REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)

2. REPORT DATE
31 Jan 2000

3. REPORT TYPE AND DATES COVERED
SBIR Final Report, Topic AF99-103

4. TITLE AND SUBTITLE
Auditory Modeling for Noisy Speech Recognition

5. FUNDING NUMBERS
Contract F41624-99-C-6019

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Standard Object Systems, Inc.
105 Lisa Marie Place
Shalimar, FL 32579

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Department of the Air Force
AFMC 311th Human System Wing/PKR
8005 9th Street
Brooks AFB, TX 78235-5353

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Report developed under SBIR contract for topic AF99-103. Standard Object Systems, Inc. (SOS) has used its existing technology in phonetic speech recognition, audio signal processing, and multilingual language translation to design and demonstrate an advanced audio interface for speech recognition in a high noise military environment. The Phase I result was a design for a Phase II prototype system with unique dual microphone hardware with adaptive digital filtering for noise cancellation which interfaces to speech recognition software. It uses auditory features in speech recognition training, and provides applications to multilingual spoken language translation. As a future Phase III commercial product, this system will perform real time multilingual speech recognition in noisy vehicles, offices and factories. The potential market for this technology includes any commercial speech and translation application in noisy environments.

14. SUBJECT TERMS
SBIR REPORT

15. NUMBER OF PAGES 79

16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT

REF D

SOS

SBIR Final Report

Auditory Modeling for Noisy Speech Recognition

Prepared by

Standard Object Systems, Inc.

January 2000

Contract:
F41624-99-C-6019

Advanced Audio Interface for Phonetic Speech Recognition in a High Noise Environment

Contents

1.0 SCOPE OF THE RESEARCH PROJECT

This Phase I SBIR research report describes the creative concept, the component designs, the experiment analysis, and the proof of concept demonstration for using a dual head-mounted noise canceling microphone with an adaptive digital filter created by Standard Object Systems, Inc. (SOS) for speech recognition in noisy environments. The primary research goal is to improve computer speech recognition in noisy environments. Both DOD and commercial users have identified the failure of computer speech recognition in real world situations as a critical problem. The following introduction provides a brief summary of sound signal processing and adaptive digital filtering to improve noisy speech recognition.

SOS has applied its technology in phonetic speech recognition, digital signal processing, software development, audio hardware engineering, and acoustic speech science to perform this research. Numerous experiments were conducted with adaptive filters, noise models, speech recognizers, auditory feature models, and dual microphone PC hardware designs. The results of this Phase I effort are documented in this report, and in the accompanying CD ROM of sound files and data, and in the proof of concept demonstration conducted at the AFRL in Wright Patterson AFB.

Section 2 of this report provides an in-depth presentation of multiple adaptive digital filter designs and the performance results from the SOS experiments for noisy speech recognition. Section 3 addresses hardware design and interface software aspects of multiple PC microphones and sound cards. Section 4 compares several available PC speech recognition software products that SOS used to test the filter speech signals, and provides an analysis of the testing results. Section 5 presents the three auditory feature models analyzed by SOS for Phase II incorporation within the SPSR tool kit. Section 6 describes the operation of the SPSR tool kit and the modifications to use auditory feature data for noisy speech recognition. Section 7 defines the SBIR Phase II prototype development research plan and the Phase III commercial product. Section 8 gives a summary of the OV10 research task accomplished in Phase I, and Section 9 concludes the report.

1.1 SIGNAL PROCESSING AND SPEECH RECOGNITION

Signal processing is the science of encoding and decoding the information inherent in a signal. A signal is a means to convey information. Most often signals are energy traveling in the form of waves. The energy can be almost any waveform, sound, radar, electrocardio, etc., and the signals are recorded in graphs. Information extracted from any waveform can be difficult to interpret if the signal has been impaired, which usually occurs because the signal is noisy, distorted or incomplete.

In human speech communication the voice conveys words as acoustic signals. Humans can hear the words, and can extrapolate meaning even if the signals have been distorted. Machines cannot hear, they only process digitized information. This requires that the curve of a wave, a non-discrete analog signal, be transformed into discreet increments, or frequency domain, that can each be assigned a numeric value and be manipulated mathematically. The frequency domain identifies the frequency components of a sound, and from these components it is possible to approximate how the human ear perceives the sound. Speech recognition uses signal processing to segment a voiced signal into digital segments of data and compare those segments to known speech patterns. This requires complex algorithmic methods that have evolved with the availability of faster computers. With complex algorithms and neural networking computers appear to be able to hear and understand. But they cannot understand if they do not receive a clear signal, thus noise in speech recognition remains a paramount problem.

1.2 Speech Recognition and Noise

Noise is any unwanted sound. That is the definition for humans. For computers the definition of noise is irrelevant or meaningless information occurring along with desired information in the input or the output. The difference is important, because computers hear things humans do not notice. Humans have the ability to tune out the sound of their own breathing, the person at the next desk, the hum of a motor, or honking horns in the traffic outside the window. Computers are not subjective. They hear and attempt to process everything, even their own electrical noise.

In speech recognition noise refers not just to the interference of external sounds. There are stutters, wheezes, lisps, and clicks which, although an integral part of an individual's speech, contain no information and actually distort the speech signals to be recognized. These artifacts identify noisy speakers, and present a challenge for speech recognition engines. Because this type of noise cannot be separated from the actual voice input, it falls outside the parameters of this research. This project is focused on the additive noise, that not coming from the speaker, and the negation of it with adaptive filtering technologies which have not been used before.

1.3 Noise and Filters

The term filter describes a device, either hardware or software, which will enhance signal processing. As mentioned, the difficulties in signal processing are when the signal has noise, is distorted or is incomplete. Filters perform three functions. They can filter, smooth, or predict a signal or segments of a signal to enhance information processing. Filtering extracts information about a signal as it is being received. The danger in filtering is the risk of negating signals that have information value. If the transmission has gaps, smoothing can delay the processing for an instant in order that subsequent signals can be patched back into the earlier gap. This is often more reliable than filtering, as it does not rely on real time decisions. Prediction involves the anticipation of the future signal based upon data already received.

Filters can be classified as either linear or nonlinear. A filter is linear if the signal at the output is a linear function of the observations applied to the filter input. Otherwise the filter is nonlinear. Linear filter theory requires the data to be formulated by some statistical criterion and approached as discrete-time signals. It is often the case that to formulate a statistical criterion about the data requires a priori information about the statistics of that data. In other words, you have to know what the noise is before you can remove it. In the real world of filtering in noisy environments, having a priori knowledge of noise is not always possible.

A different strategy employs an adaptive filter. The adaptive filter relies on a recursive algorithm that is self-designing, which allows the filter to perform in situations where complete knowledge of the relevant signal is not available. It is a process where the parameters of the adaptive filter are updated from each iteration to the next, and the parameters become data dependent. An adaptive filter is in reality a nonlinear device, however they are commonly classified as linear or nonlinear. An adaptive filter is said to be linear if the estimate of a signal is computed adaptively as a linear combination of the available set of observations applied. Otherwise the adaptive filter is said to be nonlinear.

The filtering process can be described as either a time domain computation or a frequency domain computation. A time domain filter processes the sound samples one at a time based on the sample values, it does not compute frequency estimates. A frequency domain filter processes blocks of sound samples usually estimating the frequency components using a discrete Fourier transform. Voiced speech has a signal range between 100 Hz and 3000 Hz while unvoiced speech ranges up to 6000 Hz frequency. The Nyquist sampling limit says that a digital sample rate must be at least twice the highest frequency in the signal. The experiments conducted in this research sampled the speech signal at 16,000 Hz and the plan for Phase II is to use a 20,050 Hz sample rate.

Speech recognition systems use a microphone, usually spaced just a few inches from the speaker's mouth, for speech input. Usually these microphones have some filtering device to cancel noise. They often do not work well, as the noise signals are superimposed on the speech signals. The SOS design for an adaptive filter for speech recognition employs two microphones, one positioned at the speaker's mouth and one mounted behind the speaker's head, pointing away from the speaker. Each microphone feeds into separate sound cards in the computer so the signals are processed separately. The signals from behind the speaker represent mostly noise. The signals from the microphone at the mouth contain mostly speech with noise. The adaptive filter identifies the noise signal processed on one sound card and looks for similar spectra on the signal from the other sound card. It then removes any matching noise signal from the speech signal, often by applying the inverse of the signal. This is called noise canceling.

For experimental purposes SOS has created six filters. For each filter design the hardware configuration, the microphone, sound cards, and connectors remain the same. The software carries the signal processing and statistical algorithms; and these are different in each of the six filters. Phase I, the research experiments, take the following steps:

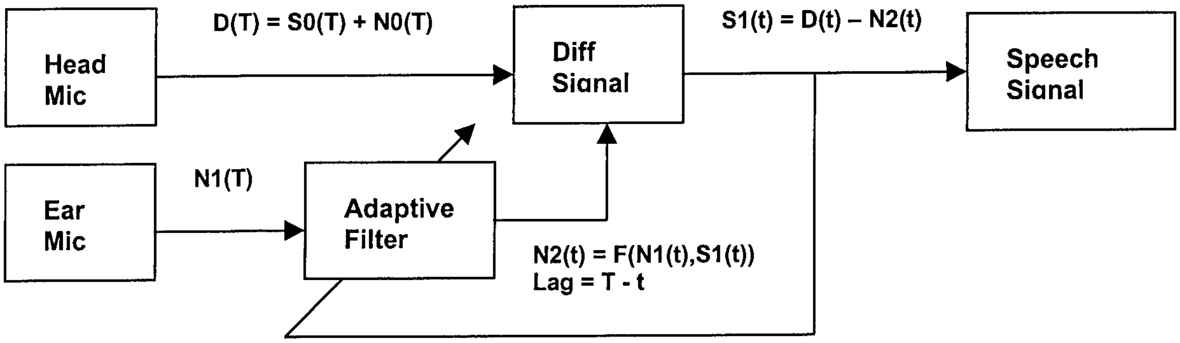
- Four speech recognition engines were selected.
- These four recognition engines were given speech samples from the TIMIT recorded speech and the results recorded.
- Four different noise levels are systematically added to the speech samples.
- The four speech recognition engines were given the same speech samples, with different levels of added noise, and the results recorded. The performance is expected to decline proportionally with the amount of added noise.
- The noise samples were processed through the six adaptive filters.
- The four speech recognition engines were tested with the filtered noise samples, and the results recorded.

The object of the experiments is to determine when one filter works better than the others, and whether any one filter may be best suited to work with a specific recognizer. The results provide a proof of concept for the Phase II design and development of a single optimized filter process for speech recognition noise cancellation.

2.0 ADAPTIVE FILTER FOR DIGITAL NOISE CANCELLATION

The ARO 1995 workshop for spoken human machine dialogue recognized voice as the logical choice for command and control in a vehicle, where it is difficult to use a mouse, touch screen, or keyboard. It also identified vehicle noise as a major factor in the mounted war, which factors battlespace, weapons firing, shock and vibration with covert operations that may require soft or whispered speech. Accurate speech recognition is hindered by noise and distortion. Distortion is due to equipment and environment. It is usually modeled as a linear effect that can be compensated for in known situations. Acoustic noise is modeled as additive to the speech signal, and special microphones are used to reduce it.

Figure 2.0-1 Adaptive Noise Canceling Filter



The adaptive filter signal enhancement for noisy environments will be accomplished as shown in Figure 2.0. Adaptive filters are used to solve four general categories of signal processing applications:

1) Identification of a linear model of a noisy process.
2) Inverse modeling to determine the best-fit parameters to a noisy process.
3) Prediction of the current value of a noisy signal.
4) Canceling interference such as echoes, noise, and beamforming.

The goal of environmental noise removal from a speech signal falls into the fourth category. Adaptive noise canceling is the removal of noise from a received signal in a changing manner to improve the signal to noise ratio. Naïve direct filtering of noise from a signal can produce disastrous results by increasing the average power of the output noise. When proper provisions are made to control filtering with an adaptive process, however, it is possible to achieve superior performance over direct filtering.

The proposed adaptive noise canceling filter will use a dual input non-causal closed loop adaptive feedback filter:

1) The signal source d(t) will be from a directional head mounted microphone.
2) The noise source n1(t) will be from an ear piece mounted wide field microphone.
3) The signal source d(t) is the speech signal s0(t) corrupted with additive noise n0(t). The signal and noise are uncorrelated and real valued.
4) The noise source receives a noise signal n1(t) that is uncorrelated to the speech s(t) but correlated to the noise n0(t) in a way that can be modeled as a cross correlation with lag.
5) The noise estimate is computed by an adaptive filter to create an estimate of the additive noise n2(t) by using the filtered signal source s1(t).

6) The filtered signal source s1(t) is created by subtracting the noise n2(t) from the signal source d(t).
7) The filter is non-causal with output time t less than input time T since the speech used in speech recognition can have a delay prior to speech recognition.

The reduction of acoustic noise in speech such as in a military vehicle can be accomplished with adaptive noise canceling. Reference microphones are placed so that they capture only the noise. For a non-causal application such as speech recognition, the weighted and smoothed estimate of the noise is removed from the signal that is delayed for input to the speech recognizer. In a digital system, the effective recognition of silence can reduce the latency delay in the speech processing by only processing speech utterance signals.

A widely published successful mathematical approach to the adaptive filtering problem is to solve for the maximum likelihood estimate of the speech wave in the presence of the noise. The general algorithm is to compute the maximum a posteriori (MAP) estimator that optimizes the probability density function of the unknown parameters given a model of the noisy observations. This algorithm has sizable computational requirements, and is sensitive to control parameters for an automatic enhancement process.

The adaptive noise removal for speech recognition enhancement uses a reference noise signal input to create an estimate of the noise to subtract from the speech signal at a time prior to the current signals. The output from the noise removal is used to control the adaptation and time lag to minimize the mean square value of the delayed speech output for recognition. This will result in the minimum mean square error speech signal for recognition.

SOS is interested in alternative approaches by comparison to this general adaptive filter and the enhanced speech signal data. With respect to the speech signal, three noise environment models will be considered: stationary, non-stationary, and quasi-stationary. A stationary noise environment is defined by the rate of change of the optimal prediction filter that is constant, such as an idling engine or electronic hum. A non-stationary noise environment is defined by the rate of change of the optimal prediction filter that changes more rapidly than the speech prediction, such as a gunshot or ringing. A quasi-stationary noise environment is defined by the rate of change of the optimal prediction filter that is of the same time order as the speech filter, such as other nearby speakers or radio chatter. SOS will investigate the relative rates of change of the optimal speech filter versus the optimal noise filter for each of these noise environments.

During Phase II testing a number of simplifying assumptions will be tested as alternative approaches to determine the best trade off between speech enhancement and operational implementation. For example: the use of an all pole model for the speech parameters; the use of a two step sequential estimation process rather than iteration; the use of spectral constraints on filter pole locations; the use of time domain smoothing for vocal tract constraints; and the use of line spectral pairs as a simpler alternative to other spectrum models. In all cases, the SOS evaluations will be made on a perceptual and spectral basis to determine the best adaptive digital filter for speech signal pre-recognition enhancement.

2.1 TRADITIONAL APPROACHES TO NOISE CANCELING

A high noise environment makes intelligible speech communication difficult, both between people, and between people and equipment. Speech intelligibility is highly affected by noisy vehicles. Although automatic speech recognition is affected dramatically by even moderate noise, a computer speech recognition system has two unusual advantages in removing noise. An adaptive non-causal filter can effectively predict and remove the vehicle noise, and modifications to the recognition process can improve the accuracy with a priori knowledge of the noise environment. There is no unique solution to linear adaptive filter problems. Rather, a set of tools exists using various recursive algorithms, each of which offers specific desirable features. The challenge in adaptive filtering is to understand the capabilities and limitations of the various adaptive filtering algorithms and to select the appropriate algorithm for the application at hand.

Numerous options exist in the classification of linear adaptive filters. The following table organizes several published algorithms into three classes by sample and block updates. The following acronyms are used in the table: LMS least mean square, DCT discrete cosine transform, GAL gradient adaptive lattice, RLS recursive least squares, SOBAF self orthogonalizing block adaptive filter, and LS least squares.

Algorithm Class	Sample Update	Block Update
Stochastic Gradient	**LMS**	**Block LMS**
Orthogonalizing	**DCT-LMS, GAL**	**SOBAF**
Least Squares	**RLS**	**Block LS**

2.1.1 Least Mean Square (LMS) Algorithm

Widrow and Hoff originated the least-mean-square (LMS) algorithm in 1960. It is a member of the stochastic gradient class of algorithms, which is different than the method of steepest descent that uses a deterministic gradient. The significant feature of the LMS algorithm is its simplicity. It does not require correlation functions or matrix inversion. In general the LMS process involves these operations:

1. The computation of a transverse filter output u(n) from tap weights w(n).
2. A known desired response function d(n).
3. Generation of an estimation error e(n) = d(n) - u(n).
4. Adaptive adjustment of the filter tap weights w(n).
5. Minimization of the estimation error e(n).

This is usually implemented as a feedback loop. The tap input and desired response are from a jointly wide sense stationary environment. For this environment the steepest descent is down the ensemble-averaged surface while the LMS behaves differently due to gradient noise. The stability of the LMS algorithm can be convergent in the mean square under certain step size conditions. In summary, the LMS algorithm is simple in implementation yet capable of high performance by adapting to its external environment. The LMS algorithm operates on stochastic inputs that makes the allowed set of directions per step of the iteration cycle quite random.

A common example of LMS use in adaptive noise canceling is the recovery of information corrupted by a known interference signal such as a sine wave noise in digital information. The traditional solution is a narrow band notch filter tuned to the sine wave frequency. But if this interference frequency drifts over time, an adaptive noise-canceling filter is required. The filter designed for this situation using the LMS algorithm has two characteristics:

1. The noise filter is an adaptive notch filter whose center frequency is determined by the interference frequency and is tunable.
2. The notch can be made very sharp at the interference frequency by choosing a small enough step size parameter.

The key questions in any adaptive filter design are the selection of the parameters to optimize the step size, the noise rejection, and the response characteristics of the system. These parameters are often chosen by modeling the desired ideal system and mathematically optimizing the model to determine the "best" parameter values. SOS proposes to replace this a priori modeling approach with the use of a Genetic Algorithm that will evolve a set of parameters for a complex changing noise environment.

The LMS algorithm computational summary:

- Parameters: number of filter taps "n", step size "s"
- Initialize: tap weight vector w(n) = 0
- Input Data: u(n) tap vector, d(n) desired response
- Compute estimated tap weights: w(n+1) at step n+1
- Compute error: e(n) = d(n) - {w(n) u(n)}
- Compute update: w(n+1) = w(n) + s u(n) e(n)

2.1.2 Frequency Domain Adaptive Filtering

The previous section described the use of the LMS algorithm for time domain filtering. A finite impulse response (FIR) filter set of weights is adapted to a changing noise source in the time domain using small step sizes. It is equally feasible to perform the adaptation of filter parameters in the frequency domain using the Fourier transform that traces back to Walzman and Schwartz in 1973. The two major reasons to use frequency domain adaptive filters are as follows:

1. Often the adaptive filter has a long impulse response that may require the use of infinite impulse response filters in the time domain, which leads to instability.
2. Frequency domain adaptive filters can improve the convergence and response of the LMS algorithm by exploiting the orthogonality properties of the discrete Fourier transform.

One approach to increasing the speed of large FIR filters is to use a block implementation that uses parallel processing. The incoming data is sectioned into L-point blocks and the adaptation is on a block basis rather than on a sample basis. The convergence property of the block LMS algorithm is similar to the standard approach. Within the conditions of long filters and slowly changing signals this method often works very well.

Given a noise removal application where the block LMS algorithm is a reasonable approach, the question is how to create a fast LMS algorithm. The computation of the block LMS algorithm involves the linear convolution of the inputs with the tap weights and the update equation is a linear correlation between the tap inputs and the error signal. The fast Fourier transform (FFT) provides a powerful tool for performing fast convolution and correlation. This implies a frequency domain method for the efficient implementation of the block LMS algorithm. Specifically the adaptation of the filter parameters is performed in the frequency domain and was called the fast LMS algorithm by Clark in 1982. The following summarizes the fast LMS computations for block data:

- Parameters: m block size, s step size
- Initialize: W(0), P(0), k block number
- Compute per input block:
- U(k) transformed input FFT[u]
- y(k) overlap and save convolution data IFFT[U W]
- e(k) = d(k) - y(k) error signal
- E(k) = FFT[e] frequency domain error
- D(k) frequency domain desired response
- P(k) = IFFT[D U E] linear correlation
- Update weights: W(k+1) = W(k) + s FFT[P]

2.1.3 Tracking Time Varying Systems

The operation of adaptive filters in nonstationary noise environments involves the real world problem of tracking time varying systems. The problem is that the minimum of the error surface is no longer fixed. The adaptive filter has to track the minimum point of the error surface, which must change slowly enough to be identified. Tracking is a steady state process as contrasted with convergence, which is a transient phenomenon. In practice an adaptive filter must pass from a transient mode to a steady state mode before tracking can be accomplished. The rates of convergence and tracking are two different properties that are not necessarily possessed simultaneously by an adaptive filter.

Nonstationary environments arise in two fundamental ways. The desired response may be time varying or the stochastic input process may vary with time. Both of these conditions have an affect on an adaptive filter implementation. For the LMS filter the variation in the desired response implies that the correlation matrix for the inputs remains constant while the cross correlation between the inputs and response is time varying. When the stochastic input process varies both the correlation and cross correlation is time varying. This is usually the case for the noise removal from speech application. A popular model for time varying systems is a first order Markov process described by the following equation:

$$w(n+1) = k\, w(n) + n(0,Q)$$

where w( ) is the tap weights vector, n is the step, k is a fixed model parameter, and n( ) is a noise vector with zero mean and Q correlation. The desired response vector d( ) is defined by:

$$d(n) = w(n)\, u(n) + v(n)$$

where u( ) is the input vector and v( ) is the measurement noise. The error signal e( ) of the process is defined by:

$$e(n) = w(n)\, u(n) + v(n) - we(n)\, u(n)$$

where we( ) is the estimated tap weight vector at step n. In order to apply the LMS algorithm in this situation the we will assume independent process noise with the input vector, and white measurement noise.Recent advances in adaptive LMS filter algorithm research have been made by Benveniste (1990) and have been supported by application designs by Brossier (1992) and a proof of convergence provided by Kushner (1995). The result is an LMS design with adaptive gain wherein we propose to set the parameters by using a Genetic Evolution method.

A number of computational experiments are planned during the development of the LMS adaptive noise filtering algorithm. These experiments will be programmed using a variety of software tools such as MATLAB, MathCAD, BASIC, C++, and other programs. There are three purposes for the experiments:

1. Verify the correctness of the filter equations by independent development.
2. Validate the performance of the filter under known test conditions.
3. Proof of concept demonstration with noisy speech data.

An algorithm design and test specification using MATLAB that describes the algorithm processing, the independent verification of computer programming, the validation of predicted performance under known conditions, and the experimental performance with noisy speech data.

2.2 Adaptive Filter Experiments

During the Phase I design, SOS constructed a number of adaptive filter experiments as a proof of concept demonstration of the feasibility of this noise canceling technology. The following table classifies the six selected prototype filters by linear and nonlinear computation, time domain and frequency domain processing, the number of microphone inputs, and the unique speech signal reconstruction method.

NUM	NAME	L/NL	TD/FD	MIC	RECONSTRUCTION
1	Linear Adaptive Filter Bank	L	FD	2	Triangular IFFT Coherent
2	Magnitude FFT	NL	FD	2	Original Phase + Filt Mag
3	Log Magnitude FFT	NL	FD	2	Original Phase + Filt Mag
4	LMS ALE	L	TD	1	None, Time Shifted Output
5	Mag FFT with Iterative Recon	NL	FD	2	Phase Iteration
6	Iterative Recon with Spectral Sub	NL	FD	1	Noise Est by Scale Function

The following sections discuss each of these prototype adaptive filters in detail. Filter 1 is a linear adaptive filter that adjusts FIR coefficients per frequency bin for overlapping signal data blocks. In general the filtered sound performs better for speech recognition programs than it sounds to the human ear. Filter 2 is a magnitude FFT experiment that accepts two sound signal inputs and produces a filtered speech signal as output. Filter 3 is a log magnitude FFT version of Filter 1 that scales the noise by the logarithm of the magnitude to approximate the response characteristic of the human ear. In general both of these filters sound clear but are not easily classified by the speech recognizer programs.

Filter 4 is a least mean square adaptive line enhancer filter that uses N speech samples to predict 2 * N samples ahead. It is a linear time domain computation that removes voiced speech from noise. The filter is limited to voiced speech signals and, as expected, performs poorly with the speech recognition programs.

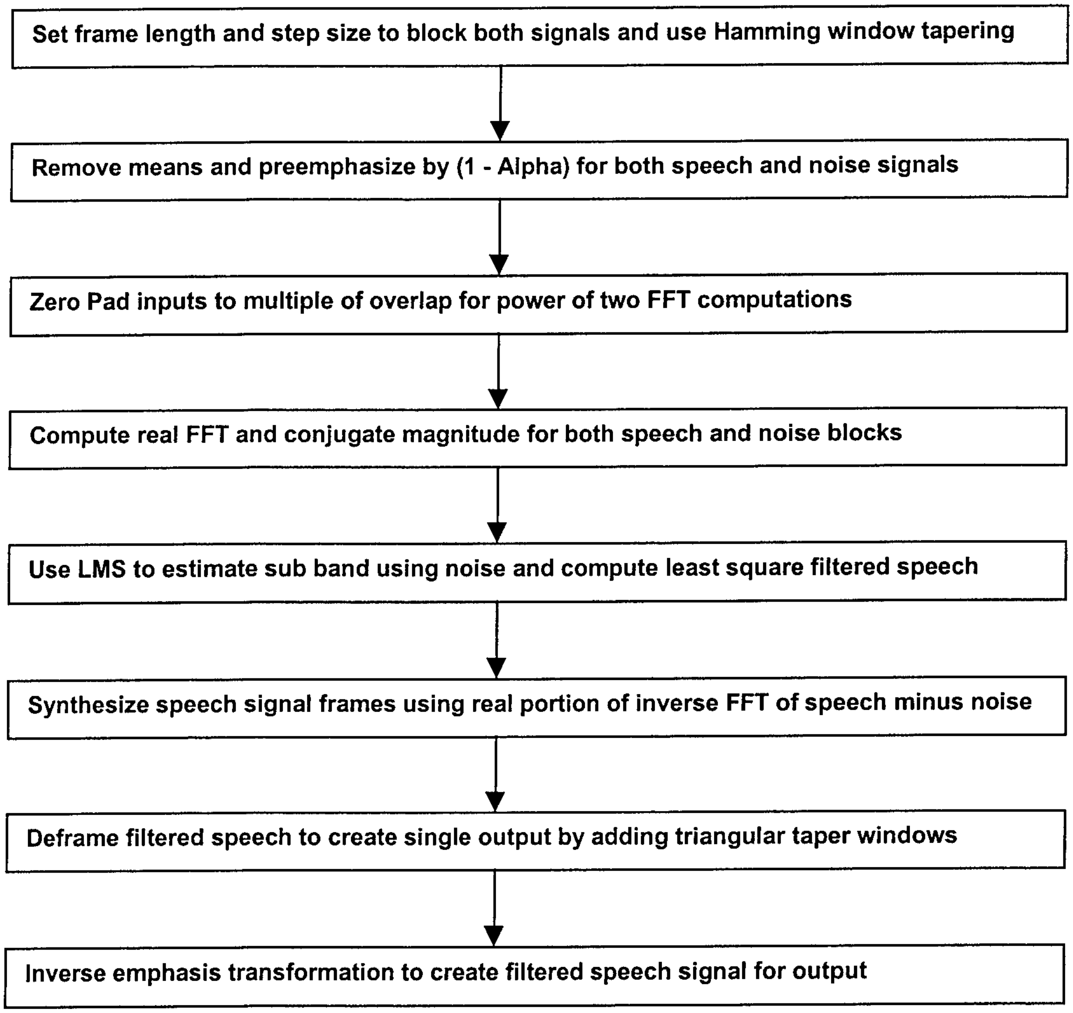
Filter 5 is a magnitude FFT adaptive filter that scales by the noise magnitude and uses iterative reconstruction for overlapping dual signal stream data blocks. Filter 6 uses the Filter 5 computation to remove the estimated noise from a single signal input stream. In general the reconstructed speech sounds of these two filters are more clear to the human ear and are easily classified by the speech recognition algorithms.

NUM	NAME	AVG %	MAX %	AVG N	MAX N
1	Linear Adaptive Filter Bank	29.58%	36.93%	90.5	113
2	Magnitude FFT	13.24%	26.80%	40.5	82
3	Log Magnitude FFT	14.46%	24.18%	44.25	74
4	LMS ALE	-22.06%	-10.46%	-67.5	-32
5	Mag FFT with Iterative Recon	6.54%	15.69%	20	48
6	Iterative Recon with Spectral Sub	18.06%	27.12%	55.25	83

The goal of the Phase I filter prototypes is to create a baseline design for the Phase II noise canceling filter development. Each of the six prototypes was tested with the same set of speech and noise signal files to produce a sound input file for speech recognition as shown in the above figure. Four speech recognizers were tested against each of the sound files to determine the correct word recognition performance. The two speech recognition performance values are for the average and maximum word recognition improvement. Based on the results of the speech recognition performance and subjective listening to the sound files, a baseline design for a Phase II adaptive noise canceling filter for speech recognition performance enhancement was created.

2.2.1 Filter 1 - Linear Adaptive Filter Bank Experiment

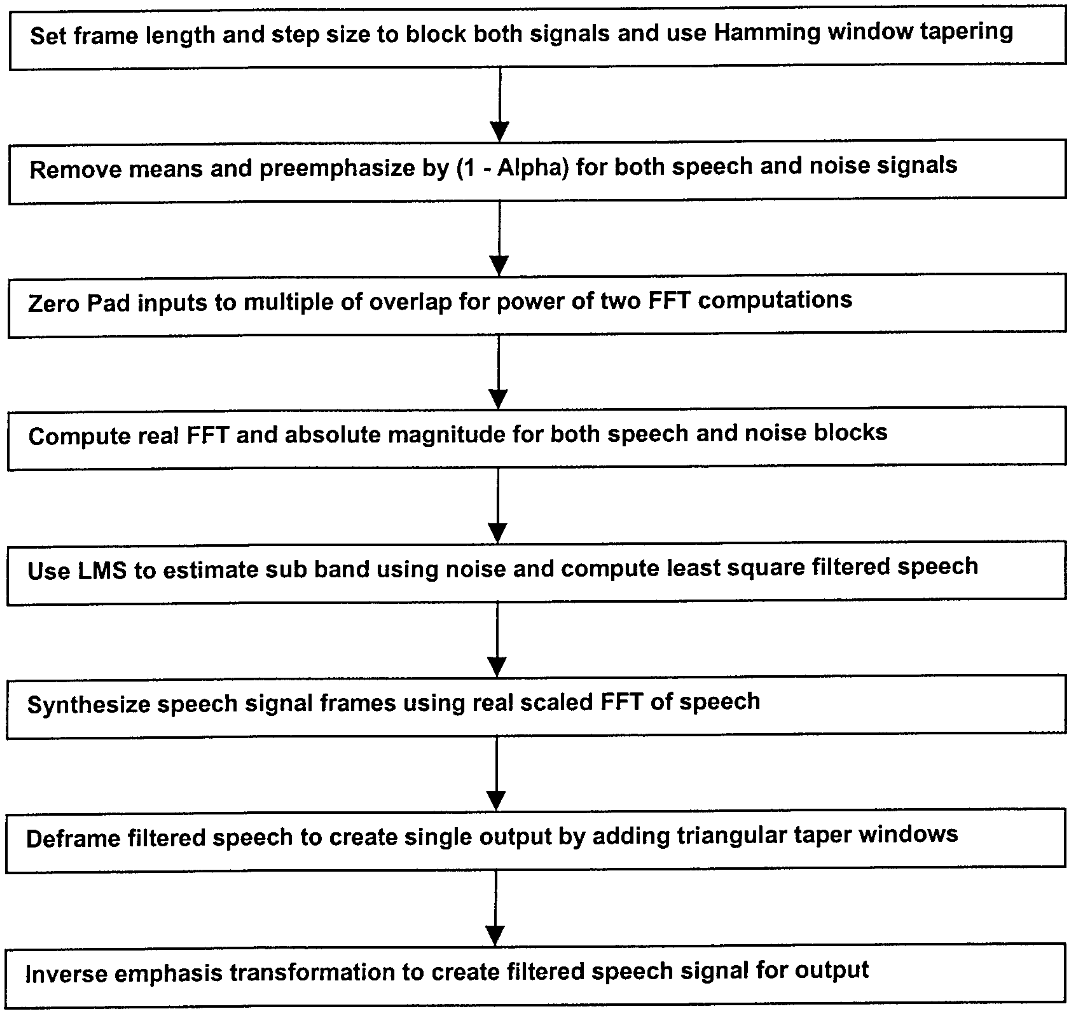
The linear adaptive filter bank experiment accepts two sound signal inputs and produces a filtered speech signal as output. This frequency domain process computes block overlapped FFTs for both the noise signal and the speech plus noise signal. This adaptive filter is designed to process noise observations from a head-mounted microphone and a desired speech signal from the boom microphone. The computational processing is shown in the following figure. This experimental program was programmed in MATLAB and the full listing is included in the appendix as file NFILT1.M.


Set frame length and step size to block both signals and use Hamming window tapering
Remove means and preemphasize by (1 - Alpha) for both speech and noise signals
Zero Pad inputs to multiple of overlap for power of two FFT computations
Compute real FFT and conjugate magnitude for both speech and noise blocks
Use LMS to estimate sub band using noise and compute least square filtered speech
Synthesize speech signal frames using real portion of inverse FFT of speech minus noise
Deframe filtered speech to create single output by adding triangular taper windows
Inverse emphasis transformation to create filtered speech signal for output

The linear adaptive filter adjusts the FIR coefficients per frequency bin for overlapping signal data blocks. It predicts the speech signal by using noise as the common portion of the two signal streams that is to be removed. The multi bin speech signal is recovered from the frequency domain using a triangular taper and an inverse complex FFT for coherent reconstruction. In general the filtered sound performs better for speech recognition than it sounds to the human ear.

2.2.2 Filter 2 - Magnitude FFT Experiment

The magnitude FFT experiment accepts two sound signal inputs and produces a filtered speech signal as output. This frequency domain process computes block overlapped FFTs for both the noise signal and the speech plus noise signal. This nonlinear adaptive filter is designed to process noise observations from a head-mounted microphone and a desired speech signal from the boom microphone. The computational processing is shown in the following figure. This experimental program was programmed in MATLAB and the full listing is included in the appendix as file NFILT2.M



Set frame length and step size to block both signals and use Hamming window tapering

Remove means and preemphasize by (1 - Alpha) for both speech and noise signals

Zero Pad inputs to multiple of overlap for power of two FFT computations

Compute real FFT and absolute magnitude for both speech and noise blocks

Use LMS to estimate sub band using noise and compute least square filtered speech

Synthesize speech signal frames using real scaled FFT of speech

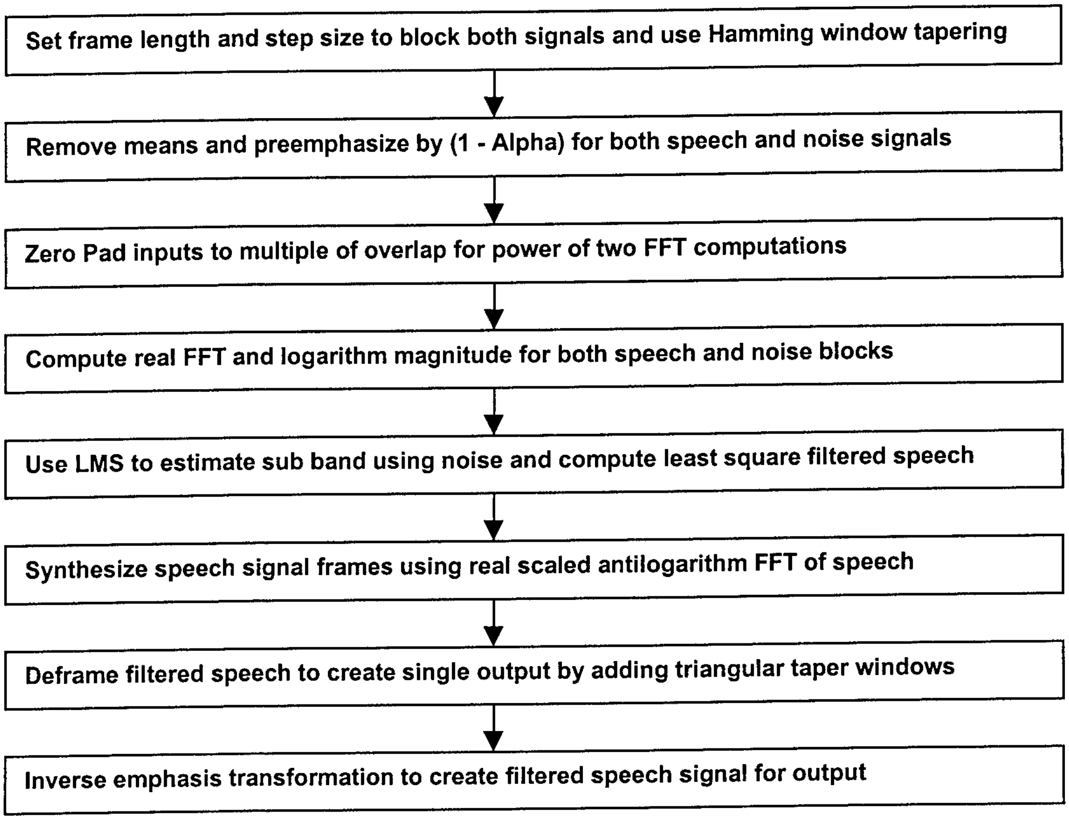
Deframe filtered speech to create single output by adding triangular taper windows

Inverse emphasis transformation to create filtered speech signal for output

The magnitude FFT adaptive filter scales the noise magnitude and uses the original phase for overlapping signal data blocks. It uses a least squares magnitude PSD estimate to remove noise as the common portion of the two signal streams. The multi bin speech signal is recovered from the frequency domain using a triangular taper and an inverse complex FFT with the original speech phase. In general the filtered speech sounds clear to the human ear but is not as easily classified by the speech recognition algorithms.

2.2.3 Filter 3 - Log Magnitude FFT Experiment

The log magnitude FFT experiment accepts two sound signal inputs and produces a filtered speech signal as output. This frequency domain process computes block overlapped FFTs for both the noise signal and the speech plus noise signal. This nonlinear adaptive filter is designed to process noise observations from a head-mounted microphone and a desired speech signal from the boom microphone. The computational processing is shown in the following figure. This experimental program was programmed in MATLAB and the full listing is included in the appendix as file NFILT3.M


Set frame length and step size to block both signals and use Hamming window tapering
Remove means and preemphasize by (1 - Alpha) for both speech and noise signals
Zero Pad inputs to multiple of overlap for power of two FFT computations
Compute real FFT and logarithm magnitude for both speech and noise blocks
Use LMS to estimate sub band using noise and compute least square filtered speech
Synthesize speech signal frames using real scaled antilogarithm FFT of speech
Deframe filtered speech to create single output by adding triangular taper windows
Inverse emphasis transformation to create filtered speech signal for output

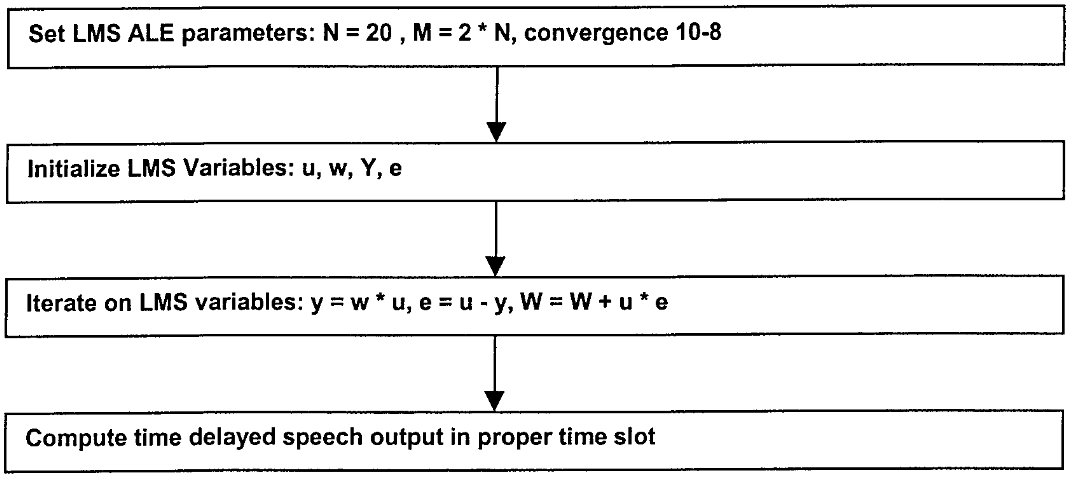
The log magnitude FFT adaptive filter scales the noise by the log magnitude to approximate the response characteristic of the human ear by the transformation:

$$\mathrm{LOG}[\ 1 + A * \mathrm{MAG}(\mathrm{FFT})\ ]$$

It uses a least squares log magnitude PSD estimate to remove noise as the common portion of the two signal streams. The multi bin speech signal is recovered from the frequency domain using a triangular taper and an inverse complex FFT with the original speech phase. In general the log magnitude filtered speech sounds clear to the human ear and is more easily classified than the magnitude filter.

2.2.4 Filter 4 - Least Mean Square Adaptive Line Enhancer Experiment

The least mean square adaptive line enhancer (LMS ALE) experiment accepts two sound signal inputs and produces a filtered speech signal as output. This time domain process uses both the noise signal and the speech plus noise signal. This linear adaptive filter is designed to process noise observations from a head-mounted microphone and a desired speech signal from the boom microphone. The computational processing is shown in the following figure. This experimental program was programmed in MATLAB and the full listing is included in the appendix as file NFILT4.M



Set LMS ALE parameters: N = 20 , M = 2 * N, convergence 10-8

Initialize LMS Variables: u, w, Y, e

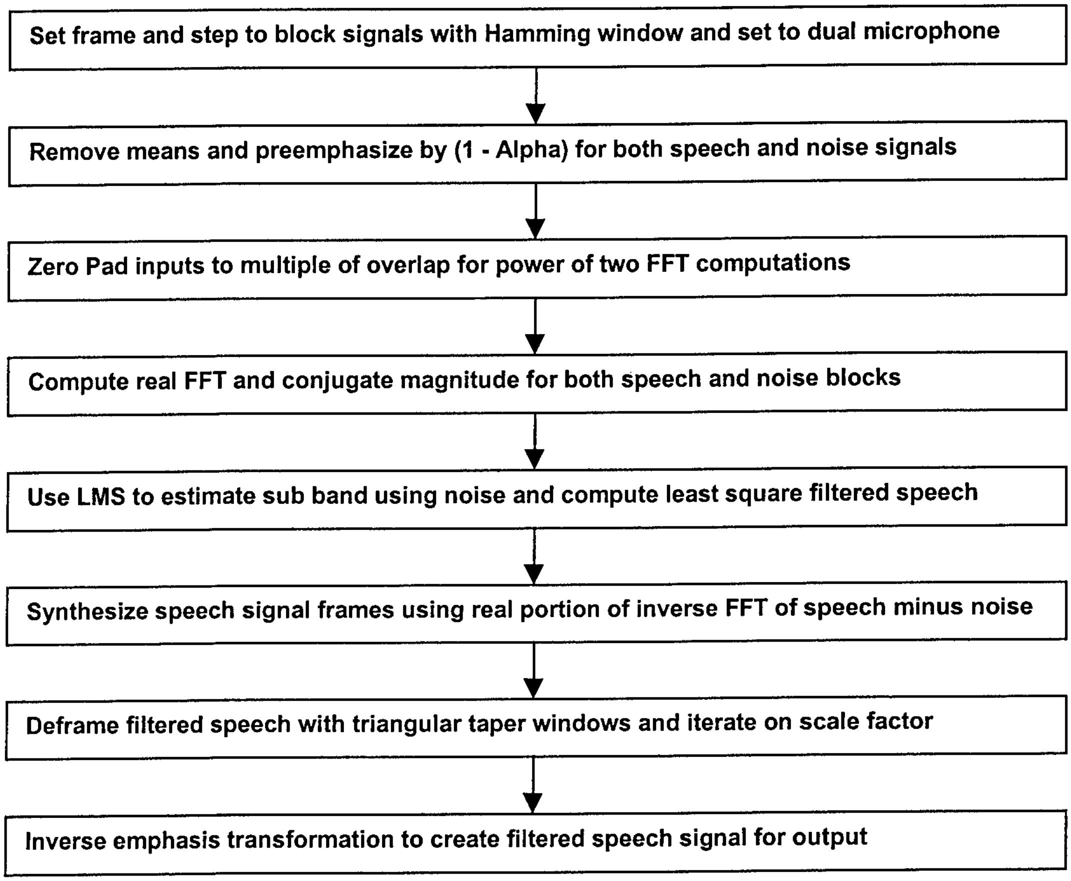
Iterate on LMS variables: y = w * u, e = u - y, W = W + u * e

Compute time delayed speech output in proper time slot

The least mean square adaptive line enhancer filter uses N speech samples to predict 2 * N samples ahead. It predicts the voiced speech signal by removing noise from the signal on a sample by sample basis. No unvoiced noise is removed and the reconstruction is on a single sample basis. In general the filtered sound performs poorly for speech recognition and it contains high frequency components in the output sound.

2.2.5 Filter 5 - Magnitude FFT with Iterative Reconstruction Experiment

The magnitude FFT filter with iterative reconstruction experiment accepts two sound signal inputs and produces a filtered speech signal as output. This frequency domain process computes block overlapped FFTs for both the noise signal and the speech plus noise signal. This nonlinear adaptive filter is designed to process noise observations from a head-mounted microphone and a desired speech signal from the boom microphone. The computational processing is shown in the following figure. This experimental program was programmed in MATLAB and the full listing is included in the appendix as file NFILT1.M



Set frame and step to block signals with Hamming window and set to dual microphone

Remove means and preemphasize by (1 - Alpha) for both speech and noise signals

Zero Pad inputs to multiple of overlap for power of two FFT computations

Compute real FFT and conjugate magnitude for both speech and noise blocks

Use LMS to estimate sub band using noise and compute least square filtered speech

Synthesize speech signal frames using real portion of inverse FFT of speech minus noise

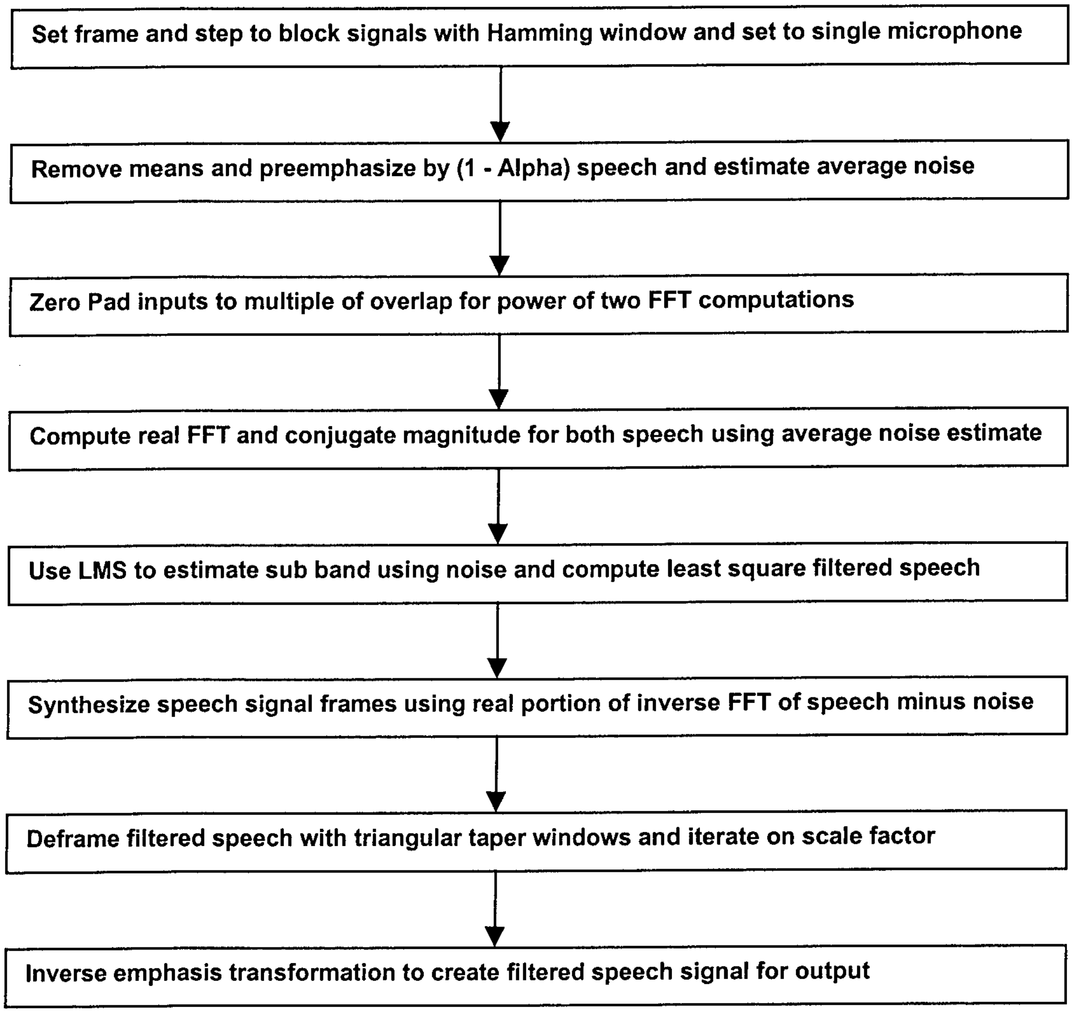
Deframe filtered speech with triangular taper windows and iterate on scale factor

Inverse emphasis transformation to create filtered speech signal for output

The magnitude FFT adaptive filter scales the noise magnitude and uses iterative reconstruction for overlapping signal data blocks. It uses a least squares magnitude PSD estimate to remove noise as the common portion of the two signal streams. The multi bin speech signal is recovered from the frequency domain using a triangular taper and an inverse complex FFT with the original speech phase. Iterative improvement is performed on the scale factor to match the magnitudes. In general the reconstructed speech sounds clearer to the human ear and is more easily classified by the speech recognition algorithms.

2.2.6 Filter 6 - Iterative Reconstruction with Spectral Subtraction Experiment

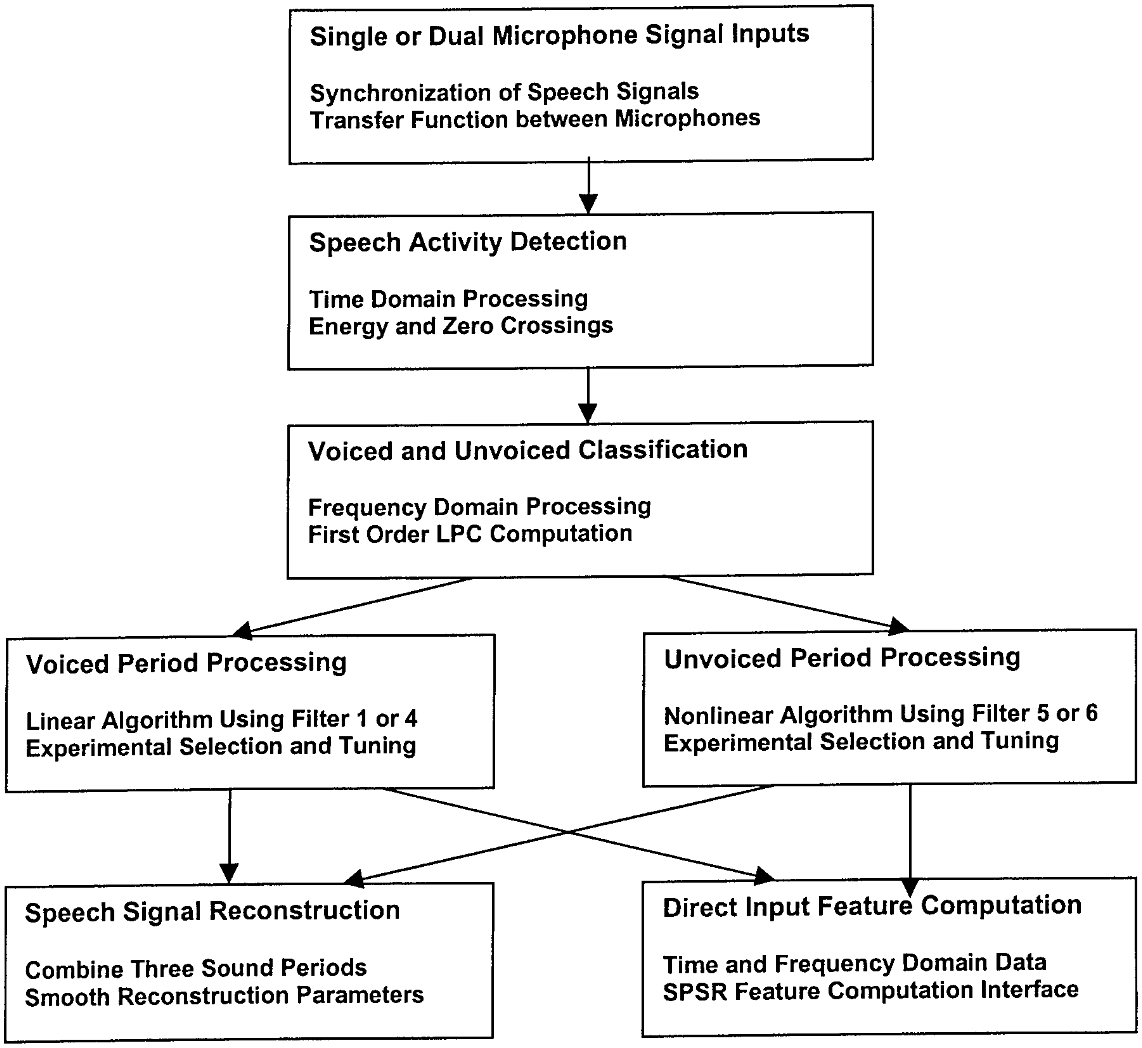
The magnitude FFT filter with iterative reconstruction and spectral subtraction experiment accepts one sound signal input and produces a filtered speech signal as output. This frequency domain process computes block overlapped FFTs for both the noise portion of the signal and the speech portion. This nonlinear adaptive filter is designed to process speech plus noise observations from a single boom microphone. The computational processing is shown in the following figure. This experimental program was programmed in MATLAB and the full listing is included in the appendix as file NFILT5.M using optional parameters for a single microphone


Set frame and step to block signals with Hamming window and set to single microphone
Remove means and preemphasize by (1 - Alpha) speech and estimate average noise
Zero Pad inputs to multiple of overlap for power of two FFT computations
Compute real FFT and conjugate magnitude for both speech using average noise estimate
Use LMS to estimate sub band using noise and compute least square filtered speech
Synthesize speech signal frames using real portion of inverse FFT of speech minus noise
Deframe filtered speech with triangular taper windows and iterate on scale factor
Inverse emphasis transformation to create filtered speech signal for output

The magnitude FFT adaptive filter scales the noise magnitude signal sample and performs iterative reconstruction with spectral subtraction for overlapping signal data blocks. It uses a least squares magnitude PSD estimate to remove the estimated noise as the common portion of the single signal stream. The multi bin speech signal is recovered from the frequency domain using a triangular taper and an inverse complex FFT using iterative reconstruction with spectral subtraction. Iterative improvement is performed on the scale factor to match the magnitudes. In general the single microphone reconstructed speech sounds clearer to the human ear and is easily classified by the speech recognition algorithms.

2.3 PHASE II ADAPTIVE FILTER DESIGN

The implementation of the adaptive filter in Phase II of this program will be based on the Phase I prototypes and test results. Two configurations are planned for delivery. First, an alpha configuration that executes on a desktop computer will be developed during year one of the project. This filter will be programmed in C++ and fully instrumented to provide test and evaluation data. This alpha configuration will be suitable for processing recorded speech input and producing recorded audio output. The second configuration is the beta delivery implemented for real-time execution possibly with a DSP unit with multiple microphone inputs and speech signal outputs. This configuration will be capable of demonstration in an operational environment test during year two of the project.


Single or Dual Microphone Signal Inputs
Synchronization of Speech Signals
Transfer Function between Microphones
Speech Activity Detection
Time Domain Processing
Energy and Zero Crossings
Voiced and Unvoiced Classification
Frequency Domain Processing
First Order LPC Computation
Voiced Period Processing
Linear Algorithm Using Filter 1 or 4
Experimental Selection and Tuning
Unvoiced Period Processing
Nonlinear Algorithm Using Filter 5 or 6
Experimental Selection and Tuning
Speech Signal Reconstruction
Combine Three Sound Periods
Smooth Reconstruction Parameters
Direct Input Feature Computation
Time and Frequency Domain Data
SPSR Feature Computation Interface

Based on the Phase I adaptive filter experiments, SOS has created a baseline Phase II design for an adaptive noise canceling filter that will improve speech recognition performance. The design is a combination of the results from the Phase I proof of concept filter experiments. This proposed Phase II design has three major processing components. First, a speech activity detector with a voiced/unvoiced state classifier. Second a linear filter for voiced period speech processing. Third, a nonlinear filter for unvoiced period speech processing. The combination of these three stages will result in the development of an innovative and unique noise cancellation system targeted to speech recognition enhancement. This design is similar to the front end processing used in low bandwidth sound compression systems such as CELP. No application references to speech recognition have been found in a preliminary search of the online PTO database and the ICASSP publication CD ROM literature. It is anticipated that SOS will apply for a provisional patent disclosure based on this novel design. The previous figure illustrates the Phase II baseline data flow and computational processes.

A number of problems have been identified while performing the Phase I filter experiments. The adaptive parameters may be over fitted to the data causing modulation of the output speech signal. Reconstruction phase errors often result in chirping artifacts in the output speech signal. Single microphone systems have problems removing impulse noises. Dual microphone systems need a good transfer function estimation between the two signal sources. Specialized processes will be needed to remove impulse noise effects and to compensate for moving noise sources including head microphone movements. The baseline Phase II design will allow experiments and improvements to correct these deficiencies.

The single microphone input allows the use of a single PC sound card so that no hardware modification is required to operate the filters. The processing for the dual microphone requires synchronization of the two signals to the sample level. Both hardware characterization and software correlation estimates will be used to accomplish this estimation. The estimation of the transfer function between the two microphones is required to compensate for gain differences. Errors in the transfer function estimate lead to a mismatch in the gain compensation for the two microphones.

The speech activity detection is common for speech recognition systems. The usual approach is to use time domain computations with a lag in the signal-input buffer. The computations include absolute signal energy estimation corrected for the mean noise level and signal zero crossings per second that estimates the fundamental speech frequency. The ratio of these two estimates is used with a noise sensitive threshold to detect speech or silence. Errors in speech activity detection will lead to substituting silence noise levels for sibilants and whisper phoneme signals.

The classification of unvoiced or voiced speech is a frequency domain computation usually performed by a first order linear prediction coefficient computation. This process is a common front end for low bandwidth speech compression such as CELP coders. Errors in voiced or unvoiced classification will only affect the transition period of these phonemes.

The voiced period filter processing will utilize either or both the linear adaptive filter bank and the LMS ALE time domain filter. The selection and tuning of these filters will be an experimental process affected by the noise environment, the speech recognition performance, and the computational loading. In general speech recognizer programs perform best on voiced phonemes, often above 90% accuracy, so only the minimum processing should be required to remove noise. The motivation for this is that voiced speech usually accounts for 30% or more of the articulation period, thus a considerable savings in computations may be possible with this approach.

The unvoiced period filter processing will utilize the magnitude FFT with interactive reconstruction. The algorithm is adjusted for the dual microphone input case or for the single microphone case. The removal of noise from short phoneme articulation signals such as plosives is a very difficult task. This is the most sensitive part of the processing and will require the most detailed computations.

The signal reconstruction will recombine the three sound periods of silence, voiced and unvoiced signals into a low noise speech signal. An alternative to this reconstruction is to use the sound period data directly to compute speech recognition feature data. This can be accomplished by a direct access to the recognition process as in the SOS SPSR tool kit or through specialized programming for the HTK or Sphinx II systems as an example. Most commercial speech recognition programs do not allow access to this level of training detail.

3.0 MICROPHONE DESIGN ANALYSIS

3.1 SIGNALS AND CONNECTIONS

There are many variables that must be considered when interfacing audio equipment to a computer sound card for either sound recording or speech recognition. It must be kept in mind that numerous sound card manufacturers have different input configurations. SOS had an experience with a PC based mobile data terminal manufacturer who wanted to add speech recognition input to their custom system. They had built a PC microphone-input interface. Unfortunately it was badly placed on the circuit board near a high frequency timer. The automatic timer generated an audio frequency square wave noise starting after the first five seconds of microphone input. Software speech recognition systems would accept the first few words and then fail. A major redesign of an in-production product was required. The lesson learned is that when the technical information supplied with the sound card is unclear, the manufacturer should be contacted

3.1.1 Signal Levels

Audio and recording microphones put out a very weak signal - less than 1/1000th of a volt, or 1 millivolt. Audio inputs on computer sound cards, even though they may be labeled "Mic In" or be identified by a small microphone-shaped icon, often are not designed to accept such a low signal level. Most sound card inputs require a minimum signal level of at least 1/100th of a volt (10 millivolts); some older 8-bit cards need 1/10th of a volt (100 millivolts). This discrepancy means that if a typical audio microphone is connected to a sound card input, the user will have to shout into the microphone or hold it just an inch or so away (or both) in order to produce a strong enough signal for the sound card to respond.

There are two possible solutions. One option is to increase the sensitivity of the sound card input, so that it can more easily detect the signal from the microphone. The software supplied with some sound cards allows the user to increase the sensitivity or "gain" of the input, either with a click-and-drag input level control or a set of check boxes that double, triple, or quadruple the sensitivity. Increasing the sensitivity of the input will always add some noise, so only as much gain as necessary should be added.

If the input sensitivity cannot be increased, it is possible to amplify the microphone signal before it goes into the sound card input. This can be done by running the microphone signal through a device called a mic preamplifier or mic-to-line amplifier. A microphone mixer can also be used if it has an output that will provide adequate signal level to the sound card input. In this case, the mixer is used only for its preamplification function and not its mixing capability. Either way, you have to know the typical output level of the microphone from the microphone's specification sheet and the sensitivity of the sound card input in order to know how much amplification is needed, and to determine whether a particular mic preamp or mixer will do the job.

3.1.2 Impedance Matching

Impedance is how much a device resists the flow of an AC signal, such as audio. Impedance is similar to resistance, which is how much a device resists the flow of a DC signal. Both impedance and resistance are measured in ohms. When referring to microphones, low impedance is less than 600 ohms, medium impedance is 600 ohms to 10,000 ohms, and high impedance is greater than 10,000 ohms.

In the early part of the 20th century, it was important to match impedance. Bell Laboratories found that to achieve maximum power transfer in long distance telephone circuits, the impedances of different devices should be matched. Impedance matching reduced the number of vacuum tube amplifiers needed. The tubes were expensive, bulky, and heat producing. In 1948, Bell Laboratories invented the transistor-a cheap, small, efficient amplifier. The transistor utilizes maximum voltage transfer more efficiently than maximum power transfer. For maximum voltage transfer, the destination device called the load should have impedance of at least ten times that of the sending device called the source, which is known as bridging. This is the most common circuit configuration when connecting audio devices, however with modern audio circuits matching impedance can actually degrade audio performance.

Audio mixers often have inputs labeled as low impedance. Actually, these inputs have impedances between 1000 ohms and 2000 ohms in order to properly bridge the low impedance microphone. A low impedance microphone may always be connected to input with higher impedance. However, the microphone may not always be able to provide enough signal strength to properly drive the mixer's audio input. Compare the microphone's output level or sensitivity to the required mixer input level. When a microphone is connected to a mixer input with lower impedance, there would be some loss of the microphone signal. As a rule of thumb, a loss of 6dB or less is acceptable.

Impedance for computer audio interfaces is important because the relationship between the impedance of a microphone and the impedance of the sound card to which it is connected can have a significant effect on how much of the microphone's signal is actually transferred to the sound card. For acceptable results, the output impedance of the microphone must be less than the input impedance of the sound card. If the impedance of the microphone is the same or higher than the input impedance of the sound card, some or all of the microphone's signal strength will be lost by an effect called loading. The higher the microphone's impedance compared to that of the sound card, the more signal will be lost. Connecting a high impedance microphone to a sound card with input impedance of 600 ohms will result in so much signal loss that the speaker's voice will be inaudible. Audio system microphones typically have output impedance of less than 600 ohms, and most sound cards have input impedance of 600 to 2,000 ohms, so impedance is not usually a problem.

3.1.3 Connector and Wiring Considerations

The most visible problem encountered when connecting an audio microphone to a sound card is that different connectors are used. Because of their limited width, computer sound cards can only accommodate very small connectors. The 3.5-mm (1/8") miniplug used on most Walkman-type personal stereos is the most popular type. The standard 1/4" and XLR connectors used on professional microphones are far too big to fit into a single card slot. Just as important as the type of connector used is the wiring scheme used. XLR connectors have three connection points (either pins or sockets). Professional microphones with XLR connectors use an industry-standard balanced wiring scheme, with two of the pins used to carry audio and the third as a ground connection. There is no standard for the wiring of the 3.5-mm miniplug connectors used on sound cards, so the actual wiring scheme varies depending on the manufacturer of the card.

The 3.5-mm miniplug is commonly available in two different configurations. Most sound cards use a three-segment version, often called a stereo connector since it is generally used to carry two separate channels of audio in addition to providing a ground connection. When used as a microphone connector, the end portion of the connector called the Tip usually carries the audio signal. The center portion of the connector, called the Ring, is sometimes used to carry low-voltage DC power required by the microphone supplied with the sound card. The third section called the Sleeve is used as the ground connection. On the two-segment or "mono" version, the Tip of the connector carries audio and the Sleeve is used for ground.

DC power cannot usually be supplied through a mono 3.5-mm miniplug. Some sound cards have an additional stereo input labeled line in. This is designed to accommodate the stereo signal from a VCR, CD player, or tape deck, and is not suitable for use as a microphone input.

3.2 Microphone Technologies

The type of power needed by the condenser microphone and the way that it is provided are important issues that may affect whether a particular professional microphone will work with a particular sound card, and how the cable connecting them together should be configured.

3.2.1 Dynamic Vs. Condenser Microphones

Different types of microphones use different methods of converting the acoustic energy created by a sound source, such as voice into electrical energy, which can be amplified, processed, recorded, or transmitted. The two most popular types of microphones are the dynamic and the condenser, sometimes called an electret. The primary difference for sound cards is that condenser microphones require a source of DC power to operate. Dynamic microphones do not require any external powering.

One type of power, called bias voltage, provides power for a small transistor inside the microphone element or head. The other type is called phantom power, and is used to operate a small preamplifier, which slightly amplifies the signal or provides frequency contouring. The preamplifier may be housed inside of the microphone handle or, in the case of small lavalier or gooseneck microphones, in an external tube or pack. The preamplifier used by professional condenser microphones is not the same as the microphone-to-line amplifier mentioned earlier, which also goes by the name preamplifier. Some audio system condenser microphones are designed to accommodate an internal battery, while others require phantom power from a microphone mixer or power supply. The microphones supplied with computer sound cards often operate on bias voltage supplied by the sound card through the Ring portion of the stereo miniplug connector. So far, sound cards cannot provide the phantom power used by many professional condenser microphones.

To connect a professional microphone with a three pin XLR output connector to the 3.5-mm miniplug mic input of a sound card, a special cable must be purchased or made. For the microphone to work properly, the cable must have the proper type of connector for the sound card with a two-conductor mono or three-conductor stereo miniplug and be wired correctly. The correct wiring scheme depends on the type of microphone and the wiring of the sound card microphone input.

3.2.2 Connecting Professional Dynamic Microphones

The wires that are connected to pins 1 and 3 of the XLR connector should both be connected to the Sleeve of the mono miniplug. The wire that is connected to pin 2 of the XLR should be connected to the Tip of the miniplug. If the soundcard uses a stereo miniplug, the configuration is slightly different. The wires that are connected to pins 1 and 3 of the XLR connector should both be connected to the Sleeve of the stereo miniplug. The wire that is connected to pin 2 of the XLR should be connected to the Tip of the miniplug. No connection should be made to the Ring of the miniplug, because dynamic microphones do not require external DC power. Sometimes it is impossible to tell if the connector on a sound card is of the mono or stereo variety. If a cable that is equipped with a mono connector is plugged into a sound card input that uses a stereo connector, the microphone should still work. This is because the Ring portion of the sound card jack will make contact with the Sleeve portion of the miniplug on the mic cable, which will connect any DC bias voltage to ground.

3.2.3 Connecting Audio Condenser Microphones

Connecting an audio condenser microphone to a sound card can be complicated, because there are so many variations between different brands of microphones in terms of bias voltage requirements. Phantom power is a defined audio industry standard and is usually the same regardless of the brand, but no sound cards are able to provide it. The following three alternatives are the possible situations. If the microphone can operate on an internal battery, no external source of power is needed and the mic can be connected to the sound card using the same wiring scheme as for a dynamic type. If the microphone is a handheld or gooseneck style with an internal preamplifier that requires phantom power because a battery cannot be accommodated, it cannot be connected directly to the sound card. These microphones must be connected to a dedicated phantom power supply or a microphone mixer that has this feature; the output of the power supply or mixer is then connected to the input of the sound card using the same method as for a dynamic mic. If the microphone is a lavalier (tie-clip), headworn, or other type with a separate tube-or box-style preamplifier that requires phantom power, it may be possible to bypass the preamplifier and connect the microphone directly to the sound card input. This is only an option if the sound card can provide the proper bias voltage that was being provided by the preamplifier.

3.2.4 Adapting Condenser Microphones to the Sound Card

Some condenser microphones can be operated on the bias voltage that is supplied by the sound card. Bias voltage is usually between 3 and 9 volts DC; some microphones can operate on a range of voltages, while others require a specific voltage. To operate a condenser microphone without its preamplifier directly from the bias voltage supplied by the sound card requires replacement or modification of the cable that connected the microphone to the preamp. It is critical to know both the requirements of the microphone and the wiring scheme and amount of bias voltage available from the sound card input. Specifically, you must know if the cable that connected the condenser microphone to the preamplifier is a one conductor shielded cable or a two conductor shielded cable. Keep in mind that signal level and electrical impedance are still important, and a condenser mic operating solely on bias voltage may have a higher impedance than one with the preamp connected. The output impedance of the mic should be less than, or equal to, the sound card input impedance. It is more common to find two conductor shielded cable, where one conductor is used to carry the audio signal and the other carries the DC power. The shield is used as the ground, and should be connected to the Sleeve of the miniplug. The bias conductor should be connected to the Ring, and the audio conductor to the Tip of the miniplug. If a condenser microphone uses only one conductor shielded cable, the conductor carries both the audio signal and the bias voltage at the same time. In this case you must add some circuitry to separate the audio signal from the bias voltage. It involves a resistor and a capacitor and will fit inside of most miniplug connectors that can be disassembled.

3.2.5 Microphone Issues

Because computer sound card inputs use the unbalanced wiring scheme, microphone cables longer than 15 feet will usually pick up electromagnetic interference or cause the sound to become muffled. To preserve sound quality, use the shortest mic cable possible. If pin 3 of the XLR connector is wired to the Tip of the miniplug instead of pin 2, the polarity of the signal will be inverted. The microphone will sound the same to the human ear, but voice recognition software will probably not recognize the sound waveform, resulting in a high error rate.

If the microphone or other audio source to be used is equipped with something other than a three-pin XLR connector, a little research must be done to find out which portion of the connector carries the audio and which is connected to ground. The audio signal should always be routed to the Tip of the miniplug connector on the sound card, and the ground should be connected to the Sleeve of this connector. No connection should be made to the Ring on stereo connectors. Cables for this application are available that terminate in a mono 1/4" phone plug on one end and a stereo 3.5mm phone plug on the soundcard end, with no connection to the Ring. A standard audio patch cable combined with an adapter can also suffice. Microphones equipped with 1/4" plugs usually have audio on the Tip and use the Sleeve as the ground. These microphones often have a high impedance (about 10,000 ohms), which means that only a fraction of their output signal will be transferred to a low impedance (600 to 2,000 ohms) sound card input.

3.2.6 Speech Recognition Microphones

For accurate speech recognition, the software must receive clear, intelligible sound from the microphone. For this to happen, the microphone must be placed in an area where it receives relatively noise-free sound from the talker. The following guidelines will help you to get the best performance from a microphone and speech recognition software.

Place the microphone close to the talker. As the background noise level increases, the ratio of signal to noise decreases and the performance of the voice recognition software degrades. The noisier the room is, the closer the microphone must be placed to the talker to provide sufficient signal-to-noise ratio for good voice recognition. In most situations, a talker-to-mic distance of less than one foot is optimum. In noisy environments, the mic should be within 6 inches of the talker's mouth for good results; a headworn, lavalier/tie-clip, or gooseneck-type microphone is usually the best choice.

Use a directional microphone. Unidirectional microphones, referred to as noise-canceling by some manufacturers, which are less sensitive to sounds coming from the rear and sides can help isolate your voice from ambient noise. Unidirectional microphones also help when the primary noise source is directly behind the microphone, such as the computer fan or hard drive. A unidirectional microphone aimed at the computer operator may still pick up noise from sources located behind the operator.

Use a windscreen or pop filter. Windscreens prevent air currents from the mouth from striking the microphone abruptly, which can cause a popping or thumping noise. These cannot be interpreted by the speech recognition software. Condenser microphones are usually more sensitive to popping than dynamic types.

3.2.7 Critical Distance and Microphone Placement

A microphone is the first component in any audio recording or speech recognition system. Its function is to convert acoustic sound waves into an equivalent electrical signal. This signal can then be recorded, transmitted, amplified, or modified. However, a microphone cannot effectively sort out desired sound of direct speech from undesired reverberation (reflected speech). Also, a microphone cannot improve the acoustic environment in which it is placed.

In every room, there is a distance measured from the talker where the direct speech and the reflected or reverberant speech are equal in intensity. In acoustics, this is known as the Critical Distance and is abbreviated Dc. If a microphone is placed at Dc or farther from a talker, the speech quality picked up will be very poor. This poor sound quality is often described as echoey, reverberant, or bottom of the barrel. The talker's words will also be hard to understand as the reflected speech overlaps and blurs the direct speech.

The estimation of Dc for a room can be computed experimentally with the following simple tools and procedure.

Tools required:

25 foot tape measure
Sound level meter (Radio Shack part #32-2050 or equivalent)
Portable "boom box" with FM radio

Dc estimation procedure:

Place the "boom box" in one end of the room in place of a talker. Tune the FM receiver between stations. This steady "white" noise will be used instead of a talker.
Extend the tape measure from the "boom box" to the far side of the room. Lock the tape measure in place. It is the reference for distances.
Set the sound level meter to "A" weighting, "slow" response, "90"dB range. Using the tape measure as a guide, place the sound level meter microphone one foot from the "boom box".
Increase the "boom box" volume until the sound level meter needle points to "0", which is 90dB of sound pressure level (SPL).
Move the sound level meter back to the 2 foot mark. The meter reading will drop 4 - 6 dB.
Reset the meter to the "80" dB range. Move the meter to the 4 foot mark. The meter reading should again drop 4 - 6 dB.
Continue to double the distance each time the meter is moved. When the distance is doubled, the meter should drop 4 - 6 dB if Dc has not been reached.
During one of these meter moves, the meter reading will not drop the predicted 4 - 6dB, but will remain relatively constant in level over several feet. Note the distance where the meter reading first remains steady.

This is Dc, the Critical Distance. In general, an omnidirectional microphone should be placed no farther from the talker than 30% of Dc, e.g. if Dc is 10 feet, an omnidirectional may be placed up to 3 feet from the talker. A unidirectional microphone (cardioid, supercardioid, or shotgun) should be positioned no farther than 50% of Dc, e.g. if Dc is 10 feet, a unidirectional may be placed up to 5 feet from the talker.

3.3 NOISE CANCELING MICROPHONES

Noise canceling microphones are advantageous in noisy environments since they pick up desired sounds that are close to the user while rejecting unwanted noise that is farther away. Earlier sections discussed the digital filter (software) portions of the noise canceling microphone. This section explains how a noise canceling microphone operates and details the mechanical construction of acoustic passive and electronic active noise canceling microphones. Frequency response, polar pattern and noise canceling performance measures are discussed for both types of noise canceling methods. Lastly, test results for both types of microphones in a speech recognition application are given and summary conclusions drawn.

3.3.1 Acoustic Noise Canceling Microphone Construction

A microphone is an acoustic to electronic transducer. Its internal diaphragm sympathetically moves from the compression and rarefaction of sound wave energy that reaches it. This movement of the diaphragm is converted to an electronic signal. A noise-canceling microphone measures the pressure difference in a sound wave between two points in space.

The construction of an "acoustic" noise canceling microphone has both sides of its diaphragm equally open to arriving sound waves. The two sides of the diaphragm are separated by the front to rear port distance "D." Because of this port separation, the magnitude of sound pressure is greater in the front than in the rear of the diaphragm and slightly delayed in time. These two effects create a net pressure difference (Pnet = Pfront - Prear) across the diaphragm that cause it to move. In this manner, an acoustic passive noise canceling microphone measures and responds to the net pressure difference in an arriving sound wave between two different points in space.

3.3.2. Electronic Noise Canceling Microphone Construction

The electronic active noise canceling microphone is similar in principal to the acoustic noise canceling microphone in that it measures the net pressure difference in a sound wave between two points in space. It does so by utilizing an array of two "pressure" microphones arrange in opposing directions with the spacing between the two front ports being a distance "D". A typical pressure microphone utilized in the array is constructed with the rear diaphragm port sealed to the acoustic wave front while the front is open. The result is the diaphragm movement represents the absolute magnitude of the compression and rarefaction of the incoming sound wave and not a pressure difference between two points. An array of two pressure microphones achieves noise canceling characteristics because the output signal of each microphone is electrically subtracted from the other by an operational amplifier. The operational amplifier output signal is Mic out Pmic1 - Pmic2. Just like the acoustic passive microphone, it represents the net pressure difference of the sound wave between the distance "D."

3.3.3 Characteristic Frequency Response

The characteristic frequency response caused by phase shift in a noise canceling microphone applies to both acoustic and electronic noise canceling microphones. The length of the front to rear port separation, distance "D," determines where the peak and dip in frequency response will occur. A larger port separation results in the characteristic peak and dip occurring at a lower frequency.

A second factor creating a net pressure difference across the diaphragm is the impact of the inverse square law. This law states that the intensity of sound emanating from a source is reduced by a factor equal to the square of the distance from the source. This means that if the sound pressure difference between the front and rear ports (Pnet) of a noise canceling microphone was measured near the sound source and again at a farther distance from the source, the near field measurement would be greater than the far field. In other words, the microphone's net pressure difference and therefore output signal, is greater in the near sound field than in the far field. The inverse square law effect is independent of frequency.

The net pressure that causes the diaphragm to move is a combination of both the phase shift and inverse square law effect. These two factors influence the frequency response of the microphone differently depending on the distance to the sound source. For distant sound, the influence of the net pressure difference from the inverse square law effect is weaker than the phase shift effect, thus the rising 20 dB per decade frequency response dominates the total frequency response. As the microphone is moved closer to the sound source, the influence of the net pressure difference from the inverse square law is greater than the phase shift, thus the total microphone frequency response is largely flat. The difference in near field to far field frequency response is a characteristic of all noise canceling microphones and applies equally to both acoustic and electronic noise canceling microphones.

The increase in frequency response, or sensitivity, in the near field compared to the far field is a measure of noise cancellation. Consequently the microphone is said to be noise canceling. The microphone is also referred to as a differential or gradient microphone since it measures the gradient difference in sound pressure between two points in space. The boost in low frequency response in the near field is also referred to as the proximity effect.

A sound wave has a maximum net pressure between two points when the axes of the points aim at the sound source. When the axis of the points turn perpendicular 90 or 270 degrees to the sound source, zero net pressure exists because both points see the same amplitude and phase of the wave front. Since a noise-canceling microphone measures pressure difference, the maximum microphone output signal occurs when the front to rear port axis points directly at the sound source. Likewise, the minimum microphone output occurs when the axis is turned 90 or 270 degrees away from the sound source. As a result, the noise-canceling microphone has a figure-of-eight or "bi-directional" polar pattern.

In summary, simple acoustic passive and electronic active noise canceling microphones are very similar in frequency response, polar pattern and noise cancellation performance because of the fundamental means of measuring the pressure difference between two points in a sound wave. The SOS development of a digital adaptive filter for noise cancellation involves far more complex acoustics, noise modeling, and signal processing than the existing technologies.

3.3.4 Speech Recognition vs Microphone Performance

The use of voice applications on personal computers is exploding. Audio applications like Internet telephony, computer telephony, videoconferencing and speech recognition are transforming the PC into the preferred communications appliance for millions of users. High quality, directional microphones are required to enable these voice applications and deliver the user benefits intended by the application developers. However, many applications are designed with the microphone as an afterthought. This often results in the selection of an incorrect microphone element and poor acoustic implementation into the product. Severe performance degradation can result when the microphone is not viewed as a critical performance element in PC speech recognition applications. By selecting the proper microphone element (unidirectional, omnidirectional, noise canceling, etc.) and implementing it correctly, developers can vastly improve the performance of their speech recogntion applications without incurring any significant additional expense.

3.4 Dual Microphone System for an Adaptive Digital Filter

The SOS adaptive digital filter for noise cancellation process is described in Section 2. The hardware to implement this process requires the following components:

- A head mounted acoustic speech input microphone
- A head mounted environment noise microphone
- Dual analog to digital converters operating up to 20KHz
- Digital signal processing to produce filtered speech data
- Computer storage and processing to analyze system performance

The Phase I desktop computing system developed for the proof of concept and design of the adaptive noise canceling filter is shown below:

Advanced Audio Interface for Phonetic Speech Recognition in a High Noise Environment

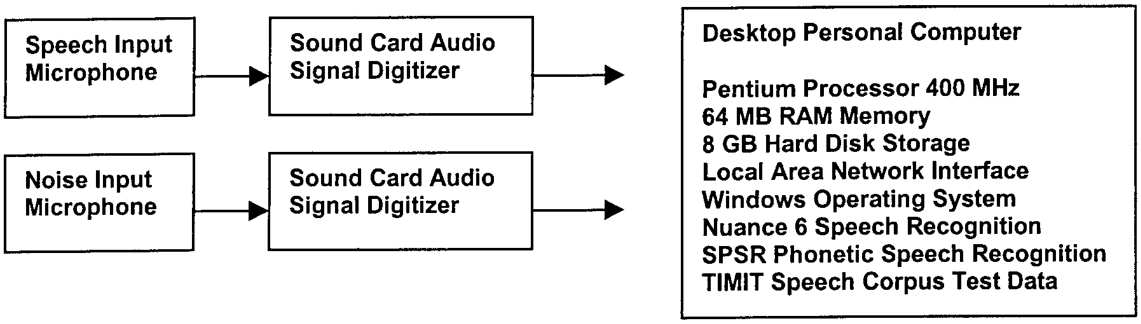

Speech Input Microphone
Sound Card Audio Signal Digitizer
Noise Input Microphone
Sound Card Audio Signal Digitizer
Desktop Personal Computer
Pentium Processor 400 MHz
64 MB RAM Memory
8 GB Hard Disk Storage
Local Area Network Interface
Windows Operating System
Nuance 6 Speech Recognition
SPSR Phonetic Speech Recognition
TIMIT Speech Corpus Test Data

Figure 3.4 Adaptive Noise Canceling Filter Development System

SOS has developed software for this system to perform the following tasks:

- Control two sound cards to input simultaneous audio data for analysis
- Store, display, playback, and analyze sound files in non real time
- Adaptive noise filter of two sound files to create speech input for Nuance
- Nuance 6 speech recognition application to test filter performance
- SPSR Tool Kit phonetic speech recognition to test filter performance

The primary goals of this computing configuration are to design and develop an adaptive noise-canceling filter with a proof of concept demonstration of this technology to justify Phase II prototype development.

4.0 SPEECH RECOGNITION SYSTEMS

Spoken communication is not a simple process; it has a wide variety of tones, accents, languages, and speaker variations. In the past, most speech recognition approaches have attempted to exploit numerous special circumstances to achieve reliable and rapid performance. These circumstances, whether limited vocabularies, discrete word pronunciation, various quantizations, or precomputed finite state models, all fail in a real world speech-processing environment. A current approach is a fundamental analysis of the basic acoustic unit of all spoken languages, the phoneme. The reliable detection, classification, and identification of a spoken phoneme is key to high accuracy speech recognition.

All speech recognition systems contain a front-end processor. This preprocessor extracts the important parameters from the speech signal and passes them to the recognition process to match up words and phrases. Ideally, preprocessor outputs should be invariant both to noise and to changes in the acoustic environment. Since humans are the best noise preprocessors and the best speech recognizers, the modeling of the human auditory system provides the clue to improved noisy environment performance over current statistical speech recognition systems.

The systems being tested are commercial in the sense they can be purchased, but only dictation systems come off the shelf, ready for a novice to employ. In order to utilize a speech recognition engine the following steps are necessary:

1. Select a speech recognition engine. Four different speech recognizers were chosen for this research. Nuance and SPSR are the topic of other ongoing research at SOS, and were therefor of interest in these tests. Whisper, the speech recognizer offered by Microsoft was selected, because all things Microsoft have the likelihood of becoming the industry standard. The IBM package was selected for similar reasons. All four of the engines chosen are SAPI compliant.

2. Implement Microsoft SAPI to interface between the speech engine and Visual Basic or C++ code. SAPI provides the communication standard between the speech engine, which is the computational core program, and the graphical user interface (GUI) which speaks to and for the user. It has become the industry standard programming interface that allows various programming languages to interface to a wide variety of speech recognition engines, and will eventually allow interoperability between the speech engines. For this implementation of SAPI, a Visual Basic GUI using ActiveX controls, formats, and scores the experiments.

3. Create grammar rules and vocabulary words for the words recognized. At a given point in the statistical recognition process, in order to reduce computation and increase accuracy, the recognizer restricts acceptable inputs from the user depending upon rules of grammar. The examples below pertain to grammar rules for air traffic controllers:
   - ♦ A defined grammar. Air traffic controllers have a defined list of words they use to convey their landing and take off commands.
   - ♦ Word order. An air traffic controller will say “ Cleared for take off," rather than “Take off cleared for.”
   - ♦ Context free. Recognition of each word is independent of the surrounding words.
   - ♦ Finite state. A sentence cannot run on forever. It will have an end point.

These tasks, which must be completed before testing can be initiated, are demonstrated in greater detail in the first speech engine section pertaining to Nuance. The individual results of each speech engine are provided in the following sections and the overall comparisons are summarized and analyzed at the end of this section.

4.1 NOISE AND SPEECH MODELS

The TIMIT corpus of read speech has been designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. TIMIT has resulted from the joint efforts of several sites under sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). Text corpus design was a joint effort among the Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI). The speech was recorded at TI, transcribed at MIT, and has been maintained, verified, and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST). This file contains a brief description of the TIMIT Speech Corpus. Additional information including the referenced material and some relevant reprints of articles may be found in the printed documentation which is also available from NTIS (NTIS# PB91-100354).

Figure 4.1-1 Dialect Distribution of Speakers

Dialect Region(dr)	#Male	#Female	Total
1	31 (63%)	18 (27%)	49 (8%)
2	71 (70%)	31 (30%)	102 (16%)
3	79 (67%)	23 (23%)	102 (16%)
4	69 (69%)	31 (31%)	100 (16%)
5	62 (63%)	36 (37%)	98 (16%)
6	30 (65%)	16 (35%)	46 (7%)
7	74 (74%)	26 (26%)	100 (16%)
8	22 (67%)	11 (33%)	33 (5%)
All	438 (70%)	192 (30%)	630 (100%)

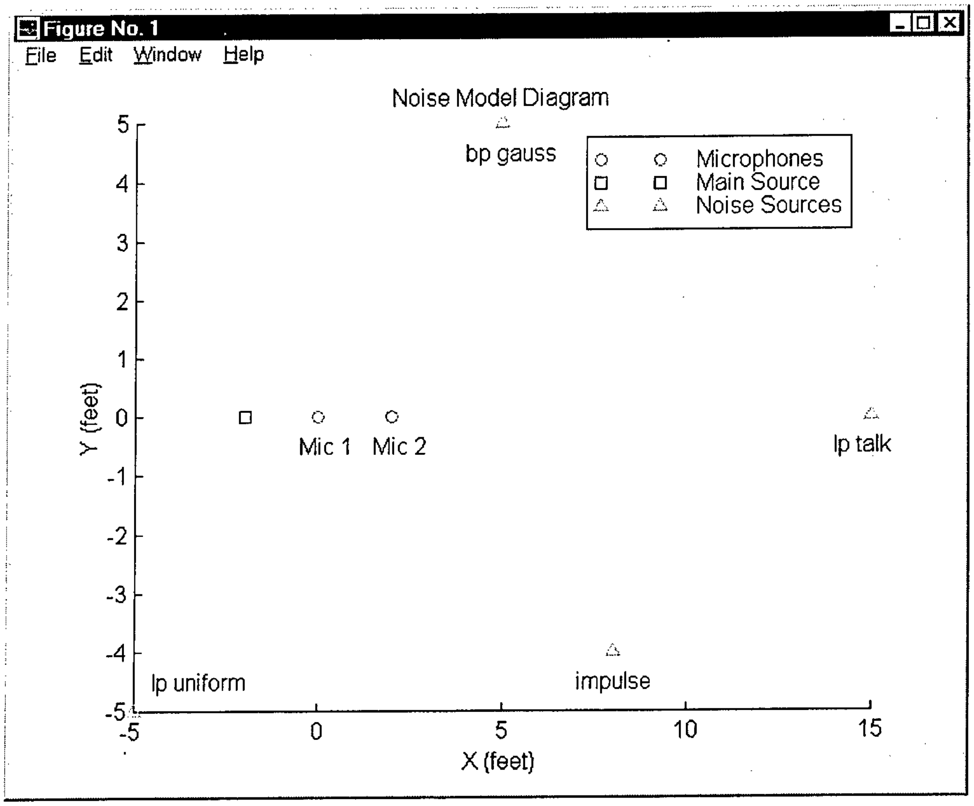
The dialect regions are:

dr1: New England
dr2: Northern
dr3: North Midland
dr4: South Midland
dr5: Southern
dr6: New York City
dr7: Western
dr8: Army Brat (moved around)

TIMIT contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. Figure 4.1-1 shows the number of speakers for the 8 dialect regions, broken down by sex. The percentages are given in parentheses. A speaker's dialect region is the geographical area of the U.S. where they lived during their childhood years. The geographical areas correspond with recognized dialect regions in U.S. (Language Files, Ohio State University Linguistics Dept., 1982), with the exception of the Western region (dr7) for which dialect boundaries are not known with any confidence, and dialect region 8 where the speakers moved around a lot during their childhood.

The SOS Noise Generation Program prototype was created using MATLAB 5.1 modeling a planar world with microphones and sound sources, Figure 4.1-2. There are four different source types and each source may be filtered. Each source can either be a wav file, a gaussian function, uniform function, or an impulsive function. The noise sources are given as (x,y) locations with in the planar space. The microphones were modeled using a frequency response filter to model the gain of the mic. The output files are the calculated sound that the mic "hears". For simplification all objects are treated as omni-directional point sources, and there are no "walls" to create sound reflections or reverberations.

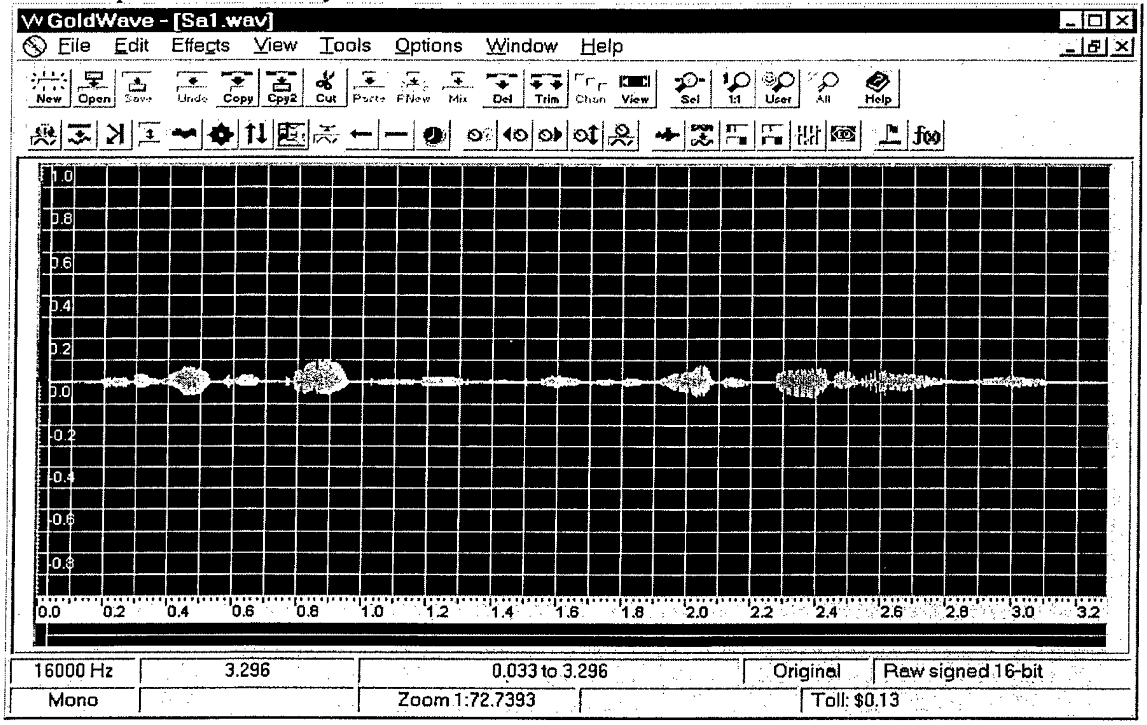
Figure 4.1-2 MATLAB Noise Model Diagram



There are two TIMIT sentences that are spoken by all speakers in the 8 different dialect regions. Using both male and female speakers will gives 16 sample sentences. Each of these 16 sentences are processed by the four recognition programs. The steps to test the effectiveness of the Adaptive Noise Filter are as follows:

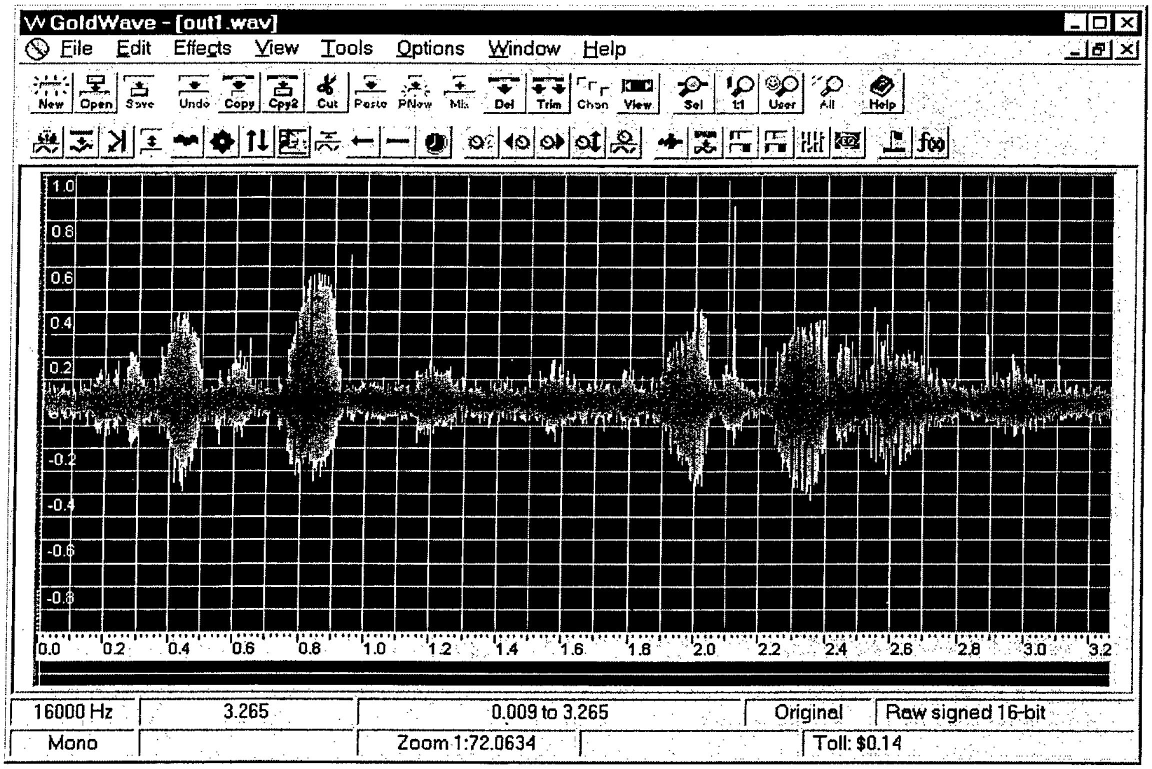
- Run 16 selected sentences through the four recognizers.
- Add noise using Noise Model Generator until recognition rate drops to an average of 15%
- Run noisy file through adaptive noise filter
- Run filtered files through the same speech recognizers and evaluate the effects of filters

Figure D.4.1-3 shows the wave form of a TIMIT sentence viewed using audio file viewer called GoldWave. The following Figure 3.7 shows a TIMIT wav file after noise has been added to it. Notice the few spikes and the noisy waveforms when loaded in GoldWave:


GoldWave - [Sa1.wav]
File Edit Effects View Tools Options Window Help
New Open Save Undo Copy Cpy2 Cut Paste PNew Mix Del Trim Chan View Sel 1:1 User All Help
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2

16000 Hz | 3.296 | 0.033 to 3.296 | Original | Raw signed 16-bit

Mono | | Zoom 1:72.7393 | | Toll: $0.13


GoldWave - [out1.wav]
File Edit Effects View Tools Options Window Help
New Open Save Undo Copy Cpy2 Cut Paste PNew Mix Del Trim Chan View Sel 1:1 User All Help
1.0
0.8
0.6
0.4
0.2
-0.2
-0.4
-0.6
-0.8
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2

16000 Hz | 3.265 | 0.009 to 3.265 | Original | Raw signed 16-bit

Mono | | Zoom 1:72.0634 | | Toll: $0.14

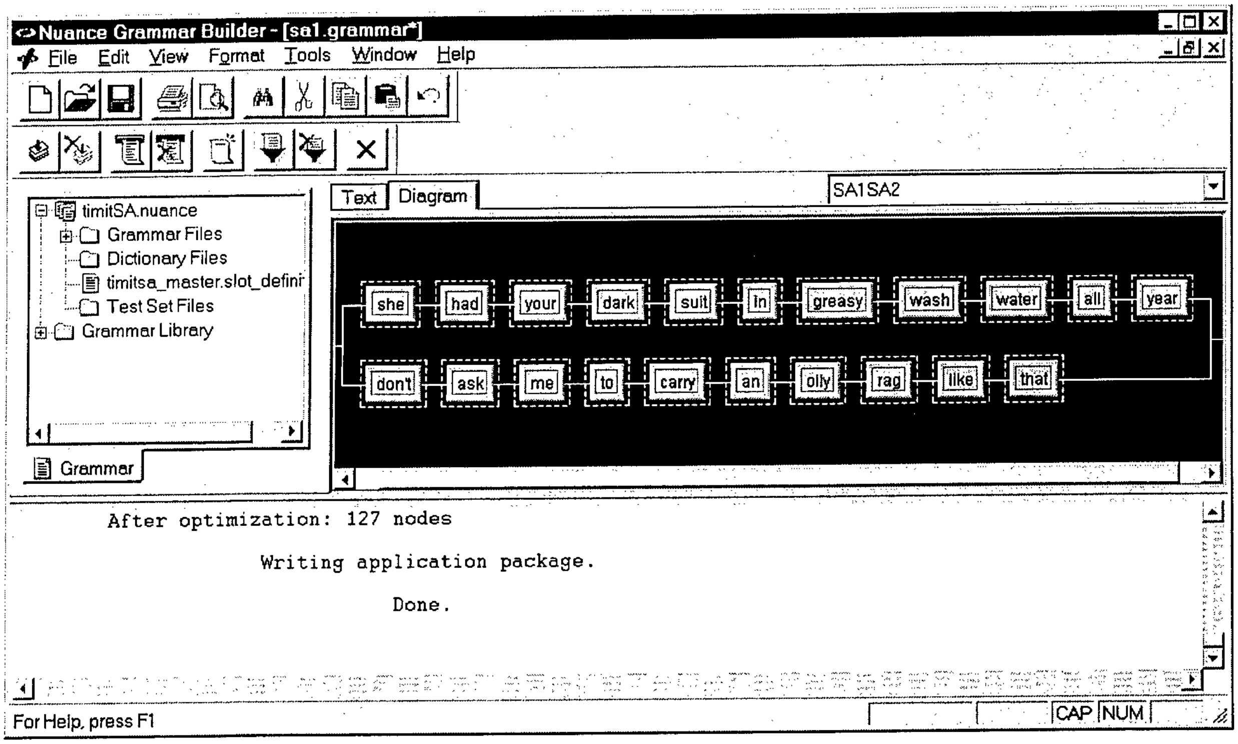
4.2 SRI NUANCE

Nuance Communications was founded in 1994 as commercial spin-off of SRI International. They lead the market in the development of speech recognition, language understanding and speaker verification software to automate access to information and services over-the-phone. Nuance focuses on customer service applications in call centers, particularly within the financial services and travel industries. Their products enable a user to speak to a computer over the telephone in everyday conversation in a variety of languages including U.S. English, U.K. English, Australian English, German, Japanese and Latin American Spanish.

Nuance 6 is recognized in the industry for its highly accurate speech recognition, available across a range of accents, languages, devices, and platforms. It employs a distributed client/server architecture to achieve scalability in large call centers for even complex systems. The Nuance Developers' Toolkit is a powerful and flexible set of tools for creating, prototyping, testing and monitoring Nuance 6 and Nuance Verifier applications. The toolkit enables any developer to create all the components of a speech application, without any prior speech recognition experience.

Part of the Nuance Developers Toolkit Version 6.2 is the ActiveX Speech Channel (NASC), a convenient and simple way to add speech recognition and control to Visual Basic programs. NASC was designed to run on Windows NT 4.0 software platform with Service Pack 3 installed, and was developed with Visual Studio 97 with Service Pack 3 installed and ATL Control.

Figure 4.2-1 Nuance Grammar Builder



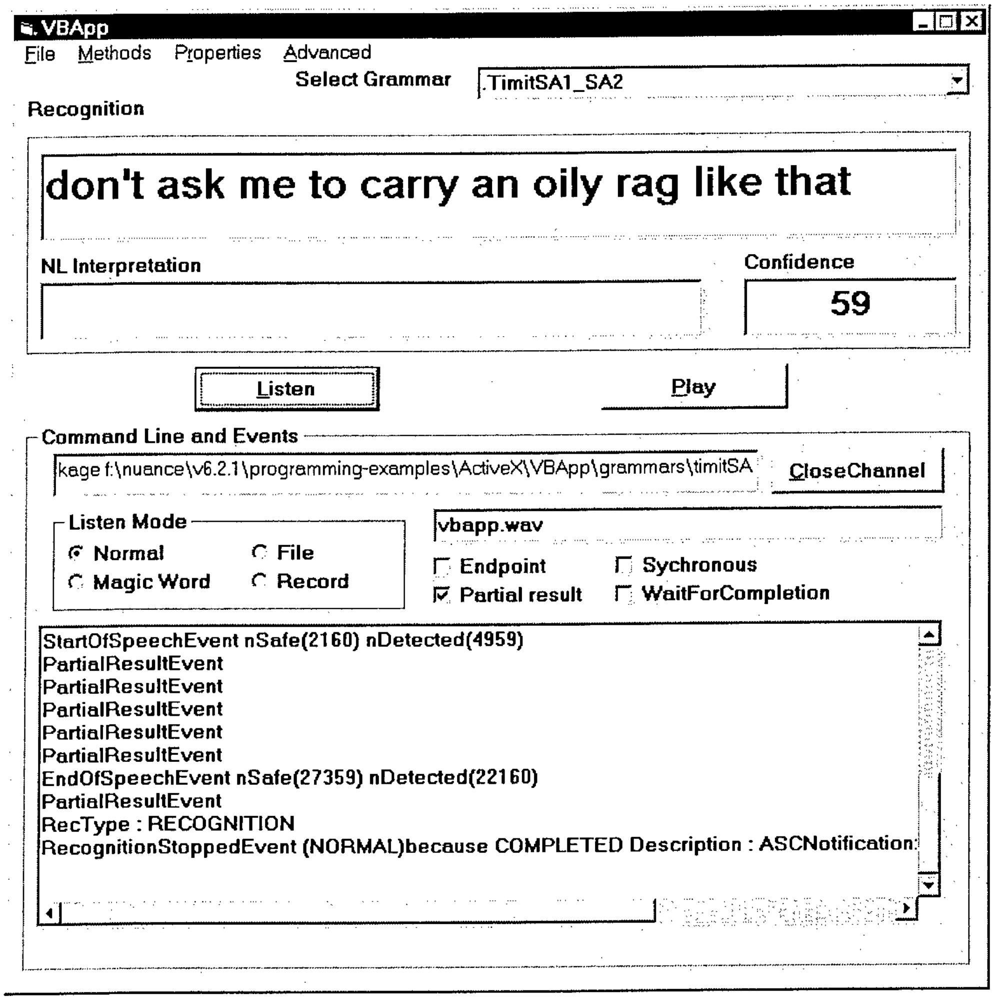
Nuance Grammar Builder - [sa1.grammar*]
File Edit View Format Tools Window Help
timitSA.nuance
Grammar Files
Dictionary Files
timitsa_master.slot_defini
Test Set Files
Grammar Library
Grammar
Text
Diagram
SA1SA2
she had your dark suit in greasy wash water all year
don't ask me to carry an oily rag like that
For Help, press F1
CAP NUM

```
        After optimization: 127 nodes

                Writing application package.

                       Done.
```

The Nuance Java based Grammar Builder lets you create, compile, and test Nuance grammar packages. The Grammar Builder provides a single environment for grammar development, including a rich graphical user interface. Figure 4.2-1 shows a simple grammar built with the Nuance's Grammar Builder. The sentences are the first two sentences from the TIMIT Speech Corpus. This grammar will recognize either of the two sentences even if a few of the words are incomprehensible.

Included in the Nuance Developers Speech Recognition Developers Toolkit is a Visual Basic ActiveX sample speech recognition program that uses the Nuance ActiveX Speech Channel (NASC). The Microsoft SAPI interface can be used to build a similar speech recognition program to recognize a set of sphere format wave files. The pertinent information this program will provide are the words recognized, the percentage recognized correct, and a Nuance statistic for confidence rating.

Figure 4.2-2 Nuance Sample VB Application



Select Grammar .TimitSA1_SA2

Recognition

don't ask me to carry an oily rag like that

NL Interpretation

Confidence 59

Command Line and Events

kage f:\nuance\v6.2.1\programming-examples\ActiveX\VBApp\grammars\timitSA

Listen Mode

● Normal ○ File

○ Magic Word ○ Record

vbapp.wav

□ Endpoint □ Sychronous

☑ Partial result □ WaitForCompletion

```
StartOfSpeechEvent nSafe(2160) nDetected(4959)
PartialResultEvent
PartialResultEvent
PartialResultEvent
PartialResultEvent
PartialResultEvent
EndOfSpeechEvent nSafe(27359) nDetected(22160)
PartialResultEvent
RecType : RECOGNITION
RecognitionStoppedEvent (NORMAL)because COMPLETED Description : ASCNotification
```

Recognition Using NUANCE

	(Words Correct out of 176)	Percent Correct
SA1	167	94.9%
SA1s	107	60.8%
SA1f1	141	80.1%
SA1f2	101	57.4%
SA1f3	96	54.5%
SA1f4	59	33.5%
SA1f5 (filt5a)	110	62.5%
SA1f6 (filt5b – single mic)	120	68.2%
	(Words Correct out of 160)	
SA2	139	86.9%
SA2s	82	51.3%
SA2f1	110	68.8%
SA2f2	68	42.5%
SA2f3	75	46.9%
SA2f4	50	31.3%
SA2f5 (filt5a)	78	48.8%
SA2f6 (filt5b – single mic)	95	59.4%

Recognition Using Nuance – Noise increased 5dB

	(Words Correct of 176)	Percent Correct
SA1	167	94.9%
SA1s	52	29.5%
SA1f1	124	70.5%
SA1f2	90	51.1%
SA1f3	98	55.7%
SA1f4	8	4.5%
SA1f5 (filt5a)	81	46.0%
SA1f6 (filt5b – single mic)	98	55.7%
SA2	139	86.9%
SA2s	44	27.5%
SA2f1	85	53.1%
SA2f2	60	37.5%
SA2f3	71	44.4%
SA2f4	9	5.6%
SA2f5 (filt5a)	63	39.4%
SA2f6 (filt5b – single mic)	76	47.5%

4.3 Microsoft Whisper

Whisper is the speech recognition engine provided by Microsoft. Whisper speech recognition fundamentally functions as a pipeline that converts PCM (Pulse Code Modulation) digital audio from a sound card into recognized speech. To enhance pattern recognition, the PCM digital audio is transformed into the frequency domain, using a windowed fast-Fourier transform. The fast Fourier transform analyzes every 1/100th of a second, and converts them into a graph of the amplitudes of frequency components, describing the sound heard for that 1/100th of a second. The speech recognizer has a database of several thousand such graphs called a codebook. The sound is identified by matching it to the closest entry in the codebook and producing a number that describes the sound. This number is called the feature number.

- The input to the speech recognizer begins as a stream of 16,000 PCM values per second. By using fast Fourier transforms and the codebook, it is boiled down into essential information, producing 100 feature numbers per second

In an ideal world, each feature number could be matched to a phoneme. If a segment of audio resulted in feature #52, it could always mean that the user made an "h" sound. If this were true, it would be easy to figure out what phonemes the user spoke. Unfortunately, this is not the case for a number of reasons. Every time a user speaks a word it sounds different, and they do not produce exactly the same sound for the same phoneme. And the sound of a phoneme can change depending on what phonemes surround it. The "t" in "talk" sounds different than the "t" in "attack" and "mist". The sound produced as a phoneme changes from the beginning to the end of the phoneme and is not constant. The beginning of a "t" will produce different feature numbers than the end of a "t".

For the speech recognizer to learn how a phoneme sounds, a training tool is passed hundreds of recordings of the phoneme. It analyzes each 1/100 th of a second of these hundreds of recordings and produces a feature number. From these it learns statistics about how likely it is for a particular feature number to appear in a specific phoneme. Hence, for the phoneme "h", there might be a 55% chance of feature #52 appearing in any 1/100 th of a second, 30% chance of feature #189 showing up, and 15% chance of feature #53. Every 1/100 th of a second of an "f" sound might have a 10% chance of feature #52, 10% chance of feature #189, and 80% chance of feature #53. The probability analysis done during training is used during recognition. The 6 feature numbers that are heard during recognition might be:

52, 52, 189, 53, 52, 52

The recognizer computes the probability of the sound being an "h" and the probability of it being any other phoneme, such as "f". The probability of "h" is:

80% * 80% * 30% * 15% * 80% * 80% = 1.84%

The probability of the sound being an "f" is:

10% * 10% * 10% * 80% * 10% * 10 % = 0.0008%

You can see that given the current data, "h" is a more likely candidate.

The sound of a phoneme will change depending upon what phoneme comes before and after. You can hear this with words such as "he" and "how". You don't speak a "h" followed by an "ee" or "ow", but the vowels intrude into the "h", so the "h" in "he" has a bit of "ee" in it, and the "h" in "how" as a bit of "ow" in it. Speech recognition engines solve the problem by creating tri-phones, which are phonemes in the context of surrounding phonemes. Thus, there exists a tri-phone for "silence-h-ee" and one for "silence-how". Since there are roughly 50 phonemes in English, you can calculate that there are 50*50*50 = 125,000 tri-phones. Because there are so many, similar sounding tri-phones are grouped together.

When the speech recognizer starts to listen it has one hypothesized state. It assumes the user is not speaking and that the recognizer is hearing the "silence" phoneme. Every 1/100th of a second it hypothesizes that the user has started speaking and adds a new state per phoneme, creating 50 new states, each with a score associated with it. After the first 1/100 th of a second the recognizer has 51 hypothesized states.

In 1/100 th of a second, another feature number comes in. The scores of the existing states are recalculated with the new feature. Then, each phoneme has a chance of transitioning to yet another phoneme, so 51 * 50 = 2550 new states are created. The score of each state is the score of the first 1/100 th of a second times the score if the 2 nd 1/100 th of a second. After 2/100 ths of a second the recognizer has 2601 hypothesized states.

This same process is repeated every 1/100th of a second. The score of each new hypothesis is the score of the parent hypothesis times the score derived from the new 1/100th of a second. In the end, the hypothesis with the best score is what's used as the recognition result. Of course, a few optimizations are introduced. If the score of a hypothesis is much lower than the highest score then the hypothesis is dropped. This is called pruning. The optimization is intuitively obvious. If the recognizer is millions of times more confident that it heard "h eh l oe" than "z z z z," then there's not much point in continuing the hypothesis that the recognizer heard, "z z z z". However, if too much is pruned then errors can be introduced since the recognizer might make a mistake about which phoneme was spoken.

4.4 IBM VoiceType

The IBM Voice Type Developers Tool Kit for Windows provides programmers with the necessary tools to develop applications that incorporate speech recognition. It includes a robust set of application programming interfaces (API) to access speech resources. It contains utility programs that enable developers to define and manage what a user can say within an application.

In this application, speech recognition is the process of translating what you say to the computer into text or commands by identifying and interpreting individual components of human speech. Voice Type provides this capability. Our units of speech are words. On paper words are made up of letters. When spoken, they are made up of sounds. When you go from the spoken word to the written word, you must make the conversion from sound to letters. There are other factors, which can make the job even more difficult. Background noise can make understanding harder.

The Voice Type developers tool kit supports both dictation and command and control interface applications. It is a speaker independent system so that most users can use it without any training, however enrollment is possible and will improve accuracy. A speech aware application is designed to respond to voice input. Some or all of the input comes from spoken words and are acted upon as commands, translated into text, or represent data.

The heart of the system is the speech recognition engine. It is a program that recognizes speech input and translates it into text for computer processing. Speech aware applications access the speech engine through the speech manager API (SMAPI). A vocabulary is a list of words that the speech engine uses to match the speech input. A word usage model provides statistical information on word sequences. An application specifies a set of active words through a vocabulary. Grammar vocabularies are words contained in a grammar created for the application that specifies word sequences to recognize.

Pronunciations are the possible phonetic representations of words. Words can have multipe pronunciations and identical pronunciations can represent multiple words. Voice Type includes a dictionary builder to add word pronunciations. It also includes a grammar compiler to specify vocabularies and word sequences.

The Voice Type speech engine handles the complex task of taking raw audio input and translating it to text. It accepts speech plus noise into the acoustic processor that has a signal processor and a labeler. The signal processor produces a set of features at one hundredth of a second intervals. The labeler converts the features into a stream of labels that identify sound categories. Word matching is performed after acoustic processing. First a fast word match performs an approximate match against all words in the vocabulary. Then a language model analyzes the probabilities of sequences of words. This is followed by a detailed acoustic match that performs a more accurate match on the smaller set of words. Last a decoder that selects the most likely sequence of words given the acoustic and language scores.

The IBM speech engine does not allow access to the training features for use of auditory model data. It does work from both stored sound files and real time speech input to perform repeatable performance tests with multiple adaptive filter outputs.

4.5 SOS SPSR Tool Kit

The SOS approach to phonetic speech recognition is based on linguistic feature processing to detect, classify, and identify phoneme signatures. The details of the phonetic speech processing cycle implemented in the SPSR Tool Kit. Phonemes are the common sound units produced by all speakers in all languages. They are created dynamically by a complex vocal tract filter applied to acoustic energy generated by pulses of air and radiated by the lips, nose, and cheeks.

The SOS approach analyzes the speech sound to detect a set of features that characterize the underlying phonemes for stable acoustic segments. A number of parallel classification algorithms use the features to estimate the actual phoneme for each segment. These independent classification methods include Bayesian statistics, metric templates, neural networks, Markov models, and fuzzy logic functions. The resulting estimates are combined to determine the most likely phonemes in an utterance using a combination of dynamic programming and statistical data processing. Linguistic and lexical methods are used to convert the resulting phonemes to text for output from the speech recognition system. This unique algorithm performs speech recognition for most human languages, is speaker independent and naturally continuous.

5.0 AUDITORY MODELS FOR NOISY SPEECH RECOGNITION

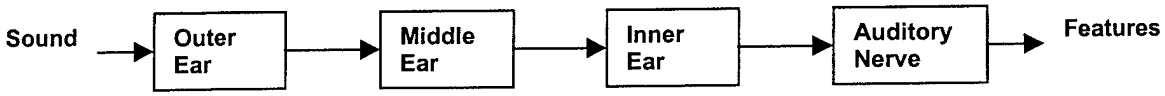
SOS is investigating three separate auditory models for application to noisy speech recognition. Each model has a different basis, but they are all similar in using physiological models of hearing for speech recognition features. This is in contrast to the statistical methods that use signal processing to transform speech into features that can be used to train and test a pattern based speech recognition system.

The first method is an auditory physical simulation, APS, developed by SOS. This model uses continuous differential equations to represent the coupled components in the hearing process. The second method is a published ensemble interval histogram, EIH, model of the cochlea and hair cell transduction to create numerical features. The third model is the auditory image model, AIM, developed by Roy Patterson and others at Cambridge University. In each case SOS has developed or acquired a computer program that will be used to analyze numerical feature data for use in phonetic speech recognition.

5.1 AUDITORY PHYSIOLOGY SIMULATION - APS

SOS has chosen to use an engineering approach to the development of an auditory physiology simulation (APS) of the hearing process for speech recognition. This approach comes from years of signal processing in industry and is based on the practical experience of modeling a number of diverse real world systems. Engineers use these modeling methods in highly competitive fields to obtain hard facts concerning the design and operation of real physical systems. A good deal of engineering consists of starting from a real physical system, such as hearing, and creating an abstract performance model. This is different from the research science methodology of starting with an abstract model and testing its fit to the real world situation. An advantage of the scientific research method is that the abstract model yields a clean single-ended problem having only one solution - the correct one. The real world often has many solutions.

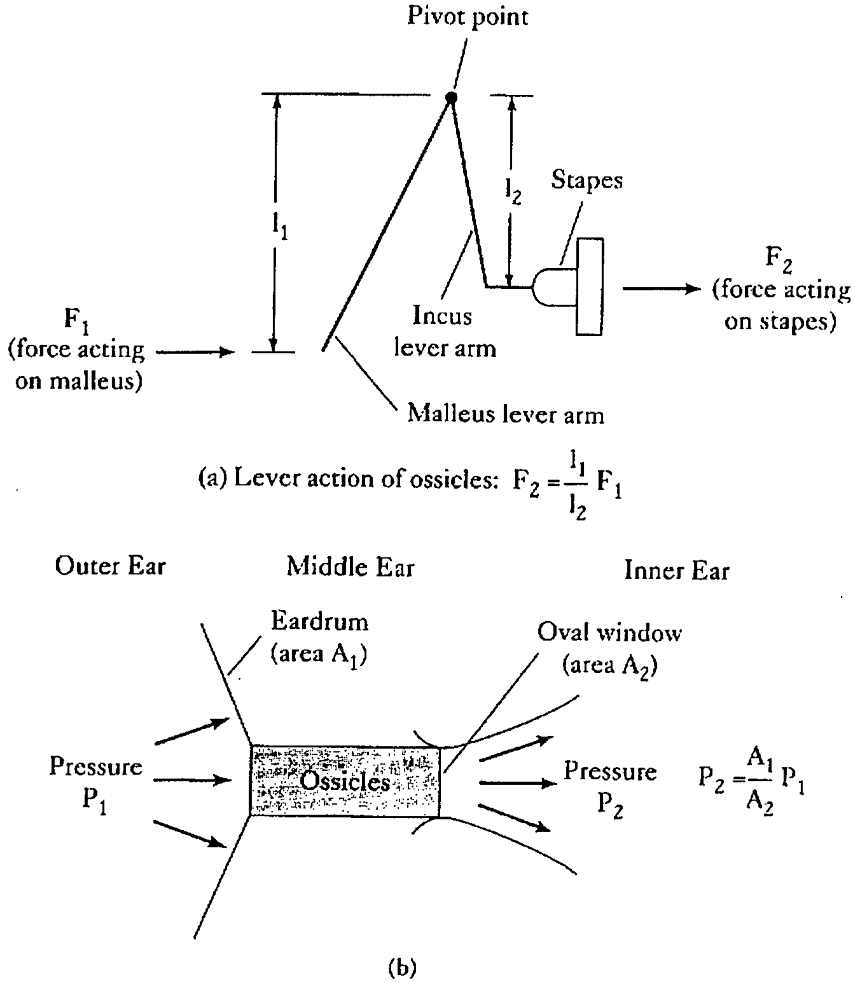
Figure 5.1-1 Physical Simulation Components of APS Model



Real engineering systems are dirty systems, cluttered with messy problems, and noisy data. In a real physical system, the ideal spring has mass, the mass is flexible, the damping is nonlinear, etc. This is especially true in the new field of biomedical engineering models of human systems such as hearing. An engineering model converts a physical system into an abstract system by making decisions on modeling each component. This requires scientific knowledge, real world data, and intuition. Such a model may be within a small or large percent error of instrumented real world data.

The physiological data presented in the first section was used to build the following APS model. The first decision was the number of coupled components to use in the simulation model. The decision to start with four physiological components is shown in Figure 5.1-1. This is based on using the mechanical coupling model for the outer ear to inner ear model shown in Figure 5.1-2.

Figure 5.1-2 Example Physical Analogues to Model Using

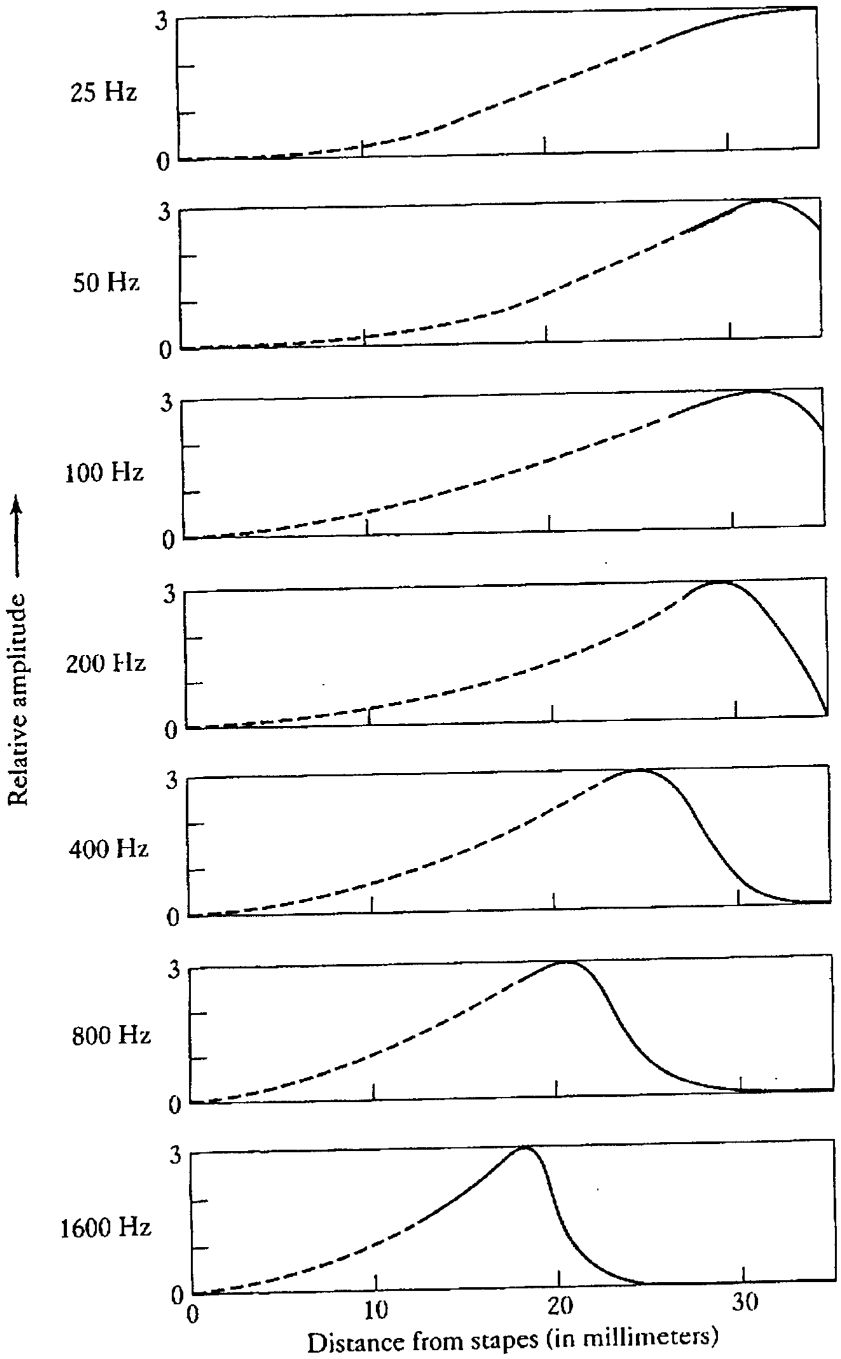


(a) Lever action of ossicles: $F_2 = \frac{l_1}{l_2} F_1$

(b)

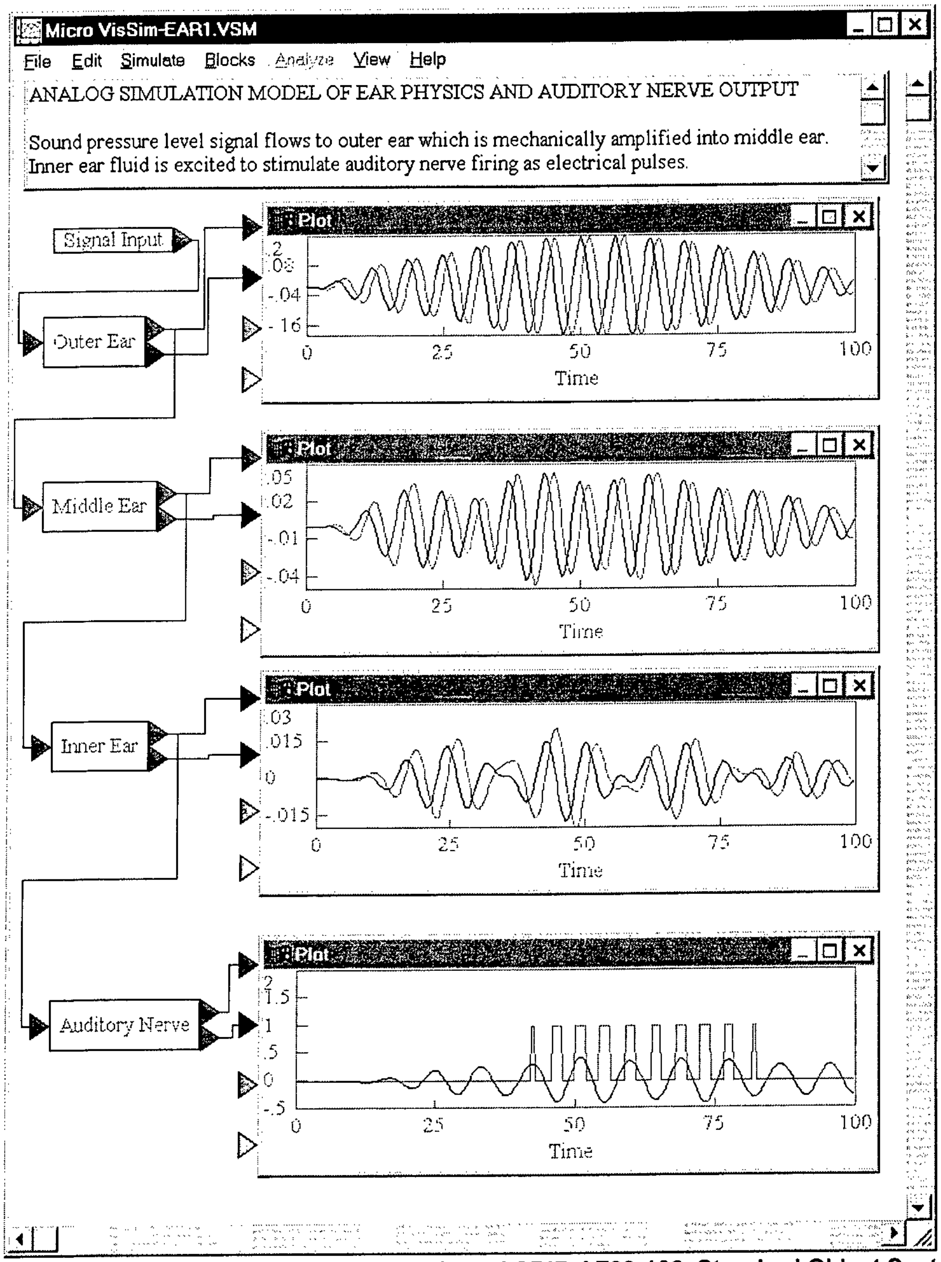
The ultimate goal of an engineering model is to design a product. In this case the product is a computer program for extracting numerical features from speech signals that represent the auditory nerve data used by the human brain to recognize speech. The physical response of the basilar membrane as a function of frequency shown in Figure 5.1-3 provides the coupling of the inner ear to the auditory nerve response.

Figure 5.1-3 Example Basilar Membrane Displacement for Stapes Excitation to Calibrate



Continuous system simulation is an important tool of engineering that is used in product design, analysis, and testing. The simulation of the components of the hearing process using differential equations is the basis of the APS model. SOS is starting with a simple coupled system as shown in Figure 5.1-4. As a first approximation, each of the four components is represented by a second order harmonic system with a forcing function. These components were modeled with the VISSIM program, and each component has a separate model.

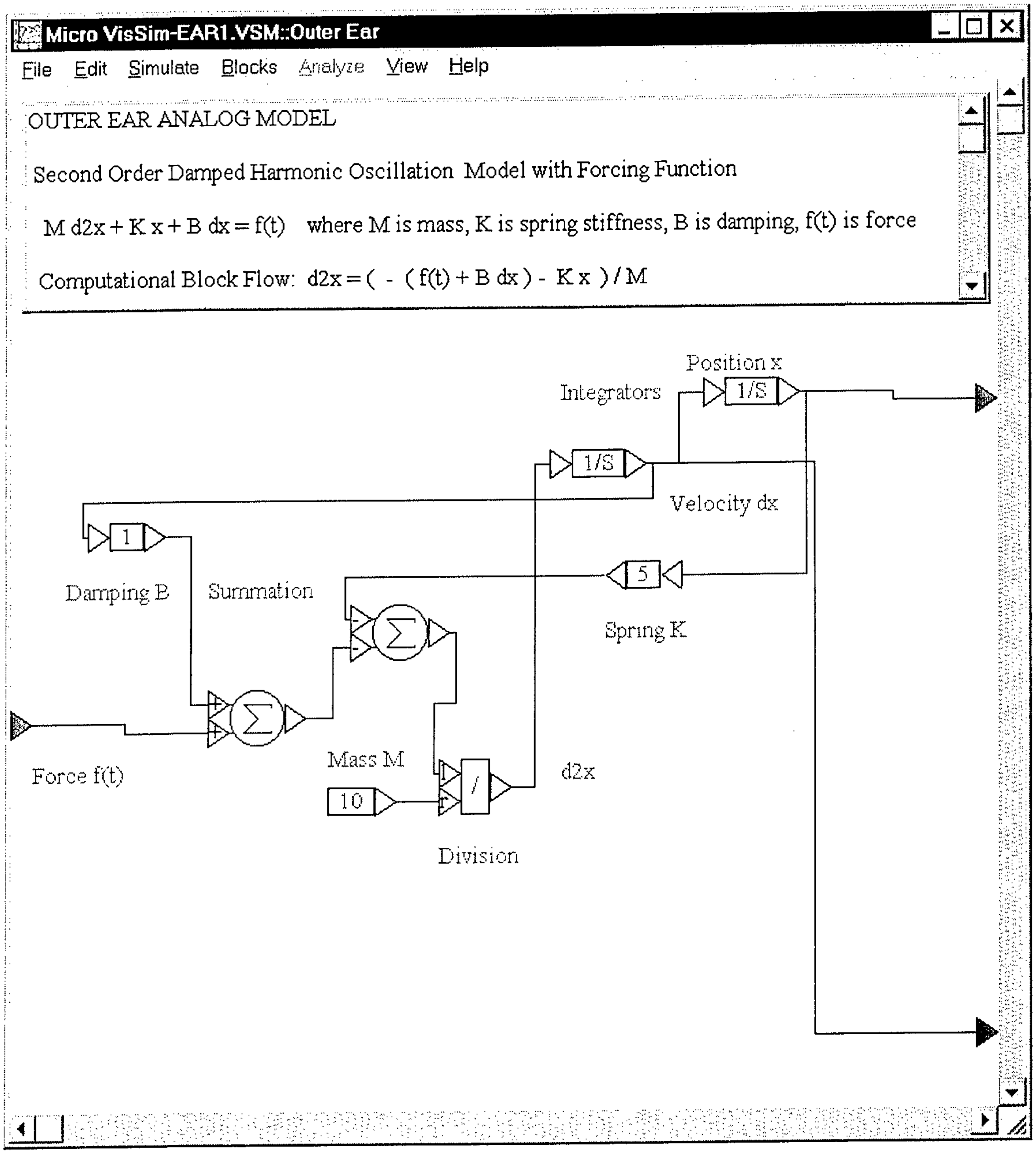
Figure 5.1-4 Example APS Model and Submodels Developed Using VISSIM



Micro VisSim-EAR1.VSM

ANALOG SIMULATION MODEL OF EAR PHYSICS AND AUDITORY NERVE OUTPUT

Sound pressure level signal flows to outer ear which is mechanically amplified into middle ear. Inner ear fluid is excited to stimulate auditory nerve firing as electrical pulses.



Micro VisSim-EAR1.VSM::Outer Ear
File Edit Simulate Blocks Analyze View Help

OUTER EAR ANALOG MODEL

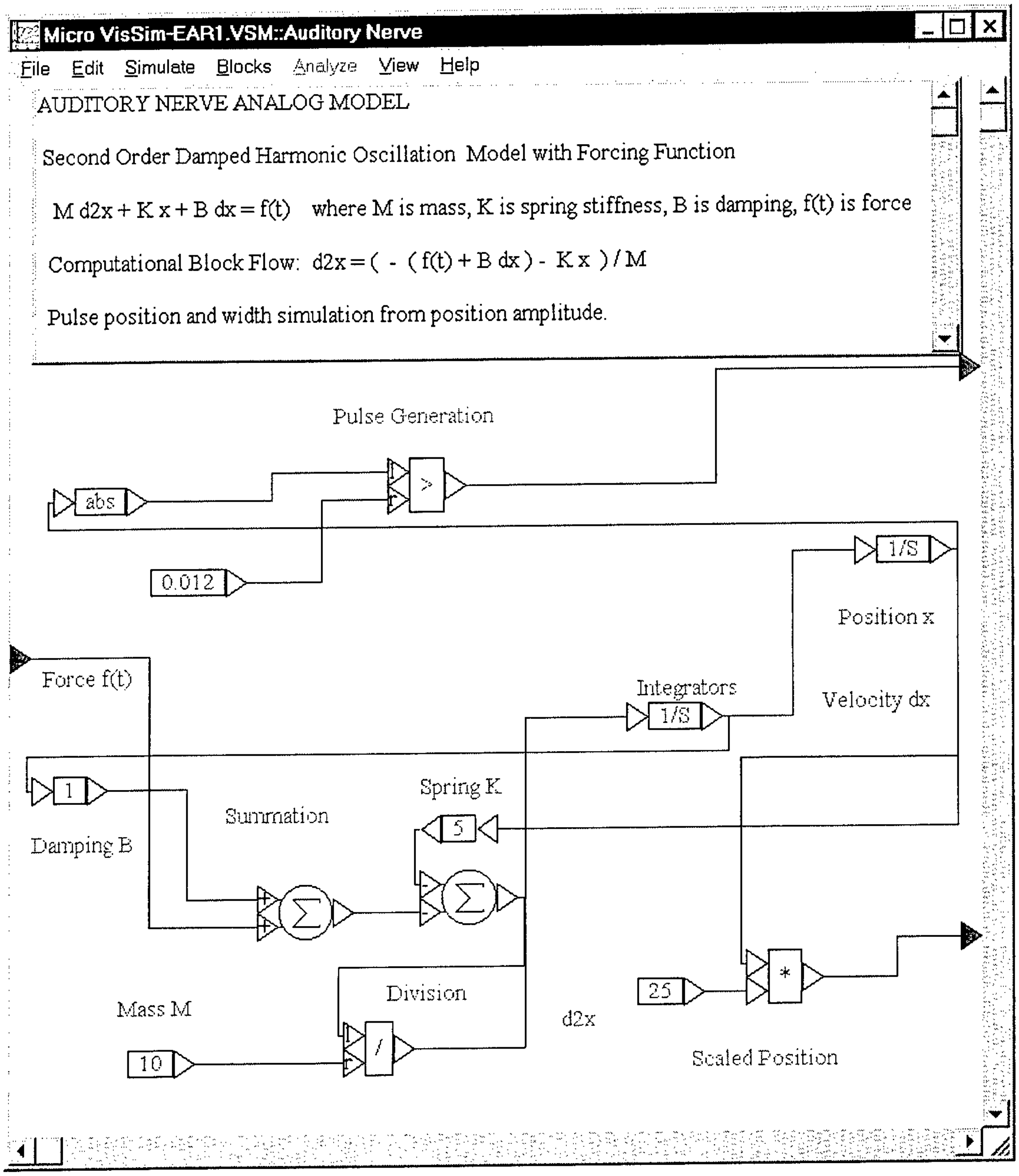
Second Order Damped Harmonic Oscillation Model with Forcing Function

M d2x + K x + B dx = f(t) where M is mass, K is spring stiffness, B is damping, f(t) is force

Computational Block Flow: d2x = (- (f(t) + B dx) - K x) / M

Figure 5.1-5 Outer Ear Model

These models represent a preliminary attempt at creating an engineering simulation model of the auditory process, Figure 5.1-5. The top level window shows the coupling of the signal input with the outer ear model shown above. The outer ear displacement is coupled to the middle ear, which is coupled to the inner ear. The inner ear is coupled to the auditory nerve to produce a series of electrical pulses as shown in Figure 5.1-6. As this research progresses, the parameters of each model will be calibrated to physical data to model the auditory process. The goal is the simplest model that captures the auditory response that can be solved in a closed form for rapid computation of feature data.



AUDITORY NERVE ANALOG MODEL

Second Order Damped Harmonic Oscillation Model with Forcing Function

M d2x + K x + B dx = f(t) where M is mass, K is spring stiffness, B is damping, f(t) is force

Computational Block Flow: d2x = (- (f(t) + B dx) - K x) / M

Pulse position and width simulation from position amplitude.

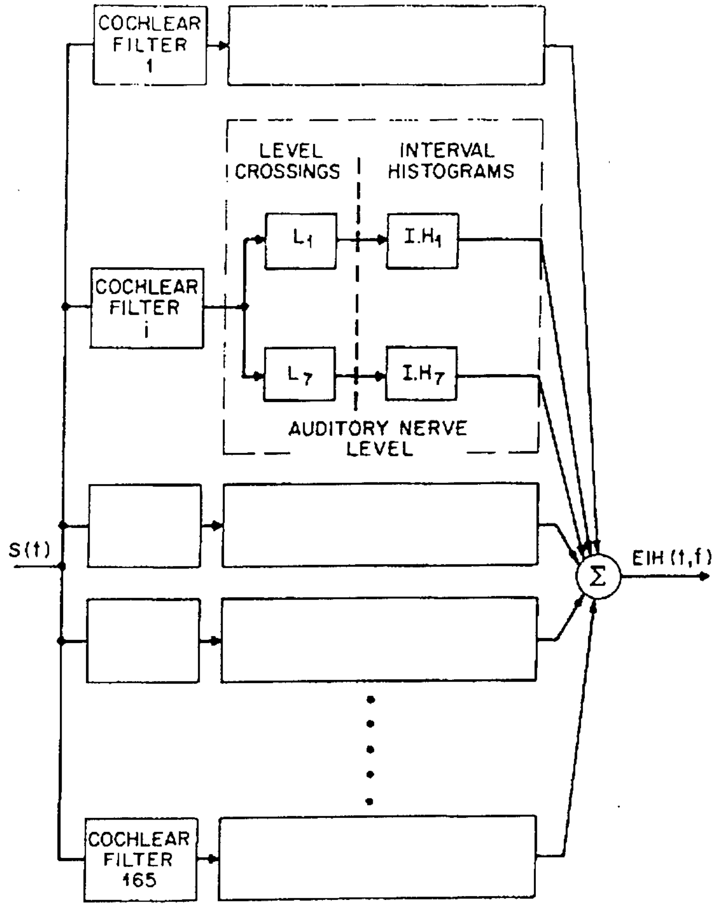
Figure 5.1-6 Inner Ear Pulses

5.2 Ensemble Interval Histogram - EIH

The ensemble interval histogram models the cochlea and the hair cell transduction to create numerical features from digitized speech signals. The signal processing consists of a filter bank that models the frequency selectivity at various points along a simulated basilar membrane, and a nonlinear processor for converting the filter bank output to neural firing patterns along a simulated auditory nerve.

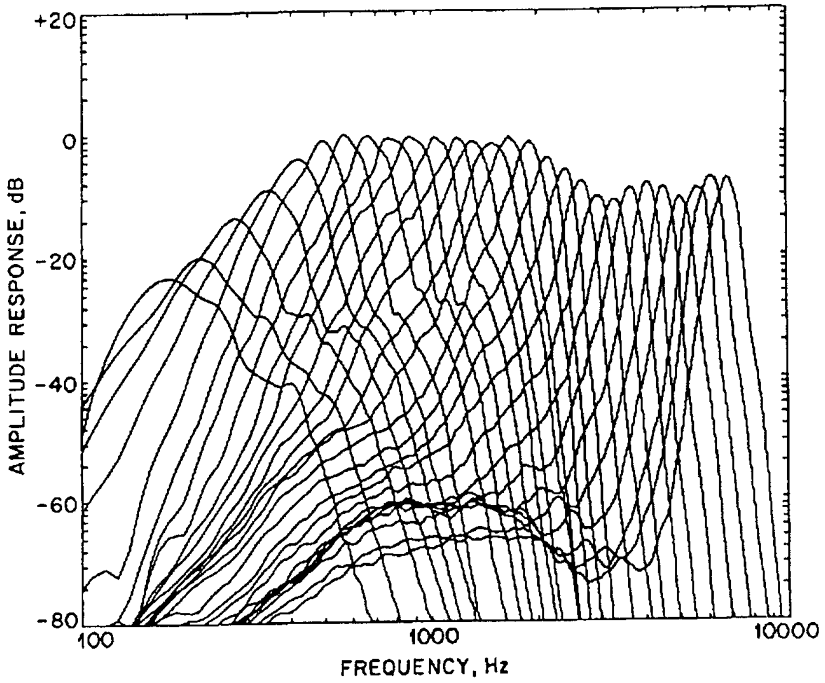
In this EIH model, the mechanical motion of the basilar membrane is sampled using 165 IHC channels, equally spaced, on a log-frequency scale, between 150 and 7000 Hz. The corresponding cochlear filters are based on actual neural tuning curves for cats. Sample amplitude responses of 28 of these filters (i.e., about 1 in 8 from the model) are shown in Figure 5.2-1. The phase characteristics of these filters is minimum phase, and the relative gain, measured at the center frequency of the filter, reflects the corresponding value of the cat's middle ear transfer function.

Figure 5.2-2 Block Diagram of EIH Computation Model



The next stage of processing in the EIH model is an array of level crossing detectors that models the motion-to-neural activity transduction of the hair cell mechanisms, Figure 5-2-2. The detection levels of each detector are pseudo-randomly distributed (based on measured distributions of level firings), thereby simulating the variability of fiber diameters and their synaptic connections.

Figure 5.2-1 Frequency Response of Cat Basilar Membrane



The output of the level-crossing detectors represents the discharge activity of the auditory nerve fibers. Figure5.2-3 shows simulated auditory nerve activity, for the first 60 msec of the vowel/o/in the word "job," as a function of both time and the "characteristic frequency" of the IHC channels. (Note the logarithmic scale of the characteristic frequency, which represents the place-to-frequency mapping on the basilar membrane.) In Figure 5.2-4, a level-crossing occurrence is marked as a single dot, and the output activity of each level-crossing detector is plotted as a separate trace. Each IHC channel contributes seven parallel traces (corresponding to the seven level-crossing detectors for each channel), with the lowest trace representing the lowest-threshold level-crossing detector. If the magnitude of the filter's output is low, only one level will be crossed, as is seen for the very top channels of the figure. However, for large signal magnitudes, several levels will be activated, creating a "darker" area of activity in the figure.

Figure 5.2-3 Magnitude of EIH Time Frequency Resolution for a Vowel

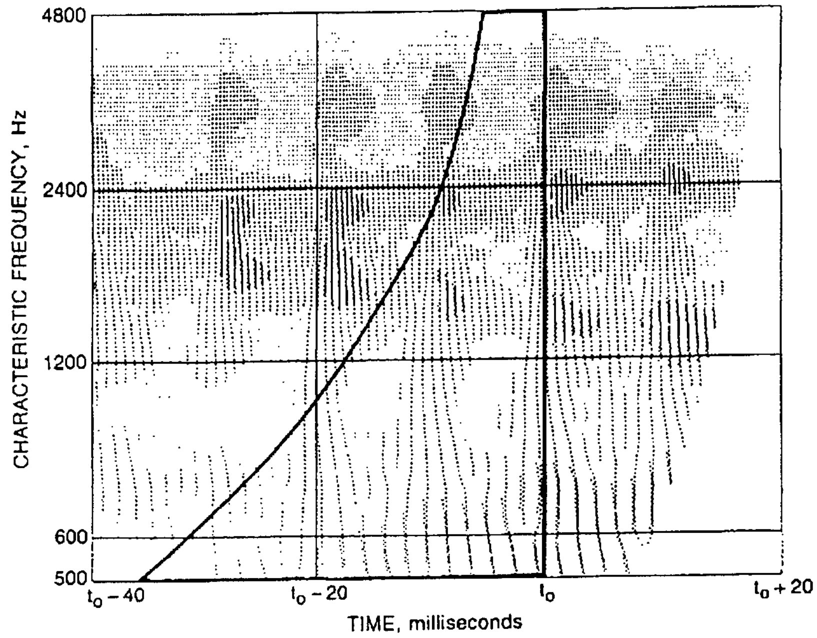
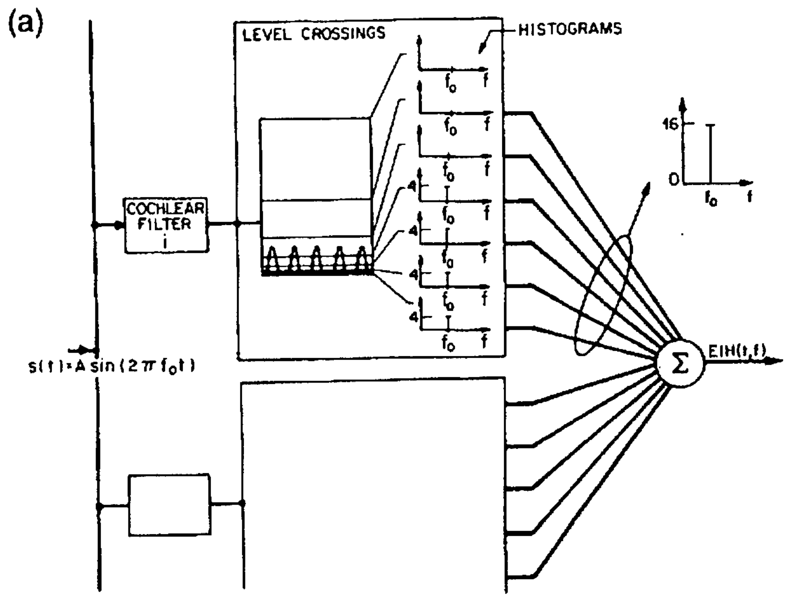


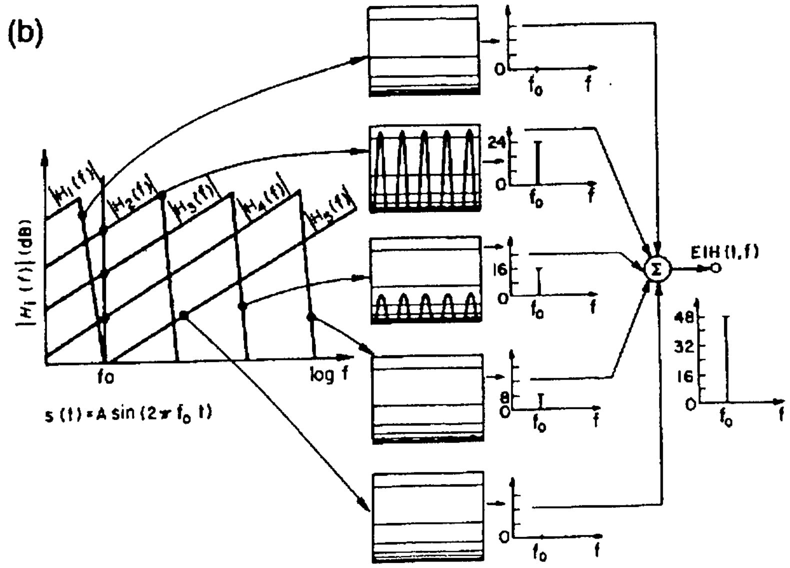
Figure 5.2-4 EIH Response Example Calculation for a



The level-crossing patterns represent the auditory nerve activity, which, in turn, is the input to a second, more central stage of neural processing, which gives the overall ensemble interval histogram (EIH). Conceptually, the EIH is a measure of the spatial extent of coherent neural activity across the simulated auditory nerve. Mathematically, it is the short-term probability density function of the reciprocal of the intervals between successive firings, measured over the entire simulated auditory nerve in a characteristic frequency-dependent time-frequency zone.

As a consequence of the multilevel crossing detectors, the EIH representation preserves information about the signal's overall energy. To illustrate this point, consider the case in which the input signal is a pure sinusoid and the characteristic frequency of a selected channel is shown, Figure 5.2-4. For a given intensity A, the cochlear filter output will activate only some low level-crossing detectors. For a given detector, the time interval between two successive positive-going level crossings is computed. Since the histogram is scaled in units of frequency, this interval contributes a count to the ## bin. For the input signal in Figure 5.2-4, all of the intervals are the same, resulting in a histogram in which the magnitude of each bin, save one, is zero.

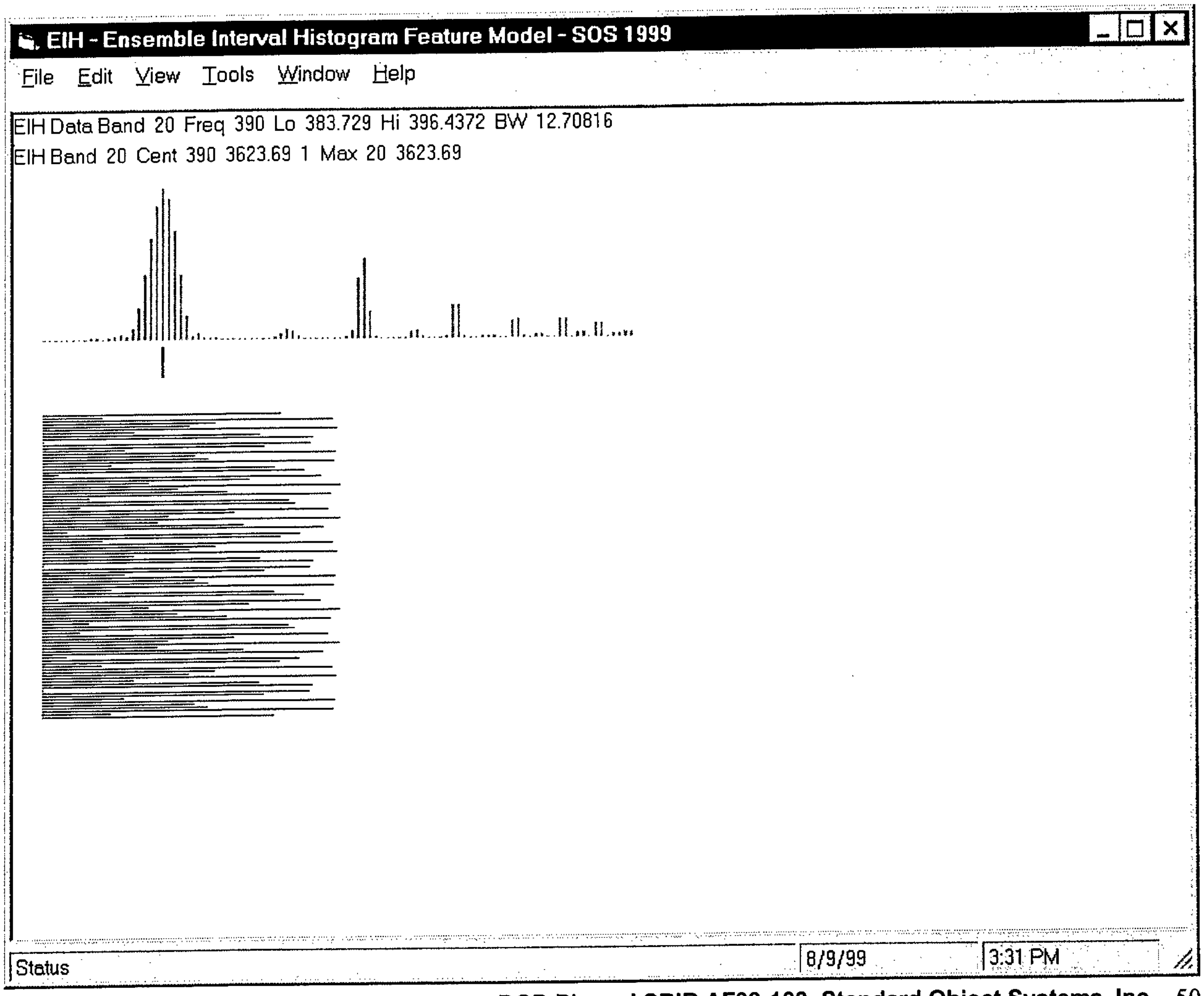
Figure 5.2-5 Example EIH Histogram Calculation for a Sinusoid



As the signal amplitude increases, more levels are activated. As a result, this cochlear filter contributes additional counts to the bin of the EIH. Since the crossing levels are equally distributed on a log-amplitude scale, the magnitude of any EIH bin is related, in some fashion, to decibel units. However, this relation is not a straightforward one because there are several sources contributing counts to the bin in a nonlinear manner. Figure E shows an input signal s(t) driving five adjacent cochlear filters with an amplitude response and a phase response. Due to the shape of the filters, more than one cochlear channel will contribute to the bin. In fact, all the cochlear filters will contribute to a bin of the EIH, provided that the signal exceeds any of the level-crossing thresholds. In Figure 5.2-5 only cochlear filters 2, 3, and 4 are contributing nonzero histograms to the EIH. The number of counts is different for each filter, depending on the magnitude of the signal.

One goal of auditory–based signal processing is to make the signal more robust to noise and reverberation than alternative spectral analysis procedures such as the filter-bank method or the LPC method. Figure 5.2-5 illustrates how well the EIH model achieves this goal. Shown in the figure are the log magnitude spectra of a clean (no noise) and a noisy (signal-to-noise ratio of 0 dB) speech signal processed by a standard Fourier filter bank (curves on the left) and by the EIH model (curves on the right). Also shown are LPC polynomial fits to the original signal spectrum (on the left) and to the EIH signal spectrum (on the right) for both the clean signal and the noisy signal. This figure clearly shows a tremendous sensitivity of the Fourier and LPC analyses to noise for the original signals. (This is especially seen in the LPC polynomial fits.) In the EIH case, the log magnitude spectra are almost unaltered by the noise, and the LPC polynomial fits are extremely close to each other. The implication of the above results for speech recognition is that the EIH model has potential for use in recognizing speech robustly in noisy and reverberant environments.

Figure 5.2-6 Example SOS Model to Compute EIH Feature



EIH - Ensemble Interval Histogram Feature Model - SOS 1999
File Edit View Tools Window Help
Status
8/9/99
3:31 PM

EIH Data Band 20 Freq 390 Lo 383.729 Hi 396.4372 BW 12.70816
EIH Band 20 Cent 390 3623.69 1 Max 20 3623.69

SOS is in the process of creating an EIH program based on the published description. Figure 5.2-6 shows the initial results from the filter bank to a square wave input (390 Hz) that detected the input frequency with the proper narrow band (12.7 Hz) filter. Initial histogram data is shown below the filter bank for the selected band. The EIH bandpass filter designs are zero phase shift FIR filters for each EIH band. Continued work will implement the interval histogram computation and feature output.

5.3 Auditory Image Model - AIM

The Applied Psychology Unit at Cambridge University has developed a time-domain model of auditory processing to simulate the auditory images produced by complex sounds like music, speech, bird song, engines, etc that represent initial sensations or perceptions of a sound rather than images of past events recalled from memory.

The Auditory Image Model (AIM) constructs its simulation of what we hear in three stages:

- Using an auditory filter bank, it converts the digitized sound wave into a simulation of the basilar membrane motion (BMM) that the sound would produce in the cochlea.
- Using a bank of haircell simulators, it 'transduces' the BMM into a simulation of the Neural Activity Pattern (NAP) that the sound would produce in the auditory nerve.
- Finally, it applies a new form of Strobed Temporal Integration (STI) to each channel of the NAP to convert the array of NAP channels into the model's simulation of our auditory image of the sound.

The NAP includes 'phase-locking' information encoded by the inner haircells because it is assumed that this information plays an important role in auditory perception and speech perception. STI performs temporal integration without destroying the phase-locking information of regular sounds -- the phase locking information that we hear. Thus, AIM is a time-domain auditory model for studying the role of phase locking and temporal fine-structure in auditory perception. Sequences of auditory images can be replayed to produce cartoons of auditory events that illustrate the dynamic response of the auditory system to everyday sounds.

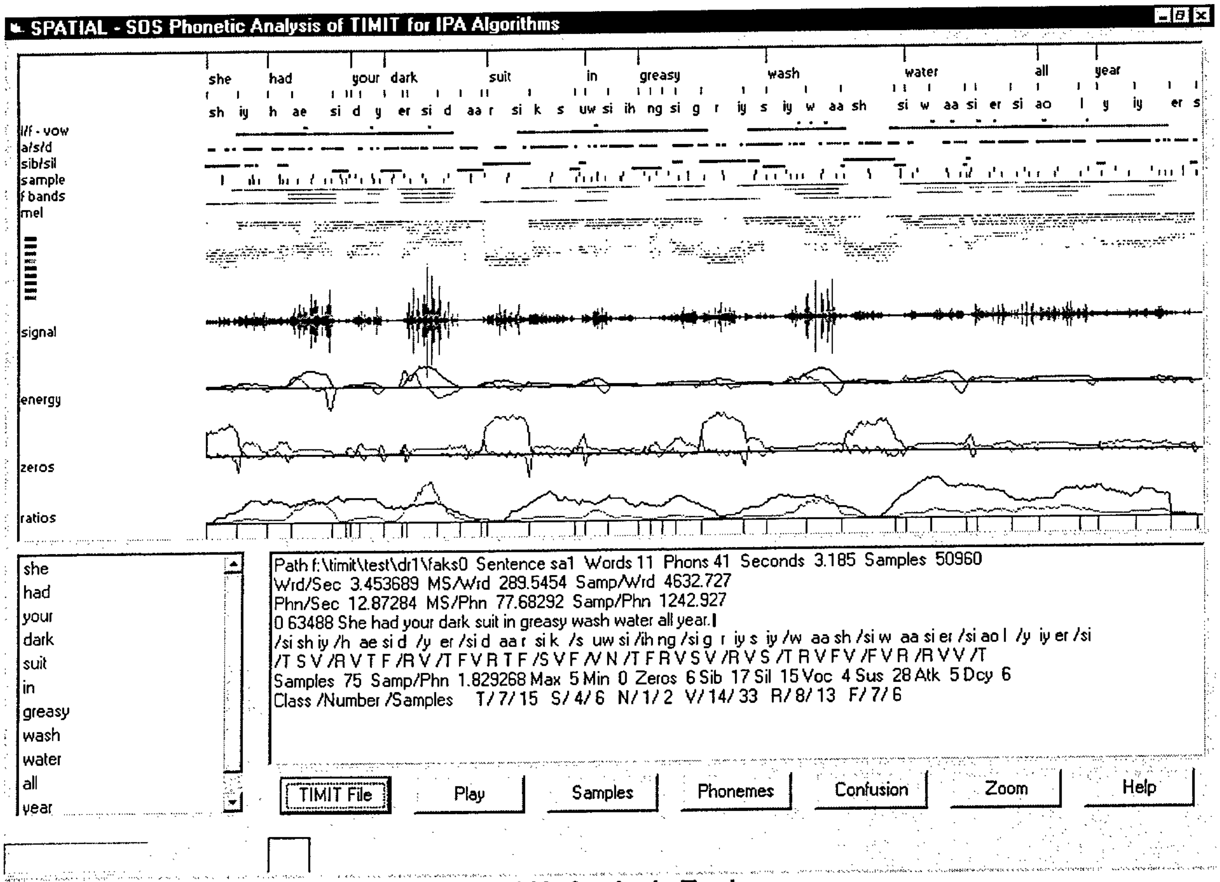
When an event occurs in the world around us, a car roaring past or a cat meowing, information about the event flows to us in light waves and sound waves. Our eyes form a visual image of the event, our ears form an auditory image of the event. The two are then combined with any other sensory inputs to produce our initial experience of the event. Auditory Image Model (AIM) research is primarily concerned with the development of a theory of auditory images and the application of that theory to speech and music perception.

At the Applied Psychology Unit, a time-domain model of auditory processing has been developed to simulate the auditory images produced by complex sounds. It converts digitized sound waves into a simulation of the Neural Activity Pattern (NAP) produced by the cochlea in response to a sound, and then applies a new form of Strobed Temporal Integration (STI) to the NAP to convert it into a high-resolution auditory image of the sound. The NAP includes the phase-locking information encoded by the inner haircells because it is assumed that this information plays an important role in auditory perception and speech perception. STI performs the temporal integration without destroying the phase-locking information of regular sounds,the phase locking information that we hear. Thus, AIM is a time-domain auditory model for studying the role of phase locking and temporal fine-structure in auditory perception.

One of the primary objectives of AIM is to explain the prominent role of octaves in music perception. The logarithmic spiral, base 2, provides means of representing octaves in time-domain models of hearing. It is simply a different mapping of the information in the auditory image, but it has the useful property of concentrating periodicity information in the auditory image. The spiral mapping is available in AIM (genspl) and it forms the basis of a module that extracts global parameters from the auditory image for pitch, pitch strength, and loudness. SOS has acquired the latest version of the AIM computer program from Cambridge University and is in the process of implementing it on a desktop PC in Windows. This will be compared to the EIH and APS models to determine the most usable auditory features to use in a speech recognition proof of concept.

6.0 SPSR PHONETIC SPEECH RECOGNITION TOOL KIT

The SOS Phonetic Speech Recognition (SPSR) software uses digital signal analysis methods for phoneme detection and identification rather than a linguistic analysis based on a specific language or dictionary. The premise of this approach is that speech consists of a set of finite length sound units that remain constant for short time periods. These sound units are referred to as phonemes, and they are the signals that will be detected and identified in the dynamic and noisy speech signal processing environment. In the first step the sound is converted to digital form and grouped into short time segments. Each segment is analyzed and a set of phonetic features are computed. The features are used to classify the segments into phonemes. Based on the classification, the actual phonemes are detected and identified for a group of segments. This is followed by a lexical process to determine syllables and text from phoneme strings. Figure 6.0-1 illustrates the acoustic segmentation process implemented in the SPATIAL speech analysis tool. This tool is aimed at using the TIMIT speech data as input and developing measures of the signal and labeled phoneme to classify acoustic segments of sound as classes of speech articulation.


SPATIAL - SOS Phonetic Analysis of TIMIT for IPA Algorithms
she had your dark suit in greasy wash water all year
sh iy h ae si d y er si d aa r si k s uw si ih ng si g r iy s iy w aa sh si w aa si er si ao l y iy er s
l/f - vow
a/s/d
sib/sil
sample
f bands
mel
signal
energy
zeros
ratios
she
had
your
dark
suit
in
greasy
wash
water
all
year
TIMIT File
Play
Samples
Phonemes
Confusion
Zoom
Help

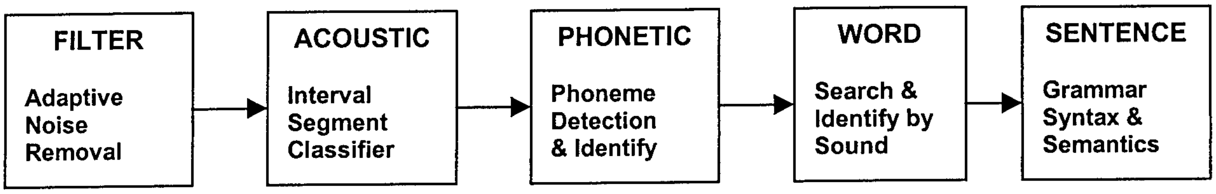
```
Path f:\timit\test\dr1\faks0 Sentence sa1 Words 11 Phons 41 Seconds 3.185 Samples 50960
Wrd/Sec 3.453689 MS/Wrd 289.5454 Samp/Wrd 4632.727
Phn/Sec 12.87284 MS/Phn 77.68292 Samp/Phn 1242.927
0 63488 She had your dark suit in greasy wash water all year.|
/si sh iy /h ae si d /y er /si d aa r si k /s uw si /ih ng /si g r iy s iy /w aa sh /si w aa si er /si ao l /y iy er /si
/T S V /R V T F /R V /T F V R T F /S V F /V N /T F R V S V /R V S /T R V F V /F V R /R V V /T
Samples 75 Samp/Phn 1.829268 Max 5 Min 0 Zeros 6 Sib 17 Sil 15 Voc 4 Sus 28 Atk 5 Dcy 6
Class /Number /Samples   T/ 7/ 15  S/ 4/ 6  N/ 1/ 2  V/ 14/ 33  R/ 8/ 13  F/ 7/ 6
```

Figure 6.0-1 SPATIAL Analysis Tool

The end to end speech recognition will combine the adaptive noise removal filter with the existing SOS phonetic speech recognition processing. The SPSR Tool Kit uses digital signal analysis methods for phoneme detection and identification rather than precomputed models based on a specific language or dictionary. The premise of this approach is that speech consists of a set of finite length sound units that remain constant for stable time segments. These sound units are referred to as phonemes, and they are the signals that will be detected and identified in the dynamic and noisy speech signal-processing environment. The steps in this process are illustrated in Figure 6.0-2, where first the sound is filtered to remove noise.

This time delayed enhanced digital speech signal is grouped into acoustic class segments. Each segment is analyzed, and a set of phonetic features is computed. The features are used to classify the segments into phonemes. Based on the classification, the actual phonemes are detected and identified for a group of segments. This is followed by a sound pattern search to determine words and a parse to determine grammatically correct phrases, and a lexical process to create text output. SOS will modify the existing SPSR Tool Kit algorithms to include EIH phonetic features that are noise resistant and the use of noisy spelling algorithms for vocabulary words. The VIGOR genetic algorithm will be used to optimize the parameters for acoustic segment identification and phoneme classification.

Figure 6.0-2 End to End Noisy Speech Recognition Processing


FILTER
Adaptive Noise Removal
ACOUSTIC
Interval Segment Classifier
PHONETIC
Phoneme Detection & Identify
WORD
Search & Identify by Sound
SENTENCE
Grammar Syntax & Semantics

6.1 PHONETIC FEATURE TRAINING

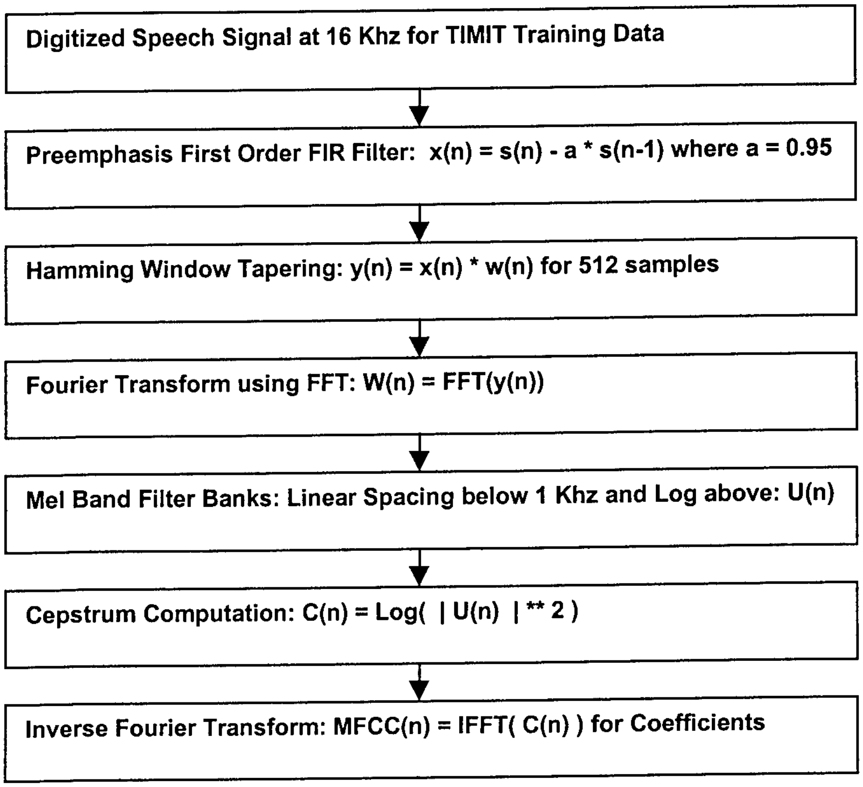
The goal of phonetic feature training is to determine a set of speech signal features that can be used to recognize spoken phonemes. These features capture the essence of the speech and reject the non-information parts of the acoustic signal. The physical voice articulators produce the speech acoustic signal in order to communicate a stream of phonemes. These physical mechanisms are in a stable position for only a short time period varying from 80 to 200 ms and then they transition to the next spoken phoneme.

A common mathematical model of the speech signal is to separate the voice excitation signal and the vocal tract filter. The excitation is assumed periodic with a pitch equal to the frequency of the voiced phonemes or to be white noise for unvoiced phonemes. The phonetic features are then defined by the vocal tract filter coefficients for each sound. The most successful speech recognition training features are the Mel Frequency Cepstral Coefficients (MFCC).

The information bandwidth of the speech signal is under 7 Khz so that the 16 Khz sample rate of the TIMIT speech results in a bandwidth of 8 Khz. The physical articulator rate allows a signal sample block size of 0.01 seconds which is approximately 512 samples at 16 Khz. This allows an efficient power of two block size for fast Fourier transform calculations. The digital signal processing to compute the MFCC data is shown in the Figure 6.1-1. The computations consist of the following digital signal processing blocks. A preemphasis FIR filter to boost the high frequency energy. The tapering of the signal data block by Hammings method to reduce the sampling artifacts in the frequency domain. A discrete Fourier transform to compute the frequency domain amplitude and phase for this signal sample block. The construction of frequency domain Mel band filters that are spaced linearly below 1 Khz and logarithmically above 1Khz. The computation of the signal cepstrum to remove the excitation signal. The inverse of the cepstrum to compute the vocal tract filter coefficients to be used as the speech training data.

The TIMIT acoustic phonetic spoken database is used as the source of phoneme training data. Each labeled phonetic speech segment is computed to determine the MFCC data values. These are accumulated to determine statistical speech models for each phoneme by gender and dialect. The resulting training database is represented as a table of phonemes that is compared to incoming speech blocks to estimate the phoneme contained in the speech signal.

Figure 6.1-1 MFCC Speech Feature Digital Signal Computation


Digitized Speech Signal at 16 Khz for TIMIT Training Data
Preemphasis First Order FIR Filter: x(n) = s(n) - a * s(n-1) where a = 0.95
Hamming Window Tapering: y(n) = x(n) * w(n) for 512 samples
Fourier Transform using FFT: W(n) = FFT(y(n))
Mel Band Filter Banks: Linear Spacing below 1 Khz and Log above: U(n)
Cepstrum Computation: C(n) = Log(| U(n) | ** 2)
Inverse Fourier Transform: MFCC(n) = IFFT(C(n)) for Coefficients

6.2 PHONEME RECOGNITION

The phonetic detection step in the recognition stage is unique to the SOS approach and consists of parallel classification processes for each acoustic segment of speech. Each classification method takes in the feature data for a speech segment and produces an estimate of which phoneme is the best match for this segment. The methods are independent and all of the estimates are combined to create the most likely estimate. The output is a matrix by acoustic segment of the probability of each phoneme called the phonetic lattice.

The key to this approach is the detection and identification of phonemes, which are seen to be universal and fundamental to human speech communication. This method differs dramatically from other speech recognition methods that are based on matching sound templates, on deriving statistical models of word structure, or on code book quantizations. This is a computationally intensive method that requires high speed digital processing to achieve real time performance. The process is scaleable so that more processing resources lead to more accurate speech recognition. The design is modular so that processing can be distributed for parallel computing execution.

This phonetic speech recognition process is implemented as standard software objects in C++. They are portable across multiple platforms including PCs, workstations, and DSPs. A prototype version of the SOS process was tested on the Japanese Hiragana language with good results. An IPA prototype is available using the NIST developed TIMIT speech corpus for eight regional accents. The phonetic feature generation is the first step in the recognition process. It consists of parallel processes for each segment of speech. The energy and zero crossings are computed from the time domain signals for the segment. An SOS designed unity gain and zero phase shift finite impulse response filter is used to represent each of the Mel scale frequency bands to estimate the features. The segment is windowed and zero filled to perform a radix two fast Fourier transform used to estimate the power spectrum density features. A correlation method is used to compute the linear prediction coefficients that are transformed into the Cepstrum coefficient features.

The phonetic detection step in the recognition stage is unique to the SOS approach and consists of five parallel processes for each segment of speech. Each method takes in the feature data for a speech segment and produces an estimate of which phoneme is the best match for this segment. The methods are independent, and all of the estimates are combined to create the most likely estimate. The first method matches the features to a table of stored features by using a minimum absolute difference metric. The second method uses a multilayer feed forward neural network to classify the input features. The third method uses a Bayesian statistical estimator to compute the conditional probability of each phoneme based on the features. The fourth method uses a probabilistic Markov model to classify the features. The fifth method uses a fuzzy logic classifier to estimate the membership function for each phoneme set. The sets of all phoneme estimates are combined for each segment to estimate the best phoneme. This probabilistic process is the ideal point to compensate for specific phonetic uncertainty due to high non-stationary noise.

The continuous speech recognition algorithm processes all of the speech segments in an utterance. The goal is to process the stream of discrete phonetic segment probabilities to determine the maximum likelihood estimate of the uttered phonemes. Each of the phoneme classification algorithms provides an estimated probability of occurrence vector for all phonemes. Three techniques are used compute the best estimate of the uttered phonemes. First, dynamic programming is used to select phonemes that maximize the probability of phoneme sequence occurrence with a Lagrange multiplier control to compensate for specific phoneme noise masking. Second, a Markov chain model is used to maximize the estimated phoneme sequence while allowing for dropouts due to noise masking. Third, an assignment based heuristic algorithm is used to select the most likely sequence of phonemes that includes the uncertainty due to the noisy environment. The result is a lattice of likely phonemes with start times and durations to input to the phonetic word pattern search and grammar parsers.

6.3 VOCABULARY AND GRAMMAR RECOGNITION

The International Phonetic Alphabet (IPA) used by SOS contains the phonemes for the speech sounds found in over 350 of the languages in the world. The exact number of distinct phonemes needed to represent a language is a matter of judgment among linguists. American English has 48 phonetic sounds in the ARPABET representation. Hiragana, the Japanese phonetic language, has only 20 phonetic sounds that can combine to form only 72 unique syllables. As another example, German is phonetically rich and has over 60 phonetic sounds. Western languages usually have over 5,000 separate syllables units that form into words. The IPA symbols selected by SOS for American English based on a study of a large phonetic dictionary, the TIMIT word set, and published pronunciation statistics. SOS exploits this data by initially segmenting a speech utterance into six acoustic categories with high accuracy. This dramatically reduces the computation as compared to other methods that often process speech in fixed time steps of one hundred per second. The SOS approach results in an average of 80% saving in computing time.

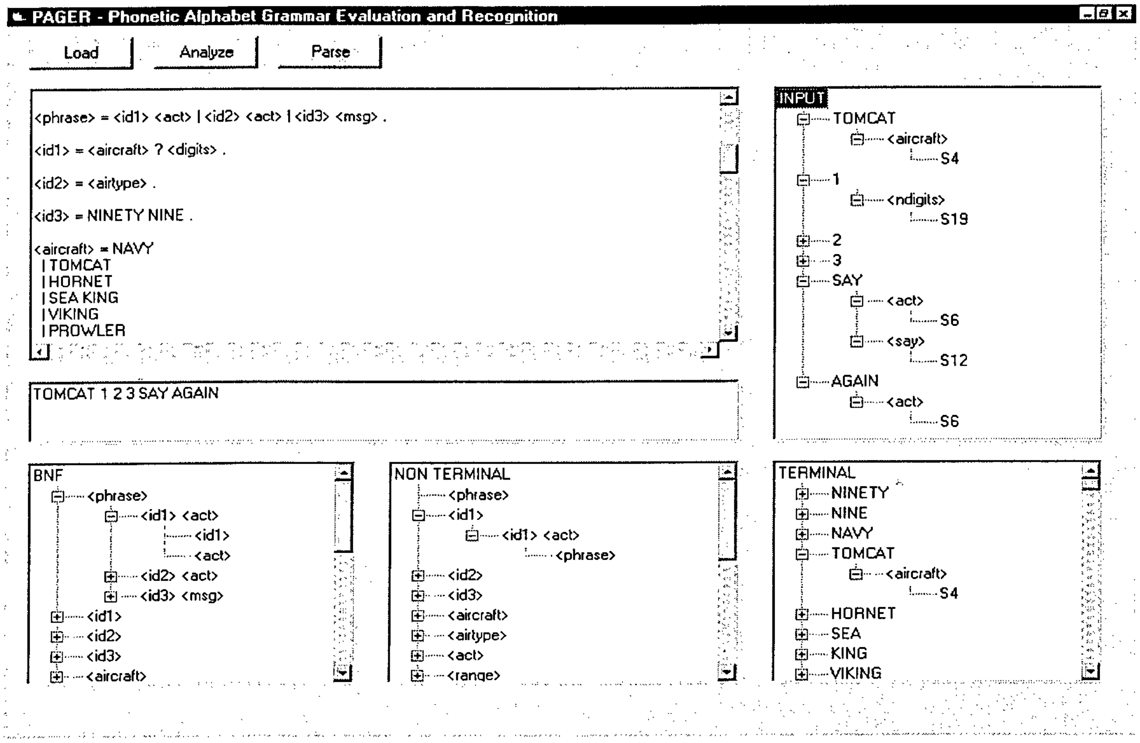
The identification of spoken words is constrained in this experiment to the TIMIT domain vocabulary. Using the TIMIT corpus the number of words is approximately 6000. SOS has a process to convert a phonetic lattice to a word lattice given a pronunciation vocabulary for the domain. In general the percent correct word recognition is a good measure of the speech recognition performance.

The 2000 TIMIT sentences can be recognized using a simple word occurrence statistic due to the low perplexity of the corpus. SOS computes the correct sentence recognition performance but does not consider this data to be a meaningful measure of the speech recognition performance. In general only two or three words need to be recognized in order to determine the correct sentence.

The techniques for searching phonetic dictionaries to transform phonemes into words have been developed by numerous stenographic dictation methods over a number of years. In general, court recorders type phonetic representations, rather than text, by using chord key stenographic equipment and proprietary phonetic encoding methods. Software exists to convert this phonetic representation to text transcriptions in many languages. SOS has modified a common public domain method for word recognition that generates output compatible with other systems for phonetic text conversion in specialized fields such as law and medicine. The unique part of the SOS implementation is the use of approximate pattern matching algorithms on the input phonetic lattice to produce a set of most likely words stored as a word lattice.

Humans make use of many non-acoustic sources of information in addition to phonetic data for speech translation including syntax, semantics, pragmatics, and dialog. Statistical methods such as trigrams have evolved to predict the next most likely word in an utterance, and are powerful approaches for perplexity reduction for dictation systems. A common measure of the difficulty of lexical processing is the perplexity of the language, which refers to the number of different branches required to identify the correct word in a full graph of the language.

One of the key problems in continuous speech recognition is the unreliable recognition of word boundaries in an utterance. The SOS approach uses a modified parsing algorithm that operates on three levels. First, the rapid recognition of a small set of control keywords such as ON, OFF, etc. Second the acoustic level using the phonetic word data produces multiple candidate phrases for an utterance. Third the grammar level selects the most likely phrase based on non-acoustic language information. This process is based on using a BNF grammar that is defined for the phrase structure of the language using the industry standard SRCL. The parser is applied in a top down approach to select the candidate phrase that best fits the language, Figure 6.3-1. Semantic tests are applied to reject nonsensical wording, format numbers, and punctuate where possible. Since people do not speak in a proper written language format, the editing to produce acceptable text is a post recognition task.


PAGER - Phonetic Alphabet Grammar Evaluation and Recognition
Load
Analyze
Parse

```
<phrase> = <id1> <act> | <id2> <act> | <id3> <msg> .

<id1> = <aircraft> ? <digits> .

<id2> = <airtype> .

<id3> = NINETY NINE .

<aircraft> = NAVY
 | TOMCAT
 | HORNET
 | SEA KING
 | VIKING
 | PROWLER
```

```
TOMCAT 1 2 3 SAY AGAIN
```

```
INPUT
  TOMCAT
      <aircraft>
          S4
  1
      <ndigits>
          S19
  2
  3
  SAY
      <act>
          S6
      <say>
          S12
  AGAIN
      <act>
          S6
```

```
BNF
  <phrase>
      <id1> <act>
          <id1>
          <act>
      <id2> <act>
      <id3> <msg>
  <id1>
  <id2>
  <id3>
  <aircraft>
```

```
NON TERMINAL
  <phrase>
  <id1>
      <id1> <act>
          <phrase>
  <id2>
  <id3>
  <aircraft>
  <airtype>
  <act>
  <range>
```

```
TERMINAL
  NINETY
  NINE
  NAVY
  TOMCAT
      <aircraft>
          S4
  HORNET
  SEA
  KING
  VIKING
```

Figure 6.3-1 Example of SOS Visual Grammar Parsing Tool for an ATC Input

The SOS recognition process has three steps where performance measurements can be taken. First the temporal filter of the phoneme probabilities increases the number of correct phoneme identifications. The gain from this filter for successful phoneme recognition is usually over thirty percent. Step two is the phonetic word recognition process using a specialized vocabulary search algorithm. Step three is the grammatical sentence recognition process which is specific to the speech recognition domain.

6.4 MULTILINGUAL SPEECH RECOGNITION AND TRANSLATION

The following sections describe the multilingual speech recognition and language translation capability planned for the SPSR tool kit. SOS has created a unique computer language translation system using a PROLOG interpreter written in C++. This integrates with the set of C++ phonetic speech recognition software objects in the SPSR tool kit. The objective is to create a translation system with a multilingual speech recognition input that will operate in noisy portable environments. The existing SOS technology will satisfy these objectives as explained in the following sections. The goal for the design is to create a prototype by combining the phonetic speech recognition, noise filtering, and language translation technologies into a Phase II prototype system. As an example, this prototype will allow an English only speaker to create a simple bilingual translator for Spanish and automatically generate the two language vocabularies with phonetic word representation for bilingual speech recognition.

SOS is prepared during Phase II of this research program to use these objectives to survey the current state of the art in spoken multilingual translation. It will search the field for current developments to analyze and incorporate the best research results available. SOS will design a specific system for language translation with speech recognition in high noise military environments for Phase II prototype development and Phase III product commercialization.

Figure 6.4-1 Technical Objectives for a Spoken Language Translation System

TECHNICAL OBJECTIVES		DESIGN APPROACH
	TRANSLATION	
Word Lexicon		Phonetic word models for pronunciation
Baseline Sentence Inputs		Statistical and logical translator generation
Predicate Logic Translation		Portable PROLOG interpreter in C++
Translator Optimization		Genetic Algorithm to evolve parameters
	RECOGNITION	
Speaker Independent Operation		Uses samples of English speakers from TIMIT
Multiple Dialects and Accents		Eight dialect regions, multiple native accents
Phonetic Unit Identification		Multiple independent feature classifiers
Near Real Time Response		Portable Pentium PC, C++, no other hardware
	VOCABULARY	
Basic Vocabulary		Phonetic dictionary with 200,000+ IPA entries
Word Recognition		Phonetic lattice search by sound patterns
Multi Lingual Capability		IPA covers over 350 spoken languages
	SPEAKER	
Identification Enrollment		Five minute phonetically rich sentence set
Continuous Adaptation		Session to session speaker characteristic file
Individual Speaker Model		Parameters for existing classification algorithms
Speaker Specific Identification		Single user tracking in multi speaker situations
	NOISE	
Noisy Environments		Model: stationary, non-stationary, quasi
Noise Removal		Adaptive digital filtering for noise cancellation
Custom Microphone		Two channel inputs with PC output
	APPLICATIONS	
Command and Control		Interrogation, check points, intelligence
Medical Information		Triage, history, interview, rounds
Borders and Immigration		Documentation, examination, boarding
Police Operations		Crowd control, questioning, directions

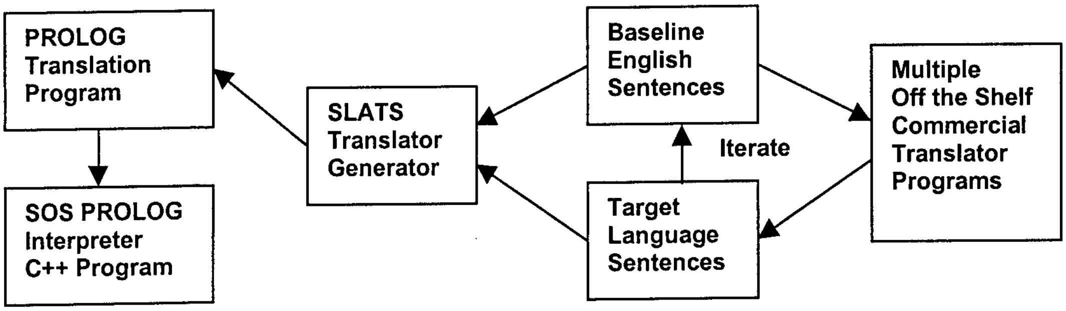
6.4.1 Technical Objectives for a Spoken Language Translation System

Figure 6.4-1 presents the specific technical objectives to be achieved by this language translation system with speech recognition in noisy environments alongside the design approach using the technology that is described in the following sections. The following are the technical objectives for Phase II of this research program:

1. Accurate translation of utterances for sample task specific dialogues
2. Effective speech recognition input in a noisy mobile environment
3. Simple authoring for translator using an English sentence set
4. Target languages: Spanish, French, Italian, Portuguese, etc.
5. Selection of commercial computing platform suitable for translation
6. Test and evaluation of translation system in a field environment

The following SOS technology description sections address each of these objectives.

Figure 6.4.2 SOS Process to Generate a Bilingual Translation Program


PROLOG Translation Program
SOS PROLOG Interpreter C++ Program
SLATS Translator Generator
Baseline English Sentences
Iterate
Target Language Sentences
Multiple Off the Shelf Commercial Translator Programs

6.4.2 Automatic Language Translator Generation

SOS has developed a novel language translator system that will be used for speech translation in noisy environments. The system is unique in that it requires no linguistic expertise or knowledge of the target language to author a translator for a task specific domain application. It is automatic in that no manual intervention is required to create a working program. This PROLOG program can be combined with the phonetic speech recognition system to produce a speech to speech translation system.

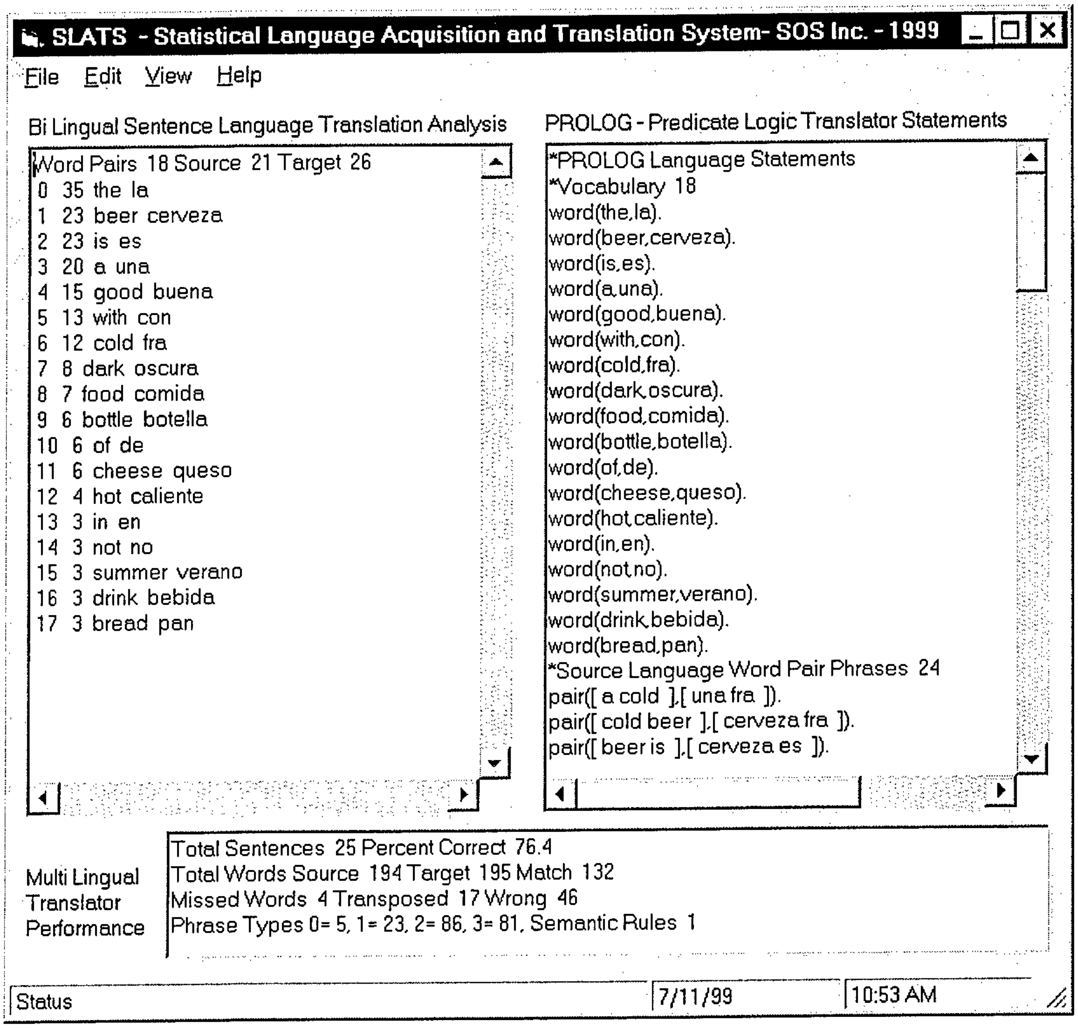
The SOS spoken language acquisition and translation system (SLATS) is based on using a single source language, currently English, for the authoring input as shown in Figure 6.4-2. To use it, a set of baseline domain sentences is created in the source language that over-describes the task domain vocabulary. This set of sentences is translated to the target language by a series of commercial off the shelf translation programs. The target translations are then retranslated to the source language, and this process is repeated until the bilingual translations are stable with no changes in vocabulary.

The SLATS program reads the bilingual sentences and automatically derives a vocabulary, a phrase substitution set, and semantic correction rules as shown in Figure 6.4-3. On the left side is the derived vocabulary data and on the right are the predicate logic translator statements in PROLOG. The bottom window has performance data for the test case based on this sentence set. The following Figure 6.4-4 illustrates the baseline sentences input in English and translated to Spanish for this example.

SOS has developed a PROLOG interpreter in C++, called SPIN, that executes the predicate logic declarations and translates source language input text to translated target language text. Initial experiments have been conducted with SLATS for target language translators, including Spanish, French, German, Italian, and Portuguese, using two popular off the shelf translation programs. The results for a small task-specific domain using the popular Wagner-Fisher evaluation algorithm from NIST, modified for translation comparisons, are shown in Figure 6.4-5. Research and development is continuing on the SLATS program based on the above results to improve the translator generation to over 90% and to evaluate it for other languages. SOS is using the 33 language Universal Translator Deluxe from Language Force to investigate the following additional languages:

Arabic, Chinese, Czech, Danish, Dutch, Esperanto, Farsi, Finnish, Greek, Hebrew, Hungarian, Indonesian, Latin, Japanese, Korean, Norwegian, Polish, Romanian, Slovak, Swahili, Swedish, Tagalog, Turkish, Ukrainian, and Vietnamese

Figure 6.4-3 Example Output of the SOS Translator Generation Program


SLATS - Statistical Language Acquisition and Translation System- SOS Inc. - 1999
File
Edit
View
Help
Bi Lingual Sentence Language Translation Analysis
PROLOG - Predicate Logic Translator Statements
Multi Lingual Translator Performance
Status
7/11/99
10:53 AM

```
Word Pairs 18 Source 21 Target 26
0  35 the la
1  23 beer cerveza
2  23 is es
3  20 a una
4  15 good buena
5  13 with con
6  12 cold fra
7  8 dark oscura
8  7 food comida
9  6 bottle botella
10  6 of de
11  6 cheese queso
12  4 hot caliente
13  3 in en
14  3 not no
15  3 summer verano
16  3 drink bebida
17  3 bread pan
```

```
*PROLOG Language Statements
*Vocabulary 18
word(the,la).
word(beer,cerveza).
word(is,es).
word(a,una).
word(good,buena).
word(with,con).
word(cold,fra).
word(dark,oscura).
word(food,comida).
word(bottle,botella).
word(of,de).
word(cheese,queso).
word(hot,caliente).
word(in,en).
word(not,no).
word(summer,verano).
word(drink,bebida).
word(bread,pan).
*Source Language Word Pair Phrases 24
pair([ a cold ],[ una fra ]).
pair([ cold beer ],[ cerveza fra ]).
pair([ beer is ],[ cerveza es ]).
```

```
Total Sentences 25 Percent Correct 76.4
Total Words Source 194 Target 195 Match 132
Missed Words 4 Transposed 17 Wrong 46
Phrase Types 0= 5, 1= 23, 2= 86, 3= 81, Semantic Rules 1
```

Figure 6.4-4 Example Baseline Sentence Set Used in Spanish Translation

original sentences	original sentencia
a cold beer is good with the food	una cerveza fría es buena con la comida
the dark beer is sweet	la cerveza oscura es dulce
a bottle of beer is good with the food	una botella de cerveza es buena con la comida
the dark beer is good in a bottle	la cerveza oscura es buena en una botella
a bottle of dark beer is refreshing	una botella de cerveza oscura está refrescándose
the beer is not good with the sweet food	la cerveza no es buena con la comida dulce
the dark beer is good in winter	la cerveza oscura es buena en invierno
a bottle of cold dark beer is a summer drink	una botella de cerveza oscura fría es una bebida de verano
a cold bottle of beer is a good drink	una botella fría de cerveza es una bebida buena
a bottle of cold beer is a good drink	una botella de cerveza fría es una bebida buena
the cold beer is good	la cerveza fría es buena
the dark beer is good	la cerveza oscura es buena
the hot beer is not good in summer	la cerveza caliente no es buena en verano
the hot bread is good with the beer	el pan caliente es bueno con la cerveza
the sweet cheese is not good with the beer	el queso dulce no es bueno con la cerveza
the cheese is good with the cold beer	el queso es bueno con la cerveza fría
the hot food is good with the cold beer	la comida caliente es buena con la cerveza fría
the cold food is good with the dark beer	la comida fría es buena con la cerveza oscura
the food is good with the beer	la comida es buena con la cerveza
a cheese is good with a cold beer	un queso es bueno con una cerveza fría
the dark beer is good with a cheese	la cerveza oscura es buena con un queso

In addition to this translation system other systems with higher performance and more accurate target language productions are being investigated. SOS will apply this novel approach of using multiple large domain translation systems to automatically produce task specific translators for this speech recognition in noisy environments project.

Figure 6.4-5 Performance Results for Multilingual Translator Program

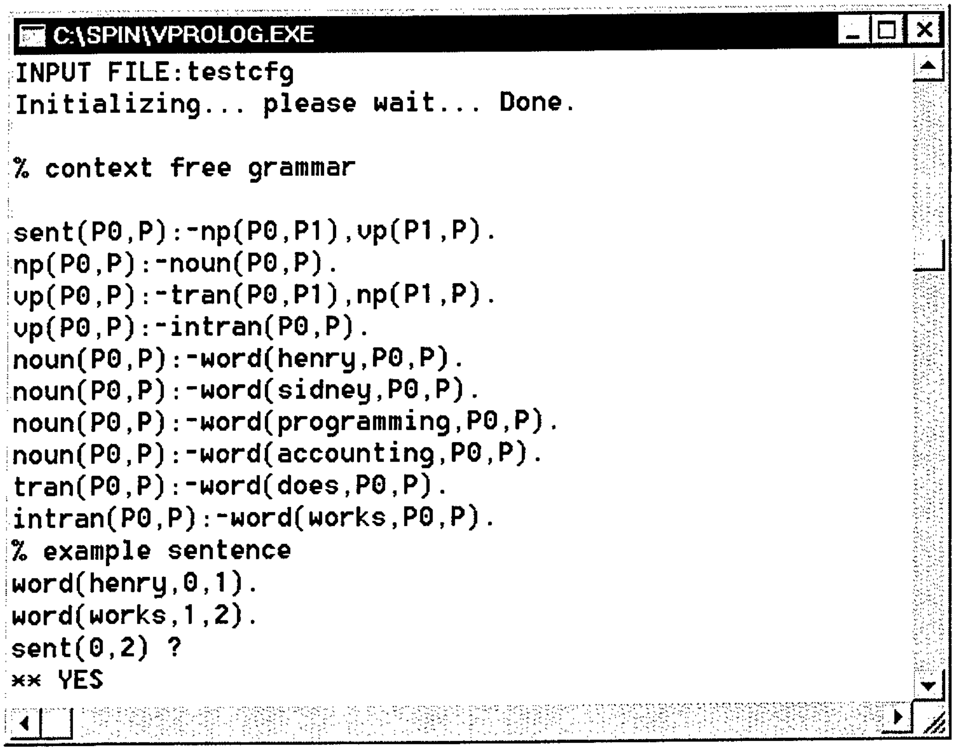
MEASURE	SPANISH	FRENCH	GERMAN	ITALIAN	PORTUGUESE
SENTENCES	27	27	27	27	27
WORDS	155	155	161	155	155
MISSED	26	31	30	30	18
INSERT	4	3	11	8	3
OMIT	2	6	1	4	2
SUB	17	19	12	20	9
ACCURACY	83.2%	80%	81.3%	80.6%	88.3%

6.4.3 Natural Language Processing and Translation with PROLOG

Natural language processing, understanding, and translation has been a computational linguistic research topic since the 1950s. One of the popular tools for NLP has been logic programming and PROLOG in particular. A primary goal of logic programming is to formally capture the notion of logical evaluation by declaring a set of predicate conditions and posing questions which can be answered by true or false, employing a formal unification procedure. The development of PROLOG to implement predicate logic in France and England in the 1970s led to significant applications in natural language processing. Predicate logic, logic programming, and PROLOG have become among the most popular research tools for natural language understanding. As a result, SOS developed a PROLOG interpreter in C++ as the natural language processing and translation interface for the output of its phonetic speech recognition tool kit. Numerous applications in spoken language understanding are possible with this development. The Figure 6.4-6 illustrates the SOS PROLOG interpreter (SPIN) execution of a simple example of a context free grammar to parse sentences.

There exists a number of commercial PROLOG programming systems, however, each is integrated with a particular operating system and computing platform which require large computer platforms. The design of the SOS PROLOG interpreter is to be a set of C++ objects that can be compiled as part of a larger spoken language system, which does not depend on any features of the operating system. Natural language applications developed on other PROLOG systems can also be executed by the SPIN program. In actuality, it is easier to develop a PROLOG application on a commercial system that has input, output, and user interface capabilities. However, the SPIN program is ideal for deploying the speech translation for a noisy environment system since it will interface with a C++ speech recognition front end program and a speech synthesis output program.

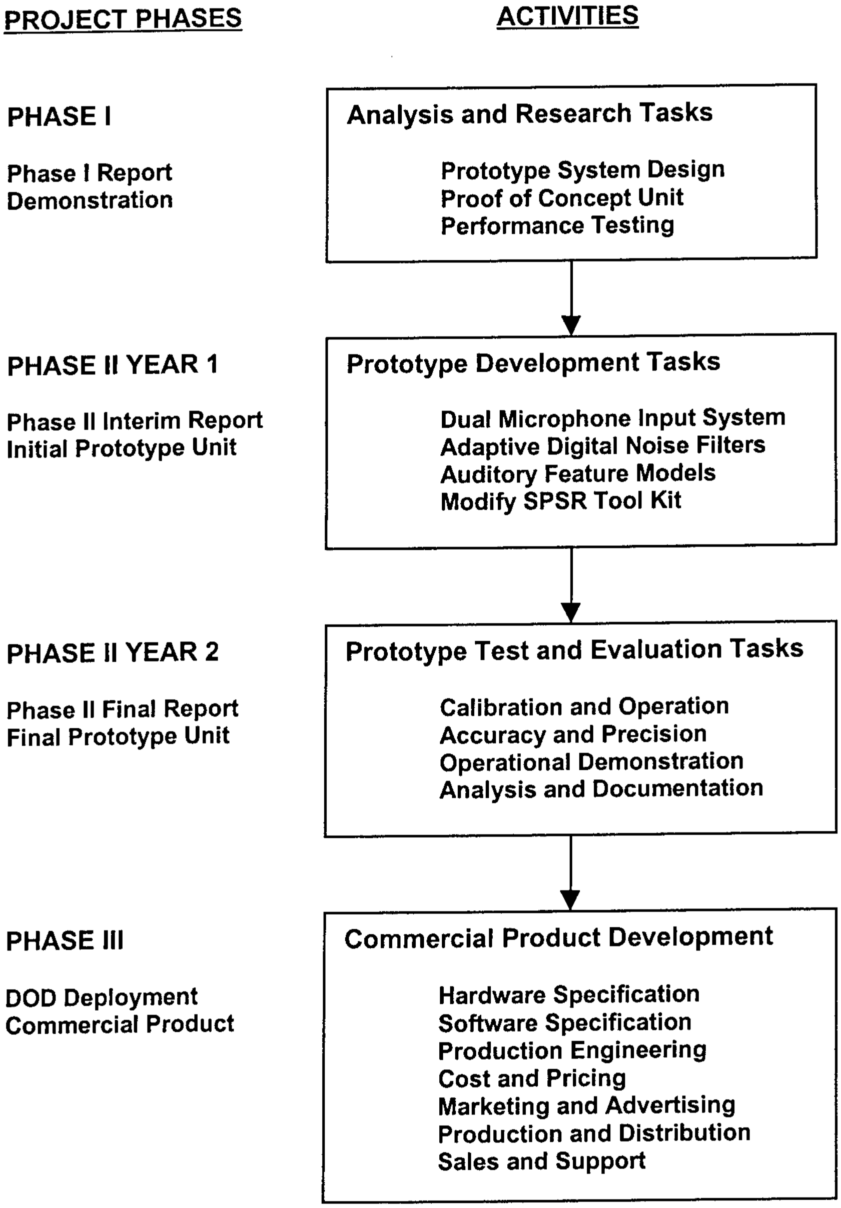
Figure 6.4-6 Example SPIN PROLOG Translation Program



```
INPUT FILE:testcfg
Initializing... please wait... Done.

% context free grammar

sent(P0,P):-np(P0,P1),vp(P1,P).
np(P0,P):-noun(P0,P).
vp(P0,P):-tran(P0,P1),np(P1,P).
vp(P0,P):-intran(P0,P).
noun(P0,P):-word(henry,P0,P).
noun(P0,P):-word(sidney,P0,P).
noun(P0,P):-word(programming,P0,P).
noun(P0,P):-word(accounting,P0,P).
tran(P0,P):-word(does,P0,P).
intran(P0,P):-word(works,P0,P).
% example sentence
word(henry,0,1).
word(works,1,2).
sent(0,2) ?
** YES
```

7.0 NOISY SPEECH RECOGNITION RESEARCH PLAN

The following research plan to evaluate the performance of the adaptive noise filter for noisy speech recognition corresponds to the three phase developments of the SBIR program. Phase I is a design and proof of concept demonstration of the feasibility of the proposed SOS technology. Phase II is a prototype development using the design in Phase I to create an engineering model for test and evaluation. Phase III is the development of a commercial product based on the engineering prototype.



PROJECT PHASES

ACTIVITIES

PHASE I

Phase I Report
Demonstration

Analysis and Research Tasks

Prototype System Design
Proof of Concept Unit
Performance Testing

PHASE II YEAR 1

Phase II Interim Report
Initial Prototype Unit

Prototype Development Tasks

Dual Microphone Input System
Adaptive Digital Noise Filters
Auditory Feature Models
Modify SPSR Tool Kit

PHASE II YEAR 2

Phase II Final Report
Final Prototype Unit

Prototype Test and Evaluation Tasks

Calibration and Operation
Accuracy and Precision
Operational Demonstration
Analysis and Documentation

PHASE III

DOD Deployment
Commercial Product

Commercial Product Development

Hardware Specification
Software Specification
Production Engineering
Cost and Pricing
Marketing and Advertising
Production and Distribution
Sales and Support

7.1 Phase I Design and Proof of Concept

The results of the Phase I design and proof of concept for using an adaptive digital filter in noisy speech recognition are as follows:

- Design and program a known noise model for speech recognition testing.
- Develop the adaptive digital filter for noisy speech signal inputs.
- Test the adaptive digital filter with the Nuance speech recognition system.
- Analyze the performance of the adaptive digital filter for speech recognition.
- Select noise-canceling microphones for the testing as Nuance system input.

7.2 Phase II Prototype Development

The anticipated tasks for the Phase II prototype developments for using an adaptive digital filter for noisy speech recognition are as follows:

- Design the end to end adaptive digital filtering for noisy speech recognition.
- Build a dual microphone and sound card PC hardware system.
- Program a baseline adaptive digital filter based on Phase I experiments.
- Test the adaptive filtering and dual microphone with speech recognition programs.
- Analyze and test the three auditory models explored during Phase I.
- Program the selected auditory models to define speech recognition features.
- Analyze the integration of these components with the SPSR Tool Kit.
- Modify the SPSR to use auditory model features for training and recognition.
- Integrate all components into a single test and evaluation system for AFRL.
- Define and conduct operational test and evaluation with USAF supplied data.

Single or Dual Microphone PC Sound Input Hardware and Software

Hardware

Dual Creative Labs Compatible 16 Bit Sound Cards
Signal Sampling Max 20,050 Hz for 8 Bits Data Per Card
Head Mounted Noise Microphone
Boom Mounted Speech Microphone
Optional Push To Talk (PTT) Speaker Button
Optional Digital Signal Processor (DSP eg TI320C30)

Software

Dual Sound Card Interface Device Driver
Dual Sound Stream Real Time Data Manager
Calibration Tone Generation and Synchronization
White and Pink Noise Test Generation

Three components were identified for Phase II development into the prototype system. The first component is the single or dual microphone hardware for a PC and software for the Windows SAPI speech recognition interface. The following figure illustrates the content of this component. This component design will be based on the proof of concept demonstration unit using commercial off the shelf microphones, sound cards, and system software interfaces. During Phase II, an integrated hardware design will be developed for a Phase III. This commercial product will use both off the shelf components and an SOS designed DSP-based add in PCI bus card to reduce the computational load and improve performance.

The second component to be developed in Phase II is the adaptive digital filter for noise cancellation in speech recognition systems. The design created during Phase I is a synthesis based on the filtering experiments performed for the proof of concept demonstration. Three functions were identified as shown in the figure below. The first function is to determine the state of the incoming signals as speech or silence, and if it is a speech signal, then to classify it as voiced or unvoiced. In the second function, two adaptive filters will be applied depending on the classification. A linear filter will be used for voiced speech and a nonlinear filter will be used for unvoiced speech. The third function is to either reconstruct a low noise speech signal for processing by commercial speech recognition programs or to transfer the filter data directly to a modified speech recognition system such as SPSR or some other usable product.

Adaptive Digital Filter for Speech Recognition Noise Cancellation

Speech Signal State Determination
- **Speech Activity Detection**
- **Voiced or Unvoiced Speech Classification**

Adaptive Noise Filters
- **Linear Voiced Speech Period Filter**
- **Nonlinear Unvoiced Speech Period Filter**

Speech Recognition Interface
- **Filtered Speech Sound Reconstruction**
- **Direct Data Transfer into Speech Recognizer**

The third component to be developed in Phase II is the computation of auditory feature data that is resistant to noisy speech. Three candidates were identified in Phase I and will be analyzed for potential performance gains as shown below. The selected models will be incorporated into the SOS SPSR Tool Kit for testing in the baseline system.

Auditory Feature Models for Speech Recognition

Auditory Physiology Simulation (APS)
- **Prototype and Test Computations**
- **Train SPSR Speech Recognition Process**
- **Test Speech Recognition Improvement**

Ensemble Interval Histogram (EIH)
- **Prototype and Test Computations**
- **Train SPSR Speech Recognition Process**
- **Test Speech Recognition Improvement**

Auditory Image Model (AIM)
- **Prototype and Test Computations**
- **Train SPSR Speech Recognition Process**
- **Test Speech Recognition Improvement**

Figure 7.2-1 illustrates the computations and data flow of the baseline system designed for Phase II prototype development. The system allows either one or two microphone speech input with or without push to talk signals. The three function adaptive digital filter designed in Phase I is implemented as separate modules that can be executed concurrently or in a DSP. The filtered signals are then processed either by commercial speech recognition systems or by the SPSR Tool Kit or other programmable recognizers.

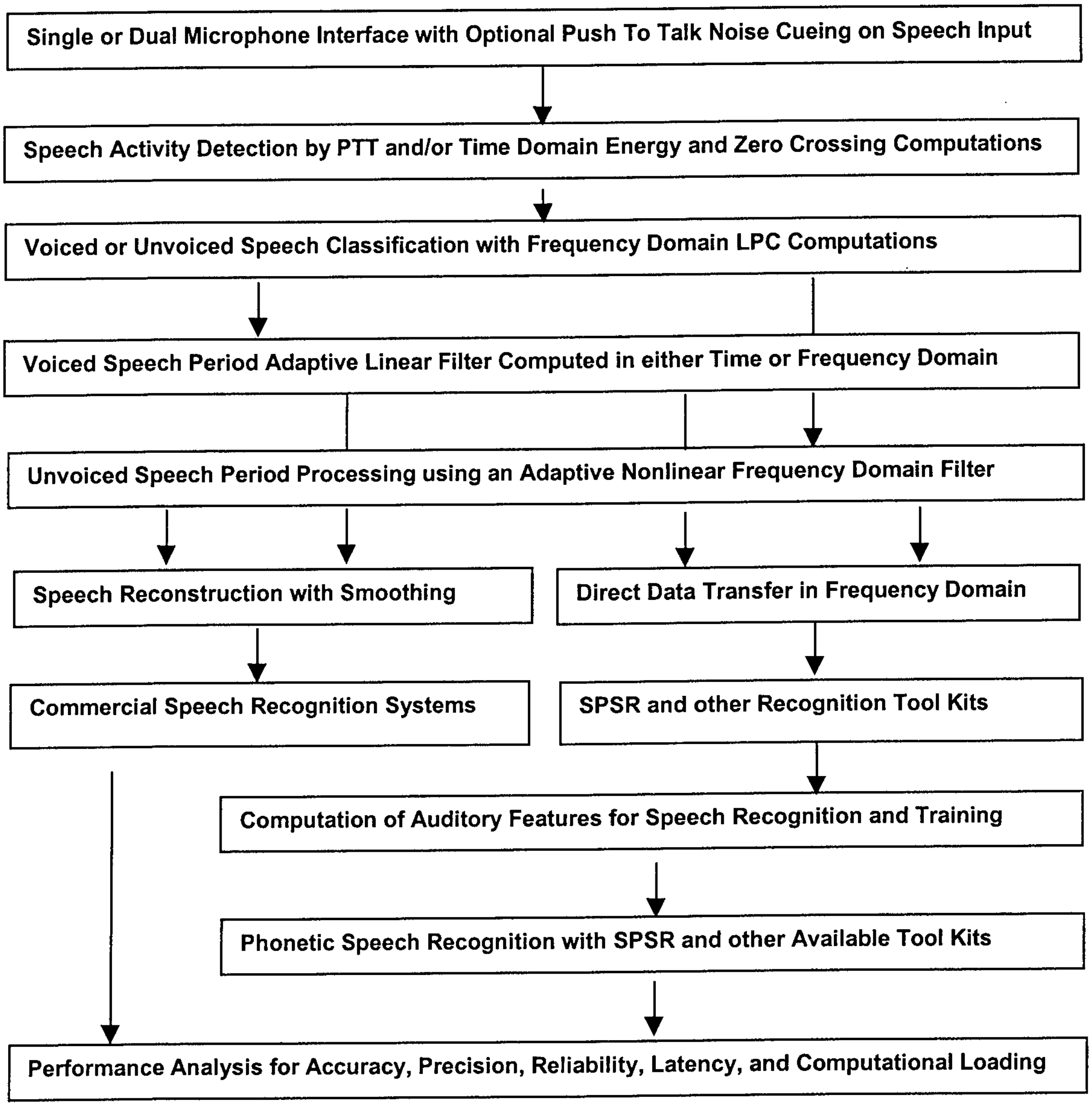

Single or Dual Microphone Interface with Optional Push To Talk Noise Cueing on Speech Input
Speech Activity Detection by PTT and/or Time Domain Energy and Zero Crossing Computations
Voiced or Unvoiced Speech Classification with Frequency Domain LPC Computations
Voiced Speech Period Adaptive Linear Filter Computed in either Time or Frequency Domain
Unvoiced Speech Period Processing using an Adaptive Nonlinear Frequency Domain Filter
Speech Reconstruction with Smoothing
Direct Data Transfer in Frequency Domain
Commercial Speech Recognition Systems
SPSR and other Recognition Tool Kits
Computation of Auditory Features for Speech Recognition and Training
Phonetic Speech Recognition with SPSR and other Available Tool Kits
Performance Analysis for Accuracy, Precision, Reliability, Latency, and Computational Loading

Figure 7.2-1 Data flow of Prototype System

7.3 PHASE III COMMERCIAL PRODUCT

The anticipated tasks for the Phase III commercial product development based on the Phase II prototype are given below. The following product configurations will be considered for production:

- Software only single microphone CD ROM for speech recognition software by mail order.
- Software only dual microphone system for higher performance by mail order.
- Dual microphone headset with sound card add in kit for retail sales.
- Dual microphone headset with high performance DSP card add in kit for retail sales.
- Custom systems for Medical, Legal, Industrial, and Government installations.

Each of these products will require the following tasks to bring it to market successfully. During Phase II these tasks will be defined in detail as part of the commercial product development anticipated in Phase III.

- Hardware Specification Task. The development of the detailed PC hardware component selection is a primary cost item in off the shelf systems. The major hardware items are microphones, sound cards, and digital signal processors.
- Software Specification Task. The selection of three types of software are critical to the success of this product; the operating system, such asWindows 95/98/NT/2000; the commercial speech recognizers such as Nuance, IBM, Microsoft, Dragon, Philips, etc.; and the tool kit software modified for the auditory features, such as SPSR, HTK, etc.
- Production Engineering Task. It is anticipated that SOS will perform the product engineering and subcontract the production of any hardware to competitive bidders including microphones, circuit boards, and complete systems.
- Cost and Pricing Task. Several models of this product will be created ranging from software only to full systems. The cost and pricing will include allowances for OEMs, distributors, mail order, and retail distribution.
- Marketing and Advertising Task. SOS will form cooperative relationships with speech recognition software companies, microphone manufacturers, and custom system developers to create marketing and advertising campaigns.
- Production and Distribution Task. SOS will minimize the in house production and distribution of any product beyond the software level.
- Sales and Support Task. SOS will provide telephone and internet direct sales and product support help.

8.0 OV10 3G NOISE REMOVAL TEST AND RESULTS

The OV10 is a noisy Vietnam era aircraft with two propeller engines used primarily for observation at low altitudes. WPAFB has instrumented one of these planes for noise data collection and research. Two CD ROMs were provided with OV10 digitized speech data and text transcriptions. The CDs contain eleven subjects recorded with various spoken utterances into a boom microphone. In addition the OV10 aircraft noise was recorded from a fixed cockpit location microphone. Five different environments were recorded for each speaker: laboratory, hanger, 1-G turn, 2-G turn, 3-G turn. Each environment has 50 digitized speech files and two text files. The noise removal and speech recognition tests were performed using the 3-G speech and cockpit noise data.

The configuration of the microphones in the OV10 is not arranged in accordance with the SOS design. The SOS design requires a fixed distance between the two microphones. The speech microphone is placed at the mouth to collect speech and environmental noise. The noise microphone is placed a fixed distance from the speech microphone, such that when the speaker turns his head or moves, that distance does not vary. In the OV10 recorded data the noise microphone located behind the speaker records the distinct propeller noise of the engines, but does not record any speech signal. Therefor the correlation between the signal received from the noise microphone and the ambient noise portion of the signal received from the speech microphone is very low.

For the SOS adaptive filter process to remove noise, the correlation between the speaker microphone noise signal and the noise microphone signal must be high. A correlation of 0.9 would yield a potential 10-dB reduction of the ambient noise in the speech signal while a value of 0.5 would yield a 3-dB reduction. A low correlation would in fact remove speech as well as noise resulting in a degradation of the recognition accuracy for the filtered speech data versus the unfiltered data. This is the phenomenon observed in this test of the OV10 data. To verify this result, the OV10 speech data was filtered with a series of bandpass filters tuned to the frequency range of speech. The hypothesis is that if the speech accuracy improves using non adaptive filters and decreases with adaptive filters, then the two signal inputs have uncorrelated noise signals.

SOS tested the adaptive digital filters on the 3G OV10 data, which had the lowest speech recognition accuracies. Matlab scripts were used to run the filters on the recorded data of the eleven speakers. Each original speech file was filtered with the corresponding background noise file to produce a set of filtered files. For example, a speech file, D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001.wav, was run through the different filters with the corresponding background file, D:\OV10-A Speech Database\Background\Sub 11 - RR\3g\001.wav. No data processing problems were encountered with the OV10 data.

From this the following five filtered files were produced by the SOS adaptive digital noise removal filters:

D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f1.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f2.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f3.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f5.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f6.wav

These were run through the Nuance speech recognition program, and the results were categorized by filter and speaker. In addition the 3G OV10 test data was also run through fixed non adaptive bandpass filters with different pass band frequency ranges. They were named according to the following naming conventions:

D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f7.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f8.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001f9.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001fa.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001fb.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001fc.wav
D:\OV10-A Speech Database\Speech\Sub 11 - RR\3g\001fd.wav

These were processed using Nuance's speech recognition program, and the results were categorized by filter and speaker. For each recognition instance, Nuance calculates a confidence rating. The sum of the confidences ratings is another indication for performance. The following tables summarize the OV10 test results beginning with the description of the five adaptive filters.

Filter Description:

NUM	NAME	L/NL	TD/FD	MIC	RECONSTRUCTION
1	Linear Adaptive Filter	L	FD	2	Triangular IFFT Coherent
2	Magnitude FFT	NL	FD	2	Original Phase + Mag
3	Log Magnitude FFT	NL	FD	2	Original Phase + Mag
4	LMS ALE	L	TD	1	None, Time Shifted Output
5	Mag FFT /Iter Recon	NL	FD	2	Phase Iteration
6	Single Mic Iter Recon	NL	FD	1	Noise Est by Scale Function
7	BandPass 100 - 4000Hz	L	TD	1	Minimum Ripple
8	BandPass 100 - 3200Hz	L	TD	1	Minimum Ripple
9	BandPass 200 - 4000Hz	L	TD	1	Minimum Ripple
A	BandPass 200 - 3200Hz	L	TD	1	Minimum Ripple
B	BandPass 100 - 2400Hz	L	TD	1	Minimum Ripple
C	BandPass 100 - 2800Hz	L	TD	1	Minimum Ripple
D	BandPass 200 - 2800Hz	L	TD	1	Minimum Ripple

This speech recognition results summary for unfiltered and filtered processing includes the average percent correct words recognized and the sum of the Nuance confidence scores. In all cases the adaptive filters (1 to 6) decreased the accuracy and confidence indicating that a portion of the speech signal was removed and a portion of the noise was not removed. The parameters were not able to be set reliably for Filter 4 and no results were obtained for this test. For the bandpass filters (7 to D) the accuracy improved in each case indicating that the low frequency and high frequency noise was affecting the speech recognition performance.

Results Summary:

Num	Filter Name	Average Percent Words Correct	Standard Deviation Taken Across Speakers	Confidence Score Sum	Unitized Confidence Score
	Original 3G OV10 Data	76.6%	±10.4%	18412	0.73
1	Linear Adaptive Filter	70.2%	±11.1%	12830	0.38
2	Magnitude FFT	67.5%	±7.2%	8004	0.07
3	Log Magnitude FFT	69.7%	±7.3%	9637	0.17
4	LMS ALE	N/A	N/A	N/A	N/A
5	Mag FFT /Iter Recon	62.0%	±8.9%	6957	0.00
6	Single Mic Iter Recon	70.2%	±6.6%	10591	0.23
7	BandPass 100 - 4000Hz	80.4%	±7.9%	21503	0.93
8	BandPass 100 - 3200Hz	80.7%	±8.0%	21603	0.94
9	BandPass 200 - 4000Hz	80.3%	±8.7%	21745	0.94
A	BandPass 200 - 3200Hz	80.9%	±8.6%	21789	0.95
B	BandPass 100 - 2400Hz	80.4%	±8.2%	21330	0.92
C	BandPass 100 - 2800Hz	80.9%	±8.1%	21952	0.96
D	BandPass 200 - 2800Hz	81.2%	±8.2%	22608	1.00

The following tables report the speech recognition accuracy of estimated words, total words, and percent correct. Each table is for a separate filter with all of the subjects listed from the OV10 data.

OV10 Original 3G Data:

Subject	Estimated	TOTAL	Percent Correct
1	143	207	69.1%
2	180	280	64.3%
3	216	280	77.1%
4	200	274	73.0%
5	205	280	73.2%
6	233	271	86.0%
7	184	275	66.9%
8	267	281	95.0%
9	247	280	88.2%
10	180	280	64.3%
11	234	280	83.6%
Average	2289	2988	76.6% (±10.4%)

Confidence Sum	**18412**

Linear Adaptive Filter
Filter 1 Data:

Subject	Estimated	Total	Percent Correct
1	133	207	64.3%
2	155	280	55.4%
3	211	280	75.4%
4	193	274	70.4%
5	196	280	70.0%
6	212	271	78.2%
7	163	275	59.3%
8	257	281	91.5%
9	219	280	78.2%
10	153	280	54.6%
11	206	280	73.6%
Average	2098	2988	70.2% (±11.1%)

Confidence Sum	**12830**

Magnitude FFT
Filter 2 Data:

Subject	Estimated	TOTAL	Percent Correct
1	140	207	67.6%
2	153	280	54.6%
3	199	280	71.1%
4	208	274	75.9%
5	180	280	64.3%
6	173	271	63.8%
7	187	275	68.0%
8	228	281	81.1%
9	181	280	64.6%
10	171	280	61.1%
11	198	280	70.7%
Average	2018	2988	67.5% (±7.2%)

Confidence Sum 8004

Log Magnitude FFT
Filter 3 Data:

Subject	Estimate	Total	Percent Correct
1	141	207	68.1%
2	155	280	55.4%
3	207	280	73.9%
4	216	274	78.8%
5	184	280	65.7%
6	179	271	66.1%
7	190	275	69.1%
8	227	281	80.8%
9	199	280	71.1%
10	175	280	62.5%
11	210	280	75.0%
Average	2083	2988	69.7% (±10.4%)

Confidence Sum 9637

LMS ALE
Filter 4 Data:

The parameters were not able to be set reliably for Filter 4 and no results were obtained for this test.

Mag FFT /Iter Recon
Filter 5 Data:

Subject	Estimated	Total	Percent Correct
1	118	207	57.0%
2	161	280	57.5%
3	167	280	59.6%
4	181	274	66.1%
5	156	280	55.7%
6	171	271	63.1%
7	150	275	54.5%
8	223	281	79.4%
9	196	280	70.0%
10	134	280	47.9%
11	196	280	70.0%
Average	1853	2988	62.0% (±8.9%)

Confidence Sum 6957

Single Mic Iter Recon
Filter 6 Data:

Subject	Estimated	Total	Percent Correct
1	148	207	71.5%
2	167	280	59.6%
3	197	280	70.4%
4	210	274	76.6%
5	177	280	63.2%
6	195	271	72.0%
7	189	275	68.7%
8	229	281	81.5%
9	204	280	72.9%
10	173	280	61.8%
11	209	280	74.6%
Average	2098	2988	70.2% (±6.6%)

Confidence Sum 10591

Band Pass Filter 100 - 4000Hz
Filter 7 Data:

Subject	Estimated	Total	Percent Correct
1	162	207	78.3%
2	204	280	72.9%
3	230	280	82.1%
4	223	274	81.4%
5	211	280	75.4%
6	228	271	84.1%
7	192	275	69.8%
8	268	281	95.4%
9	250	280	89.3%
10	196	280	70.0%
11	237	280	84.6%
Average	2401	2988	80.4% (±7.9%)

Confidence Sum 21503

Band Pass Filter 100 - 3200Hz
Filter 8 Data:

Subject	Estimated	Total	Percent Correct
1	162	207	78.3%
2	205	280	73.2%
3	230	280	82.1%
4	227	274	82.8%
5	216	280	77.1%
6	229	271	84.5%
7	192	275	69.8%
8	268	281	95.4%
9	250	280	89.3%
10	196	280	70.0%
11	236	280	84.3%
Average	2411	2988	80.7% (±8.0%)

Confidence Sum 21603

Band Pass Filter 200 - 4000Hz
Filter 9 Data:

Subject	Estimated	Total	Percent Correct
1	159	207	76.8%
2	194	280	69.3%
3	238	280	85.0%
4	225	274	82.1%
5	218	280	77.9%
6	225	271	83.0%
7	184	275	66.9%
8	266	281	94.7%
9	254	280	90.7%
10	200	280	71.4%
11	236	280	84.3%
Average	2399	2988	80.3% (±8.7%)

Confidence Sum 21745

Band Pass Filter 200 - 3200Hz
Filter A Data:

Subject	Estimated	Total	Percent Correct
1	158	207	76.3%
2	199	280	71.1%
3	238	280	85.0%
4	229	274	83.6%
5	219	280	78.2%
6	228	271	84.1%
7	184	275	66.9%
8	269	281	95.7%
9	254	280	90.7%
10	203	280	72.5%
11	235	280	83.9%
Average	2416	2988	80.9% (±8.6%)

Confidence Sum 21789

Band Pass Filter 100 - 2400Hz

Filter B Data:

Subject	Estimated	Total	Percent Correct
1	162	207	78.3%
2	203	280	72.5%
3	231	280	82.5%
4	222	274	81.0%
5	217	280	77.5%
6	230	271	84.9%
7	191	275	69.5%
8	268	281	95.4%
9	251	280	89.6%
10	193	280	68.9%
11	233	280	83.2%
Average	2401	2988	80.4% (±8.2%)

Confidence Sum 21330

Band Pass Filter 100 - 2800Hz

Filter C Data:

Subject	Estimated	Total	Percent Correct
1	162	207	78.3%
2	207	280	73.9%
3	231	280	82.5%
4	225	274	82.1%
5	215	280	76.8%
6	229	271	84.5%
7	189	275	68.7%
8	266	281	94.7%
9	253	280	90.4%
10	197	280	70.4%
11	243	280	86.8%
Average	2417	2988	80.9% (±8.1%)

Confidence Sum 21952

Band Pass Filter 200 - 2800Hz
Filter D Data:

Subject	Estimated	Total	Percent Correct
1	159	207	76.8%
2	205	280	73.2%
3	237	280	84.6%
4	229	274	83.6%
5	214	280	76.4%
6	234	271	86.3%
7	195	275	70.9%
8	268	281	95.4%
9	248	280	88.6%
10	195	280	69.6%
11	241	280	86.1%
Average	2425	2988	81.2% (±8.2%)

Confidence Sum **22608**

The speech recognition results from such a small sample of speakers and environments are statistically inconclusive. The filtered speech recognition rates were decreased as compared to unfiltered speech for the OV10 data by 7 to 9 percent. Several reasons exist for this anomaly. The correlation between the noise and speech signals was low. This is not the design environment for the adaptive filters. The pilot speech was clear with both low and high frequency noise. The Nuance speech recognizer is not designed to perform best in this situation. Further tests with other speech recognizers may provide different results. The SOS adaptive filters are not designed to remove the uncorrelated noise present in the OV10 data. The filters removed a portion of the speech signal while also removing the noise signal resulting in a speech recognition loss. It is not possible to modify the SOS designed adaptive filters for an uncorrelated noise environment that would result in improved performance of the Nuance speech recognition. The use of band pass linear filters in the time domain did improve the Nuance speech recognition accuracy. The most important result is that the it is necessary to use the SOS dual input fixed orientation head mounted microphones to collect correlated noise for successful adaptive filtering.

9.0 CONCLUSIONS

This Phase I research and development study has investigated the effect of creating an adaptive digital filter system to improve noisy speech recognition performance. The following five separate research tasks were successfully completed during the six-month research period.

1. Additive noise modeling and test data generation for adaptive filter development. Four noise models were created which degraded the speech recognition performance considerably. The TIMIT common sentences were used as test data with male and female speakers from all of the eight American dialect regions. This data provides a useful benchmark for evaluating noisy speech recognition.

2. The implementation and testing of several off the shelf commercial speech recognition systems for use with the test data in Task 1, and the use of a preliminary version SPSR system under development

by SOS for the navy as a Phase II SBIR project. Performance results were obtained for word recognition accuracy under identical test conditions for each system.

3. The design, programming, and testing of adaptive digital filter software for use with the speech recognition systems. Six variations of linear, nonlinear, time domain, frequency domain, with one or two microphones were developed. These were tested with all of the noisy speech test data and compared to create a baseline design for the Phase II system. A novel and unique three stage design was created that exploits the speech recognition aspect of the adaptive filter.

4. Development, test, and evaluation of a dual microphone and sound card PC hardware and system software unit for use in evaluating live and recorded noisy speech test data. The hardware and software engineering to create a PC based dual microphone and dual sound card test system. This system will evolve into the Phase II prototype unit and the Phase II commercial unit.

5. The investigation of auditory models to create noise resistant features for speech recognition software training and the modification of the SPSR tool kit to use this feature data for both training and recognition. Two existing scientific research auditory models, the AIM and EIH, were investigated for computational tractability and recognition feature computation. An additional model APS was developed by SOS as a new approach to auditory modeling specifically for noisy speech recognition. In addition the SPSR tool kit was investigated as the target speech recognition system to test these three feature models during Phase II.

6. A test of the SOS adaptive filters with USAF supplied OV10 data from two cockpit microphone recordings was performed. The noise data was found to be uncorrelated and not suitable for adaptive filtering. It was suitable for SOS bandpass filtering which improved the performance on the least accurate 3G data by over 25% using the Nuance speech recognizer.

The result of this Phase I research indicate that a successful Phase II prototype development can be accomplished at low risk and with a high potential value to the USAF and as a commercial product for the PC based speech recognition marketplace. SOS is fully prepared with the people and facilities to undertake this Phase II project within the SBIR schedule and budget.

REFERENCES

[1] "Speaker Dependent Speech Compression for Low Bandwidth Communication," Henry Pfister, 1996 IEEE Aerospace Applications Conference, Snowmass, 1996.

[2] "Fuzzy Logic in Speech Recognition for Japanese Hiragana," H.L. Pfister, FL 95, 1995.

[3] "State Recognition for Noisy Dynamic Systems," H.L. Pfister, Tech 2005, Chicago, 1995.

[4] "Experiences Using MODSIM and C++," H. L. Pfister, IC Mod & Sim Symp, 1992.

[5] "Object Oriented Design in C++," H. L. Pfister, THITI, Bankok, Thailand, 1991.

[6] "Object Oriented Simulation," H. L. Pfister, 1st Aerospace Conf on AI, Los Angeles, 1987.

[7] "Object Oriented Planning," H. L. Pfister, 2nd Aerospace Conf. on AI, Los Angeles, 1988.

[8] TIMIT CD-ROM, U.S. Department of Commerce, 1991.

[9] "From Text to Speech: The MITalk System," Allen, et al, Cambridge University Press, 1987.

[10] "Advanced Algorithms for Neural Networks," T. Masters, Wiley, 1995.

[11] "Noise Control for Engineers," H. Lord, W. Gately, & H. Evenson, McGraw-Hill, 1980.

[12] "Spoken Human-Machine Dialogue Workshop," ARO and TRADOC, May 30, 1995.

[13] "Adaptive Filter Theory - Third Edition," Simon Haykin, Prentice Hall, 1996.

[14] "Advanced Signal Processing and Digital Noise Reduction," S. Vaseghi, Wiley, 1996.

[15] "Introduction to Filter Theory," D. Johnson, Prentice-Hall, 1976.

[16] "Fundamentals of Speech Recognition," L. Rabiner and B Juang, Prentice-Hall, 1993.

[17] "Speech Recognition," C. Becchetti and L. Ricotti, Wiley, 1999.

[18] "Practical Genetic Algorithms," R. Haupt and S. Haupt, Wiley, 1998.

[19] "Fundamentals of Speech Synthesis and Speech Recognition," E. Keller, Wiley, 1994.

[20] "Discrete Time Processing of Speech Signals," J. Deller, J. Proakis, J. Hansen, MacMillan, 1993.

[21] "Electronic Speech Recognition," G. Bristow, McGraw Hill, 1986.