptions can be made about the information in the signal, because these are the only values needed in the computation process.[2]

12.4. Gain

Another feasible relation when discussing linear prediction is to relate the gain constant, represented by G, to the excitation signal and the error in the prediction. The excitation signal, represented by Gu(n), is

$$Gu(n) = s(n) - \sum_{k=1}^{p} A_k s(n-k)$$

and the predictor error signal, represented by e(n), is

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

It can therefore be seen that when $A_k = \alpha_k$, e(n)=Gu(n).

A more reasonable assumption is

$$G^2\left(\sum_{m=0}^{N-1} u^2(m)\right) = \left[\sum_{m=0}^{N-1} e_n^2(m)\right]$$
$$= E_n$$

Simply stated, the energy in the error signal is equal to the energy in the excitation input.

Now, assumptions about u(n) must be made in order to relate G to the known quantities. Remember, the known quantities are the α_k's and the correlation coefficients. For voiced speech, u(n)=δ(n). Also, *p* has to be large enough to account for the vocal tract and the glottal effects. For unvoiced speech, we assume that u(n) is a zero mean, unity variance, static, Gaussian process. For voiced speech, Gδ(n) is the input and

$$h(n) = \sum^{p} a_k h(n-k) + G\delta(n)$$
$$\tilde{R}(m) = \sum_{h=o}^{\infty} h(n)h(n+m)$$

and satisfies

$$\tilde{R}(m) = \sum_{k=1}^{p} a_k \tilde{R}(|m-k|)$$
$$m = 1, 2, \ldots, p$$

and

$$\tilde{R}(0) = \sum_{k=1}^{p} a_k \tilde{R}(k) + G^2$$

Also,

$$\tilde{R}(m) = R_n(m)$$
$$1 \le m \le p$$

Because

$$R(0) = \tilde{R}(0)$$

we get

$$G^2 = R_n(0) - \sum_{k=1}^{p} a_k \tilde{R}_n(k) = E_n \quad [2]$$

12.5. Levinson-Durbin Recursion

The Levinson-Durbin recursion process for the autocorrelation method is shown below. The code is written in the C programming language.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <math/integral/integral.h>

float_4 levdur(float_4 *pCoeff, int_4 order, float_4 *rn,
float_4 *energy)

{
  int_4 M = order + 1;
  float_4 *rCoeff = new float_4[M];
  float_4 *tCoeff = new float_4[M];

// rn = autocorrelation
  energy[0] = rn[0];

// recursion for i = 1, 2, ..., M
  for (int l = 1; l < M; l++)
  {
    //initialize l = 0
      rCoeff[l] = 0;
      // compute the ith reflection coefficient
      for (int j = 1; j < l; j++)
      {
         rCoeff[l] -= tCoeff[j] * rn[l - j];
      }
      rCoeff[l] += rn[l];
      rCoeff[l] = rCoeff[l]/energy[l-1];
      // compute the lp parameters
      pCoeff[l] = rCoeff[l];
      for (int j = 1; j < l; j++)
      {
         pCoeff[j] -= (rCoeff[l] * tCoeff[l - j]);
      }
      energy[l] = energy[l-1] * (1 - pow(rCoeff[l],2));

for (int j = 0; j <= l; j++)
      {
tCoeff[j] = pCoeff[j];
      }
  } // end for
  delete [] rCoeff;
  delete [] tCoeff;
return (10 * log10 (energy[order]));
}// end Levinson-Durbin recursion[5]
```

Levinson first published his algorithm for solving *Ax=b* where *A* is Toeplitz, symmetric and positive definite, and *b* is arbitrary back in 1947. It should be noted that autocorrelation equations are exactly of this form. In 1960, Durbin came along and published a slightly more efficient algorithm. Hence, the name Levinson-Durbin recursion. Levinson-Durbin recursion is a recursive-in-model-order solution for the autocorrelation equations.[5]

12.6. Lattice Formulations

Lattice formulations have sort of combined the matrix correlation values with linear equations to form a recursive algorithm for finding the linear predictor parameters. Remember that for the *i*th stage of this procedure, the set of coefficients $\{a_j^{(i)}, j = 1, 2,..., i\}$ are the coefficients of the *i*th order linear predictor. Therefore,

$$A^{(i)}(z) = 1 - \sum_{k=1}^{i} a_k^{(i)} z^{-k}$$

is the system function of the *i*th order inverse filter. If the input is

$$s_n(m) = s(m+n)w(m)$$

then the output it the predictor error

$$e^{(i)}(m) = s(m) - \sum_{k=1}^{i} \alpha^{(i)}{}_k s(m-k)$$

and

$$E^{(i)}(z) = A^{(i)}(z)S(z)$$

If we make a simple substitution, we obtain

$$A^i(z) = A^{(i-1)}(z) - k_i z^{-1} A^{(i-1)}(z^{-1})$$

and

$$E^i(z) = (A^{(i-1)}(z)S(z) - k_i z^{-1} A^{(i-1)}(z^{-1})S(z))$$

If we define

$$B^i(z) = z^{-1}A^i(z^{-1})S(z)$$

then, the inverse transform can be shown to be

$$b^i(m) = s(m-i) - \sum_{k=1}^{i} a_k^i s(m+k-i)$$

This equation implies that we are trying to predict s(m-i) form the *i* samples of the input that follow s(m-i). Thus, b^i(m) is called the backward predictor error sequence. The *i* samples involved in the prediction are the same ones that are used in the prediction of s(m) in terms of *i* past samples. The error e^i(m) can then be expressed as

$$e^i(m) = e^{(i-1)}(m) - k_i b^{(i-1)}(m-1)$$

Again, making a simple substitution, we get

$$B^i(z) = (z^{-1}B^{(i-1)}(z) - k_i E^{(i-1)}(z))$$

Thus, the *i*th stage backward predictor error is

$$b^i(m) = b^{(i-1)}(m-1) - k_i e^{(i-1)}(m)$$

Therefore, the forward and backward predictor error sequences for an *i*th order linear predictor in terms of the corresponding predictor errors of an (*i-1*)th order predictor, which use a *zero*th order predictor, is equal to using no predictor at all. This implies

$$e^0(m) = b^0(m) = s(m)$$

Finally, the lattice structure is defined.[2]

12.7. The Prediction Error Signal

A by product of linear prediction analysis is the generation of the error signal

$$e(n) = Gu(n)$$

$$= s(n) - \sum_{k=1}^{p} a_k s(n-k)$$

e(n) is a good approximation of the excitation source. It is expected that the prediction error will be somewhat large for voiced speech.[2] Figure 4 shows the linear predictor error waveform for a sample speech file.

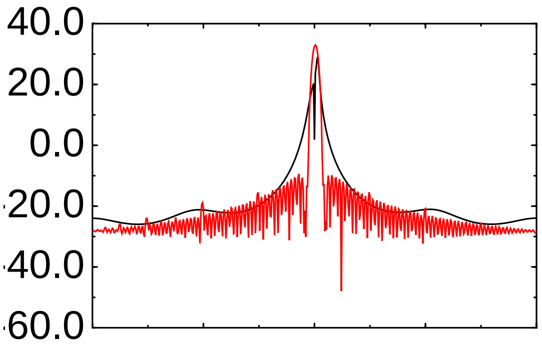

Figure 4. LP Waveform.

12.8. Parameter Transformations

One desirable feature of linear prediction is to be able to easily transform between parameters without loss of information. For this reason, a pair by the names of Markel and Gray developed a series of FORTRAN code that made conversions between parameters relatively easy. The FORTRAN code developed by Markel and Gray can be found in *Linear Prediction of Speech*. Currently, C compatible code is being developed to performed these conversions. The code is still in the debugging stage. Once the code has been fully debugged, it will be a user-friendly menu based system that allows you to select which type of parameters you wish to “convert to” from a selected “convert from” parameter.[4]

13. FFT

The Fourier transform is one of the most important mathematical applications to signal processing. It has many widespread applications outside the signal processing area as well.[feb78,236] A discrete Fourier transform (DFT) is of the following form

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k\frac{n}{N}}$$
$$k = 0, 1, \ldots, N-1$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j2\pi k\frac{n}{N}}$$
$$n = 0, 1, \ldots, N-1$$

The calculation of the DFT is one of the central operations in digital signal processing (DSP).[26] It is important to remember that when using a DFT representation, all sequences behave as if they were periodic when represented by a DFT. The DFT is widely used for computing spectrum estimates, correlation functions, and for implementing digital filters.[book] The DFT can be implemented as a filter bank in a way which reduces the number of filter coefficients.[feb78, 56] A digital filter is a discrete-time linear shift-invariant system that relates the output to the input. y(n) represents the output, x(n) the input, and h(n) the unit sample response. h(n) is the convolutoin of x(n) with y(n). H($e^{j\omega}$) represents the frequency response of the system (the Fourier transform of the unit impulse response).[2]

The poles lie inside the unit circle for stability, bounded input bounded output, and the system is causal (h(n) = 0). For a finite impulse response (FIR) filter, no nonzero poles exist, only zeros exist. The FIR can be exactly linear in phase. Being of linear phase is very useful in speech processing applications where precise time alignment is essential. This property of FIR filters also can greatly simplify the approximation problem since it is only necessary to be concerned with approximating a desired magnitude response. The drawback, however, is that a large impulse response duration is required to adequately handle sharp cut-off filters. Another type of filter is the infinite impulse response filter. This filter has poles as well as zeros and cannot have exact linear phase. The orders of magnitude are more efficient in realizing sharp cut-off filters than in the case of the FIR filter.[2]

A question still remains about the amount of computation required in computing the DFT of N points. For many years, up until the mid-1960's, the answer was:

Define W as the complex number $W = e^{2\pi\frac{i}{N}}$.

Then

$$H(n) = \sum_{k=0}^{N-1} W^{nk} h_k$$

Stated more clearly, the vector of h_k's is multiplied by a matrix whose *(n,k)*th element is the constant W to the power n*k. The matrix multiplication produces a vector result whose components are the H_n's. N^2 complex multiplications are required for this matrix multiplication in addition to a smaller number of mathematical operations to produce the required powers of W. So, the DFT appears to be an O(N^2) process. This is highly deceiving. The DFT can be computed on O($N\log_2 N$) operations with an algorithm dubbed the fast Fourier transform (FFT).[36] The development and widespread use of the FFT, stimulated by the paper of Cooley and Tukey, has had a major impact on signal processing.[26] The key difference between $N\log_2 N$ and N^2 is extensive. With N=10^6, it is the difference between 30 seconds of CPU time and 2 weeks of CPU time on a microsecond cycle time computer. The very existence of an FFT became generally known in the mid-1960s, from the work of J.W. Cooley and J.W. Tukey. Looking back, we now know that efficient ways for computing the DFT had been independently discovered... even implemented... by perhaps a dozen people, starting with Gauss way back in 1805![36]

One discovery of the FFT, that of Danielson and Lanczos in 1942, provides a clear derivation of the algorithm. This duo showed that the DFT of length N can be rewritten as the sum of two DFTs, each of length N/2. One of the two is formed from the even-numbered points of the original N, the other from the odd-numbered points. The proof is as follows:

$$F_k = \left(\sum_{j=0}^{\frac{N}{2}-1} e^{2\pi 2jk\frac{i}{N2f_j}} + \sum_{j=0}^{\frac{N}{2}-1} e^{2\pi(2j+1)k\frac{i}{N2f_{(2j+1)}}} \right)$$

$$= (F_k^e + W^k F_k^o)$$

In the last line, W is a complex constant. F^e_k represents the *k*th component of the Fourier transform of length N/2 from the even components of the original f_j's while F^o_k is formed from the odd components. One should also notice that *k* varies from 0 to N, not just to N/2. The transforms F^e_k and F^o_k are periodic in *k* with length N/2 which has the effect of repeating through two cycles to obtain F_k.[36]

The good thing about the Danielson-Lanczos Lemma is that it an be used recursively. Once F_k has been reduced to computing F^e_k and F^o_k, we can reduce F^e_k to computing the transform of its N/4 even parts and N/4 odd parts.[36]

Although there are ways of treating other cases, the easiest, most useful case is the case in which the original N is an integer power of 2. It is normally recommended that the FFT be used only in cases where N is a power of 2. If the length of the signal is not a power of 2, it can be padded with zeros up to the next power of two so that the FFT can be used. With this restriction on N, it is shown that we can keep using the Danielson-Lanczos Lemma until a transform of length 1 has been found. The Fourier transform of length one is the identity operation that copies its one input number into its one output slot. In other words, for each $\log_2$N even and odd part, there exists a one-point transform that is just one of the input numbers f_n.[36]

The next trick is to decide which value of *n* corresponds to which pattern of even and odd parts. Simply reverse the even and odd parts and set the even part equal to zero. You should then set the odd part equal to one. This provides you with in binary value of *n*. This works because successive subdivisions of the signal into even and odd parts are test of successive low-order bits of *n*. This idea of "bit reversal" can be exploited in a very resourceful way which, with the Dnaielson-Lanczos Lemma, makes the FFT very practical.[36]

Now, we have the structure of the FFT algorithm. It is composed of two key sections. The first section sorts the signal into bit-reversed order, taking up no additional storage or memory because you are simply swapping elements, so to speak. The second section has an outer loop that is executed $\log_2$N times and computes transforms of length 2, 4, 8,..., N. Two nested inner loops that range over the subtransforms already computed and the elements of each transform, implementing the Danielson-Lanczos Lemma, compose each stage of this process. The mathematics is made more efficient by restricting external calls for trigonometric sines and cosines to the outer loop.[36]

Whenever data is rearranged into a bit-reversed order, it is referred to as a *decimation-in-time* process, or *Cooley-Turkey FFT.* It is also possible to compute the FFT algorithm that first sorts through a set of $\log_2$N iterations on the input data and then rearranges the output values into bit-reversed order. These are called *decimation-in-frequency*, or *Sande-Turkey FFT.* For some applications, like convolution, a data set is taken into the Fourier domain, and forced back out again after a fair amount of manipulation. In these cases, it is possible to avoid bit reversing. The decimation-in-frequency algorithm minus the bit reversing can be used to obtain the "scrambled" Fourier domain, perform you operations, then use an inverse algorithm minus its bit reversing to return to the time domain. This procedure however does not save much computation time because the bit reversals represent only a small fraction of an FFT's mathematical calculations count.[36]

The FFT code to compute the radix-2 FFT is shown. This code is written in the C programming language.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <math/integral/integral.h>

void find_fft(int_4 fft_len,float_4* xr,float_4*
xi,float_4* wr, float_4* wi)

{
int_4 m,l,i1,i2;
int_4 n1,n2;
float_4 real_twid,imag_twid;
float_4 tempr, tempi;

m = (int)(log10(fft_len)/log10(2));
n2 = fft_len;

for(int i=1; i<m+1; i++)
  {
    n1 = n2;
    n2 = (int_4)(n2/2);
    i1 = 1;
    i2 = (int_4)((fft_len)/n1);
    for(int j=1; j<n2+1; j++)
      {
real_twid = wr[i1];
imag_twid = wi[i1];

for(int k = j; k <= fft_len; k = k + n1)
  {
    l = k + n2;
    tempr = xr[k] - xr[l];
    xr[k] = xr[k] + xr[l];
    tempi = xi[k] - xi[l];
    xi[k] = xi[k] + xi[l];
    xr[l] = real_twid*tempr + imag_twid*tempi;
    xi[l] = real_twid*tempi - imag_twid*tempr;
  }

i1= i1 + i2;
      }
  }

//procedure for bit reversal
float_4 temp = 0;
int j = 1;

for (int i = 1; i < fft_len; i++)
  {
    if (i < j)
      {
temp = xr[j];
xr[j] = xr[i];
xr[i] = temp;
temp = xi[j];
xi[j] = xi[i];
xi[i] = temp;
     }

int_4 k = fft_len/2;

while (k < j)
     {
      j = j - k;
      k = k/2;
     }
   j = j + k;
  }
}//end FFT routine[4]
```

13.1. Cepstral Analysis

“Cepstral” analysis is primarily motivated by the problems that focus on voiced speech. Voiced speech are sounds produced by forcing air through the glottis with the tension of the vocal cords adjusted so that they vibrate in a relaxation oscillation thus producing quasi-periodic pulses of air which excite the vocal tract. Some examples of voiced sounds are: |u|, |d|, |w|, |i|, and |e|.[5]

The spectrum of a speech waveform is the representation of the signal with which we can assess the “separation” of the component parts. The eventual derivation of needed information about those specific components may also result. The cepstrum represents a transformation on the speech signal where the representatives of the component signals are separated in the cepstrum. These representatives are linearly combined. The cepstrum may also serve to be sufficient enough to provide the needed information on the properties of the component signals. Linear filters can also be used to then remove undesired cepstral components.[5]

When using a cepstrum, the typical scale used is the *mel* spaced cepstrum. A mel is a unit of measure of perceived pitch or frequency of a tone. The mapping of the mel scale is linear up to 1 kHz and then logarithmic above 1 kHz. The mel scale does not correspond linearly to the physical frequency of the tone as the human auditory system fails to perceive pitch in a linear manner. Perception of a particular frequency by the auditory system is influenced by energy in a critical band of frequencies around that particular frequency. The bandwidth of the critical band will vary with frequency, beginning around 100 Hz for frequencies below 1 kHz and then increasing logarithmically above 1 kHz.

To use the FFT in computing the cepstrum, the following equations must beIn order to define the complex cepstrum by the FFT, we have to define the complex logarithm of the Fourier transform. A constraint that the complex cepstrum of a real input sequence is also a real sequence must also be imposed. One should recall that the Fourier transform is an even function, and that the imaginary part of the transform is odd. For the complex cepstrum to be a real sequence, the log magnitude function has to be an even function of ω, and the phase must be an odd function of ω.[5]

$$Xp(e^{jw}) = \sum_{n=0}^{N-1} x(n)e^{-jwn}$$

$$N \le k \le N-1$$

$$Xp(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi k\frac{n}{N}}$$

The FFT can be used to approximate the cepstrum equations efficiently. Thus, the approach for computing the complex cepstrum is to replace all of the Fourier transform operations with the corresponding FFT operations. The following equations show how to find the cepstral coefficients based on the FFT analysis approach. c(n) represents the cepstrum.[5]

$$c(n) = \frac{1}{N}\sum_{k=0}^{N-1} (\log|Xp(k)|)e^{-j2\pi k\frac{n}{N}}$$

$$0 \le n \le N-1$$

Because of aliasing that is inherent in the use of the FFT in the computation of cepstral coefficients, it is necessary to use a rather large value of N, the number of points of the FFT or stated another way, the high rate of sampling the Fourier transform.[5]

If the original signal is defined to be symmetrical, the FFT used in cepstral analysis can be replaced by a discrete cosine transform (DCT). This principle is applied to the evaluation of the real and complex “pseudocepstrum” of speech signals. In both the real and the complex cepstrum cases, it is found that the use of the DCT does not degrade the information contained in the cepstrum, but it does substantially reducing the computational complexity.[19]

13.2. LP Cepstrum vs. FFT Cepstrum

Based on studies performed on LP and FFT based cepstrum analysis, it has been shown that the spectral envelope derived from the LP cepstrum is slightly different from the spectral envelope derived from the FFT cepstrum. Several experiments were performed using six utterances. The size of the time window was fixed to 256 samples to extract the FFT cepstrum. The window size for the LP cepstrum measurements was set to 30 msec. The FFT cepstrum computational time was almost twice that of the LP computational time. A key reason for this difference is that two Fourier transformations are included in the FFT based approach. In terms of distance ratio measures, the FFT and the LP results were relatively the same, except for the fact that the FFT results focused on the first-order cepstrum. The LP cepstrum produces almost the same results in speech recognition as the conventional FFT cepstrum. LP based analysis tends to perform poor in noisy enviroments whereas the FFT performs well.[21]

14. FEATURE SELECTION

The final goal of all speech recognition systems is to devise an automatic, time-independent, unbiased system that can duplicate the human ability to perform fast, accurate, and text-independent speaker recognition. The task is not a trivial one. Once the acoustic attributes of speech have been extracted and compared with a reference set, the attributes will be recognized if there is a close enough correlation between the two sets. Thus, the speaker is said to be recognized. In text-dependent speech recognition in which the test and the reference features are derived from the same text material, meaningful comparisons between the two sets can be made after time aligning the utterances. The same situation fails for text-independent cases where the test and reference sets bear no linguistic relationship to one another. The success of text-independent speech recognition must therefore depend on the extraction of a set of acoustic properties that can characterize each speaker independent of the text.[15]

By time averaging the acoustic attributes of speech over different speech text, it was shown that some features (pitch, reflection coefficients, gain, etc.) exhibited large inter-speaker variability regardless of the context. Therefore, long-term averaging of acoustic features seems suitable for text-independent speech recognition. Not all acoustic features of speech are useful to speech recognition. A selection procedure must be composed to keep only those features which give the best results[15].

To select the k best features from an entire set N, the optimal method is to consider all the combinations of N objects taken k at a time and exhaustively search for the best one. This method requires a tremendous amount of computation which leads to the implementation of suboptimal schemes, such as search without replacement (“knock-out” strategy). These algorithms start with the evaluation of the N features one at a time and “knocks-out” the most effective feature. This feature is then coupled one at a time with the remaining N-1 attributes in the set. These feature pairs are then evaluated resulting in the knock-out of the best pair of features. The disadvantage of this approach is that the resulting subset which contains the best features is not necessarily the optimal subset of features.[15]

In automatic speech recognition, the features used are measured from the speaker’s speech, and each measurement of these features can be represented by a point in the N-dimensional feature space. Through repetition of the measurement process, a cluster of points are generated in the space and they are distributed according to some N-dimensional probability density function (pdf) which characterizes the variance in the speaker’s voice.[15]

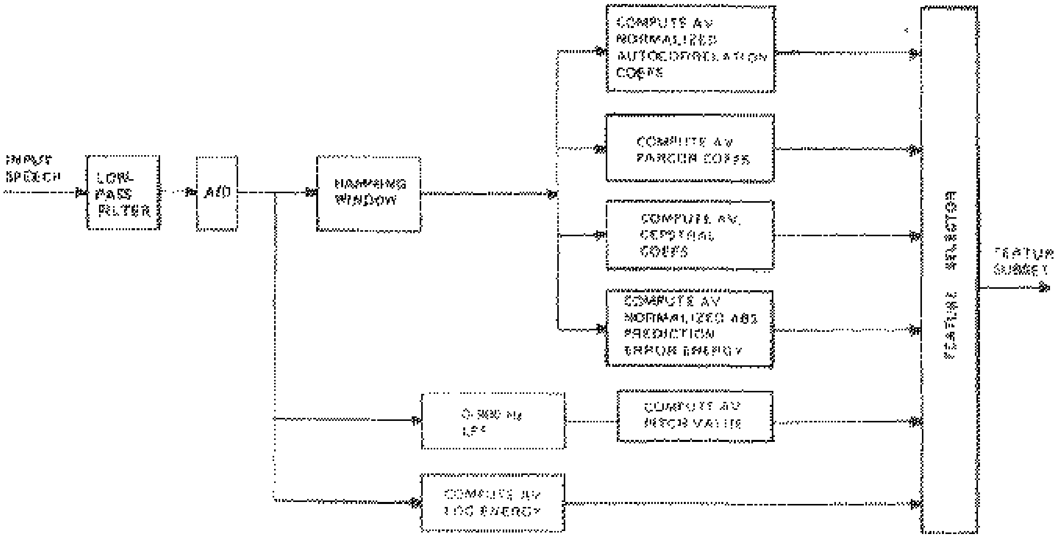

Figure 4. Feature Selection Flow Chart.

Before making a decision about the goodness of some features, the classifier used to distinguish the speaker pdf's has to be established. Once this decision logic has been formed, the features can be evaluated. For identifying unknown speaker's, a meaningful effectiveness measure is the error performance of the features over some test data. The set of features that commits the most errors in identifying a group of speaker's is said to be the least effective one. An effectiveness measure of this stature can be determined experimentally by using the attributes in the identification experiment and summing up the mistakes made, sometimes termed a scoring function. Implementation of this measure requires an extraordinary amount of computation. Another method is to exploit the statistical properties of the features and compute the probability of the error from the speaker's pdf. This scheme involves the estimation of the multidimensional distribution from a set of labeled training samples. If the distribution happens to be Gaussian, the probability or the error is found by integrating over the error range. This calculation is difficult to perform and is a tedious process.[15]

Figure 4 shows a schematic of feature selection process.

For example, suppose four lists of sentences were read by ten different speakers to comprise a database. These sentences bear no linguistic relationship to one another, but they are phonetically balanced. Each list is composed of ten sentences. The first two lists were used as the training set while the last two lists were used as the test set. From these lists, a set of 32 features can be found from the input speech. These features are: pitch value, log energy, ten PARCOR coefficients (partial correlation coefficients), ten cepstral coefficients, normalized absolute prediction error energy, and nine normalized autocorrelation coefficients. Pitch value, M, can be determined by using the average magnitude difference function. The log energy (in decibels) is computed with the following formula

$$E = 10\log\frac{1}{L}\langle\sum_{i=1}^{L} S_i^2\rangle$$

where S_i is the input speech and L is the frame length (frame duration * sample frequency). The linear prediction coefficients can be found through the use of a 10th order linear predictive analysis on the Hamming -windowed speech waveform using the autocorrelation method. The normalized autocorrelation coefficients Ri were calculated with

$$R_i = \frac{1}{R_0}\sum_{t=0}^{L-i-1} S_t S_{t+1}$$
$$i = 2, 3, \ldots, p$$

where *p* = the order of the linear predictor, and Ro is given by

$$R_0 = \sum_{t=0}^{L-1} S_t^2$$

The PARCOR coefficients, K_i, and the normalized absolute prediction error energy, |e|, were determined using the Levinson-Durbin recursion algorithm. The cepstral coefficients, C_i, were derived from the predictive coefficients.[15]

To allow the feature set to be applicable to the text-independent speech recognition system, each feature was averaged over some input test

$$\langle x_i \rangle_j = \frac{1}{L_v}\sum_{j=1}^{L_v} x_{ij}$$

where x_{ij} was the *i*th feature derived from the *j*th speech fame. L_v was the number of frames used in the averaging process. Also of note is the fact that only voiced frame features were considered because silence, voice and unvoiced speech are assumed to be sample functions of different random processes.[15]

15. SUMMARY

Because of the vital role that speech plays in the everyday lives of people, speech recognition is a much desired area of research. Whether this research is being performed simply to make the everyday, routine tasks of life easier for people or whether it's purpose is to aid the physically challenged, speech research will be carried out well into the next century. From voice activated automatic teller machines to operator assistance to controlling your personal computer with your voice instead of a mouse, the need for robust speech recognition systems is evident.

In order for a speech recognition system to be the best quality, most robust system, strong emphasis must be placed on the signal modeling component of the speech recognizer. If you do not start off on the right foot so to speak, how will your end result be the best that it can be? For that reason, the front-end of the speech recognition system must be carefully constructed. Many useful DSP tools enable the front-end to be duly constructed to minimize the error in the performance of the overall recognition system.

Once the front-end of a speech recognizer has been thoroughly tested, thoroughly meaning exhaustive tests on extensive databases, the front-end can be incorporated into a speech recognition system that is composed of the front-end, search, and language modeling.

16. ACKNOWLEDGEMENTS

The author is grateful to Dr. Joseph Picone for all the support he has given towards this project and also supplying the needed software tools and environment to perform this project. Additional thanks goes out to the fellow students in the EE8993 Speech Recognition class at Mississippi State University. Without the aid and support of the students in the class, certain aspects of development and research would not have been possible.

REFERENCES

18. Picone, J., “Signal Modeling Techniques in Speech Recognition,” *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215-1246, Sept. 1993.
19. Rabiner, L.R. and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Inc., 1978.
20. Proakis, J. G. and D. G. Mandakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Macmillian Publishing, 1988.
21. Markel, J.D. and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag Publishers, 1982.
22. Deller, J. R., Jr., J. G. Proakis, and J. H.L. Hansen, *Discrete-Time Processing of Speech Signals,* Macmillian Publishing, 1993.
23. Flanagan, J. L. and L.R. Rabiner, *Speech Synthesis*, Dowden Hutchinson and Ross, Inc., 1973.
24. http://www.uninova.pt/~tr/home/tooldiag.html
25. http://www.speech.su.oz.au/comp.speech/Section6/Q6.5.html
26. ftp://ftp.cs.cmu.edu/project/fgdata/speech-compression/LPC/
27. http://www.lhs.com/
28. Pan, R. and C.L. Nikias, “The Complex Cepstrum of Higher Order Cumulants and Nonminimum Phase System Identification,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 2, pp. 186-205, Feb. 1988.
29. Chen, Y., “Cepstral Domain Talker Stress Compensation for Robust Speech Recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 433-439, Apr. 1988.
30. Lee, C.H., “On Robust Linear Prediction of Speech,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 5, pp. 642-650, May 1988.
31. Furui, S., “Comparison of Speaker REcognition Methods Using Statistical Features and Dynamic Features,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 448-449, June 1981.
32. Cheung, R.S. and B.A. Eisenstein, “Feature Selection via Dynamic Programming for Text-Independent Speaker Identification,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 5, pp. 397-402, Oct. 1978.
33. Picinbono, B. and J.M. Kerilis, “Some Properties of Prediction and Interpolation Errors,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 525-531, Apr. 1988.
34. Stoica, P. and A. Nehorai, “On Linear Prediction Models Constrained to Have Unit-Modulus Poles and Their Use of Sinusoidal Frequency Estimation,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 6, pp.

940-941, June 1988.

35. Shichor, E. and H.F. Silverman, "An Improved LPC Algorithm for VOiced-Speech Synthesis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 1, pp. 180-182, Feb. 1984.
36. Hassenein, H. and M. Rudko, "On the Use of Discrete Cosine Transform in Cepstral Analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 4, pp. 922-923, Aug. 1994.
37. Kobzyashi, T. and S. Imai, "Spectral Analysis Using Generalized Cepstrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1235-1237, Dec. 1984.
38. Furui, S., "Cepstral Analysis Technique for Automatic Speaker Verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 254-272, Apr. 1981.
39. Schroeder, M.R., "Direct (Nonrecursive) Relations Between Cepstrum and Predictor Coefficients," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 297-301, Apr. 1981.
40. Jaroslavski, L.P., "Comments on 'FFT Algorithm for Both Input and Output Pruning'," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 3, pp. 448-449, June 1981.
41. Barnwell, T.P. III, "Recursive Windowing for Generating Autocorrelation Coefficients for LPC Analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 5, pp. 1062-1066, Oct. 1981.
42. Steiglitz, K. and B. Dickenson, "The Use of Time-Domain Selection for IMproved Linear Prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 1, pp. 34-39, Feb. 1977.
43. Kolba, D.P. and T.W. Parks, "A Prime-Factor FFT Algorithm Using High Speed Convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 4, pp. 281-294, Aug. 1977.
44. Carayannis, G., "An Alternative Formulation for the Recursive Solution of the Covariance and Autocorrelation Equations," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 6, pp. 574-576, Dec. 1977.
45. Gupta, V.N., J.K. Bryan, and J.N. Gowdy, "A Speaker-Independent Speech Recognition System Based on Linear Prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol 26, no. 1, pp. 27-33, Feb. 1978.
46. Bruun, G., "Z-Transform DFT Filters and FFTs," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 56-63, Feb. 1978.
47. Agrawal, J.P. and J. Ninan, "Hardware Modifications in Radix-2 Cascade FFT Processors," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 2, pp. 171-172, Apr. 1978.
48. Webster, R.J., "A Generalized Hamming Window," I*EEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 2, pp. 176-177, Apr. 1978.
49. Webster, R.J., "A Generalized Hamming Window," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 3, p. 269, June 1978.
50. Tadakoro, Y. and T. Higuchi, "Discrete Fourier Transform Computations via the Walsh Transform," *IEEE Transactions*

on Acoustics, Speech, and Signal Processing, vol. 26, no. 3, pp. 236-239, June 1978.

51. Jackson, L.B. and S.L. Wood, “Linear Prediction in Cascade Form,” *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 26, no. 6, pp. 518-528, Dec. 1978.
52. Meisel, W.S. “Commericial Applications of Speech Recognition,” TMA Associates, Dec. 1995.
53. *Numerical Recipes in C: The Art of Scientific Computing,* Cambridge University Press, http://world.std.com/~nr.
54. Picone, J. “Continuos Speech Recognition Using Hidden Markov Models,” *IEEE ASSP Magazine,* pp. 26-41, July 1990.
55. Reddy, R.N. and C.A. Ziegler, *FORTRAN 77 with Applications for Scientists and Engineers*, West Publishing Company, 1989.
56. Etter, D.M., *Structured FORTRAN 77 for Engineers and Scientists*, The Benjamin/Cummings Publishing Co., 1993.
57. *The Merriam-Webster Dictionary of Quotations*. Merriam-Webster, INC, 1992.