IMPROVING SPEECH RECOGNITION IN MULTIPLE SPEAKER ENVIRONMENTS USING A BSS ALGORITHM

BY KOSTAS ATHANASOGLOU

COMMITTEE:

Professor Vasilios Digalakis (Supervisor)

Assistant Professor Alexandros Potamianos

Professor Nikolaos Sidiropoulos

Submitted to the Department of Electronic and Computer Engineering in partial fulfillment of the requirements for the degree of Diploma in Engineering

Technical University of Crete

July 2007

By Kostas Athanasoglou: *Improving Speech Recognition in Multiple Speaker Environments Using a BSS Algorithm,* Submitted to the Department of Electronic and Computer Engineering in partial fulfillment of the requirements for the degree of Diploma in Engineering, © July 2007

ABSTRACT

We constructed a system, which can be used to evaluate the speech recognition performance of noise-removal algorithms, when the noise is generated from interfering speakers. We chose the widely used AURORA4 and TIMIT corpora as our data sets, so that the results obtained are credible and easily compared to related work. The system uses AURORA4 utterances as the clean signals, while noise can be selected from either AURORA4 or TIMIT.

Additionally, we used this system to prove the suitability of a blind source separation (BSS) algorithm in removing the noise of an interfering speaker, compared to the spectral subtraction (SS) method, which is shown not to be effective in removing this type of noise. In fact, results show that SS yields worse results than not applying any noise-removal algorithm in the frontend of the recognizer.

By applying the BSS algorithm we managed to drop the word error rate by 73.5% and after retraining the recognizer using similar accoustic models for the training and the test sets, the word error rate dropped by 78.44%.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [6]

ACKNOWLEDGMENTS

Above all, I would like to thank my parents for their lifelong support and the useful advice not only in my years in the university, but in my life in general. Without them, all these beautiful years learning and studying the field of Electronics and Computer Engineering would not be possible.

Special thanks to my supervisor Vasilis Digalakis for trusting me this work, for his valuable recommendations regarding the progress of the work and his useful guidance, whenever I had problems.

Big thanks to Kleanthis Mokios for our flawless cooperation and the useful insight he gave me regarding the operation of the BSS algorithm he developed during his Master Thesis.

Many thanks to all the professors of the Telecommunications Division for their support: Mr. Alexandros Potamianos for his help on speech recognition topics and for reviewing this work, Mr. Nikolaos Sidiropoulos for reviewing this work, Mr. Athanasios Liavas and Mr. George Karystinos for their help on signal processing topics.

Lastly but not least, I would like to thank Vlasis, Kostas and Nikos for their company, the enlightening conversations, the sharing of knowledge and information, the successful (and the not so successful) projects we attempted and above all, the fun we had all these years.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Speech recognition performance is greatly degraded when the conditions of the training and the test set differ widely. For example, a recognizer which is trained using sentences recorded in a noise-free environment, does not exhibit satisfying recognition rates when used in noisy environments. But even when the acoustics of the training and the test environments are similar, if the speakers used in each set have very different characteristics, the recognition rates will be poor.

To make things worse, systems trained with one speaker and tested with the same speaker and with the same noise conditions may not perform well when the speaker exhibits variabilities in his voice (e.g. the speaker has a cold which has affected his vocal tract).

To counteract all these problems, several methods have been devised. To make the acoustic model of the noisy test sets similar to the clean training sets, we try to model the properties of the noise and create systems that remove it optimally. Another approach is to create recognizers using data from many acoustic environments. To make the recognizers speaker-independent, we use speakers with very different characteristics in our training sets to create a *speaker-independent* recognizer.

An important property of the noise source is whether it has static or slow-varying statistical properties (e.g. white-noise, mumbles or traffic noise from cars, trains and airplanes) or rapid-varying statistical properties (e.g. human voice). We focus on cases where noise is of the second type.

A broad category of algorithms used to remove that type of noise are known as blind source separation algorithms. These algorithms require

no prior knowledge of the sources and try to separate them looking only at their mixtures. The only requirement of these algorithms is that they need as input at least as many mixtures as the number of sources that produced them.

There are various ways to evaluate the performance of such algorithms. According to [10] some are: speech recognition rates, plots of separated signals, plots of cascaded mixing/unmixing impulse responses and signal to noise ratios. As we're interested in speech recognition applications, we selected the speech recognition rates as a performance indicator.

To successfully test such an algorithm, an appropriate system must be setup which will contain the signal of each individual speaker, the conditions under which these signals will be mixed and any other interesting parameters, such as the modification of the relative energy between the two speakers.

In this work, we constructed such a system that can be used to evaluate the performance of these algorithms and compare them under various scenarios (e.g. using different positions for the speakers or the receiving microphones). The system can be used to simulate scenarios where several speakers talk simultaneously in a room, create the appropriate mixtures, as they would be recorded from actual microphones, feed them to such an algorithm and present the results to the user.

We selected our speech signals from the widely used AURORA4[9] and TIMIT[2] databases. AURORA4 was made by the DARPA[1] Spoken Language System (SRS) community. It was created by recording speakers reading articles from the Wall Street Journal. It consists of several test sets corresponding to different noise sources digitally added to the clean speech recordings. TIMIT, on the other hand, was designed to further acoustic-phonetic knowledge and automatic speech recognition systems. It was commissioned by DARPA and worked on by many

1 Defense Advanced Research Projects Agency

sites, including Texas Instruments (TI) and Massachusetts Institute of Technology (MIT), hence the corpus' name.

Utterances from the AURORA4 database were selected as the training set, while the test set contains utterances from the AURORA4 database, contaminated with noise either from the TIMIT or the AURORA4 database, according to the type of scenario we are using.

After creating this system, we used it to compare two different kinds of algorithms: the blind source separation (BSS) algorithm presented in [8] and the spectral subtraction (SS) method presented in [3]. The two algorithms differ greatly to the way they model the noise. The SS method assumes that noise has slow-varying statistical properties, which is not the case when noise comes from an interfering speaker.

As it was expected, the results show the suitability of the BSS algorithm for these types of environments. As an extension, we try to further improve the performance of the recognizer with the BSS algorithm, by retraining the recognizer to achieve similar acoustic models for the training and the test sets.

In Chapter 2 we present the necessary background material the reader must be familiar with to understand the concepts used later. An overview of our speech recognition system is presented and the BSS and SS methods are discussed briefly.

In Chapter 3 we discuss the implementation of our system and the actual implementation conditions used to compare the BSS and the SS methods.

In Chapter 4 we present analytic results containing word error rate graphs, as well as tables showing substitution, deletion and insertions errors.

In Chapter 5 we discuss the implications of the results, the suitability of the BSS algorithm in real-world applications and propose future work.

2

BACKGROUND MATERIAL

This chapter presents all the theoretical background used throughout the rest of the work. An overview of the system constructed is presented and the algorithms used (blind source separation and spectral subtraction) are discussed. The algorithms are presented briefly, but appropriate references have been placed, for the interested reader to study them in detail.

Both blind source separation and spectral subtraction techniques were used as noise removal algorithms and were implemented at the front-end of our recognition system. The familiar reader may note that an alternative technique for increasing recognition rates is using a training data set with similar acoustic conditions with the test set (same noise conditions). However, this technique requires either training by individuals in different environments, which is generally considered expensive or a variety of background sounds that could be added artificially, which may not be available in all situations.

2.1 SPEECH RECOGNITION BASICS

Speech recognition systems generally assume that the speech signal is a realization of some message encoded as a sequence of one or more symbols as shown in Figure 1 [11].

To effect the reverse operation of recognizing the underlying symbol sequence given a spoken utterance, the continuous speech waveform is first converted to a sequence of equally spaced discrete parameter vectors. This sequence of parameter vectors is assumed to form an

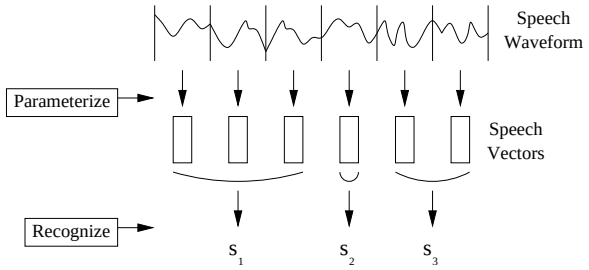

Figure 1. Message Encoding/Decoding.

exact representation of the speech waveform on the basis that for the duration covered by a single vector (typically 10ms or so), the speech waveform can be regarded as being stationary. Although this is not strictly true, it is a reasonable approximation. The role of the recognizer is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences.

Let each spoken word be represented by a sequence of speech vectors or observations $\mathbf{O}$, defined as:

$$\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T$$

where $\mathbf{o}_t$ is the speech vector observed at time t. The word recognition problem can then be regarded as that of computing

$$\arg\max_i \{P(w_i|\mathbf{O})\}$$

where w_i is the i'th vocabulary word. This probability is not computable directly but using Bayes' Rule gives

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})}$$

Thus, for a given set of prior probabilities $P(w_i)$, the most probable spoken word depends only on the likelihood $P(\mathbf{O}|w_i)$. Given the dimensionality of the observation sequence $\mathbf{O}$, the direct estimation

of the joint conditional probability $P(\mathbf{o}_1, \mathbf{o}_2, \ldots, |w_i)$ from examples of spoken words is not practicable. However, if a parametric model of word production such as a Markov model is assumed, then estimation from data is possible since the problem of estimating the class conditional observation densities $P(\mathbf{O}|w_i)$, is replaced by the much simpler problem of estimating the Markov model parameters.

In HMM based speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word is generated by a Markov model as shown in Figure 2. A Markov model is a finite state machine which changes state once every time unit and each time t that a state j is entered, a speech vector $\mathbf{o}_t$ is generated from the probability density $b_j(\mathbf{o}_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability α_{ij}. In HTK, the entry and exit states of a HMM are non-emitting. This is to facilitate the construction of composite models. In practice, only the observation sequence $\mathbf{O}$ is known and the underlying state sequence X is hidden. This is why it is called a *Hidden Markov Model*.

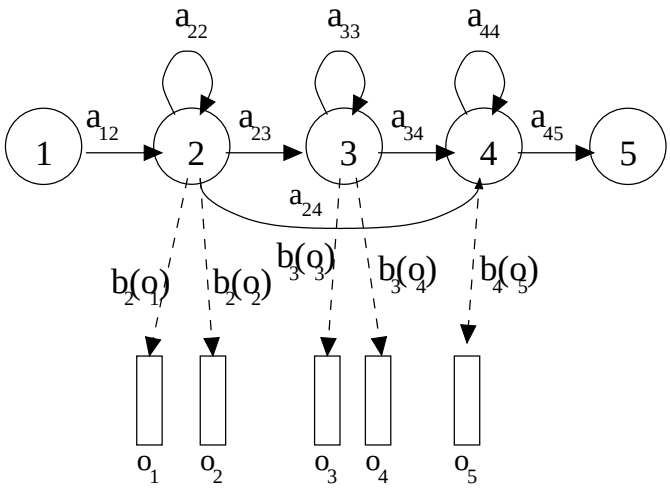

Figure 2. The Markov Generation Model.

In Figure 3 you can see a block diagram of the recognizer we are using. In the feature analysis step, a spectral analysis of the signal

using Mel Frequency Cepstral Coefficients (MFCCs) is performed and the vectors obtained are used to train our HMMs (each phoneme is modeled by one HMM). In the decoding phase, the recognized utterance is selected considering the Speech Recognition Units (which phonemes are allowed), the Word Dictionary (which words are allowed) and the Grammar (which sequences of words are allowed).

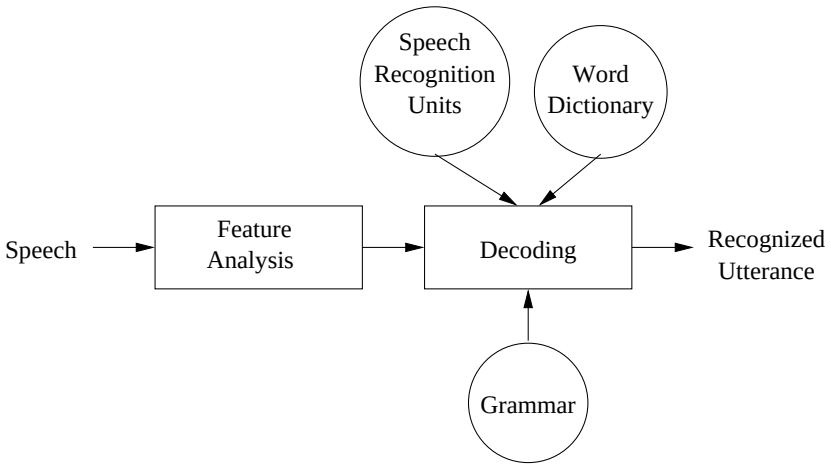

Figure 3. Block diagram of our speech recognizer.

2.2 THE BLIND SOURCE SEPARATION METHOD

Imagine multiple speakers speaking simultaneously in a room and several microphones, scattered in the room, recording them. Each microphone will obtain a mixture of the signals (that is, a linear combination of them). The blind source separation (BSS) method is a method that tries to obtain the initial signals of the speakers, looking only at the obtained mixtures, that is without having prior knowledge of each signal's properties (hence blind source).

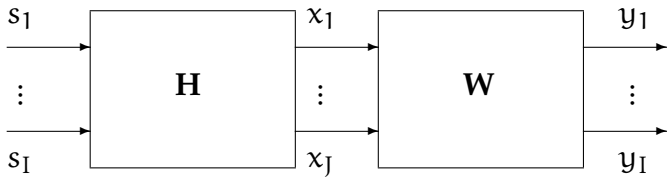

Figure 4. A model of the blind source separation problem.

2.2.1 *Noise-Removal formulated as a BSS Problem*

In Figure 4 you can see a model of the BSS problem. We assume I mutually uncorrelated speaker signals[1] $s_i(t)$, $i = 1, \ldots, I$ and $t = 1, \ldots, N$. There are J microphones that record these speakers. As all speakers talk simultaneously, each microphone is recording a mixture of them. If $\alpha_{ji}(t)$ is the impulse response with length L, of the path between the j-th microphone and the i-th speaker then each microphone signal will be given by:

$$x_j(t) = \sum_{i=1}^{I} \alpha_{ji}(t) \star s_i(t), \quad t = 1, \ldots, N \tag{2.1}$$

The case with two speakers and four microphones is shown in Figure 5.

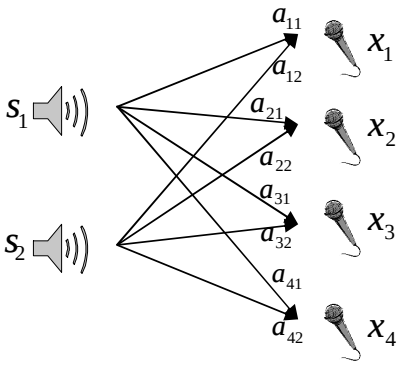

Figure 5. The case with two speakers and four microphones.

1 all signals are discrete-time ones.

Note that normally if we convolve two sequences with length L_1 and L_2 respectively, the output will be of length $L_1 + L_2 - 1$, but we truncate the output to N samples (same length as the input signals). We also don't consider added noise. When noise is not negligible, its power can be estimated from silence periods and subtracted from correlation matrix estimates.

Equation 2.1 can be written as:

$$\begin{bmatrix} x_1(t) \\ \vdots \\ x_J(t) \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{J1} \\ \vdots & \ddots & \vdots \\ \alpha_{1J} & \cdots & \alpha_{JI} \end{bmatrix} \star \begin{bmatrix} s_1(t) \\ \vdots \\ s_I(t) \end{bmatrix}$$

or in a more compact form:

$$\mathbf{x}(t) = \mathbf{A} \star \mathbf{s}(t) \tag{2.2}$$

where the $\star$ operation is computed the same way as the matrix multiplication operator, but instead of multiplication we use convolution. The above equation is called a multi-input, multi-output (MIMO) linear model for the received signals.

Theoretically, if $\boldsymbol{A}$ is invertible, by left multiplying both sides of 2.2 with $\boldsymbol{W} = \boldsymbol{A}^{-1}$ we can retrieve the original signals:

$$\mathbf{s}(t) = \boldsymbol{W} \star \mathbf{x}(t) \tag{2.3}$$

2.2.2 *Description of the BSS Algorithm Used*

Various approaches have been proposed to solve this problem, which can be divided into time-domain and frequency-domain methods. The algorithm we used [8], is a frequency-domain one. Frequency domain methods have the advantage over the time-domain ones that they are not too expensive to compute and the mixing channel order (L)

need not be known. The frequency-domain BSS methods decompose the time-domain convolutive BSS problem into multiple independent instantaneous BSS problems, one at each frequency bin. However, there is an inherent frequency-dependent permutation and scaling ambiguity problem in all frequency-domain BSS methods which does not exist in time-domain ones.

The algorithm we used is composed of two separate stages. In the first stage, parallel factor analysis (PARAFAC) [7] is employed in order to separate the speech signals, while in the second stage the task of matching the arbitrary permutations in the frequency domain is performed, via a novel integer-least-squares-based method.

To transform Equation 2.2 into the frequency-domain, we use the property of the DFT that allows us to express circular convolutions as products. In 2.2, we assume linear convolution. But a linear convolution can be approximated by a circular convolution if the size T of the DFT's frame is much larger than the length of the convolution sum. In such a case we can write:

$$\mathbf{x}(f,t) \approx \mathbf{A}(f)\mathbf{s}(f,t) + \mathbf{n}(f,t) \;\text{ for }\; L \ll T$$

where:

$$\mathbf{x}(f,t) = \sum_{\tau=0}^{T-1} \mathbf{x}(t+\tau)e^{\frac{-2\pi j f \tau}{T}}$$

is the DFT of the frame of size T starting at t.[2] The same holds for $\mathbf{s}(f,t)$. And finally,

$$\mathbf{A}(f) = \sum_{\tau=0}^{T-1} \mathbf{A}(\tau)e^{\frac{-2\pi j f \tau}{T}} = \sum_{\tau=0}^{L} \mathbf{A}(\tau)e^{\frac{-2\pi j f \tau}{T}}$$

because $\mathbf{A}(\tau) = \mathbf{0}^{J\times I}$ for $\tau > L$. The i-th column of $\mathbf{A}(f)$ represents now the spatial signature of the i-th speaker in the frequency domain, at frequency f.

2 For simplicity, we use the same symbol to denote time-domain and frequency-domain representation of a signal or a filter, depending on its argument.

The algorithm used, uses three assumptions:

1. The speaker signals $\mathbf{s}(t)$ are zero mean, second-order quasi-stationary (their variances are slowly varying, such that over short time intervals they can be assumed stationary). For speech signals, that is a reasonable assumption.

2. The number of speakers is known.

3. The contribution of the noise therm $\mathbf{n}(t)$ is negligible compared to the speaker signals.

Recall from 2.3 that we need the inverse matrix $\boldsymbol{W}(f) = \boldsymbol{A}^{-1}(f)$. There are two ways one can tackle the BSS problem in the frequency domain. One is to try to estimate the direct channel $\boldsymbol{A}(f)$ and then invert it to solve the problem, and the other one is to try to estimate the inverse channel $\boldsymbol{W}(f)$ directly. We used the direct estimation approach.

We can obtain estimates of $\boldsymbol{A}(f)$, from the autocorrelation of the received signals from the microphones:

$$\begin{aligned} \mathbf{R}_x(f,t) &= E[\mathbf{x}(f,t)\mathbf{x}^H(f,t)] \approx \boldsymbol{A}(f)E[\mathbf{s}(f,t)\mathbf{s}^H(f,t)]\boldsymbol{A}^H(f) \quad (2.4) \\ &= \boldsymbol{A}(f)\mathbf{D}_s(f,t)\boldsymbol{A}^H(f) \quad (2.5) \end{aligned}$$

Since we assume mutually uncorrelated speaker signals we postulate diagonal autocorrelation matrix $\mathbf{D}_s(f,t)$. As discussed in [8], proper processing of the autocorrelation data via PARAFAC analysis, for all frequency bins and all intervals over which the measured signals are assumed stationary, enables us to specify the matrices $\boldsymbol{A}(f)$, $f = 0, \ldots, T-1$.

We use a modified version of the algorithm, which assumes that all frequency ambiguity and scaling problems are solved perfectly.

2.3 THE SPECTRAL SUBTRACTION METHOD

Spectral subtraction is a method for removing noise from signals. The method estimates the noise spectrum in regions of the signal that are considered "noise-only" (eg. when the speaker has not started speaking yet) and removes it from the entire signal. It follows, that for this method to work properly, the noise must remain relatively constant during the whole speech activity.

2.3.1 *Noise-Removal formulated as a SS Problem*

In Figure 6 you can see schematically how spectral subtraction formulates the problem we need to solve. The algorithm takes as input the noisy signal $x(t)$ which is properly sampled and quantized. The noisy signal is considered the sum of the valuable signal $s(t)$ and the noise $n(t)$, so that:

$$x(t) = s(t) + n(t) \tag{2.6}$$

The method tries to solve the problem in the frequency domain by estimating the power spectral density of the noise and subtracting it from the signal. Equation 2.6 in the frequency domain becomes:

$$X(e^{-2\pi jk}) = S(e^{-2\pi jk}) + N(e^{-2\pi jk}) \tag{2.7}$$

Thus, by estimating $N(e^{-2\pi jk})$ we can subtract it from $X(e^{-2\pi jk})$ to get $X(e^{-2\pi jk})$ and after inverse DFT the original signal $x(t)$.

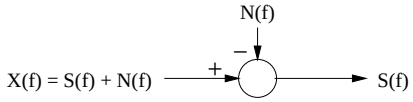

Figure 6. A model of the spectral subtraction problem.

2.3.2 *Description of the SS Algorithm Used*

The spectral subtraction implementation we used was the one in [3]. In Figure 7 you can see the steps of the algorithm in a block diagram form. Each step is explained in more detail in each own paragraph:

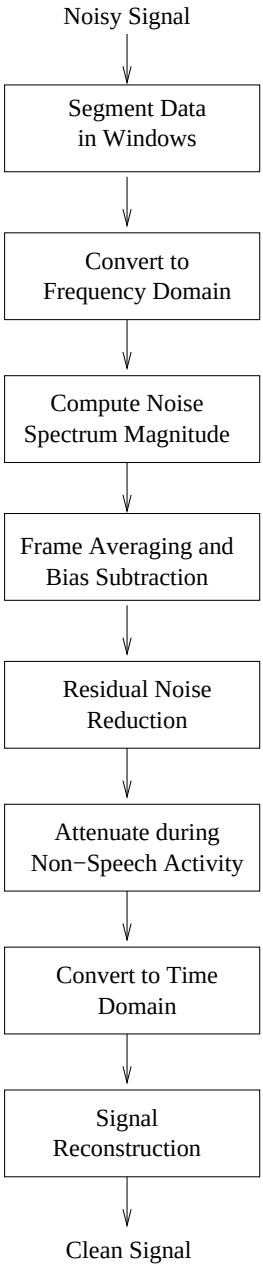

Figure 7. Block diagram of the spectral subtraction method.

NOISY SIGNAL This is the noisy signal as presented in Equation 2.6.

SEGMENT DATA IN WINDOWS Before we process the signal and take its Fourier transform, we must first segment it in small windows, where the speech signal can be considered stationary. The specific algorithm uses half-overlapped Hanning windows of 10 ms. With the sampling frequency of 16000 Hz that we used, each windows contains 160 samples.

CONVERT TO FREQUENCY DOMAIN After segmentation, we transform the signal into the frequency domain. For that, a 512-point Discrete Fourier Transform (DFT) was used. Thus, from each window a 512-point vector is obtained, which its two halfs are symmetrical because the signal is real valued.

COMPUTE NOISE SPECTRUM MAGNITUDE This is the step where we estimate the spectrum of a "noise-only" area. We select the first 50 ms (corresponding to 5 windows) of the signal as the noise-only area[3].

To compute the noise spectrum magnitude, we took the average of the DFTs of the first 5 windows (we consider only the first 256 points from here on, as the DFTs are symmetric). We denote this average noise spectrum as $\mu(e^{j2\pi k})$.

FRAME AVERAGING AND BIAS SUBTRACTION The spectral subtraction estimator $\hat{S}(e^{j2\pi k})$ is:

$$\hat{S}(e^{j2\pi k}) = X(e^{j2\pi k}) - \mu(e^{j2\pi k}) \tag{2.8}$$

3 All AURORA4 sentences were verified not to contain any speech activity in the first 50ms, so this assumption holds for our signals.

The error that results from this estimator is given by:

$$\epsilon(e^{2\pi jk}) = \hat{S}(e^{j2\pi k}) - S(e^{j2\pi k}) \tag{2.9}$$

and if we combine Equations 2.7 and 2.8, it becomes:

$$\epsilon(e^{2\pi jk}) = N(e^{j2\pi k}) - \mu(e^{j2\pi k}) \tag{2.10}$$

A technique to reduce the error in Equation 2.10 is used, which is called local averaging. Because $\epsilon(e^{2\pi jk})$ is the difference between $\mu(e^{j2\pi k})$, which is an average of 5 windows and $N(e^{j2\pi k})$ or $|N(e^{j2\pi k})|e^{j\theta_x}$ (we consider the phase of noise the same as the signal's phase, as these two signals have the same delay), it could be better if instead of $|N(e^{j2\pi k})|$ we used $|\overline{N(e^{j2\pi k})}|$. We just replace $|X(e^{j2\pi k})|$ with $|\overline{X(e^{j2\pi k})}|$:

$$|\overline{X(e^{j2\pi k})}| = \frac{1}{M}\sum_{i=0}^{M-1}\left|X_i(e^{2j\pi k})\right| \tag{2.11}$$

where $X_i(e^{2j\pi k})$ is the i-th time-windowed transform of $x(t)$. In our implementation, we used $M = 3$, because according to [3], averaging over more than three windows, will weaken intelligibility.

The new estimator will be:

$$\hat{S}_A(e^{j2\pi k}) = \left[|\overline{X(e^{j2\pi k})}| - \mu(e^{j2\pi k})\right] e^{j\theta_x}$$

and the new spectral error:

$$\epsilon(e^{2\pi jk}) = |\overline{N(e^{j2\pi k})}| - \mu(e^{j2\pi k})$$

Thus, the sample mean of $|N(e^{j2\pi k})|$ will converge to $\mu(e^{j2\pi k})$ as a longer average is taken.

RESIDUAL NOISE REDUCTION When there is no speech present in a given signal, the difference between $N(e^{j2\pi k})$ and $\mu(e^{j2\pi k})$ is called noise residual and will demonstrate itself as disorderly spaced narrow bands of magnitude spikes. When we transform the signal back into the time domain, these spikes will sound like the sum of tone generators with random frequencies. This is a phenomenon known as the "musical noise effect".

Because the spikes fluctuate from frame to frame, we can reduce the audible effects of the noise residual by replacing the current values from each frame with the minimum values chosen from the adjacent frames.

The motivation of the approach is threefold: If the amplitude of $\hat{S}(e^{j2\pi k})$ lies below the maximum noise residual, and it varies radically from frame to frame, there is a high probability that the spectrum at that frequency is due to noise; therefore, suppress it by taking the minimum value. Second, if $\hat{S}(e^{j2\pi k})$ lies below the maximum but has a nearly constant value, there is a high probability that the spectrum at that frequency is due to low energy speech, therefore taking the minimum will retain the information. Third, if $\hat{S}(e^{j2\pi k})$ is greater than the maximum, there is speech present at that frequency, therefore removing the bias is sufficient.

ATTENUATE DURING NON-SPEECH ACTIVITY The energy of $\hat{S}(e^{j2\pi k})$ compared to $\mu(e^{j2\pi k})$ is indicative of the presence of speech activity contained in a given analysis frame:

$$T = 20\log_{10}\left[\frac{1}{2N+1}\sum_{k=0}^{N}\left|\frac{\hat{S}(e^{j2\pi k})}{\mu(e^{j2\pi k})}\right|\right] \tag{2.12}$$

where $N = 160$ the size of each frame.

According to [4], if $T < 12$ dBs, we can classify this frame as without speech and attenuate it by a reasonable factor of 30 dBs (according

again to [4]). So, we must multiply each sample of the vector with a constant c, such as $20\log_{10} c = -30$ dB. That means $c = 10^{-3/2}$.

CONVERT TO TIME DOMAIN After the above processing and optimizations performed to each frame at the frequency domain, we can apply the inverse Fourier transform to each frame, so we can take back the original signal. The result will be a "clean" signal with the presence of the noise suppressed.

3

CREATING AN ENVIRONMENT TO TEST THE RECOGNIZER

Before we could embed either spectral subtraction or blind source separation methods in the front-end, we first had to properly setup a proper environment for the recognizer. That is, decide on what utterances the speakers will use (from which database), the room conditions where the mixing of the signals was done, the relative energies of the speaker signals, etc.

The two speech corpora we used was AURORA4 and TIMIT.

AURORA4 is a first general-purpose English, large vocabulary, natural language, high perplexity, corpus containing significant quantities of both speech data (400 hrs.) and text data (47M words), thereby providing a means to integrate speech recognition and natural language processing in application domains with high potential practical value. That is why we used it to train our recognizer and also the speakers we want to recognize read passages from the AURORA4 corpus.

The DARPA TIMIT speech database was designed to provide acoustic phonetic speech data for the development and evaluation of automatic speech recognition systems. It consists of utterances of 630 speakers that represent the major dialects of American English. Because TIMIT is a general text, phonetically balanced corpus, we used it as the "noise" database. That is, it was used for the interfering speakers that we can safely ignore (after separating their signals from the main speaker's, we can discard them).

To differentiate the case where we are only interested in recognizing the main speaker (discarding the rest of the speakers) from the case where we want to recognize all the speakers simultaneously, we created

two kinds of "scenarios".

The first one, is the *office* scenario. This scenario involves a speaker who dictates speech to his computer, while in the same environment there are interfering speakers. We are interested in recognizing only what the first speaker says. The other speakers are ignored; after their signals have been separated from the main speaker's, they are discarded. The main speaker uses utterances from the AURORA4 database, while the interfering speakers use utterances from the TIMIT corpus.

The second one, is the *conference* scenario. In this scenario several people talk simultaneously in a conference room and a transcriber tries to recognize them all at once. In this case, all speakers are considered as valuable signals and use the AURORA4 database.

In our experiments we used the *office* scenario, only.

3.1 CUSTOMIZING THE SPEECH DATABASES

The initial format of the databases (AURORA4 and TIMIT) we used to obtain the utterances, were not in the appropriate format. The files were saved in a big-endian format (Sun platforms) and the files did not have the appropriate WAV headers, so they could only play in a very limited number of players. Additionally, as our main work focuses on the blind source separation algorithm, according to [8], the algorithm shows better separability, when speech signals are of length greater than 30 seconds (instead of 7 seconds, which is the average length of an AURORA4 file).

As a measure indicator, we can use the Signal-to-Interference (SIR) ratio. SIR indicated how louder is the signal we are interested to separate, relative to all the other signals. If $P^{y}_{j,i} = \sum_{t=0}^{T-1} x_j^2(t)$, where T the number of samples, represents the power of the recorded signal at the j-th output of the algorithm when only source i is active, the SIR in dB

is defined as:

$$\mathrm{SIR}_y(j) = 10 \log \frac{\max_i P^y_{j,i}}{\sum_{i=1}^{I} P^y_{j,i} - \max_i P^y_{j,i}}$$

When the speech signals have a length between 22 and 35 seconds, the average SIR in output $\mathrm{SIR}_y = \frac{\sum_{j=1}^{J} \mathrm{SIR}_y(j)}{J}$ is approximately 21.5 dBs, while speech signals of small length (between 5 and 15 seconds), have an average SIR of 15.5 dBs. To exploit this difference of 6 dBs, we concatenated sentences of the same speakers together to create utterances with an average length greater than 30 seconds.

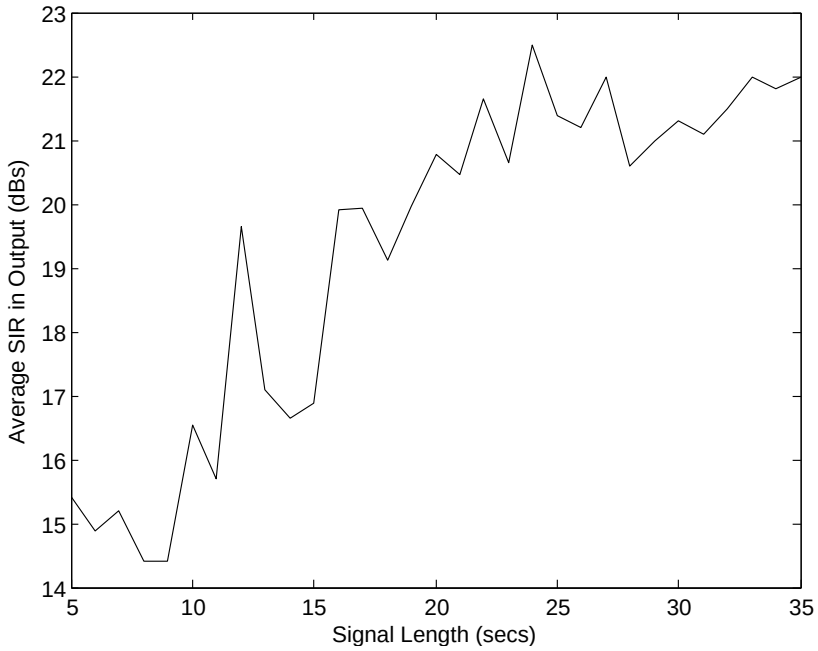

Figure 8. How signal length affects the average output SIR.

After concatenating the AURORA4 utterances, we had sentences with an average length of 35-40 seconds. The same thing had to be done in the TIMIT database. However, because the average length of a TIMIT utterance is 3 seconds, in many cases even after concatenating all the sentences of one speaker, the length was shorter than 35 seconds and the two speakers would not overlap each other during the whole time.

There were two options available. Either to let it happen (zero padding the second sentence) or to replicate the smaller sentence from samples from the beginning to make it overlap the main speaker for the whole time. We selected the second option.

Each source had to be normalized to fit in the range $[-1, 1]$, so it can be processed by MATLAB. Then we created the relative difference of our choice, between the energies of the two signals. Let's say we want the second source to be n (new ratio) dBs lower than the first one. We first compute the power of the signals:

$$P_x = \frac{1}{N}\sum_{i=1}^{N} x_i^2$$

where N the length of the signal. Accordingly, we compute P_y. The current ration in dBs between the two signals will be:

$$\left(\frac{Y}{X}\right)_{dB} = 10\log_{10}\frac{P_y}{P_x}$$

We want to change signal y so the new ratio will be n:

$$10\log_{10}\frac{P'_y}{P_x} = n \Leftrightarrow \frac{P'_y}{P_x} = 10^{n/10} \Leftrightarrow P'_y = 10^{n/10}P_x$$

So, we want to scale P_y be a factor $c^2 = \frac{P'_y}{P_y}$. So we must scale y with:

$$c = \sqrt{\frac{10^{n/10}P_x}{P_y}}$$

After all the initial setup was done, we could finally mix the two sources.

3.2 SIMULATING A ROOM AND MIXING THE SPEAKER SIGNALS

Because we were unable to perform the recordings ourselves in a real room, we simulated a room using Douglas Campbell's Roomsim [5]

program. In Figure 9 you can see a sketch of the room showing the dimensions of the room and the positions of the speakers and the microphones. The green-colored speaker is the primary speaker, while the red-colored speaker is the interfering one. The mixed file we kept was the one received by the first microphone (top left).

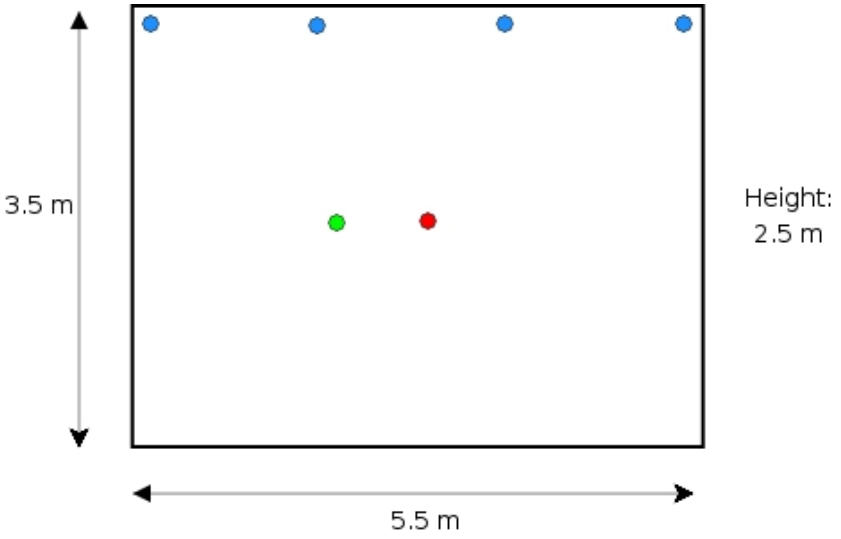

Figure 9. The simulated room, depicting the positions of the speakers and the microphones.

3.3 THE TRAINING PROCESS

The training process begins by parameterizing the raw speech waveforms into sequences of feature vectors. The sampling frequency of our speech databases was 16 KHz (that is a sampling period of 6.25 μs).

We also removed the DC mean from the source waveform. This was to ensure that any DC offset was removed from the signal. Note that this method is applied to each window individually so that it can be used both when reading from a file and when using direct audio input.

It is usually beneficial to taper the samples in each window so that discontinuities at the window edges are attenuated. We used a Ham-

ming windowing approach which applies the following transformation to the samples $\{s_n, n = 1, \ldots, N\}$:

$$s'_n = \left\{ 0.54 - 0.46 \cos\left(\frac{2\pi(n-1)}{N-1} \right) \right\} s_n$$

To compensate for the signal attenuation in speech signals caused by the lips, we amplified high frequencies by using a pre-emphasis filter:

$$s'_n = s_n - k s_{n-1}$$

with $k = 0.97$.

We used windows of 25 ms and a frame period between the windows 10 ms so there is a window overlap to extract the feature vectors. Figure 10 shows how the windows are deployed in time.

First filterbank analysis is done to the signal using 26 channels and then 12 Mel Frequency Cepstral Coefficients are extracted (MFCCs) from each window according to:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N} (j - 0.5) \right)$$

where N = 26, the number of filterbank channels. To augment the spectral parameters we appended the zeroth cepstral parameter C_0.

We also added time derivatives of the basic static parameters, to enhance the performance of the recognizer (see [11], pages 63-64). We added first order regression coefficients (delta coefficients) and second order regression coefficients (acceleration coefficients).

MFCCs are the parameterization of choice for many speech recognition applications. They give good discrimination and lend themselves to a number of manipulations. In particular, the effect of inserting a transmission channel on the input speech is to multiply the speech spectrum by the channel transfer function. In the log cepstral domain, this multiplication becomes a simple addition which can be removed

by subtracting the cepstral mean from all input vectors. In practice, of course, the mean has to be estimated over a limited amount of speech data so the subtraction will not be perfect. Nevertheless, this simple technique is very effective in practice where it compensates for long-term spectral effects such as those caused by different microphones and audio channels. We used this so called *Cepstral Mean Normalization* (CMN) (see [11], pages 61-62) in our experiments.

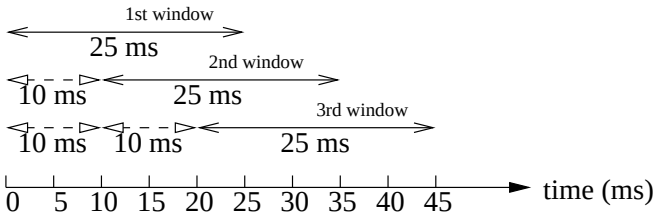

Figure 10. Windows used for feature extraction. There is an overlap between adjacent windows. Window size is 25 ms and frame period (distance between adjacent windows) is 10 ms.

The Hidden Markov Model used for each phoneme was a left-right, with three states. HTK also adds a non-emitting starting and end state, so a model of an HMM was similar to the one found in Figure 2.

Because in our speech databases we did not have speech data available for which the location of the sub-word (i.e. phone) boundaries were marked, we used a so-called *flat start*. In this case, all of the phone models are initialized to be identical and have state means and variances equal to the global speech mean and variance.

Once the initial set of models had been computed, we performed *embedded training* using the entire training set. For each training utterance, the corresponding phone models are concatenated and then the forward-backward algorithm is used to accumulate the statistics of state occupation, means, variances, etc., for each HMM in the sequence. When all of the training data has been processed, the accumulated statistics are used to compute re-estimates of the HMM parameters.

We used the philosophy of system construction in HTK, which is that

HMMs should be refined incrementally. We started with a simple set of single Gaussian context-independent phone models and then iteratively refined them by expanding them to include context-dependency and use multiple component Gaussian-mixture distribution (we used 6). The formula for computing the output distributions $b_j(\mathbf{o}_t)$ is:

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{6} c_{jm} \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

where c_{jm} the weight of the m'th component and $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, that is:

$$\mathcal{N}(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{o}-\boldsymbol{\mu})}$$

where $n = 39$ (13 coefficients along with their first and second derivatives) is the dimensionality of $\mathbf{o}$.

When building context-dependent HMM systems, there is always a problem of data insufficiency. To overcome this problem, we tied parameters together which allows data to be pooled so that the shared parameters can be robustly estimated.

As the focus of this paper is on noise robustness, we consider only a subset of the Aurora4 task. This subset corresponds to training and testing on the recordings from the Sennheiser microphone at 16 kHz and processed by a P.341 filter [1]. The use of the P.341 filter simulates the transmission characteristics for wideband telephony. In particular, 7138 utterances from 83 speakers in the *training clean sennh* set are used for training the acoustic model.

3.4 THE TESTING PROCESS

In HTK an alternative formulation of the Viterbi algorithm is used called the *Token Passing Model*. In brief, the token passing model makes the concept of a state alignment path explicit. Imagine each state j of a

HMM at time t holds a single movable token which contains, amongst other information, the partial log probability:

$$\psi_j(t) = \max_i \left\{ \psi_i(t-1) + \log(a_{ij}) \right\} + \log(b_j(\mathbf{o}_t))$$

This token then represents a partial match between the observation sequence $\mathbf{o}_1$ to $\mathbf{o}_t$ and the model subject to the constraint that the model is in state j at time t. The key steps in the equivalent token passing algorithm which is executed at each time frame t is:

1. Pass a copy of every token in state i to all connecting states j, incrementing the log probability of the copy by $\log[\alpha_{ij}] + \log[b_j(\mathbf{o}_t)]$.

2. Examine the tokens in every state and discard all but the token with the highest probability.

The point of using the Token Passing Model is that it extends very simply to the continuous speech case. For example, Figure 11 shows a simple network in which each word is defined as a sequence of phoneme-based HMMs and all of the words are placed in a loop. In this network, the oval boxed denote HMM instances and the square boxed denote *word-end* nodes. This composite network is essentially just a single large HMM and the above Token Passing algorithm applies. The only difference now is that more information is needed beyond the log probability of the best token. When the best token reaches the end of the speech, the route it took through the network must be known in order to recover the recognized sequence of models.

The history of a token's route through the network may be recorded efficiently as follows. Every token carries a pointer called a *word end link*. When a token is propagated from the exit state of a word (indicated by passing through a word-end node) to the entry state of another, that transition represents a potential word boundary. Hence a record called a *Word Link Record* is generated in which is stored the identity of the word from which the token has just emerged and the current value of

the token's link. The token's actual link is then replaced by a pointer to the newly created WLR.

Once all of the unknown speech has been processed, the WLRs attached to the link of the best matching token (i.e. the token with the highest log probability) can be traced back to give the best matching sequence of words. At the same time the positions of the word boundaries can also be extracted if required.

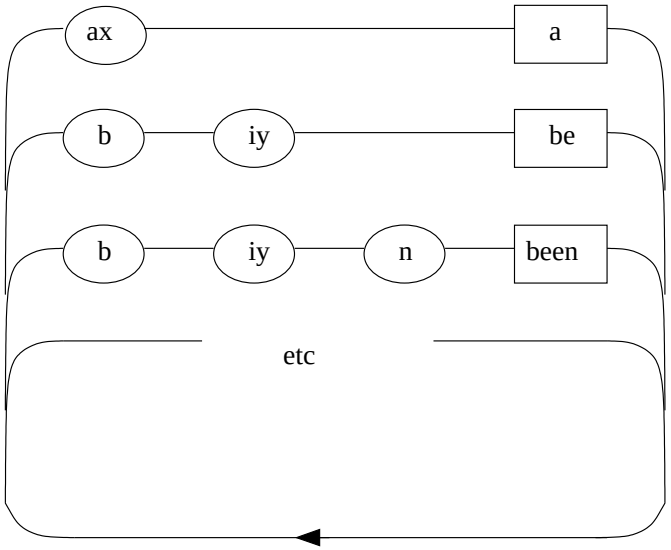

Figure 11. A Recognition Network.

There were two test sets available in the AURORA4 database. One with 160 and one with 330 utterances. We selected the larger test set for greater credibility.

4

RESULTS

4.1 PERFORMANCE ON CLEAN DATA

Before we can apply any noise-removal algorithms in the test sets, we first had to evaluate the performance of the speech recognizer when both the training and the test sets contain clean data; that is data not contaminated with any kind of noise. Then, by looking at these baseline results we can evaluate the performance of either noise-removal algorithm.

Decoding was performed on a test set containing 330 utterances of the AURORA4 database. The data used were sampled at 16 KHz and used 16 bits per sample.

In Table 1 you can see the recognition results on the clean test set. The first column indicates the percentage of words correctly recognized, columns 2-3 indicate false substitutions (SUB), false deletions (DEL) and false insertions (INS) of the recognizer. The percent of correct words recognized (CORR) does not consider the false insertions. That is why we do not use it, as it is not indicative of the overall error of the recognizer. Alternatively, we use the word error rate (ERR) which is simply the sum of false insertions, deletions and substitutions. As you can see from Table 1, the word error rate (WER) on clean data is 11.13%.

As expected, the results yield a very good recognition rate as the conditions of the training and the test sets are similar.

CORR	SUB	DEL	INS	ERR
91.84	7.42	0.75	2.97	11.13

Table 1. Recognition results on clean data

4.2 RESULTS AFTER APPLYING SS

Before we could test the spectral subtraction algorithm, we first had to artificially add noise to the test set from interfering speakers. The process is the one explained in 3.2. We used several relative energies between the two speakers. The second speaker (considered as the noise) was initially at the same energy level as the main speaker (0 dBs difference between them). Then, we were gradually lowering the energy of the second speaker by 5 dBs each time.

In Figure 12 we illustrate the effects of applying spectral subtraction in the front-end. In Table 2 you can see the recognition results on the mixed data for the various dB levels, while in Table 3 you can see each case after applying spectral subtraction in the front-end.

As you can see, spectral subtraction fails to separate the signals of the two speakers due to its assumption that the noise emanating from the second source has stationary statistical properties. Not only the performance of the recognizer remains the same, compared to the mixed signals case, but it is also degraded because of the spectral corruptions the signal suffered. The word error rate increases from 1.55 and 3.34% when the two speakers speak with relatively equal energy (0 and 5 dBs difference respectively), to 18% when the unwanted speaker virtually doesn't exist (25 dBs lower than the first one).

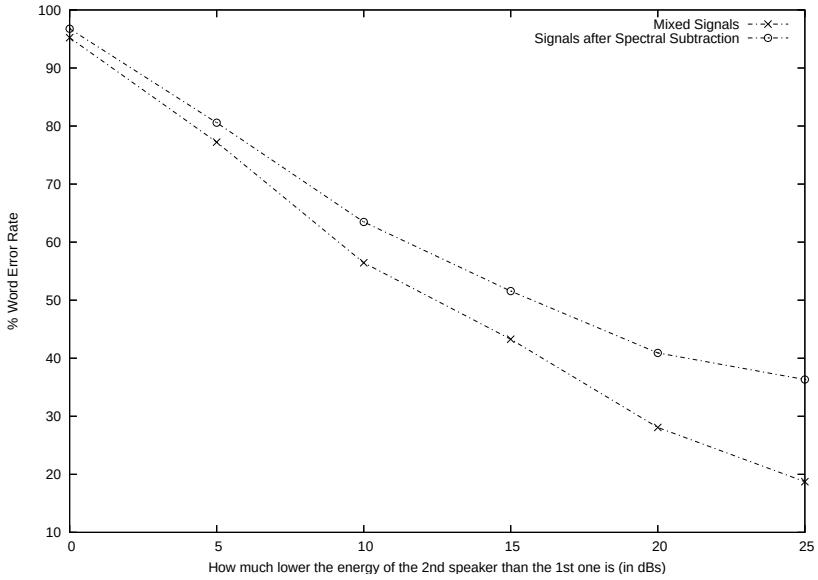

Figure 12. Results with and without applying spectral subtraction (SS) in the frontend.

DBS	CORR	SUB	DEL	INS	ERR
0	33.72	56.64	9.64	28.94	95.22
-5	53.37	39.47	7.15	30.60	77.23
-10	71.31	24.90	3.79	27.76	56.45
-15	80.96	16.50	2.54	24.25	43.28
-20	86.76	11.83	1.42	14.83	28.08
-25	89.15	9.77	1.08	7.88	18.74

Table 2. Recognition results in mixed data

DBS	CORR	SUB	DEL	INS	ERR
0	26.51	61.11	12.39	23.28	96.77
-5	44.52	46.14	9.34	25.09	80.57
-10	60.55	33.31	6.15	24.02	63.48
-15	69.77	25.63	4.60	21.33	51.56
-20	75.85	20.44	3.72	16.76	40.91
-25	78.03	18.76	3.21	14.37	36.33

Table 3. Recognition results after spectral subtraction

4.3 RESULTS AFTER APPLYING BSS

The same process was used to test the blind source separation algorithm. In Figure 13, we illustrate the benefits of the blind source separation method and we compare it to spectral subtraction. In Table 4 you can see the results in more detail. Blind source separation reduces the word error rate significantly by 73.5% when the two speakers speak at the same energy level, compared to the mixed signals case. The improvement of BSS starts to degrade, as we lower the energy of the second speaker. But even when the unwanted speaker is 5 and 10 dBs lower than the primary one, BSS still improves the recognition by 49.71 and 14.14% respectively. When we further lower the energy of the second speaker, the algorithm exhibits an adverse behavior and degrades the performance of the recognizer.

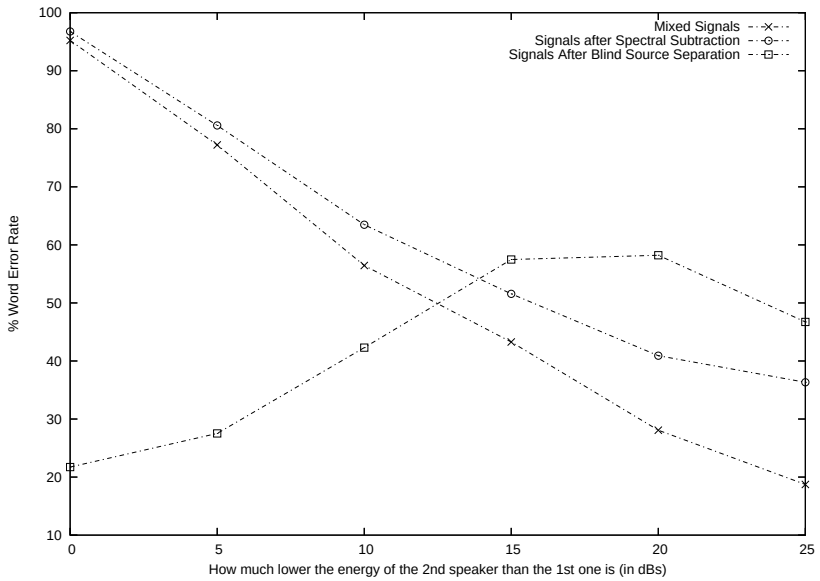

Figure 13. Results with and without applying blind source separation (BSS) in the frontend and comparing it to spectral subtraction.

DBS	CORR	SUB	DEL	INS	ERR
0	84.71	13.64	1.66	6.43	21.72
-5	80.95	16.70	2.35	8.46	27.52
-10	69.25	25.44	5.31	11.56	42.31
-15	54.36	35.01	10.63	11.84	57.48
-20	52.70	36.05	11.25	10.91	58.21
-25	61.26	29.76	8.99	7.98	46.72

Table 4. Recognition results after blind source separation

4.4 RETRAINING THE RECOGNIZER

To further improve the recognition rate, we can retrain the recognizer using similar acoustic environment as the one used in the test set, that is signals that have been separated using the blind source separation algorithm.

As a baseline scenario, we can create a training and a test set that contain convolutive mixtures of the two speakers. Then we can create a training and a test set with convolutive mixtures that have been separated using the BSS algorithm and compare the results to the baseline scenario. That way, we'll examine how the retraining procedure affects the results and how the BSS algorithm improves them.

As explained in 3.1, when creating mixtures we can choose the relative energy of the two speakers. So, one option is to use the same energy difference in the training and the test set and another one is to create a "general" training set, which contains mixtures where the relative energy between the two speakers varies. In Figure 14 you can see the results using either of the two options. Detailed results are given in Tables 5 and 6 respectively.

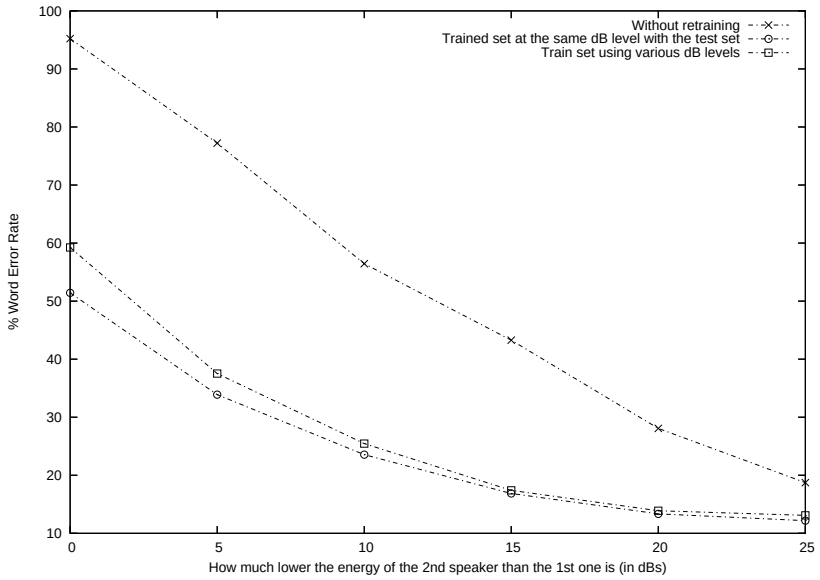

Figure 14. Results after retraining the recognizer using mixed data.

DBS	CORR	SUB	DEL	INS	ERR
0	60.00	31.05	8.95	11.43	51.43
-5	76.44	19.37	4.18	10.33	33.89
-10	84.72	12.61	2.67	8.28	23.56
-15	88.32	9.51	2.17	5.19	16.87
-20	90.34	8.43	1.23	3.70	13.36
-25	90.96	7.92	1.12	3.14	12.18

Table 5. Recognition results after retraining at the same dB levels

4.4.1 *Retraining Using BSS Data*

Now a similar process can be done by separating the convolutive mixtures using the BSS algorithm. Again, we have the two options described in Section 4.4. In Figure 15 you can see the performance of the recognizer when using the BSS algorithm. Detailed results are presented in Tables 7 and 8.

DBS	CORR	SUB	DEL	INS	ERR
0	60.99	32.24	6.76	20.23	59.24
-5	76.11	19.24	4.65	13.62	37.51
-10	83.19	13.60	3.21	8.65	25.46
-15	87.13	10.45	2.41	4.55	17.41
-20	88.42	9.55	2.04	2.30	13.88
-25	88.92	9.23	1.85	2.02	13.10

Table 6. Recognition results after retraining using various dB levels

DBS	CORR	SUB	DEL	INS	ERR
0	88.66	10.18	1.16	3.36	14.70
-5	86.79	11.40	1.81	3.53	16.74
-10	83.86	13.69	2.45	3.81	19.95
-15	79.25	16.87	3.89	6.65	27.41
-20	80.42	15.88	3.70	4.32	23.89
-25	84.20	13.08	2.73	3.51	19.32

Table 7. Recognition results after BSS retraining at the same dB levels

DBS	CORR	SUB	DEL	INS	ERR
0	86.72	11.71	1.57	3.49	16.78
-5	85.50	12.67	1.83	4.04	18.53
-10	82.59	14.78	2.63	5.44	22.85
-15	78.63	17.15	4.22	6.84	28.21
-20	77.71	17.71	4.58	4.89	27.18
-25	80.76	15.26	3.98	3.19	22.44

Table 8. Recognition results after BSS retraining using various dB levels

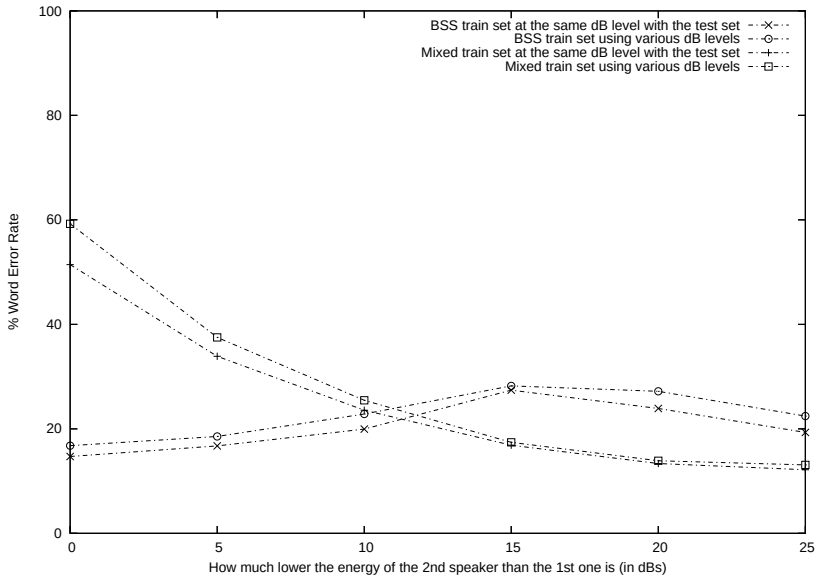

Figure 15. Results after retraining the recognizer using BSS data.

4.5 IMPLEMENTING A REAL RECOGNIZER

To successfully implement the BSS algorithm in a real recognizer, we must choose these options that give the best recognition rates, and also can be implemented in a real recognizer.

The recognizers that are trained at the same noise levels as the test sets cannot be used, because we do not know the relative energies of the two speakers in a real scenario. Looking at Figure 15, you can see that when the energy of the second speaker is equal or lower than 10 dBs from the first, the recognizer trained using BSS data from various dB levels is the best selection in a real scenario. In contrast, when the difference of the two speakers is greater than 10 dBs, the recognizer trained using mixed data at various levels is the best choice in a real scenario.

Ideally, if we knew this difference we could select the appropriate recognizer from the two and achieve the recognition results shown in Figure 16.

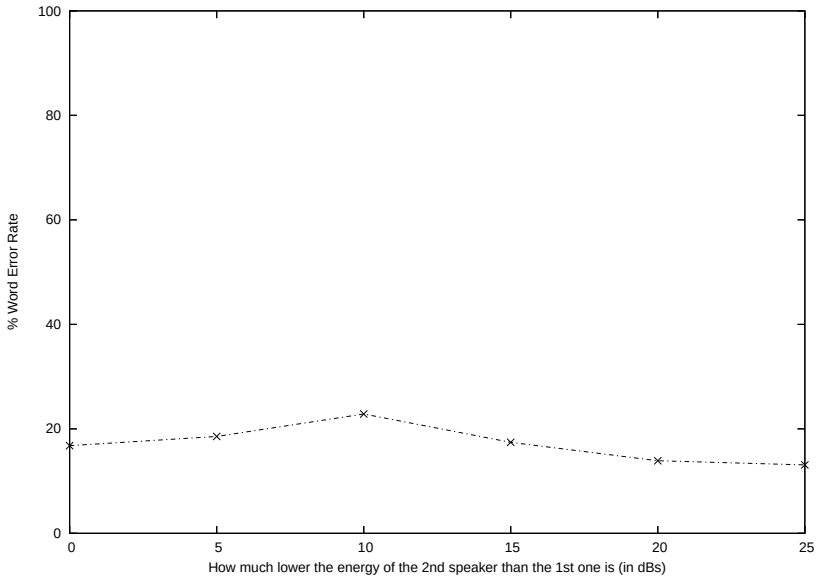

Figure 16. The ideal recognizer (considering we know the energies of the two signals).

Because we do not know this difference in a real scenario, we tested some methods that can be used to make the best decision between the two recognizers and we present them below.

4.5.1 *Choosing Recognizer Using Energy Difference of Separated Signals*

One possible option is to get the mixture of the signals, separate each source and then calculate the ratio between the separated sources to decide. Unfortunately, the BSS algorithm amplifies the sources, so the difference between them is reduced greatly. In Table 9, we present three signals we created, from each noise level[1] before and after separation. We can see, that even when the second signal is 25 dBs lower than the first one before separation, after the separation process this difference drops to 3-4 dBs, which is the same difference when the initial noise level was -5 dBs, so a separation of the cases by this method is not

1 We define noise level to be the difference in dBs between the energy of the first speech signal, relative to the second.

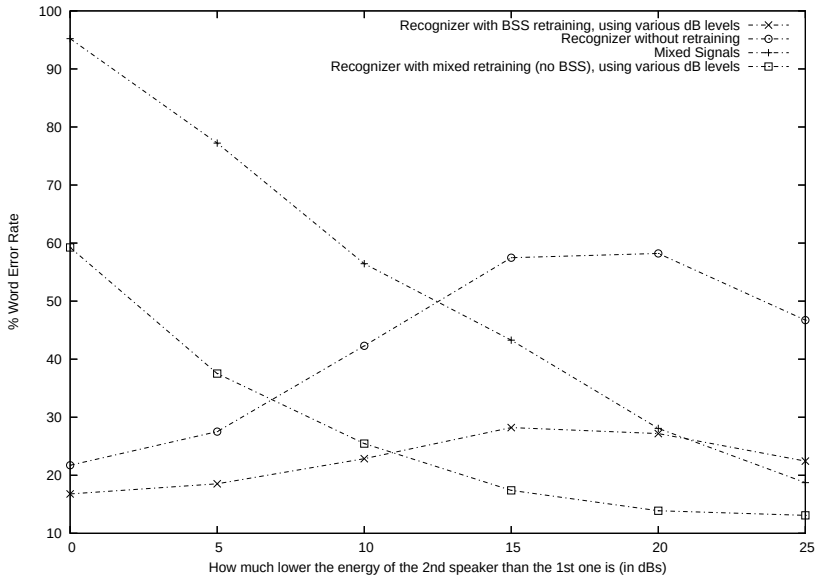

Figure 17. Comparing the available options for our final recognizer.

possible.

4.5.2 *Choosing Recognizer Using Forced Alignments*

An alternative option is to use both recognizers for each sentence and get a hypothesis from each recognizer (see Figure 18), where hypothesis is the most likely word string for a spoken utterance. Then we can force align (see [11], pages 186-187) hypothesis 2 with the BSS recognizer and hypothesis 1 with the mixed recognizer and see with what probability each recognizer would give the output of the other recognizer. From these, we can compute an estimate of confidence for each sentence from each recognizer:

$$\frac{p(\mathrm{hyp}_1|\mathrm{BSS})}{p(\mathrm{hyp}_2|\mathrm{BSS})} \lessgtr \frac{p(\mathrm{hyp}_2|\mathrm{mixed})}{p(\mathrm{hyp}_1|\mathrm{mixed})}$$

The output of the recognizer with the greatest confidence will be used as our final output.

BEFORE	AFTER
0	1.71
0	0.97
0	2.12
-5	-2.70
-5	-3.76
-5	-2.61
-10	-5.78
-10	-7.62
-10	-5.61
-15	-4.95
-15	-6.77
-15	-4.05
-20	-3.59
-20	-3.95
-20	-4.02
-25	-4.23
-25	-3.51
-25	-4.24

Table 9. Noise Levels before and after separation

This method was also found inadequate. We obtained confidence scores for the sentences at 0 dBs (where the BSS recognizer should be selected) and at -25 dBs (where the mixed recognizer is better), and the selection based on this criterion was correct in 45%-50% of the cases (which is the same as a random choice).

4.5.3 *Choosing Recognizer Using N-best Lists*

The final option we tested was to let each recognizer produce N-best outputs instead of only one (the most probable). If we name $\mathrm{hyp}_1, \mathrm{hyp}_2, \ldots, \mathrm{hyp}_N$ the N-best outputs then $p(\mathrm{hyp}_i|M_A) = p(\mathbf{o}, \boldsymbol{w}_i|M_A)$.

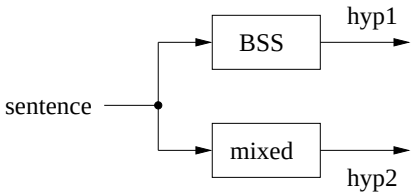

Figure 18. Each recognizer (one trained with BSS separated data and the other with mixed data) produce a hypothesis for each sentence they take as input.

We can calculate $p(\boldsymbol{w}_1|\mathbf{o}, M_A)$ by:

$$p(\boldsymbol{w}_1|\mathbf{o}, M_A) = \frac{p(\mathbf{o}, \boldsymbol{w}_1|M_A)}{p(\mathbf{o}|M_A)}$$

We can approximate $p(\mathbf{o}|M_A) \approx \sum_i p(\mathbf{o}, \boldsymbol{w}_i|M_A)$, so:

$$p(\boldsymbol{w}_1|\mathbf{o}, M_A) = \frac{p(\mathbf{o}, \boldsymbol{w}_1|M_A)}{\sum_i p(\mathbf{o}, \boldsymbol{w}_i|M_A)}$$

Using the á-posteriori products as confidences we can choose the best recognizer based on this criterion:

$$p(\boldsymbol{w}_A|\mathbf{o}, M_A) \lessgtr p(\boldsymbol{w}_B|\mathbf{o}, M_B) \tag{4.1}$$

We used $N = 100$ as the number of best hypotheses to use. HTK also allows to change the number of active tokens in each state (cf Section 3.4, page 26). Increasing the number of active tokens, increases the accuracy of the results but also increases the computation time. In Table 10 you can see the confidence scores for each noise levels for three different number of active tokens.

In Figure 19 you can see the word error rates in an ideal case, where we choose in each case the best recognizer (because we know the energies of the two signals) and in the case where we choose which recognizer to use from criterion 4.1. In Tables 11 you can see the results in greater detail. As you can see, using the confidence scores obtained from the N-Best List method, helps us choose a recognizer with a slight

Noise Level (dBs)	Number of active tokens		
	n = 4	n = 8	n = 12
0	85.7%	85.2%	85.0%
-5	80.2%	81.2%	83.1%
-10	69.3%	82.1%	81.9%
-15	66.2%	83.4%	84.3%
-20	73.3%	78.2%	78.4%
-25	70.3%	75.0%	75.3%

Table 10. In what percent of the cases was the correct recognizer chosen for each noise level and for the different number of active tokens per state.

DBS	IDEAL WER (%)	REALISTIC WER (%)
0	16.78	22.02
-5	18.53	23.44
-10	22.85	21.53
-15	17.41	20.91
-20	13.88	16.40
-25	13.10	14.44

Table 11. Recognition results of ideal and realistic recognizers.

increase in WER of about 2%-5%.

In the ideal case, we choose which recognizer is best in each noise level, comparing the performance of the two recognizers in all the 330 sentences for this noise level. Using the N-Best List criterion, we choose the best recognizer for each *sentence*. This can sometimes give us better results than the "ideal" case, as shown in the case where the second speaker is 10 dBs lower than the first one!

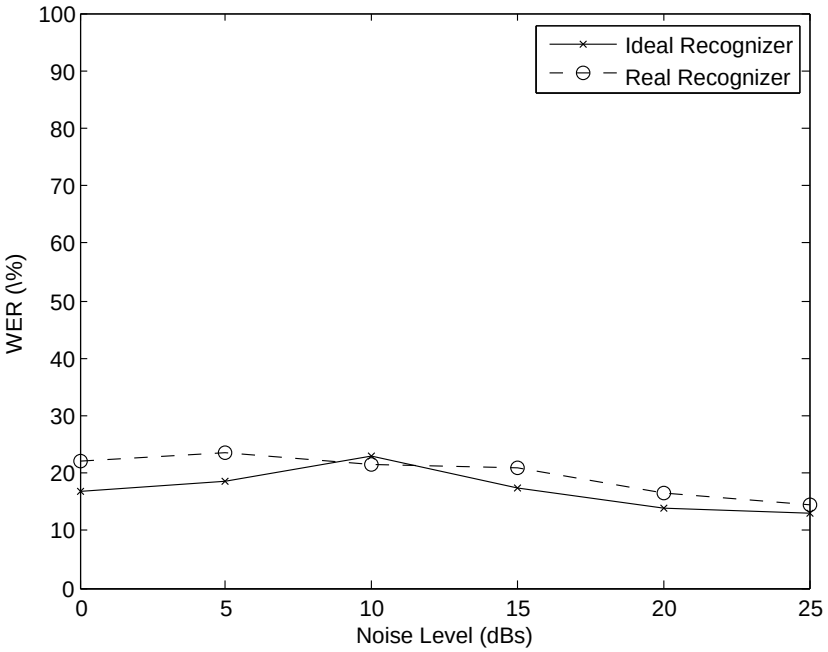

Figure 19. Recognition results of our final recognizer (BSS retraining for 10 dBs and lower and mixed retraining for 15 dBs and above).

5

DISCUSSION

5.1 FINAL RECOGNIZER

As the results show, spectral subtraction is not suitable for removing the type of noise we are studying in this work. The method assumes that the properties of the noise remain approximately static throughout the whole speech activity, which does not hold true for speech signals. So, the only viable option for removing this kind of noise is the blind source separation method.

Retraining the recognizer using mixed data gives better results and even better results can be obtained by training the recognizer using mixed data that have been separated using the BSS algorithm.

In a realistic case, we do not know the relative energies of the speaker signals, so we must use the recognizers trained using the "general" training sets that were discussed in Section 4.4 and also use a criterion that selects which recognizer to choose in each case. The criterion based on N-Best Lists gives good indications as to which recognizer to choose with an accuracy of above 70% in all cases and even as high as 85% when the two signals have same energy levels.

The final results show that the BSS algorithm can be used in realistic recognizers giving recognition results that range from 14.5% to 22%. However one problem that needs to be solved, is the computation time that is needed to separate each sentence (approximately 1 minute to separate each sentence with an average length of 30 secs per sentence) and to run the N-Best List criterion using 12 active tokens per state, which yields the best results (approximately 10 minutes to produce

outputs for 1 sentence).

5.2 FUTURE WORK

As discussed in Section 2.2, we used a modified version of the algorithm which assumes perfect solution of frequency ambiguity and scaling problems. As a future work, we can study how each of these problems affect the recognition rates specifically and which strategies can be used to counteract the effects.

We can also use different recording setups (different room acoustics, different positions of microphones and speakers) to evaluate the performance of the separation in various environments.

Finally, we could try to separate more than two speakers using the BSS algorithm and investigate the performance in each case.

BIBLIOGRAPHY

[1] ITU-T, "Recommendation P.341," in Transmission characteristics for wideband (150-7000 hz) digital handsfree telephony terminals. The International Telecommunication Union, 2005. (Cited on page 26.)

[2] The TIMIT database: http://www.mpi.nl/world/tg/corpora/timit/timit.html. (Cited on page 2.)

[3] S. George Ayanah. Using Spectral Subtraction to Enhance Speech and Improve Performance in Automatic Speech Recognition. Master's thesis, The Florida State University, Spring Semester 2006. (Cited on pages 3, 13, and 16.)

[4] Steven F. Boll. Suppression of Acoustic Noise in Speech Using Spectral Subtraction. Master's thesis, University of Utah Salt Lake City, April 1979. (Cited on page 17.)

[5] Douglas R. Campbell, Kalle J. Palomäki, and Guy J. Brown. Roomsim, a MATLAB Simulation of "Shoebox" Room Acoustics for use in Teaching and Research. (Cited on page 22.)

[6] Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, December 1974. (Cited on page v.)

[7] Lieven De Lathauwer. Principal Component, Independent Component and Parallel Factor Analysis: http://homes.esat.kuleuven.be/ imarkovs/workshop/s7l3.pdf. (Cited on page 11.)

[8] Kleanthis N. Mokios, Nicholas D. Sidiropoulos, and Potamianos Alexandros. Blind Separation of Multichannel Speech Mixtures Using PARAFAC Analysis and Integer Least Squares in acoustics, speech and signal processing, ICASSP 2006 proceedings. 2006 IEEE international conference on publication date: 2006, volume: 5, on page(s): V-V, location: Toulouse. (Cited on pages 3, 10, 12, and 20.)

[9] D. Paul and J. Baker. The Design of Wall Street Journal-Based CSR Corpus. *Proc. International Conference on Spoken Language Processing '92*, pages 899–902, 1992. (Cited on page 2.)

[10] D. Schobben, K. Torkkola, and P. Smaragdis. Evaluation of Blind Signal Separation Methods, 1999. (Cited on page 2.)

[11] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershw, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. The HTK Book (for HTK Version 3.3. (Cited on pages 5, 24, 25, and 38.)