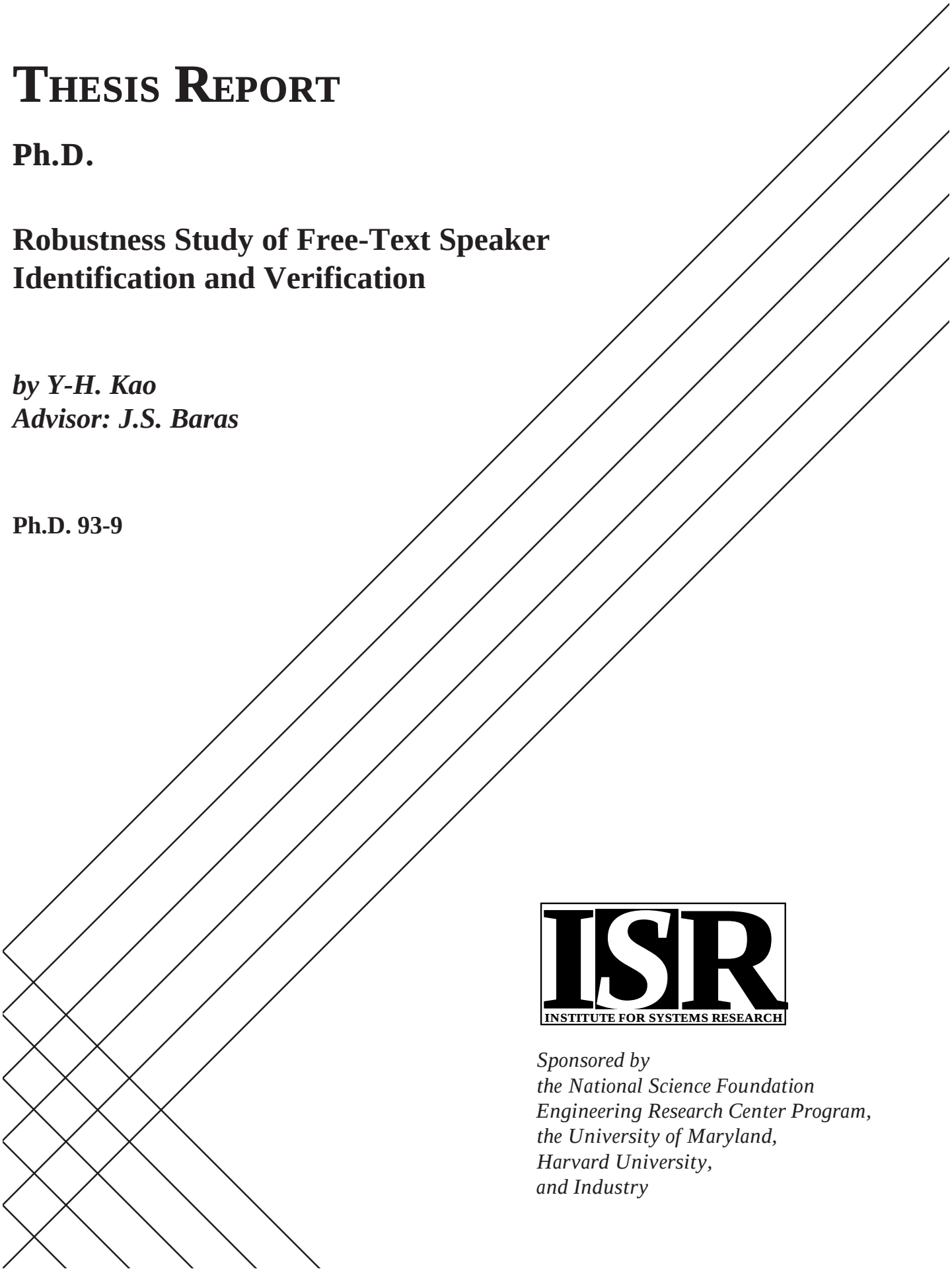
THESIS REPORT

Ph.D.

Robustness Study of Free-Text Speaker Identification and Verification

by Y-H. Kao
Advisor: J.S. Baras

Ph.D. 93-9



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Robustness Study of Free-Text Speaker Identification and Verification

by

Yu-Hung Kao

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1992

Advisory Committee:

Professor John S. Baras, chairman/advisor
Professor Thomas Fuja
Professor Benjamin Kedem
Professor Prakash Narayan
Doctor Raja Rajasekaran

ABSTRACT

Title of Dissertation: Robustness Study of Free-Text
Speaker Identification and Verification

Yu-Hung Kao, Doctor of Philosophy, 1992

Dissertation directed by: Professor John S. Baras
Department of Electrical Engineering
Martin Marietta Chair in Systems Engineering

Usable free-text speaker identification and verification systems must exhibit robustness under varying operational conditions. We studied the degree of robustness provided by various signal processing techniques - spectrum subtraction, bandpass liftering, RASTA filtering, ISDCN, and stereo database normalization. The experiments were performed on a widely used, challenging long distance telephone database. This database consists of data recorded at two different sites, with data from one site much poorer in quality than the other; further, the recording equipment had been inadvertently changed for the later half of the sessions resulting in a significantly changed environment. Our study identifies the combination of techniques that provides consistent and significant improvements; our results surpass other published results on the same task. We further verified the results on two other databases and achieved consistent improvements. Detailed results on exhaustive experimentation are presented along with appropriate discussions.

Acknowledgements

There are a number of people to whom I am indebted, and whom I would like to thank. First, there are those who had taught me and inspired me, whether through courses, books, papers or conversations; I cannot possibly name them all, but I remember every precious moment I spent with them. Without their contributions, this work will not be possible. I would like to thank Professor Baras, my very resourceful advisor, for his guidance and support all through my graduate study. I would like to thank Dr. Rajasekaran, for providing the opportunity for me to work with the speech group at Texas Instruments. This wonderful experience made my life more productive and enjoyable. Finally, I would like to thank my parents, for their love and encouragement. Through life's ups and downs, joy and pain; they showed me there is always hope and told me never to give up. I would like to dedicate this thesis to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The purpose of ASI (Automatic Speaker Identification) is to output the identity of a person based on this person's speech data. While fingerprints or retinal scans are usually more reliable than ASI, ASI has the convenience of easy data collection over the telephone.

There are two sources of variation among speakers : 1) Difference in vocal cords and vocal tract shapes; 2) Difference in speaking styles. There are no acoustic cues specifically or exclusively dealing with speaker identity. Most of the parameters and features used in speech analysis contain information useful for the identification of both speakers and spoken messages. ASI can be categorized into two cases : text-dependent and text-independent. In the text-dependent case, we can obtain the features attached to a specific sound (e.g. phoneme) and the remainder of the identification process is just template matching. In the text-independent case, we usually use long-term statistics averaged over whole utterances. The text-independent case, which is the object of this thesis, is particularly important, because ASI usually deals with un-cooperative situations, where it is impossible to ask the person being tested to speak a particular text.

1.2 Basic problems in ASI

1.2.1 Efficiency

In theory, a speaker recognizer could be as simple as a large dictionary where each entry is a stored waveform associated with a speaker identity. Given an input utterance, the dictionary would be searched to find an exact match (or a close match), and then the system can output the corresponding identity. However, this is impractical due to the enormous memory and computation required. We need to reduce the amount of data and the computational complexity to make ASI practical. These problems are solved by feature extraction (Section 1.2.2) and speaker modeling (Section 1.3).

1.2.2 Feature Extraction

To reduce the amount of data, we need to extract features instead of using raw speech waveforms. However, it is not currently known what features carry speaker-dependent information. There is no simple answer to this question. Since we know that people have different vocal cords and vocal tracts, formant frequencies and linear predictive coefficients (LPC, LSP, cepstrum, mel-cepstrum ...) provide some initial good candidates. Besides extracting useful features, in order to improve performance, we also have to find out which features have "better discriminating ability". There are ways to "rank" features according to their "effectiveness" in classification [6] [23] [9]. Evaluation of the effectiveness of cepstral coefficients can be found in Chapter 7, Figure 7.1.

We found that the LPC cepstrum coefficients give us the best performance; the description of our LPC front end can be found in Appendix A.

1.2.3 Distance Measures

In a classification task, after the features are extracted, we need to define a distance measure. Actually, this step is inseparable from feature extraction. **The effectiveness of a feature depends on the distance measure and vice versa.** For example, Euclidean distance or other more complicated distance measure (Chapter 2) are used in non-parametric classification (e.g. vector quantization); however, in parametric modeling the underlying *pdf* is often used (Chapter 5).

1.2.4 Channel Normalization

Because ASI data are often collected via telephone, and different handsets have different responses and noises, we need to find ways to offset these undesirable effects. This is important in order to obtain a robust ASI algorithm. To offset linear response, we can either choose features invariant to it [11] or compute offset values [22]. To remove noise, we can find a noise model for a particular channel and then try to remove the noise according to this model [12].

We devoted most of this thesis in addressing this problem. Without channel robustness, ASI is useless in real world applications - where the most potential of ASI technology can be expected. We studied the following techniques:

- Noise reduction (spectrum subtraction): Chapter 3.
- Normalization using stereo database training: Chapter 6.
- Bandpass liftering: Chapter 7.
- RASTA filtering: Chapter 8.

- ISDCN: Chapter 9.

We found that the stereo database technique is very useful in channel normalization; however, it is not usually available. Noise reduction helps in some cases (King database, 26 San Diego speakers) but hurts in others (for speech data with either very good or very bad qualities it does not predict the noise well and will degrade recognition performance). Bandpass liftering and RASTA filtering not only give us significant and consistent improvement on the three databases (King, Total voice, and CSR) we evaluated, but they are also the easiest (ideas are simple and computation is low). ISDCN gives us consistent improvement, but not significant. Compared to bandpass liftering and RASTA filtering, the "price performance" of ISDCN seems low.

1.2.5 Fixed-Text vs. Free-Text

Most speaker identification / verification systems use fixed-text mode, because fixed-text mode has higher identification accuracy. However, our requirement is free-text mode. We investigate the idea of "converting" free-text mode to fixed-text mode by automatic phonetic segmentation (Chapter 4). Our investigation showed inconclusive results:

- Detailed phonetic segmentation performed worse than no phonetic segmentation.
- Broad class phonetic segmentation performed as well as no phonetic segmentation.

The results suggest two problems that need further study:

1. The accuracy of phonetic segmentation should be improved.

2. Training data for each phonetic class may be insufficient.

1.3 Modeling Methods

1.3.1 Parametric

Features can be assumed Gaussian distributed, speaker models can be built by training the mean and covariance of the distribution for each speaker (Chapter 5). After the models are built, the classification is simply to compute a score (e.g. log likelihood) with respect to each speaker model:

$$\text{Score}_j = (1-\alpha)\text{COV}(\Sigma_j, S) + \alpha\text{MEAN}(\mu_j, x; \Sigma_j) \tag{1.3.1}$$

Here the COV term is proportional to the log likelihood of observing the data (with mean removed) under the assumption that the data is Gaussian distributed with covariance Σ_j. The MEAN term is the log likelihood of observing x given model j is true. (S is the sample covariance and x is the sample mean, Σ_j and μ_j are the covariance and the mean of speaker model j, [11]).

Parametric modeling is used in Chapter 5 only; all other results are based on VQ.

1.3.2 Non-parametric

The advantage of non-parametric approaches is that they do not assume any particular distribution (any such assumption can be wrong). The disadvantage is that they are more susceptible to noise and linear distortion. Vector Quantization is the most widely used approach; however, there are many VQ structures, from the classic LBG [18] [10] to tree structure [6] [9], plus some

other modifications ([7] ...). Examples using this approach can be found in [25] and [14].

Classic VQ faces the high computational complexity problem. Tree structured VQ is efficient in computation and needs very small space to store classification rules. However, when a new speaker is enrolled, rules have to be re-computed, which is very expensive in computation.

1.4 Summary

This thesis has the following components :

- Database: Three databases are evaluated:
  - King: telephone quality, Section 10.1.
  - Total voice: telephone quality, Section 11.3.1.
  - CSR (part): studio quality, stereo recording, Section 6.3.
- Robustness: As described in Section 1.2.4, Chapters 3, 6, 7, 8, and 9.
- Modeling: Gaussian Mixture vs. VQ, Chapter 5.
- Phonetic segmentation: Chapter 4.
- Speaker verification: Speaker verification is more useful than speaker identification in real world applications. We dedicate Chapter 11 to describe the difference between verification and identification, and address some facts that we have learned from the study of the identification problem and apply them to the verification problem.

Chapter 2

Transformation Space

2.1 Decorrelation Matrix

First we consider the **within class** distribution; the ideal feature vector should not have redundancy in representing signal. Suppose $\vec{p}$ is the parameter vector. Its components may be correlated, but we don't want redundancy among these components, so we use a linear transformation matrix to map this parameter vector (with correlated components) into a feature vector (with uncorrelated components).

In the following we describe how we compute this decorrelation transformation matrix. Suppose $\vec{p}$ is the parameter vector with correlated components, and $[C] = E[\vec{p}\vec{p}^T]$ is the within class covariance matrix. First we compute the eigenvalues and eigenvectors of $[C]$, λ_i and $\vec{e}_i, \forall i = 1, \cdots, n$,

$$([C] - \lambda_i[I])\vec{e}_i = 0 \tag{2.1.1}$$

$$[C]\,\vec{e}_i = \lambda_i \vec{e}_i \tag{2.1.2}$$

$$[C]\,[\vec{e}_1 \vdots \vec{e}_2 \cdots \vec{e}_n] = [\vec{e}_1 \vdots \vec{e}_2 \cdots \vec{e}_n][\Lambda] \tag{2.1.3}$$

where

$$[\Lambda] = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots \\ 0 & \lambda_2 & 0 & \cdots \\ 0 & 0 & \lambda_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.1.4}$$

$$[\varepsilon]^T = [\vec{e}_1 \vdots \vec{e}_2 \cdots \vec{e}_n] \tag{2.1.5}$$

Then, using $[\varepsilon] = ([\varepsilon]^T)^{-1}$,

$$[C]\,[\varepsilon]^T = [\varepsilon]^T\,[\Lambda] \tag{2.1.6}$$

$$([\varepsilon]^T)^{-1}\,[C]\,[\varepsilon]^T = [\Lambda] \tag{2.1.7}$$

$$[\varepsilon]\,[C]\,[\varepsilon]^T = [\Lambda] \tag{2.1.8}$$

Suppose $\vec{v} = [\varepsilon]\,\vec{p}$ is the transformed feature vector, then

$$E[\vec{v}\vec{v}^T] = E[[\varepsilon]\,\vec{p}([\varepsilon]\,\vec{p})^T] = E[[\varepsilon]\,\vec{p}\vec{p}^T\,[\varepsilon]^T] = [\varepsilon]\,[C]\,[\varepsilon]^T = [\Lambda] \tag{2.1.9}$$

Up to here, we have the decorrelation transformation matrix, $[\varepsilon]$, which can convert parameter vectors $\vec{p}$ into feature vectors $\vec{v}$, and the feature vectors have a diagonal covariance matrix $[\Lambda]$. Furthermore, we can modify the transformation to make the covariance matrix an identity matrix $[I]$. Suppose the new transformation is $[X] = [\Lambda^{-1/2}]\,[\varepsilon]$, where

$$[\Lambda^{-1/2}] = \begin{bmatrix} \sqrt{1/\lambda_1} & 0 & 0 & \cdots \\ 0 & \sqrt{1/\lambda_2} & 0 & \cdots \\ 0 & 0 & \sqrt{1/\lambda_3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.1.10}$$

Then $\vec{v} = [X]\vec{p}$, and

$$E[\vec{v}\vec{v}^T] = E[[\Lambda^{-1/2}]\,[\varepsilon]\,\vec{p}\vec{p}^T\,[\varepsilon]^T\,[\Lambda^{-1/2}]] = [\Lambda^{-1/2}][\Lambda][\Lambda^{-1/2}] = [I] \tag{2.1.11}$$

We thus obtain the transformation $[X] = [\Lambda^{-1/2}]\,[\varepsilon]$ which maps parameter

vectors into feature vectors with uncorrelated components, and unit variances for all components.

2.2 Discriminant Matrix

After considering the within class distribution to remove the redundancy in signal representation, we now consider the **inter class** distribution because our goal is to discriminate between classes. Decorrelation only removes redundancy in signal representation. Since our goal is to discriminate between different classes, we need to find out which of the decorrelated components contribute more to discrimination. This will help us to construct the best feature space for discrimination.

After decorrelation, all dimensions have equal weights (unit variances), which means that we give each dimension equal significance regarding the distance measure (using Euclidean distance), which is not necessarily the optimal feature space regarding the discrimination problem at hand. There are several ways to solve this problem. Usually one selects those dimensions with large eigenvalues, because the eigenvalue is the variance of that dimension, and from the view point of information theory, those dimensions with large variances usually carry more information. However, this is not necessarily true. It is possible that large variances are due to large noise, and we may want to choose those dimensions with small variances, because they are stable and thus can reliably represent signal information. This later choice also has a serious problem: although stability is a good property, what if the corresponding dimensions don't discriminate? Stable but non-discriminating features are useless!

That is why we need to consider the **inter class** distributions, we let the

statistics decide the best feature space. Here is how we do it:

Suppose $[C_x]$ is the inter class covariance matrix, and $[X]$ is the decorrelation matrix derived in the previous section. First we compute $[C'_x] = [X][C_x][X]^T$, and then perform eigen-analysis on $[C'_x]$, to obtain the eigenvalues λ_i^x and the eigenvectors $\vec{e}_i^x$, $\forall i = 1, \cdots, n$. We define matrices $[\Lambda_x]$ and $[\varepsilon_x]$ similarly as in the previous section. The final discriminant matrix is $[X_d] = [\varepsilon_x][X]$. We can factor it into $[D][V]$, where $[V]$ is normalizing each row of $[X_d]$, and $[D]$ is a diagonal matrix with the length of each row as diagonal entries.

This discriminant matrix $[X_d]$ has two desirable properties :

- For within class parameter vectors $\vec{p}$,

$$E[[X_d]\vec{p}([X_d]\vec{p})^T] = E[[\varepsilon_x][X]\vec{p}\vec{p}^T[X]^T[\varepsilon_x]^T] \quad (2.2.12)$$

$$= [\varepsilon_x][X][C][X]^T[\varepsilon_x]^T \quad (2.2.13)$$

$$= [\varepsilon_x][I][\varepsilon_x]^T = [I] \quad (2.2.14)$$

- For inter class parameter vectors $\vec{r}$,

$$E[[X_d]\vec{r}([X_d]\vec{r})^T] = E[[\varepsilon_x][X]\vec{r}\vec{r}^T[X]^T[\varepsilon_x]^T] \quad (2.2.15)$$

$$= [\varepsilon_x][X][C_x][X]^T[\varepsilon_x]^T \quad (2.2.16)$$

$$= [\varepsilon_x][C'_x][\varepsilon_x]^T = [\Lambda_x] \quad (2.2.17)$$

As shown, this discriminant matrix in addition to decorrelating the within class parameters, also identifies the discrimination ability of each dimension. Indeed the covariance matrix of the transformed, inter class parameter vectors is $[\Lambda_x]$, and a feature dimension with associated large variance means that this dimension is wide spread, therefore it has good discrimination ability. Consequentially

we can choose dimensions with large λ_i^x's, and construct the best discriminant space.

Below we discuss some motivation insights and justify the use of the discriminant matrix as described above:

1. Why do we want the feature vectors to have uncorrelated components? Because we want to remove the redundancy in signal representation. For a typical distance measure, say Euclidean distance, we sum the square of the differences of each component. If the components are correlated, then this summation does not accumulate independent information properly. It is just like covariance weighted distance : $(\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y})$ with respect to Euclidean distance : $(\vec{x}-\vec{y})^T(\vec{x}-\vec{y})$.

2. Why do we normalize the variances of all dimensions into unity (in the decorrelation matrix derivation)? Because we want a fair comparison in λ_i^x's (inter class variances) in the discriminant matrix computation. Dimensions with large within class variances (λ_i's) will tend to have large inter class variances (λ_i^x's); in this case, a large λ_i^x does not represent better discrimination ability, because its corresponding large λ_i (within class variances) means it is unstable, and this will hurt discrimination. In order to have a fair comparison among the λ_i^x's, we have to first normalize all within class variances to be the same.

3. Why do we want to choose those dimensions with large inter class variances (λ_i^x's)? This is obvious; components with large inter class separation can discriminate better. Usually the more information we have, the better the performance will be. So why not use all the dimensions? Based on

experiments, we know that as the information amount (the number of parameters) increases, the performance will not increase indefinitely. Instead, it will reach a maximum and then decrease. The reason for this is that after a certain point contaminated information is used. This provides the reason for using the discriminant matrix: to keep only good information and throw away bad information.

After all the derivations, the idea behind this discriminant matrix is very simple : suppose $[C]$ is the within class covariance matrix, and $[C_x]$ is the inter class covariance matrix, we just want to maximize :

$$\frac{[C_x]}{[C]} \tag{2.2.18}$$

2.3 Insufficient Training Data Problems

2.3.1 Sub-space Grouping Based on Human Knowledge

Usually we need an amount of data equivalent to 10 times the dimension of training data to compute reliable statistics, e.g. to compute a 10 * 10 covariance matrix, we need at least about 100 vectors (dimension 10) of training data. However, enough training data are not easy to get in real applications. When we don't have enough training data, human knowledge is very important in order to control the disturbances in empirical statistics caused by insufficient training data.

For example, in the computation of the decorrelation matrix, if we have enough training data, we can input all dimensions into the computation, and then the covariance matrix will designate what dimensions correlate with each other. However, if training data are insufficient, then it is very likely that

the covariance matrix will have non-zero entries where those dimensions don't correlate. These errors are caused by insufficient training data.

One simple way to fix this problem is to decorrelate only those dimensions that are likely to be correlated. For example, if we have both spectrum coefficients and absolute energy in our parameters, we should not put them into a decorrelation computation together, because the spectrum coefficients are normalized, so they are not likely to be correlated with absolute energy. If we compute their covariance values without any rationale, then we are likely to get non-zero values between uncorrelated parameters due to insufficient training data, and these erroneous non-zero covariance values will hurt the good statistics.

The idea is to select a sub-space that is likely to contain correlated dimensions, based on knowledge, and then perform decorrelation in this sub-space only. Suppose the full covariance matrix of the whole parameter space is $[C]$, dimension 10 * 10, but we know that the first 5 dimensions are independent of the last 5 dimensions. Therefore we want to decorrelate these two sub-spaces separately. We can define a pre-transformation matrix selecting the first 5 dimensions :

$$[P_1] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.3.19}$$

First we compute,

$$[C'] = [P_1][C][P_1]^T \tag{2.3.20}$$

which is the covariance matrix of the subspace, then we perform eigen analysis

on $[C']$, and get eigenvector matrix $[\varepsilon]$, and eigenvalue matrix $[\Lambda]$. Then the transformation matrix

$$[X_1] = [\Lambda^{-1/2}][\varepsilon][P_1] \tag{2.3.21}$$

will transform the first 5 dimensions into uncorrelated, unit variance features.

Similarly, we can compute the transformation matrix for the last 5 dimensions, simply by replacing the pre-transformation matrix with,

$$[P_2] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3.22}$$

Suppose the transformation matrix is $[X_2]$, then appending $[X_2]$ to $[X_1]$, results in a 10 * 10 transformation matrix, which decorrelates only within each subspace.

The pre-transformation matrix can be made to not only select a sub-space, but also transform the space. The decorrelation can be performed in a cascading fashion, using human knowledge and experiences, to construct the best transformation matrix under the constraint of limited training data.

2.3.2 Covariance Pooling

Sometimes the number of training instances is even less than the dimension of the parameter vector; thus the covariance matrix is not even positive definite (singular). In that case, we can pool (average) many singular matrices to form a pooled matrix, and usually this pooled matrix is non-singular and has all positive eigenvalues. Even if the matrix is positive definite, by pooling matrices

we typically can get better estimates.

This situation often occurs when we compute within class covariance matrices, as the number of classes increases, because the amount of training data is fixed, so the training data falling into a particular class decreases. When the training data is not enough to form a non-singular matrix, we can pool all the within class covariance matrices and usually get a non-singular matrix. Covariance pooling not only solves the singular matrix problem, it also has good physical meaning interpretation : averaged covariances can smooth out disturbances and give a good estimate of the covariances among all those parameters.

2.3.3 Stabilization

Covariance pooling is not the only way to solve the singular covariance matrix problem, with a little twiddling, we can convert a singular matrix into a non-singular matrix and still retain the property of the original matrix without too much distortion.

First we need to understand the effects of scalar multiplication and identity addition to the eigen-analysis of a matrix. Suppose an eigenvalue and eigenvector of the matrix $[C]$ are λ and $\vec{e}$. And suppose λ_α and $\vec{e}_\alpha$ are the eigenvalue and eigenvector of $\alpha[C]$.

$$(\alpha[C] - \lambda_\alpha[I])\vec{e}_\alpha = 0 \tag{2.3.23}$$

Let $\lambda_\alpha = \alpha\lambda'$, then

$$(\alpha[C] - \alpha\lambda'[I])\vec{e}_\alpha = 0 \tag{2.3.24}$$

$$\alpha([C] - \lambda'[I])\vec{e}_\alpha = 0 \tag{2.3.25}$$

$$([C] - \lambda'[I])\vec{e}_\alpha = 0 \tag{2.3.26}$$

Therefore,

$$\vec{e}_\alpha = \vec{e} \tag{2.3.27}$$

$$\lambda_\alpha = \alpha\lambda' = \alpha\lambda \tag{2.3.28}$$

Next, suppose λ_β and $\vec{e}_\beta$ are the eigenvalue and eigenvector of $[C] + \beta[I]$,

$$([C] + \beta[I] - \lambda_\beta)\vec{e}_\beta = 0 \tag{2.3.29}$$

$$([C] - (\lambda_\beta - \beta)[I])\vec{e}_\beta = 0 \tag{2.3.30}$$

Let $\lambda' = \lambda_\beta - \beta$,

$$([C] - \lambda'[I])\vec{e}_\beta = 0 \tag{2.3.31}$$

Therefore,

$$\vec{e}_\beta = \vec{e} \tag{2.3.32}$$

$$\lambda_\beta = \lambda' + \beta = \lambda + \beta \tag{2.3.33}$$

Based on these simple computations, if we have a covariance matrix $[C]$, which has some negative eigenvalues, or we are not very confident about because of insufficient training data, we can modify it into $\alpha[C]+(1-\alpha)[I]$. Its eigenvectors

will be the same, and eigenvalues will change to $\alpha\lambda + (1 - \alpha)$. For $\lambda = 1$, there is no change, but for $\lambda < 1$, the changed eigenvalue becomes larger, while for $\lambda > 1$, it becomes smaller. That means outliers are modified towards 1, thus stabilizing the covariance matrix.

Chapter 3

Noise Reduction

3.1 Bandpass Filtering Is Not Enough

For telephone speech data sampled at 8 KHz (4 K spectrum), we usually use 300 to 3300 Hz bandpass filtering to remove noises outside the normal human speech spectrum. However, bandpass filtering can only handle noises with specific frequency ranges, e.g. 60 Hz static humming. To deal with more diffusive (white) noises, as in our case, simple filtering based on frequency contents won't work. We have to model the noise differently, therefore we choose the spectrum subtraction approach [5].

3.2 Spectrum Subtraction

We use the following assumption to develop this algorithm: **the background noise is acoustically or digitally added to the speech.** The noise environment remains stationary (at least in a short time sense) so the noise spectrum can be estimated from the nonspeech period just before the speech begins. For a slowly varying nonstationary noise environment, we can use a speech detector to decide when to use the data to model the noise, and when to subtract the noise from the noise inflicted speech.

3.2.1 Algorithm

1. Input speech frame, 256 samples per frame, shifted by 128 samples; so half of the frame is overlapped with the previous frame. Multiply by Hanning window to smooth the frame.

2. FFT the windowed signal to compute the complex spectrum.

3. Compute the energy magnitude and phase of the spectrum. In the following steps, we want to modify the energy but save the original phase for IFFT reconstruction.

4. Subtract noise energy profile from the energy of the signal, where the noise energy profile is updated from time to time to accommodate possible time varying noise.

5. Half wave rectification: Set those coefficients with negative energy to zero.

6. Residual noise reduction: In each frequency bin, if the energy is smaller than the maximum of the noise estimate, then it is very likely to be noise. We set its value to be the minimum in its time neighborhood. The rationale behind this smoothing is as follows: If it is noise, then it is likely to be a burst, setting it to be the minimum within its neighbors will remove this burst; if it is speech, then it will be stable for more than one frame, setting it to be the minimum within its neighbors will still retain its value.

7. Nonspeech test. If it is nonspeech, then update the noise energy profile, and further attenuate the energy. Nonspeech detection is the most difficult step.

8. (Optional) Bandpass filtering.

9. Use the modified energy and original phase to reconstruct the complex spectrum.

10. IFFT on the complex spectrum to reconstruct the time signal.

11. Overlap addition with the previous frame to compute the output signal. Note that the Hanning window overlapping is equivalent to the original.

Since we have spectrum values in the middle of the processing, it is very easy to do bandpass filtering - simply attenuate the values at those unwanted bands.

3.2.2 Nonspeech Detection

Because the noise level can vary a lot, any fixed threshold algorithm will not work well. One reliable way to check for noise is using a "two-pass" approach. The first pass scans the signal and builds the histogram of frame energies, decides the noise level; then the second pass performs the noise subtraction.

1. Compute energy of each frame.

2. Sort.

3. Pick the value at 20th percentile as the threshold.

4. Designate those frames with energy larger than 2 * threshold as well as their neighbors with energy larger than 1.5 * threshold as speech frames. Designate the others as nonspeech frames.

The rationale behind this algorithm is that speech frames are likely to be continuous, so we choose those frames with large energy and their neighbors as speech.

3.3 Results

We refer to section 4.3 for a detailed description of the test database used. We compared results before and after noise reduction, for both within and across the great divide. The results are on King database, 26 San Diego speakers.

- No noise reduction: Table 3.1 and Table 3.2.

training session	test session	recognition rate
1 2 3	4	18/26
1 2 3	5	19/26
6 7 8	9	24/26
6 7 8	10	24/26
average		81.7%

Table 3.1: Identification results before noise reduction, within the great divide

training session	test session	recognition rate
6 7 8	4	2/26
6 7 8	5	4/26
1 2 3	9	1/26
1 2 3	10	2/26
average		8.7%

Table 3.2: Identification results before noise reduction, across the great divide

- After noise reduction: Table 3.3 and Table 3.4.

training session	test session	recognition rate
1 2 3	4	25/26
1 2 3	5	25/26
6 7 8	9	23/26
6 7 8	10	23/26
average		92.3%

Table 3.3: Identification results after noise reduction, within the great divide

As we can see in the Tables 3.1, 3.2, 3.3, and 3.4, noise reduction improves the performance in the case of "within the great divide"; however, in the "across

training session	test session	recognition rate
6 7 8	4	2/26
6 7 8	5	2/26
1 2 3	9	2/26
1 2 3	10	2/26
average		7.7%

Table 3.4: Identification results after noise reduction, across the great divide

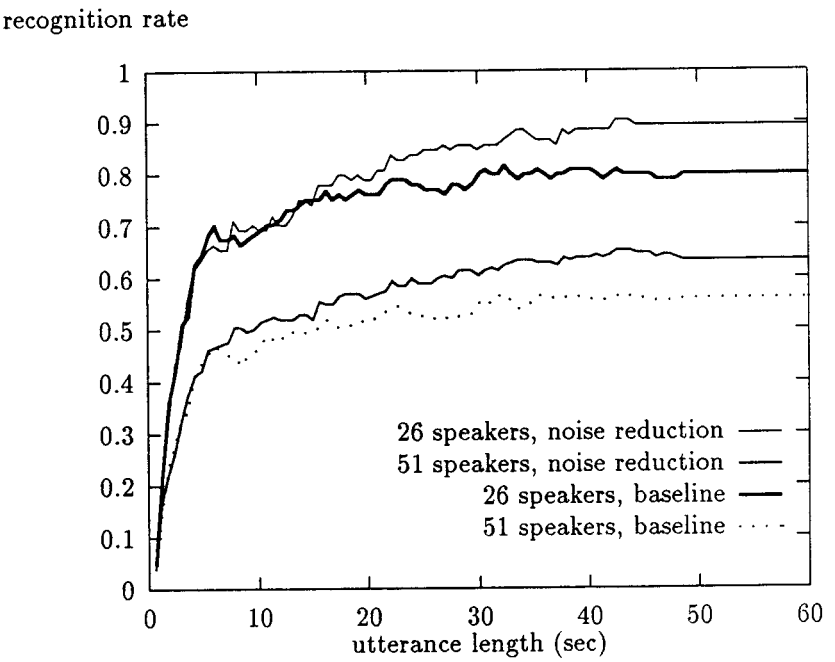


Figure 3.1: Recognition rate increases as input utterance length increases

the great divide" case, performance is still very poor. We need to use channel normalization techniques to deal with this problem. Figure 3.1 shows that the recognition rate increases as input utterance length increases; it includes the complete 51 speakers result. Note that the utterance length includes silence; and for the King database, about half of the materials are silence or nonspeech.

Chapter 4

Phonetic Segmentation

4.1 Introduction

In this chapter we present the results of experimental investigations into several aspects of free-text speaker identification and speaker authentication using a long-distance telephone database, described in [11]. Specifically, the following speaker identification experiments were carried out to analyze the effects of the following:

1. Using phonetic segments provided by a speaker independent recognizer.
2. Using broad phonetic class segments, obtained by pooling classes from the segments used in item 1 above.
3. VQ codebook size for the feature vector.
4. Using various parameters as feature vectors: log area ratios, LPC cepstral coefficients, and reflection coefficients.
5. Noise reduction including bandpass filtering.

Open set speaker verification was carried out using broad phonetic class speaker models along with noise reduction.

We conducted experimental investigation of free-text speaker identification methods based on long-term statistics with widely-used long distance telephone database [11]. On a 26-speaker subset, we obtained an average correct identification of 93.3%. On the complete 51-speaker set, we obtained 67.6% correct identification. Our speaker verification experiments on the database provided receiver operating characteristics (ROC) comparable to or better than the ones available in open literature.

4.2 Algorithms

4.2.1 Speaker Models

A speaker model consists of one or more vector quantization (VQ) codebooks of a speech parameter vector (for example, LPC cepstral coefficients) derived from the training utterances of the speaker. A single codebook is used when no phonetic hypotheses are to be exploited; all non-speech frames are eliminated by energy thresholding, and the speech frames are utilized to build the VQ codebook. Multiple codebooks are used for phonetic hypothesization experiments. The untranscribed training utterances are segmented by a speaker-independent continuous speech recognizer into phonetic categories. A VQ codebook for each phonetic category (or broad phonetic category) is built from the corresponding phonetic segments. A phonotactic grammar was used to improve the performance of the phonetic hypothesization, Appendix C. Phonetic models used in the recognizer were derived from the Voice Across America (VAA) database described in [28].

4.2.2 Speaker Identification

For our non-phonetic method (single VQ codebook speaker model), each frame of the test utterance is first classified as speech or non-speech; then the features of the speech frames are compared with the model of each of the speakers in the population to generate the distance score (distortion) for each candidate speaker according to a pooled speaker discrimination metric (refer to Chapter 2). The candidate speaker with the least accumulated distortion is declared as the identified speaker. The pooled speaker discrimination metric is derived from the training data by maximizing the F-ratio to improve separability between speaker classes (Chapter 2). We also developed an alternative method employing a speaker-independent continuous speech recognizer, whose output consists of hypothesized phonetic segments. The frames of speech from a phonetic category is compared with the candidate speaker model for that category as per the pooled speaker discrimination metric. Thus, a distortion for each of the observed phonetic categories in the utterance is calculated. These distortions are summed (it is possible to do so in a weighted manner; but this was not done) over all observed phonetic categories to provide the total distortion for each candidate speaker. Again, the candidate speaker with the least distortion is declared as the identified speaker.

4.2.3 Speaker Verification

VQ speaker models described in Section 4.2.1 were used along with Euclidean distance as the metric. We chose not to use the pooled speaker discrimination metric because it would have provided an unfair statistical knowledge of the impostors. In this paradigm, half the population (by choosing alternate speak-

ers) was used as impostors, and the target speaker and "normalizing" speakers came from the other half. The total distortion over the input test utterance is computed for the target speaker as well as for each of the "normalizing" speaker. If the total distortion provided by the target speaker is lower than that of the "normalizing" speaker models, the test utterance is verified as belonging to the target speaker; otherwise the test utterance is declared as belonging to an impostor.

4.3 Database

The database utilized in this study is the digitized subset of speech data collected in 10 sessions from 51 speakers, speaking on several topics (so that the speech is natural) over a long distance telephone line. Of the 51 speakers, 26 were based in San Diego, CA, and 25 in Nutley, NJ. The data from Nutley speakers were considerably noisier than that from San Diego speakers. Further, the equipment used for recording had changed from session 6 onwards, establishing a division of the database into two portions - sessions 1 through 5 (Div1), and sessions 6 through 10 (Div2). The nominal duration of the utterances in each session was about 45 seconds; when non-speech frames were eliminated, the average duration of the speech segments was about 23 seconds. When the speaker-independent continuous speech recognizer with the phonotactic grammar was used, and non-phonetic categories (silence, background, inhalation, exhalation etc.) were eliminated, the average duration of the speech segments was only about 29 seconds. Our experiments were conducted for both the 26-speaker (San Diego) subset and the total 51-speaker set. Training and test material were mostly restricted to Div1 or Div2 (within the "great divide"). We describe

in Chapter 10 our complete experiments including both "within" and "across" the great divide.

4.4 Experiments and Results

4.4.1 Phonetic Segmentation

Previous published research [8] [19] [24] and informal discussions with various speech researchers in this area provided a mixed review of the value of automatic phonetic segmentation. This observation, along with the belief that the act of converting free, unknown text to known, but unfixed text (performed with acoustic consistency) should help, motivated us to investigate this aspect. Allowing 49 phonetic categories, with a 10-element VQ codebook for each category (20 LPC cepstral coefficients), as the speaker model, an average correct speaker identification of 89.4% was obtained for the 26-speaker (San Diego) set; over all the 51 speakers, it was 63.2%. By collapsing the phonetic categories into 11 broad classes, and retraining the 10-element VQ codebook for each of the 11 classes, an improved average recognition of 93.3% was obtained for the 26-speaker set; an average recognition of 67.6% resulted for the 51-speaker data. When no phonetic marking was used with a 110-element VQ codebook, the results were nearly identical to that of broad phonetic class method, and better than that of detailed phonetic marking (49-category), the results are shown in Tables 4.1 and 4.2.

Figure 4.1 shows the correct speaker identification rate as a function of input speech durations (including non-speech).

These above results were based on data where noise suppression was used (refer to Chapter 3).

Session No.		No. of speakers correctly identified		
Training	Test	A	B	C
1, 2, 3	4	25	25	24
1, 2, 3	5	24	24	24
6, 7, 8	9	24	24	23
6, 7, 8	10	24	24	22
Average		93.3%	93.3%	89.4%

- A : Non-phonetic
- B : Broad Phonetic (11 classes)
- C : Detailed Phonetic (49 phones)

Table 4.1: Identification results: 26-speaker San Diego data, noise reduced

4.4.2 VQ Codebook Size

With non-phonetic models, the codebook size was varied between 5 and 110. The performances were virtually identical for codebook sizes of 25, 55 or 110. With a codebook of size 10, it was noticeably worse, Figure 4.2.

4.4.3 Features

The effect of using various speech parameters for speaker identification was studied with the detailed phonetic model approach. LPC cepstral coefficients of dimension 20 yielded 89.4%, 10-th order log area ratio coefficients 86.5% and 10-th order reflection coefficients, 83.7% for the 26 San Diego speaker experiments. For the total 51-speaker experiment, the corresponding performances were 63.2%, 61.3%, and 59.3% respectively. All other results presented in this thesis were obtained with LPC cepstral coefficients.

Session No.		No. of speakers correctly identified		
Training	Test	A	B	C
1, 2, 3	4	36	36	34
1, 2, 3	5	31	34	31
6, 7, 8	9	37	37	34
6, 7, 8	10	33	31	30
Average		67.2%	67.6%	63.2%

- A : Non-phonetic
- B : Broad Phonetic (11 classes)
- C : Detailed Phonetic (49 phones)

Table 4.2: Identification results: 51-speaker complete data, noise reduced

4.4.4 Noise Reduction and Filtering

Preliminary listening and spectrographic analyses of the database clearly showed the noisy nature of the data, especially that of the Nutley speakers. The spectral subtraction [5] method of noise suppression, along with bandpass filtering (300 - 3300 Hz), was used to preprocess the database. Experimental results indicate that noise suppression increased the speaker identification rate by an additional 10% for both non-phonetic and phonetic methods. What is surprising is that the performance with the relatively cleaner San Diego data (26-speaker set) also showed the improvement. The effects of noise suppression are brought out in the performance results shown in Table 4.3 and 4.4.

4.4.5 Speaker Verification

A set of preliminary speaker verification experiments were carried out on the database. The following three sets of data were considered: (i) 26-speaker San Diego speakers, (ii) 25-speaker Nutley speakers, and (iii) 51-speaker total population. Speaker models were based on broad phonetic classes, and Euclidean

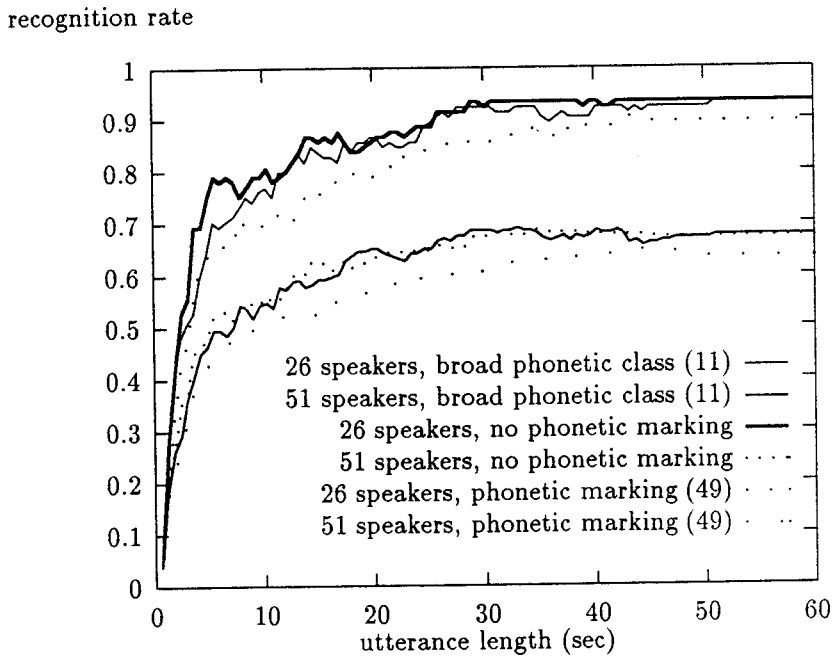


Figure 4.1: Recognition rate vs. input utterance length

distance was used as the metric. Half of the speakers are registered targets, the other half are impostors. Targets claim their own true identities (to compute detection rate), impostors claim all the registered target identities (to compute false alarm rate). The speaker models were derived from sessions 1, 2 and 3 for Div1 data experiments, and from sessions 6, 7 and 8 for Div2 data experiments. Test data came from sessions 4 and 5 for Div1 experiments and 9 and 10 for Div2 experiments. Figure 4.3 presents the receiver operating characteristics (ROC, prob(detection) vs. prob(false alarm)) for the three data sets.

4.5 Discussions

Our experiments indicate that detailed phonetic hypothesization for building speaker models and identification did not provide improvement over a non-phonetic model approach. We believe that this result may be due to inadequate

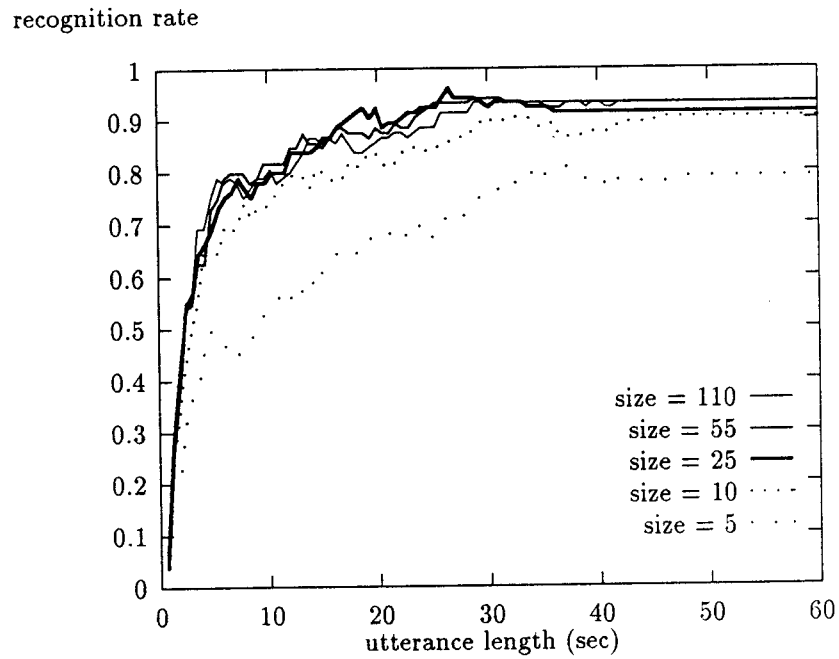


Figure 4.2: Recognition rate vs. input utterance length for different VQ codebook sizes

training data, and possibly due to some poor phonetic segmentation. A thorough experiment with expert-marked speech data will indeed be very revealing; such data is usually limited (resulting in poor training of speaker models) and hard to obtain. However, if orthographic transcription of the data is available, we could perform supervised recognition [27] to obtain a reasonably good phonetic marking to enable us to determine the value of this approach. A suitable candidate would be the Switchboard database [8], where the orthographic transcriptions along with word-level segmentation (guided by a pronunciation dictionary) are available.

Broad phonetic class speaker models performed as well as non-phonetic models with equivalent size codebook, and much better than phonetic models. Broad phonetic models also provide computational advantage in the distortion computation portion because of the smaller size codebooks (11 10-element codebooks

Session No.		No. of speakers correctly identified			
Training	Test	A'	A	B'	B
1, 2, 3	4	25	22	24	20
1, 2, 3	5	24	20	24	18
6, 7, 8	9	24	23	23	24
6, 7, 8	10	24	22	22	21
Average		93.3%	83.7%	89.4%	79.8%

- A : Non-phonetic
- B : Detailed Phonetic (49 phones)
- A' and B' are noise suppressed versions of A and B

Table 4.3: Identification results: 26-speaker San Diego data

vs. 110-element codebook); but the computational burden of speaker independent recognition will have to be taken into account in the overall computational requirements. Noise suppression proved to be very valuable indeed. The average correct identification increased by 10%. We were surprised that it improved the results for even the relatively clean data from San Diego speakers. Preliminary speaker verification experiments provided very encouraging results compared to other published results [15]. By incorporating a speaker discrimination scheme specific to the verification paradigm, we hope to improve our performance.

Session No.		No. of speakers correctly identified			
Training	Test	A'	A	B'	B
1, 2, 3	4	36	31	34	28
1, 2, 3	5	31	25	31	29
6, 7, 8	9	37	33	34	30
6, 7, 8	10	33	29	30	27
Average		67.2%	57.8%	63.2%	55.9%

- A : Non-phonetic
- B : Detailed Phonetic (49 phones)
- A' and B' are noise suppressed version of A and B

Table 4.4: Identification results: 51-speaker complete data

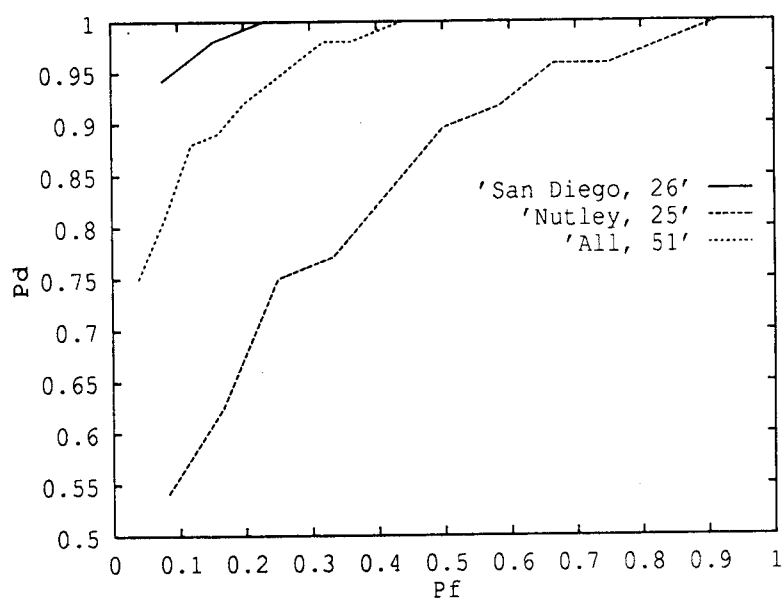


Figure 4.3: Open set speaker verification ROC

Chapter 5

Gaussian Mixture

5.1 Introduction

While vector quantization makes it possible to model speech with non-parametric discrete distributions, a severe loss of information may occur. Using VQ modeling, we only know the centroid of each cluster, but have no idea about the variance of each cluster or the weighting of each cluster compared to the whole training set. In contrast, continuous parameter modeling is able to preserve more acoustic information, at the cost of more computational complexity. Gaussian distribution is commonly used in modeling random processes; however, a single Gaussian distribution can only model unimodal behavior, thus resulting in modeling inaccuracies when the random process under consideration has more than one mode. In the case of speaker identification, each speaker model usually has more than one mode (because different phonetic events will cluster into different modes), and it is thus very clear that single mode Gaussian distributions are not adequate in building speaker models. If single mode Gaussian distributions are used to build speaker models, the resulting failure to model multimode behavior can easily destroy any leverage from the additional acoustic information gained from continuous modeling.

Because of this trade-off between discrete models (VQ) and continuous models (Gaussian), a modeling approach (Gaussian Mixture) that combines VQ and Gaussian has been proposed [4]. It has been first used in HMM speech recognition to model the observation distributions of those states with multimode behavior, which cannot be properly modeled by a single Gaussian distribution. It has also been applied to the speaker identification problems [26] [22] [21] [20], where although the VQ approach can describe a speaker's feature space quite well, the Gaussian Mixture approach can provide more **texture** to the models.

5.2 Model Parameters

The model is described by a mixture of finite number of pdf's, each pdf is a Gaussian distribution. Suppose there are M such pdf's (cluster) for a model. Each cluster is represented by Gaussian parameters: $s_m = \{c_m, \mu_m, R_m\}, m = 1, \ldots, M$, then the whole model is represented by

$$\Lambda = \{c_m, \mu_m, R_m | m = 1, 2, \ldots, M\} \tag{5.2.1}$$

where c_m, μ_m, R_m are respectively the cluster weight, the mean vector, and the covariance matrix of the **m-th** cluster. It has to satisfy the probability constraint: $\sum_{m=1}^{M} c_m N(x_t; \mu_m, R_m) = 1$. The probability of a feature vector x_t given a speaker model Λ is

$$p(x_t|\Lambda) = \sum_{m=1}^{M} c_m N(x_t; \mu_m, R_m) \tag{5.2.2}$$

where $N(x_t; \mu_m, R_m)$ is a multivariate Gaussian probability density function of observing a feature vector x_t given model parameters: μ_m and R_m, which can

be written explicitly as

$$N(x_t; \mu_m, R_m) = \frac{1}{(2\pi)^{N/2}|R_m|^{1/2}} e^{\frac{1}{2}(x_t-\mu_m)^T R_m^{-1}(x_t-\mu_m)} \tag{5.2.3}$$

Compared to VQ modeling, Gaussian Mixture provides not only cluster centroids μ_m (as in VQ), but also cluster weights c_m and covariance matrices R_m.

5.3 Model Parameter Training Using Expectation Maximization Algorithm

A Maximum Likelihood (ML) training procedure, the Expectation Maximization (EM) algorithm can be used to train the Gaussian Mixture parameters. Suppose we have a set of training vectors, $X = \{x_t | t = 1, \ldots, T\}$, and we decide there are M mixtures in this model. The iterative algorithm is as follows:

1. Initialization: provide some initial estimate of the Gaussian mixture model

$$\Lambda^{(0)} = \{c_m^{(0)}, \mu_m^{(0)}, R_m^{(0)} | m = 1, 2, \ldots, M\} \tag{5.3.4}$$

2. Expectation: compute the posterior probability (the probability of being in cluster s_m of model $\Lambda^{(i)}$ given the observation x_t), $p(s_m | x_t, \Lambda^{(i)})$, as

$$p(s_m | x_t, \Lambda^{(i)}) = \frac{p(x_t | \mu_m^{(i)}, R_m^{(i)}) c_m^{(i)}}{\sum_{m=1}^{M} p(x_t | \mu_m^{(i)}, R_m^{(i)}) c_m^{(i)}} \tag{5.3.5}$$

   where $p(x_t | \mu_m^{(i)}, R_m^{(i)}) = N(x_t; \mu_m^{(i)}, R_m^{(i)})$.

3. Maximization: re-estimate the Gaussian Mixture parameters by adjusting the parameters, $c_m^{(i)}$, $\mu_m^{(i)}$, and $R_m^{(i)}$, such that the total log likelihood function, $L = \sum_{t=1}^{T} \log p(x_t | \Lambda)$, is maximized. The resulting re-estimation formulae are:

$$c_m^{(i+1)} = \frac{1}{T} \sum_{t=1}^{T} p(s_m | x_t, \Lambda^{(i)}) \tag{5.3.6}$$

$$\mu_m^{(i+1)} = \frac{\sum_{t=1}^{T} p(s_m|x_t, \Lambda^{(i)}) \cdot x_t}{\sum_{t=1}^{T} p(s_m|x_t, \Lambda^{(i)})} \tag{5.3.7}$$

$$R_m^{(i+1)} = \frac{\sum_{t=1}^{T} p(s_m|x_t, \Lambda^{(i)}) \cdot (x_t - \mu_m)(x_t - \mu_m)^T}{\sum_{t=1}^{T} p(s_m|x_t, \Lambda^{(i)})} \tag{5.3.8}$$

4. Check convergence criterion: if converge, stop; if not, goto step 2.

5.4 Result

We used feature vectors after noise reduction (Chapter 3) and bandpass liftering (Chapter 7). Diagonal covariance matrices were used in Gaussian Mixture models instead of full covariance matrices. The same number of clusters (30 clusters) was used in each speaker model for both VQ and Gaussian Mixture; therefore the model size for the Gaussian Mixture model is about twice that of a VQ model. We compare the Gaussian Mixture modeling with VQ modeling. The results are provided for the King database, 26 speakers, for both within and across the great divide.

- Gaussian Mixture modeling: Table 5.1 and Table 5.2.

training session	test session	recognition rate
1 2 3	4	25/26
1 2 3	5	25/26
6 7 8	9	26/26
6 7 8	10	24/26
average		96.2%

Table 5.1: Identification results using Gaussian mixture, within the great divide

- VQ modeling: Table 5.3 and Table 5.4.

As shown above, Gaussian Mixture does provide better performance in the "within the great divide" test; however, in the "across the great divide" test,

training session	test session	recognition rate
6 7 8	4	6/26
6 7 8	5	8/26
1 2 3	9	3/26
1 2 3	10	4/26
average		20.2%

Table 5.2: Identification results using Gaussian mixture, across the great divide

training session	test session	recognition rate
1 2 3	4	24/26
1 2 3	5	24/26
6 7 8	9	23/26
6 7 8	10	24/26
average		91.3%

Table 5.3: Identification results using VQ, within the great divide

VQ performs a little better.

Since ISDCN (refer to Chapter 9) can improve the "across the great divide" performance on a VQ system, we suspect that it can also be applied to the Gaussian Mixture system. By replacing the VQ centroids in the normalization codebook with Gaussian Mixture centroids (μ_m), we can perform the ISDCN algorithm exactly the same as in the VQ system. Here we do not consider the distortion in covariances, it has been reported [11] that variances are less affected (than means) under change of environments. The identification results after performing ISDCN are:

training session	test session	recognition rate
6 7 8	4	6/26
6 7 8	5	9/26
1 2 3	9	3/26
1 2 3	10	5/26
average		22.1%

Table 5.4: Identification results using VQ, across the great divide

- Gaussian Mixture modeling: Table 5.5 and Table 5.6.

training session	test session	recognition rate
1 2 3	4	25/26
1 2 3	5	25/26
6 7 8	9	26/26
6 7 8	10	23/26
average		95.2%

Table 5.5: Identification results using Gaussian mixture, after ISDCN, within the great divide

training session	test session	recognition rate
6 7 8	4	14/26
6 7 8	5	13/26
1 2 3	9	10/26
1 2 3	10	8/26
average		43.3%

Table 5.6: Identification results using Gaussian mixture, after ISDCN, across the great divide

- VQ modeling: Table 5.7 and Table 5.8.

training session	test session	recognition rate
1 2 3	4	25/26
1 2 3	5	25/26
6 7 8	9	26/26
6 7 8	10	26/26
average		98.1%

Table 5.7: Identification results using VQ, after ISDCN, within the great divide

As shown above, ISDCN can also improve the "across the great divide" performance as much as it can in VQ. (Although the "within the great divide" performance degrades a little bit, this perturbation is not unusual when the performance is already high before normalization).

training session	test session	recognition rate
6 7 8	4	14/26
6 7 8	5	14/26
1 2 3	9	11/26
1 2 3	10	13/26
average		50%

Table 5.8: Identification results using VQ, after ISDCN, across the great divide

Chapter 6

Channel Normalization Using Stereo Database

6.1 Introduction

The easiest way to normalize channel variations is to "learn" the channel characteristics and then compensate it according to what we have learned. The question is how to characterize the channels?

We can put the same signals into different channels and then measure the output signals; of course, the output signals will be different because of the different channel effects. From these different output signals we can learn the characteristics of different channels. For example, we can compute the spectra of these output signals, and then compare them with one another. This way we can learn that, for example, channel "A" amplifies low frequency components while channel "B" amplifies high frequency components; we can measure the different degrees of amplification and then compensate the spectra.

6.2 The Channel Model

Since we use cepstral coefficients as features, the channel effects become "additive". As shown in Figure 6.1, the signal spectrum $S(f)$ passes through channel

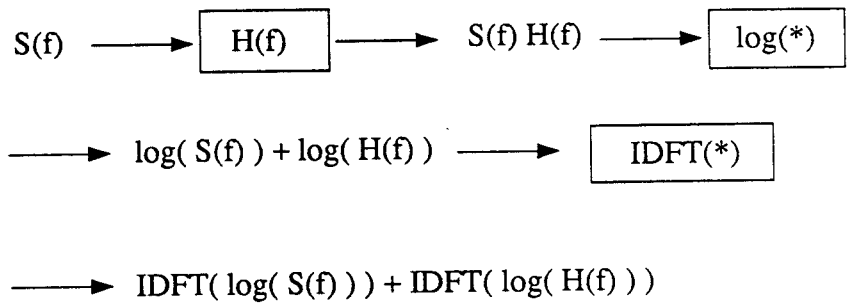


Figure 6.1: Cepstrum after channel effect

$H(f)$ and becomes $S(f)H(f)$. Cepstrum is the inverse Fourier transform of log spectrum, so the channel affected Cepstrum is $\mathrm{IDFT}(S(f)) + \mathrm{IDFT}(H(f))$.

Now that the channel effect $H(f)$ is an additive term, we can simply estimate this term and then subtract to compensate for it.

In the following sections, we will introduce the stereo database used for the channel characterization and two different normalization schemes.

6.3 Stereo Database

To estimate the channel effect $H(f)$, a stereo database is needed. By stereo database we mean that for one utterance, two simultaneous recordings are made. This way we have the same input utterances to both channels (e.g. two different microphones with different frequency responses); and then get two different output recordings. These two recordings are of the same length, and can be compared "sample by sample".

A portion of the CSR (Continuous Speech Recognition) database was used in our experiments. CSR is a stereo database, studio quality, recorded by two different microphones simultaneously (Sennheiser vs. Crown or SONY; Sennheiser is a better microphone). We used 30 speakers in this particular experiment

(including male and female), each speaker was asked to speak 40 adaptation sentences (designed to be phonetically rich and balanced). There are totally 30 (speakers) * 40 (adaptation sentences) * 2 (microphones) = 2400 sentences. Figure 6.2 and 6.3 shows the waveforms of the same utterance recorded by two different microphones, it is clear that Sennheiser's waveform is better than Crown's.

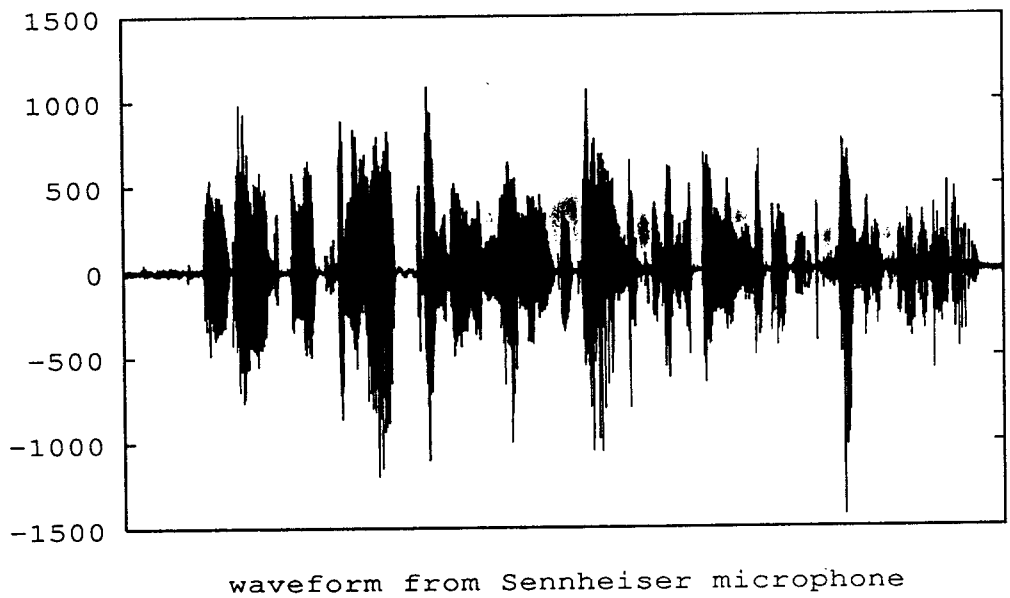


Figure 6.2: Sennheiser microphone waveform

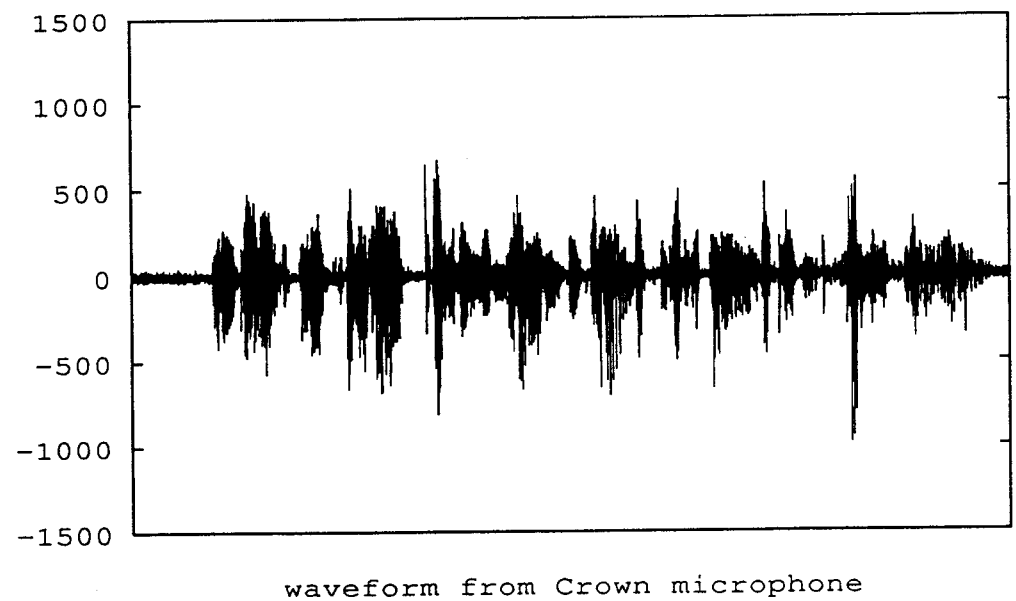


Figure 6.3: Crown microphone waveform

CSR is a much "easier" database compared to "King" or "Total-Voice" database. It is:

- Studio quality: as opposed to telephone quality.
- Including both male and female: as opposed to all male; for speaker identification, it is easier to tell form male to female than from the same sex.
- Phonetically balanced: as opposed to spontaneous speech; it contains rich phonetic contents for training and recognition.

We use the first 10 Sennheiser sentences for speaker model training, and the other 30 Crown (or SONY) sentences for testing. For each test sentence, an identification decision was made. It would be too easy if we use all 30 sentences to make one identification decision.

When the first 10 Sennheiser sentences were used for training and the other 30 Sennheiser sentences were used for testing (thus the training and testing data are from the same microphone, they are still different sentences with different speech contents), then 100% identification rate was achieved for all 30 (speakers) * 30 (sentences) = 900 tests. In the following sections, the experiments are exclusively "cross microphone" tests.

6.4 SNR Dependent Normalization

In Figure 6.1, suppose the cepstrum from mic-1 is

$$\text{IDFT}(log(S(f))) + \text{IDFT}(log(H_1(f))) \tag{6.4.1}$$

and the cepstrum from mic-2 is

$$\text{IDFT}(log(S(f))) + \text{IDFT}(log(H_2(f))) \tag{6.4.2}$$

The difference (normalization vector) between each stereo pair of cepstra can be represented as:

$$\text{IDFT}(log(H_2(f))) - \text{IDFT}(log(H_1(f))) \tag{6.4.3}$$

and they should be all the same (at least similar) for all the frames. However, this is not the case. The normalization vectors are different for different frames.

We can use an idea borrowed from speech coding - "perceptual weighting". In the analysis by synthesis coding process, speech residual (error) spectra are weighted less at high SNR and weighted more at low SNR. This is because at frequency bins with high SNR, the error is likely to be masked by the strong signal, so the error is not as "perceivable", therefor weighted less; while at low SNR, it is not masked and thus needs to be weighted more.

Following this concept, we can classify the normalization vectors (6.4.3) according to their segmental SNR's.

$$\vec{n}_{snr_i} = \frac{\sum_{\text{all training vectors}} \sum_{\text{all snr}} (\vec{c}_2 - \vec{c}_1)\delta(snr - snr_i)}{\sum_{\text{all training vectors}} \sum_{\text{all snr}} \delta(snr - snr_i)} \tag{6.4.4}$$

where $\vec{c}_1$ is the cepstrum vector from mic-1, and $\vec{c}_2$ is its counterpart from mic-2. $\vec{n}_{snr_i}$ is the normalization vector for frames with SNR at level snr_i; $\delta(x) = 1$ only when $x = 0$, otherwise it is 0. Basically we collect all the $\vec{c}_1$ and $\vec{c}_2$ pairs with SNR = snr_i, the normalization vector $\vec{n}_{snr_i}$ is computed by averaging all the difference vectors $\vec{c}_2 - \vec{c}_1$ at this SNR level. Then we can use the normalization vectors to normalize test vectors according to their SNR values.

6.5 Codeword Dependent Normalization

Similarly, we can classify the normalization vectors according to the codeword clusters in the speaker model codebooks. This requires a little more computation because we have to find which cluster the training pair, $\vec{c}_1$ and $\vec{c}_2$, belongs to.

The reason that we classify the normalization vectors according to their codeword clusters is because similar feature vectors are likely to be affected by the channel in the same way. If we cluster the difference vectors, $\vec{c}_2 - \vec{c}_1$, we can see that they cluster pretty much according to how $\vec{c}_1$ and $\vec{c}_2$ cluster.

$$\vec{n}_{cw_i} = \frac{\sum_{\text{all training vectors}} \sum_{\text{all cw}} (\vec{c}_2 - \vec{c}_1)\delta(cw - cw_i)}{\sum_{\text{all training vectors}} \sum_{\text{all cw}} \delta(cw - cw_i)} \tag{6.5.5}$$

Basically we collect all $\vec{c}_1$ and $\vec{c}_2$ pairs that belong to cluster cw_i; the normalization vector $\vec{n}_{cw_i}$ is computed by averaging all the difference vectors $\vec{c}_2 - \vec{c}_1$ in this codeword cluster. Then we can use the normalization vectors to normalize test vectors according to their cluster allocations.

6.6 Results and Conclusions

As shown in Table 6.1, both normalization algorithms perform well, error was reduced from 35 (out of 900) to 9 and 3 for SDN and CDN respectively. Because this is an easy database, with performances this close, it is difficult to decide which algorithm performs better by looking at identification rates only. To compare the performance more precisely, we defined another measure - distance ratio:

$$\text{distance ratio} = \frac{\text{d(test utterance, correct model)}}{\text{d(test utterance, incorrect models)}} \tag{6.6.6}$$

Smaller distance ratio means winning by a "larger" margin, which is better. As shown in Table 6.1, CDN indeed performs better than SDN judging from both identification rate and distance ratio. This comes at a price - more computations.

Equipped with a stereo database, it is very easy to characterize the channel differences. We presented two ways to classify the normalization vectors - SNR dependent and codeword dependent. They all perform pretty well. There exist other modeling methods, it should be easy to come up with different modifications according to different recognition structures.

We can actually make the normalization vectors depend on **both** SNR and codeword, thus increase the modeling resolution. However, there exists a trade off between channel modeling resolution and the amount of training data. Higher resolution can characterize channel statistics more precisely, but it also means fewer training data. Suppose we have 10000 training frames, if we have 20 SNR bins, in average 500 frames will be available to train the normalization vector of each SNR bin. However, if 200 SNR bins were used, in average only 50 frames will be available to train each normalization vector, which may be insufficient. Higher resolution is better only when there exist enough training data.

	baseline	SDN	CDN
ID-rate	96.111%	99%	99.67%
distance-ratio	0.64899	0.63238	0.61414

Table 6.1: Identification results of baseline vs. SDN vs. CDN

Chapter 7

Bandpass Liftering

7.1 Introduction

Most speech recognition systems use some type of spectral analysis as front ends. The two spectral analysis methods most frequently used are filter bank analysis and linear prediction.

Filter bank approaches typically use a bank of 8 to 32 bandpass filters, uniformly spaced or warped according to the sensitivity of the ear to different frequency bands. These bandpass filters are generally highly overlapped. One advantage of the filter bank approach (or other FFT based methods) is that each frequency channel is treated independently, i.e. there are no global constraints on the spectrum, thus there are no constraints on the filter bank outputs. Distortions caused by different microphones, different background noises, and different transmission channels can be dealt with quite easily without complication caused by model assumptions. However, time and frequency resolution is a big limitation on the filter design: we cannot have high resolution on both time and frequency. The filter bank approach is also very susceptible to variations in speech excitation, such as the change of fundamental frequency, which is inevitable in natural speech from utterance to utterance.

The alternative to filter bank analysis is linear prediction. It assumes an all pole model of the speech production mechanism to model the short time speech spectrum. The advantage of this approach is that it leads to a consistent and meaningful resolution of the source-tract interaction, and therefore it alleviates the excessive sensitivity to fundamental frequency variations in speech excitation. However, this approach introduces its own artifacts.

Suppose we use a Gaussian process to drive an all-pole filter, and then analyze the output waveform to compute the cepstral coefficients. We also collect mixed data from the real speech. If we plot the ratio of the variances of the simulated fixed filter data to the variances of the mixed data, it is clearly seen that the ratio increases as the cepstrum index increases. The increase in variance ratio with increasing cepstral coefficient index indicates the **diminishing** discriminating power of the higher order coefficients. This shows that the variability of higher order coefficients is an inherent disadvantage of the linear prediction process.

Figure 7.1 shows the variance analysis for all 20 dimensions of cepstral coefficients. It is clear that both within class and inter-class variances decrease as cepstrum index increases. We now consider the variable:

$$\text{F-ratio} = \frac{\text{inter-class variance}}{\text{within class variance}} \tag{7.1.1}$$

Larger F-ratio means better discrimination power. As shown in Figure 7.1, the F-ratio is larger in the middle of the cepstrum and lower on both sides.

On the other hand, channel variation often affects the lower order coefficients much more than higher order coefficients. For example, the effect of different channel frequency responses is usually most prominent in the first couple of cepstral coefficients.

7.2 Bandpass Liftering

Because the higher order coefficients have less discriminating power and the lower order coefficients are more susceptible to channel variation, as an engineering compromise, we use a window function to de-emphasize both higher and lower order coefficients. We call this approach bandpass liftering [16]. The window used is:

$$w(k) = 1 + h \sin(\pi k / L) \tag{7.2.2}$$

where $h = L/2$, $k = 1, 2, \ldots, L$ and $w(k) = 0$ for other k.

We use 20-dimension cepstral coefficients as feature vectors, the bandpass liftering weighting is shown in Figure 7.2.

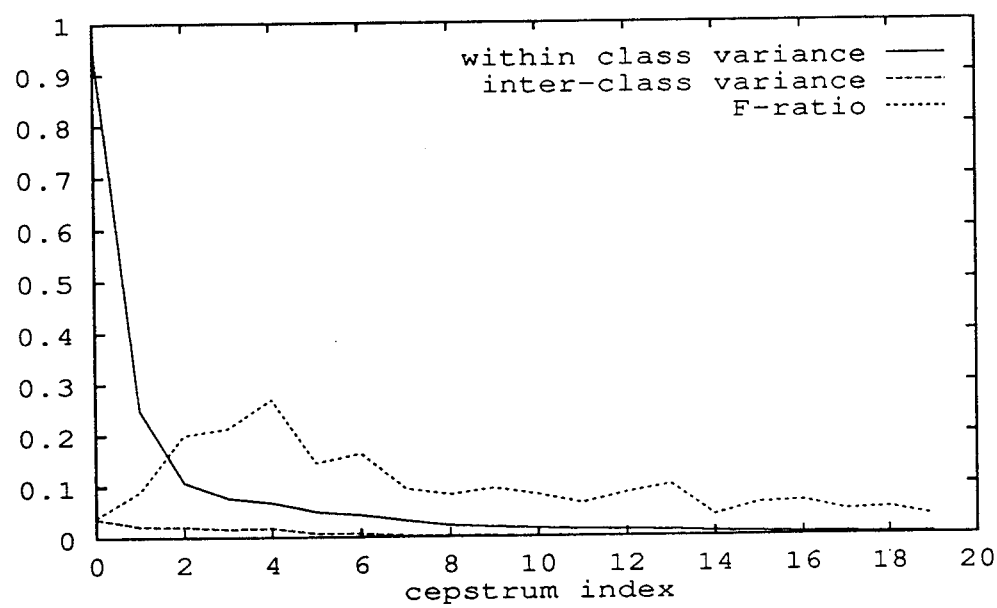


Figure 7.1: Variance analysis for cepstrum

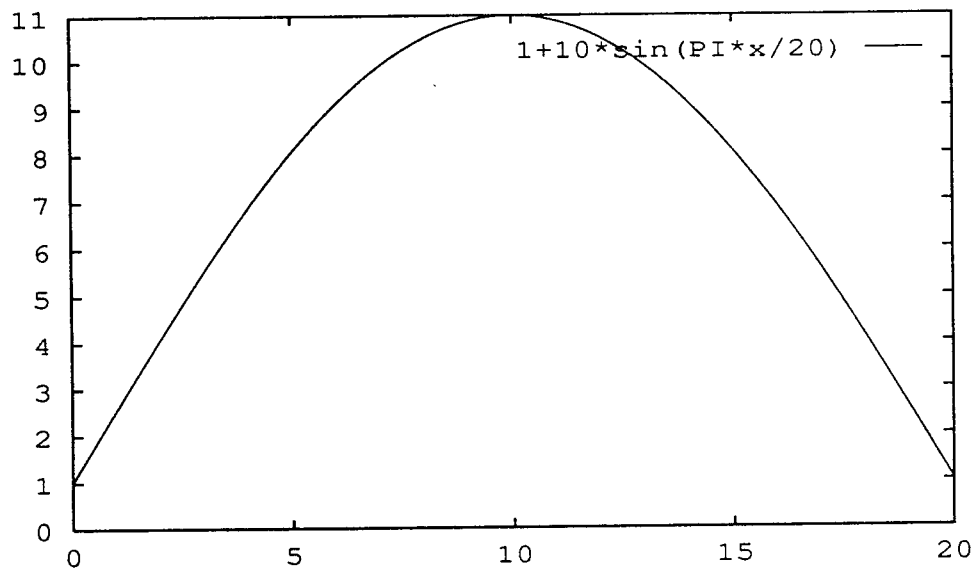


Figure 7.2: Bandpass liftering window

7.3 Results and Conclusion

We compared results before and after bandpass liftering, for both within and across the great divide. The results are on the King database, 26 San Diego speakers.

- Noise reduction, no weighting of coefficients: Table 7.1 and Table 7.2.

training session	test session	recognition rate
1 2 3	4	25/26
1 2 3	5	25/26
6 7 8	9	23/26
6 7 8	10	23/26
average		92.3%

Table 7.1: Identification results before bandpass liftering, within the great divide

- Noise reduction, and bandpass liftering: Table 7.3 and Table 7.4.

training session	test session	recognition rate
6 7 8	4	2/26
6 7 8	5	2/26
1 2 3	9	2/26
1 2 3	10	2/26
average		7.7%

Table 7.2: Identification results before bandpass liftering, across the great divide

training session	test session	recognition rate
1 2 3	4	24/26
1 2 3	5	24/26
6 7 8	9	23/26
6 7 8	10	24/26
average		91.3%

Table 7.3: Identification results after bandpass liftering, within the great divide

We can see the obvious improvement in the "across the great divide" training-test case; however, the "within the great divide" performance suffers a little bit, which is common in channel normalization operations.

More extensive experimental results are presented in Chapter 10.

training session	test session	recognition rate
6 7 8	4	6/26
6 7 8	5	9/26
1 2 3	9	3/26
1 2 3	10	5/26
average		22.1%

Table 7.4: Identification results after bandpass liftering, across the great divide

Chapter 8

RASTA Filtering

8.1 Introduction

Although a stereo database is a very powerful tool for channel normalization, it is not usually available; therefore the algorithms introduced in Chapter 6 are not applicable.

RASTA (RelAtive SpecTrA) filtering does not require stereo database; yet in our evaluation, it performs better than algorithms that require stereo database, Table 8.2. Moreover, the idea of RASTA is very simple, and the computational complexity is very low.

8.2 RASTA Filtering

Channel variations come from different frequency responses of the channels and also from background noises. Compared to speech signals, these channel effects are "stationary" or "slowly varying". We can use a highpass filter to remove the lower frequency components of the spectrum and still preserve the speech. Of course, we cannot remove too much of the lower frequency components, they may contain important speech information. The proper design of the filter requires experiments.

To be able to "filter out" the slowly varying channel effect, $H(f)$, it must be an "additive" term instead of a "multiplicative" term as in the case of the corrupted spectrum, $S(f)H(f)$; Thus the operation log is required on spectrum for this conversion. Figure 8.2 shows RASTA filtering being applied to log spectrum. If RASTA is applied on spectrum directly, it will not work.

We use cepstrum as feature vector, which is the inverse Fourier transform of log spectrum, as shown in Figure 8.3. The channel effect term is an "additive" term, RASTA filtering can be applied directly on the cepstral coefficients.

We use a bandpass filter:

$$H(z) = \frac{0.2 + 0.1z^{-1} - 0.1z^{-3} - 0.2z^{-4}}{(1 - 0.98z^{-1})z^{-4}} \tag{8.2.1}$$

with its frequency response shown in Figure 8.1.

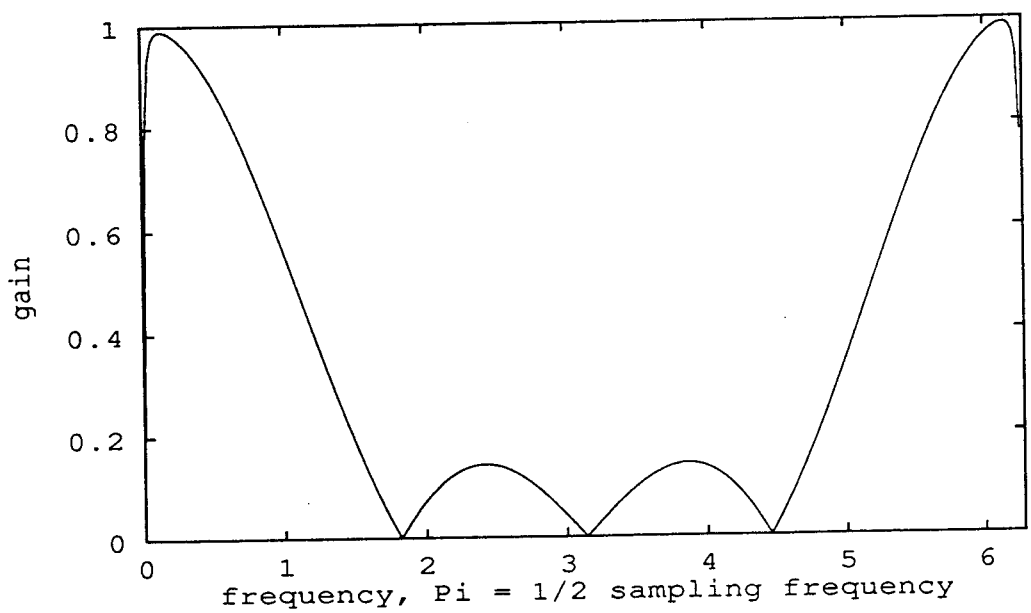


Figure 8.1: Frequency response of RASTA bandpass filter

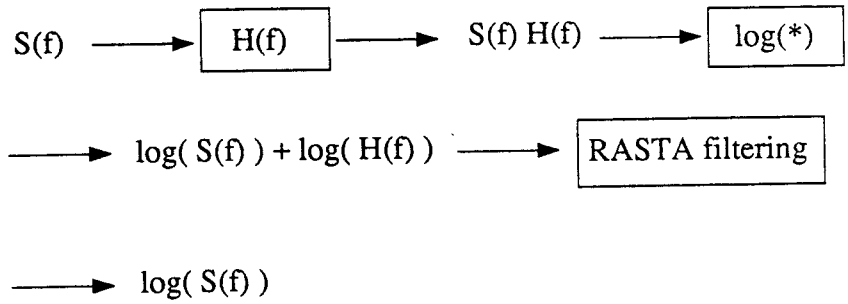


Figure 8.2: The idea of RASTA filtering

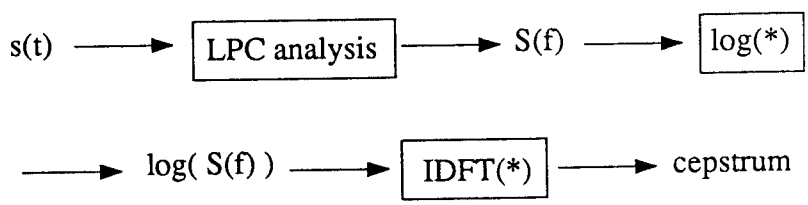


Figure 8.3: The computation of cepstrum

We collect all the cepstrum vectors in time order, $\vec{c}^t, t = 0, 1, 2, \ldots, N$, and then filter each dimension i, $i = 0, \ldots, 19$, respectively with the RASTA filter; i.e. feed the sequence - $c_i^t, t = 0, 1, 2, \ldots, N$ to the filter, and do this for all $i, i = 0, \ldots, 19$.

8.3 Scattergrams

One way to measure the effectiveness of channel normalization algorithms is by inspecting the "scattergrams". However, stereo database, where speech data transmitted through two channels are recorded simultaneously, is required to compute the scattergrams. In our experiments, we used the CSR (Continuous Speech Recognition) database, where each utterance is recorded by two different

microphones (Sennheiser vs. Crown or SONY) simultaneously. Because of the simultaneous recording of two channels, "sample by sample" match makes the exact comparison of channel effects possible. Here we compare the cepstral coefficients, frame by frame, dimension by dimension.

Ten sentences were used in the following analysis. The average length of each sentence is about 5 seconds. For 10 ms frames length, 16 KHz sampling rate, it amounts to $10*5/(10*10^{-3}) = 5000$ frames. 20-dimensional cepstral coefficients were computed, c_0 to c_{19}. For each frame, we used c_i from microphone 1 as x coordinate, c_i from microphone 2 as y coordinate to construct a point in the scattergram, thus 5000 points were constructed for dimension i. From Figure 8.4 to Figure 8.43 we show the scattergrams from dimension 0 to 19, the left figure is baseline, the right figure is after RASTA filtering normalization. It is clear to see that the scattergrams after RASTA filtering are more aligned with the line: $x = y$ (the most significant change can be observed in dimension 1), which means the cepstral coefficients from two different microphones are "normalized" to be equal to each other.

It is not easy to obtain the "quantitative" improvements from RASTA filtering by simply inspecting the scattergrams, we can only know it is "better". To measure the improvements, we can compute the distance of each point to the line: $x = y$, i.e.

$$d^2 = \frac{\|\vec{c}\|^2 \|(1,1)\|^2 \sin^2(\theta)}{2} \tag{8.3.2}$$

where $\vec{c}$ is the point, θ is the angle between $\vec{c}$ and the vector $(1,1)$. We call this d^2 deviation, and the deviation values (accumulated for all 5000 points) are

listed with each scattergram. As shown, RASTA filtering improves (decreases) the deviation values.

Because of the "filtering" nature of the RASTA processing, the cepstral coefficients will become "smaller" after the processing; therefore the "deviation" measure mentioned above is **not** a fair comparison. Because the absolute values of the cepstral coefficients tend to be smaller after RASTA filtering, the distances of the points to the line: $x = y$ also tend to be smaller consequentially. This decrease in distances has nothing to do with whether the points conform to the line: $x = y$ or not. To offset this effect and have a fair comparison, we normalize the deviation with variance. We can compute the variance of all the points in each figure, (σ_x, σ_y), $\sigma = (\sigma_x + \sigma_y)/2$, and then adjust the deviation, i.e.

$$deviation_{adjusted} = \frac{deviation}{\sigma} \tag{8.3.3}$$

Now the comparison is "fair". Table 8.1 lists all the deviations and adjusted deviations for each cepstral dimension. The deviation ratios (RASTA to baseline) are listed, too. As shown in Table 8.1, although the adjusted deviation ratio is not as good as the unadjusted, they are all smaller than 1, which means RASTA filtering improves the scattergrams for all dimensions. Figure 8.44 is the diagram for deviation ratios before and after the variance adjustment. Dimension 1 shows the most significant improvement from RASTA.

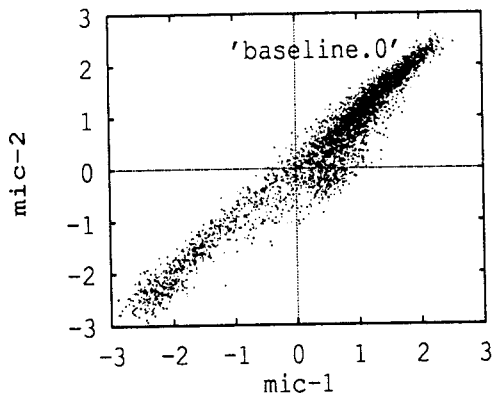


Figure 8.4: baseline, Deviation = 210.5

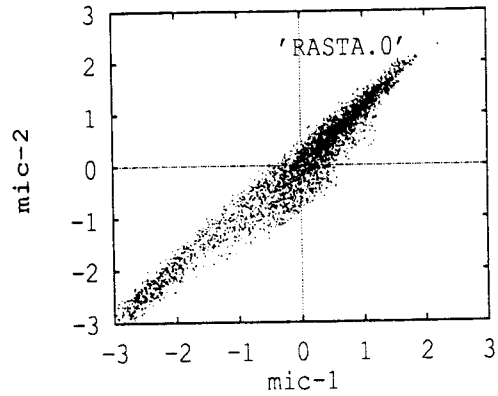


Figure 8.5: RASTA, Deviation = 163.6

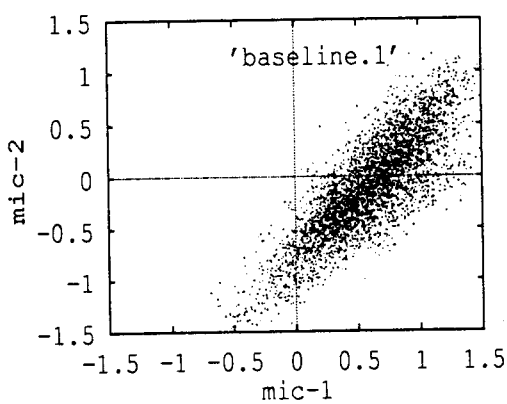


Figure 8.6: baseline, Deviation = 1305.2

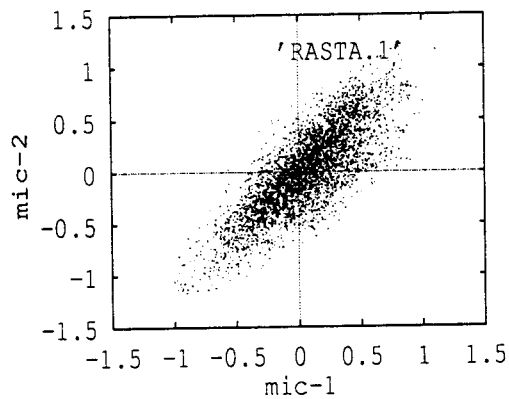


Figure 8.7: RASTA, Deviation = 124.1

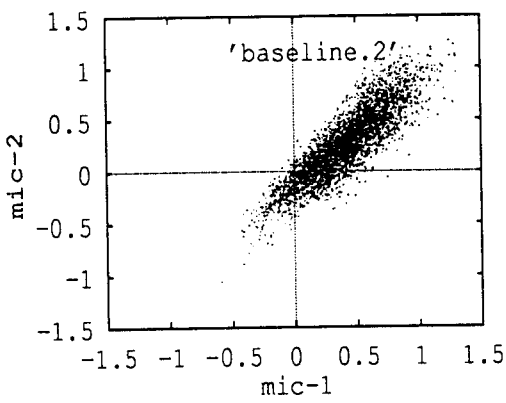


Figure 8.8: baseline, Deviation = 103.3

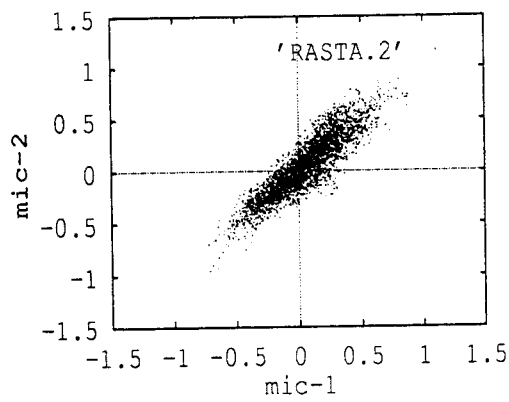


Figure 8.9: RASTA, Deviation = 32.3

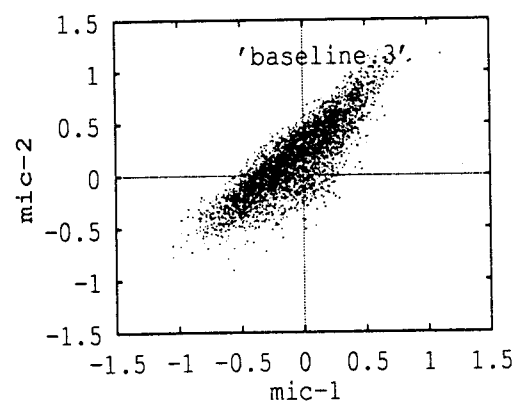


Figure 8.10: baseline, Deviation = 200.5

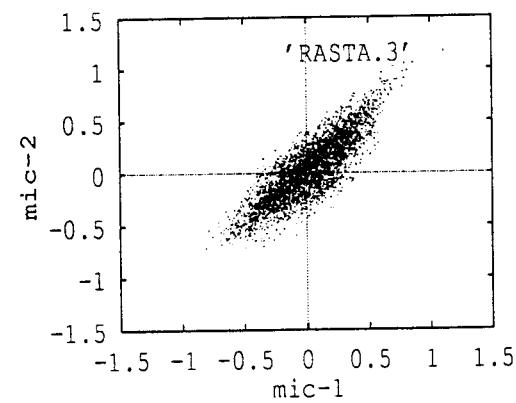


Figure 8.11: RASTA, Deviation = 51

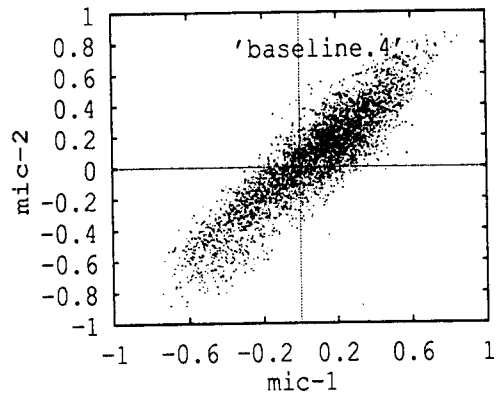


Figure 8.12: baseline, Deviation = 37.7

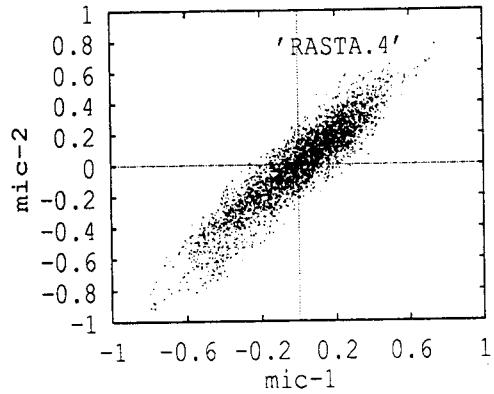


Figure 8.13: RASTA, Deviation = 21

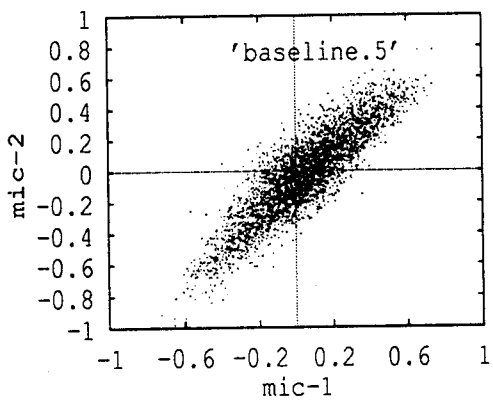


Figure 8.14: baseline, Deviation = 44.8

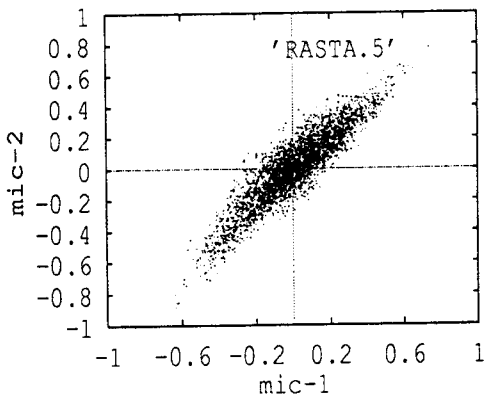


Figure 8.15: RASTA, Deviation = 18.7

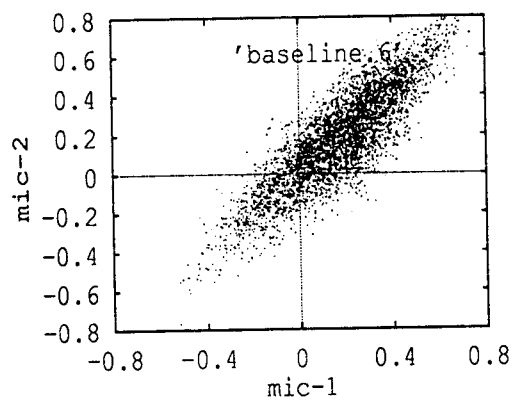


Figure 8.16: baseline, Deviation = 32.4

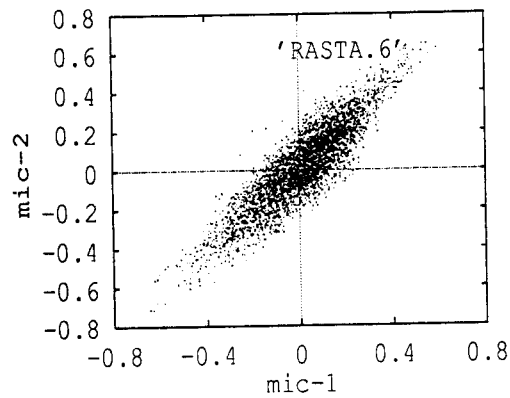


Figure 8.17: RASTA, Deviation = 19.2

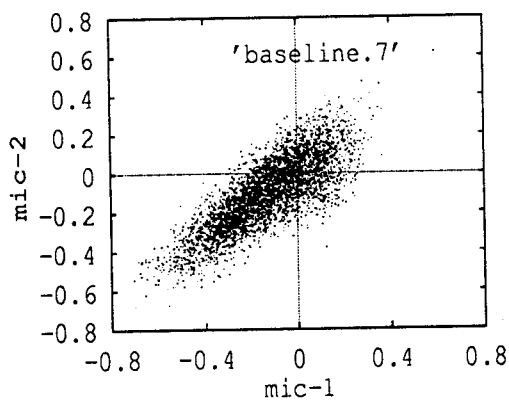


Figure 8.18: baseline, Deviation = 31

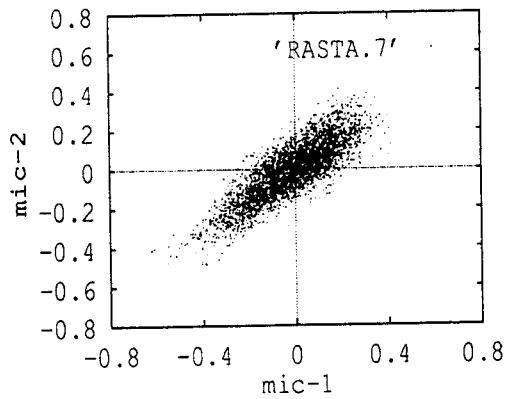


Figure 8.19: RASTA, Deviation = 16.4

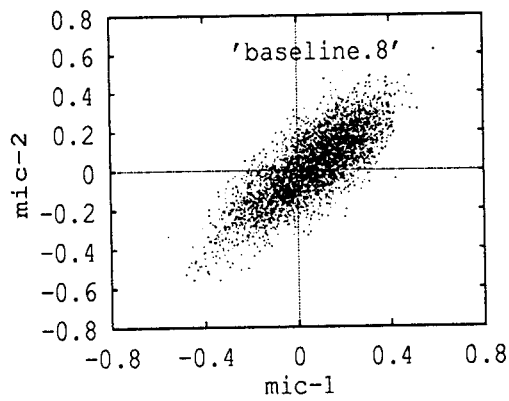


Figure 8.20: baseline, Deviation = 29.3

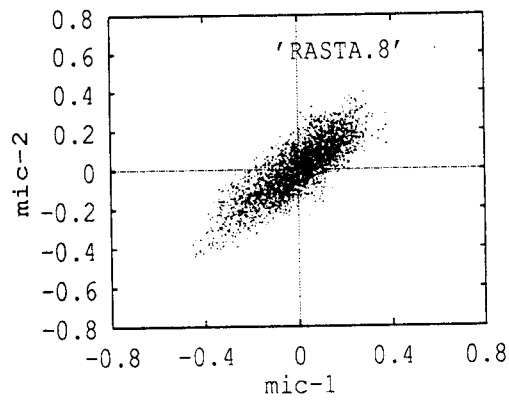


Figure 8.21: RASTA, Deviation = 13.4

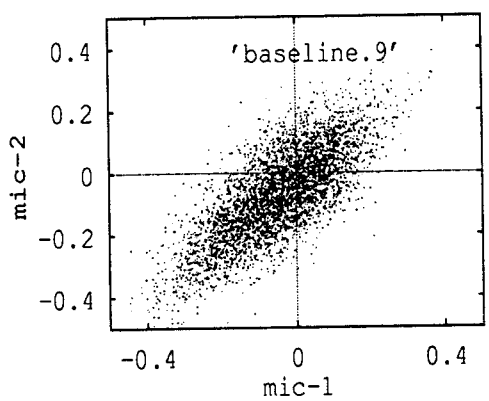


Figure 8.22: baseline, Deviation = 21

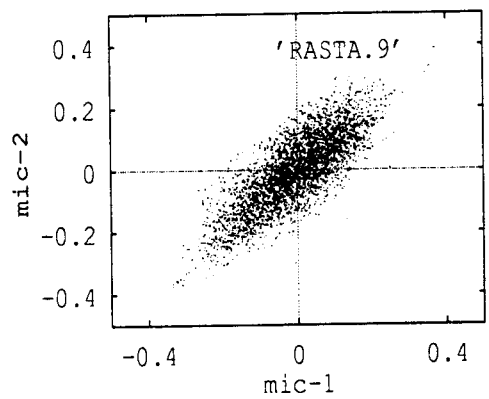


Figure 8.23: RASTA, Deviation = 10

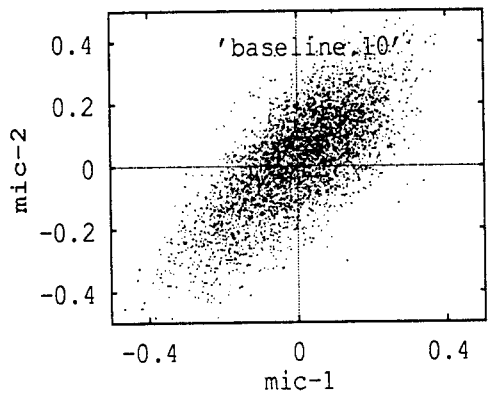


Figure 8.24: baseline, Deviation = 27.5

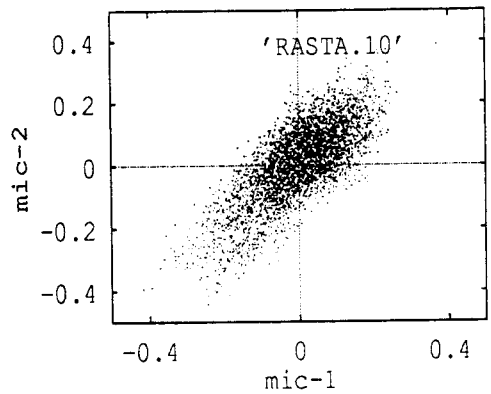


Figure 8.25: RASTA, Deviation = 14.6

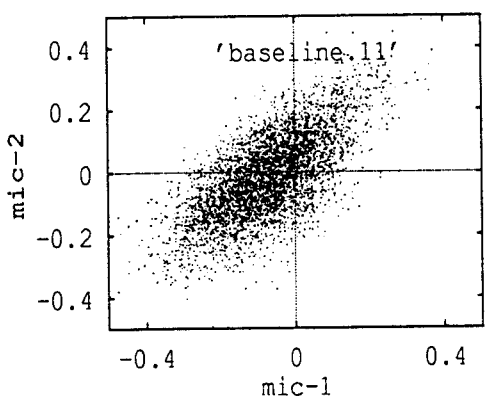


Figure 8.26: baseline, Deviation = 31.7

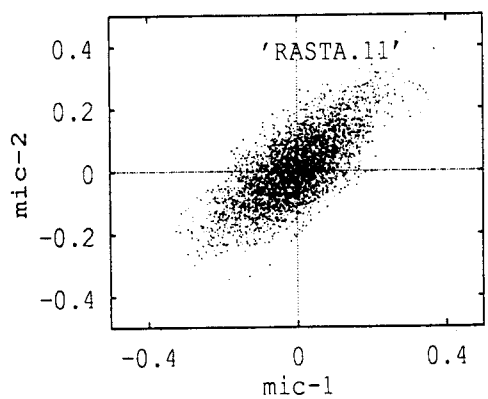


Figure 8.27: RASTA, Deviation = 13.2

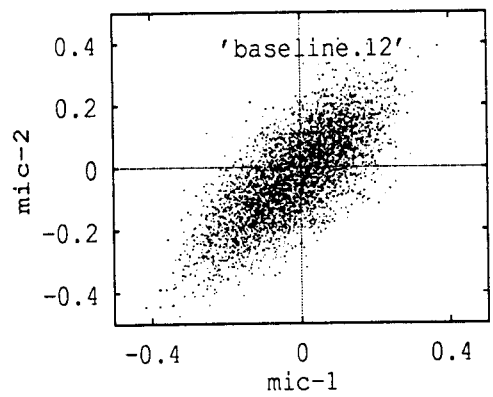


Figure 8.28: baseline, Deviation = 19.9

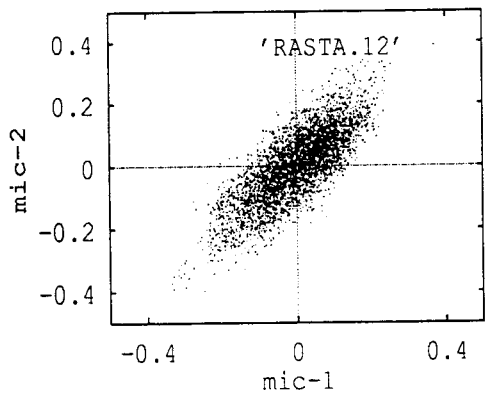


Figure 8.29: RASTA, Deviation = 10.4

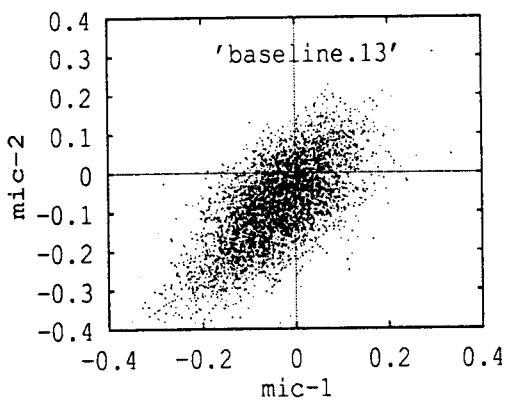


Figure 8.30: baseline, Deviation = 19.1

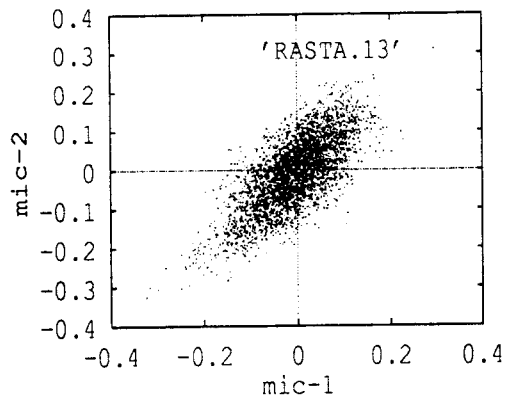


Figure 8.31: RASTA, Deviation = 8

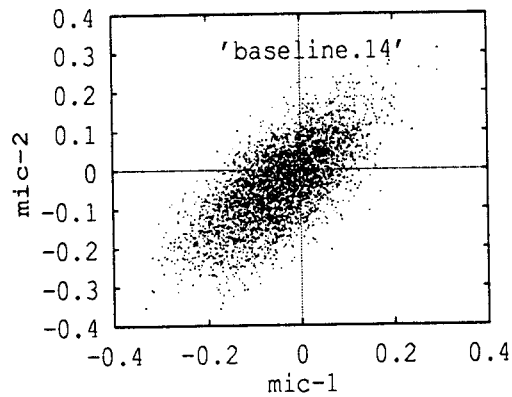


Figure 8.32: baseline, Deviation = 12.2

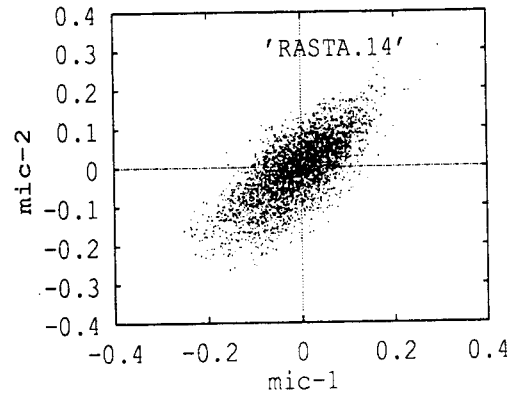


Figure 8.33: RASTA, Deviation = 6.9

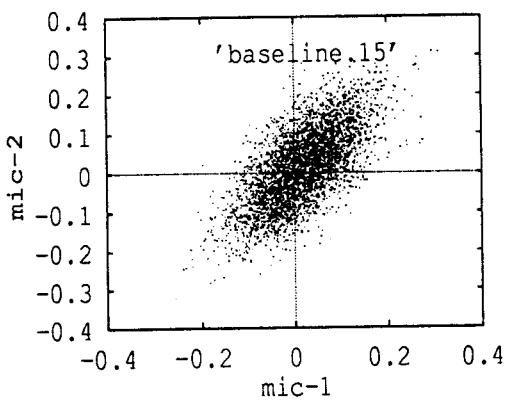


Figure 8.34: baseline, Deviation = 10.6

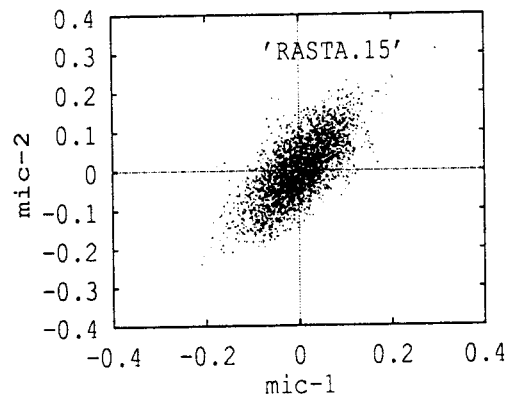


Figure 8.35: RASTA, Deviation = 5.9

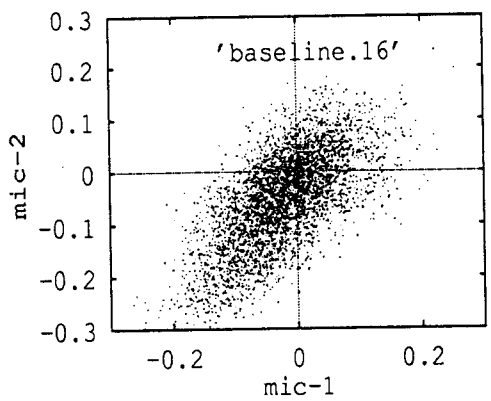


Figure 8.36: baseline, Deviation = 11.7

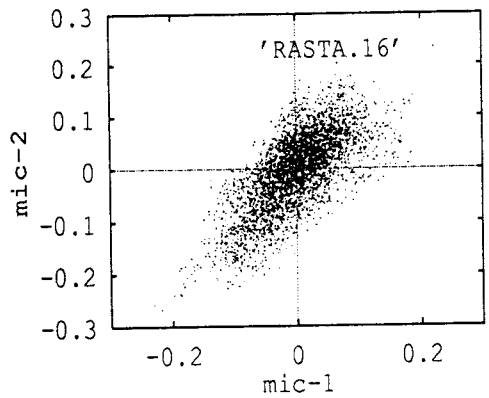


Figure 8.37: RASTA, Deviation = 5.6

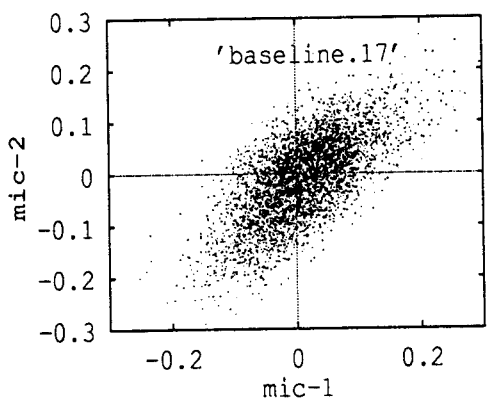


Figure 8.38: baseline, Deviation = 8.9

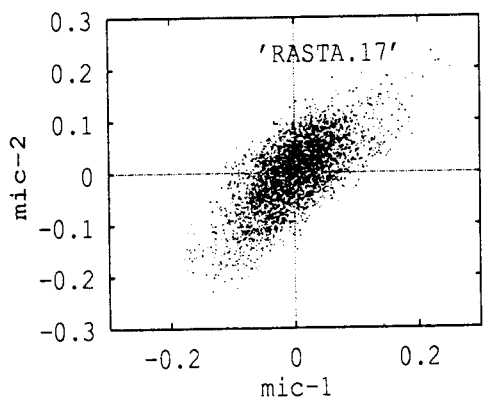


Figure 8.39: RASTA, Deviation = 4.8

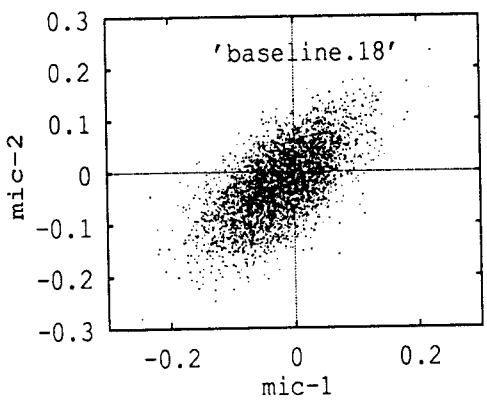


Figure 8.40: baseline, Deviation = 6.2

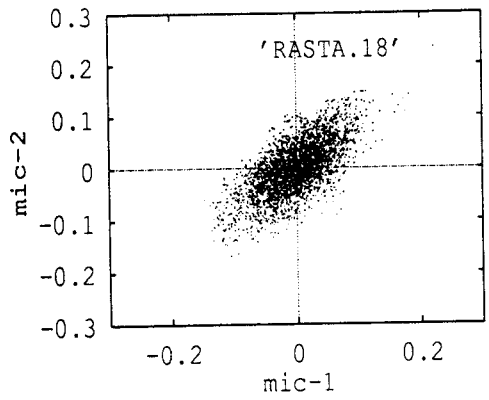


Figure 8.41: RASTA, Deviation = 3.1

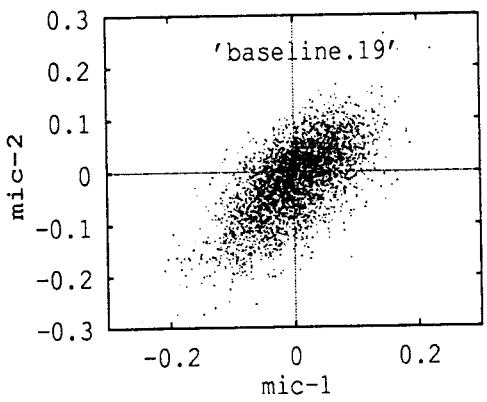


Figure 8.42: baseline, Deviation = 6.6

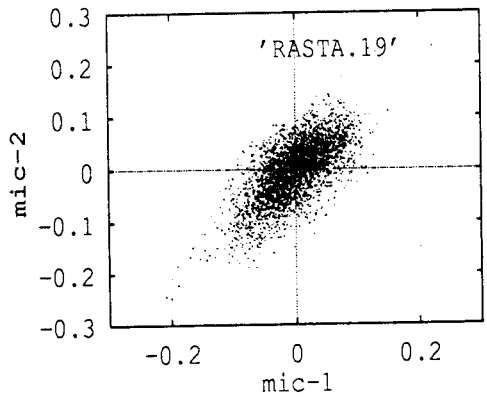


Figure 8.43: RASTA, Deviation = 3.3

	Baseline deviation		RASTA deviation		Deviation ratio	
Index	un-adjusted	adjusted	unadjusted	adjusted	unadjusted	adjusted
0	210.51	158.80	163.56	145.47	0.78	0.92
1	1305.21	7977.61	124.14	1021.88	0.10	0.13
2	103.30	1189.76	32.29	508.24	0.31	0.43
3	200.47	2058.08	50.99	726.97	0.25	0.35
4	37.66	481.46	20.96	372.07	0.56	0.77
5	44.77	772.89	18.73	449.11	0.42	0.58
6	32.38	693.58	19.17	564.26	0.59	0.81
7	30.96	1021.96	16.37	785.69	0.53	0.77
8	29.33	1182.58	13.44	812.77	0.46	0.69
9	20.95	1204.73	10.08	921.07	0.48	0.76
10	27.49	1445.22	14.64	1194.29	0.53	0.83
11	31.68	2075.94	13.24	1371.18	0.42	0.66
12	19.86	1255.89	10.45	984.49	0.53	0.78
13	19.13	2057.98	8.05	1407.45	0.42	0.68
14	12.23	1506.05	6.89	1322.53	0.56	0.88
15	10.61	1628.67	5.85	1482.21	0.55	0.91
16	11.74	1996.81	5.64	1484.82	0.48	0.74
17	8.87	1746.98	4.80	1496.69	0.54	0.86
18	6.24	1771.86	3.14	1546.79	0.50	0.87
19	6.61	1871.62	3.33	1515.66	0.50	0.81

Table 8.1: Deviations from *line* : $x = y$ before and after RASTA

8.4 Results and Conclusions

We compared the RASTA filtering with normalization algorithms using stereo database - SNR dependent normalization and codeword dependent normalization; Chapter 6. Table 8.2 shows the results. This experiment was performed on the CSR database, 30 speakers. The test conditions were exactly the same as in Chapter 6 - train on the first 10 Sennheiser sentences, test on the other 30 Crown (or SONY) sentences. The results are averaged from 900 tests.

RASTA filtering performs even better than SDN and CDN, which is quite phenomenal, because RASTA does not "learn" the characteristics of different channels as SDN and CDN do, what it does is "blindly" filter out slowly varying

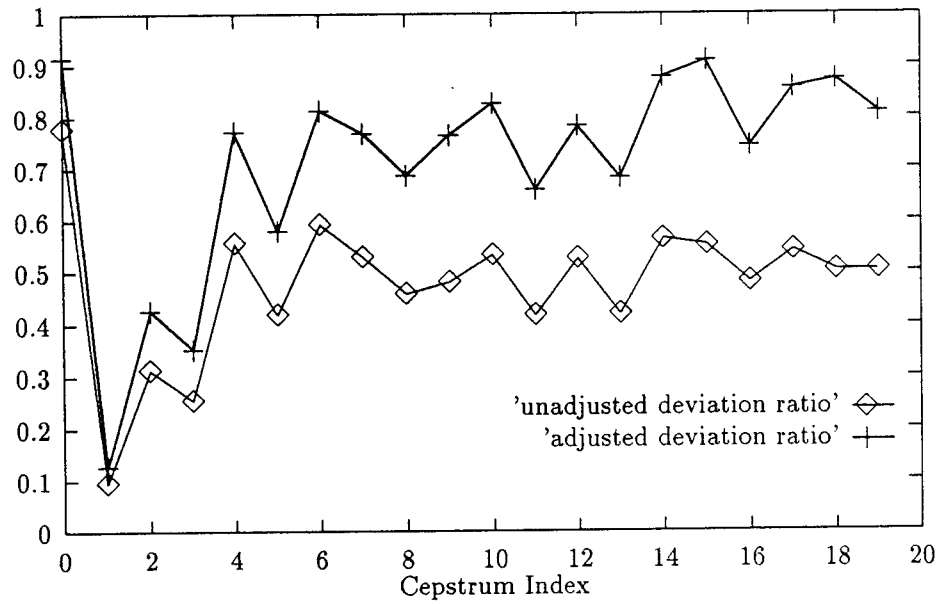


Figure 8.44: unadjusted and adjusted deviation ratios

	baseline	SDN	CDN	RASTA
ID-rate	96.111%	99%	99.67%	99.78%
distance-ratio	0.64899	0.63238	0.61414	0.73322

Table 8.2: Identification results of baseline vs. SDN vs. CDN vs. RASTA

components. The distance ratio of RASTA is a little worse than baseline; this is perceivable because larger scores improve (decrease) more than smaller scores; we refer to Chapter 11 for further discussion of this point.

More extensive results of RASTA filtering are presented in Chapter 10.

Chapter 9

ISDCN

9.1 Introduction

The purpose of this chapter is to evaluate yet another channel normalization algorithm: ISDCN (Interpolated SNR Dependent Cepstrum Normalization) [1] [2] [3]. We assume that there are two kinds of environmental variations: additive background noise and spectral tilt (introduced by linear filtering). A robust system needs to compensate for these two degradations, which are caused by collecting training and test data under different environments. ISDCN does not need a stereo database to estimate the compensation vectors; the noise and spectral tilt vectors are estimated while vector quantizing the distorted test vectors to the standard (clean) codebook by minimizing the accumulated VQ distortion iteratively. Another difference of ISDCN from traditional methods was motivated by perceptual consideration: instead of being computed independently, noise and spectral tilt vectors are combined into one single compensation vector with adjustable weights depending on instantaneous SNR. At higher SNR, more of the compensation comes from spectral tilt; while at lower SNR more of the compensation comes from noise. This idea is very similar to the perceptual weighting widely used in speech coding, where noise is masked by strong

components in the spectrum. We have tried the ISDCN algorithm on the King database, with 26 speakers, 10 sessions per speaker. Sessions 1 to 5 and sessions 6 to 10 are recorded under different environments. The identification rate of cross-environment test was improved from 8.6% to 28.8%; in the same-environment case, performance was also improved from 81.7% to 86.5%. Because both environments in the King database are noisy, after adding noise reduction pre-processing and then ISDCN channel normalization, the performance was further improved to 46.2% (across the great divide) and 89.4% (within the great divide). When bandpass liftering weighting was applied to the cepstral vectors, the performance was further improved to 50% and 98.1%.

ISDCN (Interpolated SNR Dependent Cepstrum Normalization) does not need stereo database. Basically it assumes that there are two components contributing to channel variation: background noise $\vec{n}$ and spectral tilt $\vec{q}$. The degree of contributions of these two components (to the final correction vector) depends on SNR, i.e., the correction vector is a linear combination of these two. According to this assumption, we can compute $\vec{n}$ (which is fixed through all iterations) , and estimate $\vec{q}$ iteratively by minimizing accumulated vector quantization error. Finally we get the correction vector and subtract it from the input vectors to get the normalized output vectors.

9.2 ISDCN

In the following we define all the terms used in this algorithm:

- $\vec{z}_i$, i = 0 to N-1: input frames.
- $\hat{x}_i$, i = 0 to N-1: output frames (normalized).

- $\vec{c}[i]$, i = 0 to M-1: codebook vectors, M is the codebook size.

- $\vec{w}$: correction vector.

- $f_i(x)$: interpolation function (between $\vec{n}$ and $\vec{q}$).

$$\hat{x}_i = \vec{z}_i - \vec{w}(\vec{n}, \vec{q}, SNR_i) \tag{9.2.1}$$

$$\vec{w}(\vec{n}, \vec{q}, SNR_i) = \vec{n} + (\vec{q} - \vec{n}) f(SNR_i) \tag{9.2.2}$$

$$f_i(x) = \frac{1}{1 + \exp(-\alpha_i x + \beta_i)} \tag{9.2.3}$$

Note that $f_i(x)$ interpolates between the noise $\vec{n}$ at low SNR and the spectrum tilt $\vec{q}$ at high SNR. At low SNR, $f_i(x) \approx 0$, and the correction vector $\vec{w}$ is mostly from $\vec{n}$; at high SNR, $f_i(x) \approx 1$, and $\vec{w}$ is mostly from $\vec{q}$. $f_i(x)$ is monotonic and smooth.

$\vec{n}$ can be reliably estimated by averaging noise frames (we used an energy threshold to determine noise frame). $\vec{q}$ needs to be estimated iteratively. In order to do that, we want to minimize the accumulated vector quantization error,

$$D(\vec{n}, \vec{q}, k_i) = \sum_{i=0}^{N-1} \|\vec{z}_i - \vec{w}(\vec{n}, \vec{q}, SNR_i) - \vec{c}[k_i]\|^2 \tag{9.2.4}$$

where $\vec{c}[k_i]$ is the best codeword for frame i.

Therefore, we use the following algorithm to estimate $\vec{q}$:

1. Start with an estimate for $\hat{q}^{(0)}$, and set j = 1.

2. Encode all frames, i.e. find k_i that minimize the distortions:

$$\|\vec{z}_i - \vec{w}(\vec{n}, \hat{q}^{(j-1)}, SNR_i) - \vec{c}[k_i^{(j)}]\|^2 \ , \ 0 \le i \le N-1 \tag{9.2.5}$$

3. Estimate $\hat{q}^{(j)}$ from all the frames:

$$\hat{q}^{(j)} = \vec{n} + \frac{\sum_{i=0}^{N-1}(\vec{z}_i - \vec{n} - \vec{c}[k_i^{(j)}])f(SNR_i)}{\sum_{i=0}^{N-1} f^2(SNR_i)} \tag{9.2.6}$$

4. If $\hat{q}$ has converged, stop; otherwise, goto step 2.

In the implementation, we choose $\alpha_i = \beta_i = 3$, and $\hat{q}^{(0)} = 0$. We believe that setting $\hat{q}^{(0)}$ to zero vector may not be a good choice, there should be a way to estimate a better initial value. When the spectral tilt is large, this algorithm may converge to incorrect codebook regions. If we have a good estimate for $\hat{q}^{(0)}$ as a starting point, it will converge to better regions.

9.3 Results

In the "King" database, sessions 1 to 5 and sessions 6 to 10 are recorded under different environments (different background noises and different microphone responses) - the great divide. The following results are on 26 San Diego speakers, for both within and across the great divide.

- Raw data, no noise reduction, no ISDCN: Table 9.1 and Table 9.2.

training session	test session	recognition rate
1 2 3	4	18/26
1 2 3	5	19/26
6 7 8	9	24/26
6 7 8	10	24/26
average		81.7%

Table 9.1: Identification results, baseline, within the great divide

- After channel normalization (ISDCN), no noise reduction: Table 9.3 and Table 9.4.

- Noise reduction and ISDCN: Table 9.5 and Table 9.6.

training session	test session	recognition rate
6 7 8	4	2/26
6 7 8	5	4/26
1 2 3	9	1/26
1 2 3	10	2/26
average		8.7%

Table 9.2: Identification results, baseline, across the great divide

training session	test session	recognition rate
1 2 3	4	21/26
1 2 3	5	21/26
6 7 8	9	24/26
6 7 8	10	24/26
average		86.5%

Table 9.3: Identification results, ISDCN, within the great divide

- Bandpass liftering, noise reduction and ISDCN: Table 9.7 and Table 9.8.

As shown in the Tables 9.1 to 9.8, the ISDCN channel normalization algorithm improves the "across the great divide" performance, and it does not disturb the "within the great divide" performance (it even improves it). We will try a better estimate of initial spectrum tilt $\hat{q}^{(0)}$, we believe it will help the estimate to converge to a better final value.

training session	test session	recognition rate
6 7 8	4	10/26
6 7 8	5	8/26
1 2 3	9	5/26
1 2 3	10	7/26
average		28.8%

Table 9.4: Identification results, ISDCN, across the great divide

training session	test session	recognition rate
1 2 3	4	24/26
1 2 3	5	24/26
6 7 8	9	23/26
6 7 8	10	22/26
average		89.4%

Table 9.5: Identification results, noise reduction and ISDCN, within the great divide

training session	test session	recognition rate
6 7 8	4	12/26
6 7 8	5	13/26
1 2 3	9	14/26
1 2 3	10	9/26
average		46.2%

Table 9.6: Identification results, noise reduction and ISDCN, across the great divide

training session	test session	recognition rate
1 2 3	4	25/26
1 2 3	5	25/26
6 7 8	9	26/26
6 7 8	10	26/26
average		98.1%

Table 9.7: Identification results, bandpass liftering, noise reduction and ISDCN, within the great divide

training session	test session	recognition rate
6 7 8	4	14/26
6 7 8	5	14/26
1 2 3	9	11/26
1 2 3	10	13/26
average		50%

Table 9.8: Identification results, bandpass liftering, noise reduction and ISDCN, across the great divide

Chapter 10

Results

10.1 Database

The database utilized in this study is the narrowband portion of the "King" database, collected in 10 sessions from 51 male speakers, 26 from San Diego and 25 from Nutley. The speakers were asked to talk about several topics, so that the speech is natural and spontaneous. The data were collected over long distance telephone line, and the data for the 25 Nutley speakers were much noisier than that of the 26 San Diego speakers. The speech material from each session is approximately 45 seconds long; the data were digitized at 8 kHz and 12-bit resolution. Sessions 1 to 5 and sessions 6 to 10 were collected under different environments. This division of data, "the great divide", results in serious degradation of performance as observed in [11] when training on one set and testing on the other. Also the Nutley data are much noisier than San Diego data. We performed our experiments in three contexts: San Diego alone (26 speakers), Nutley alone (25 speakers), and all 51 speakers combined. Further, the experiments were carried out across "the great divide" for the most challenging test condition. Figure 10.1 and Figure 10.2 show the typical waveforms of San Diego data and Nutley data. The quality of Nutley is much worse.

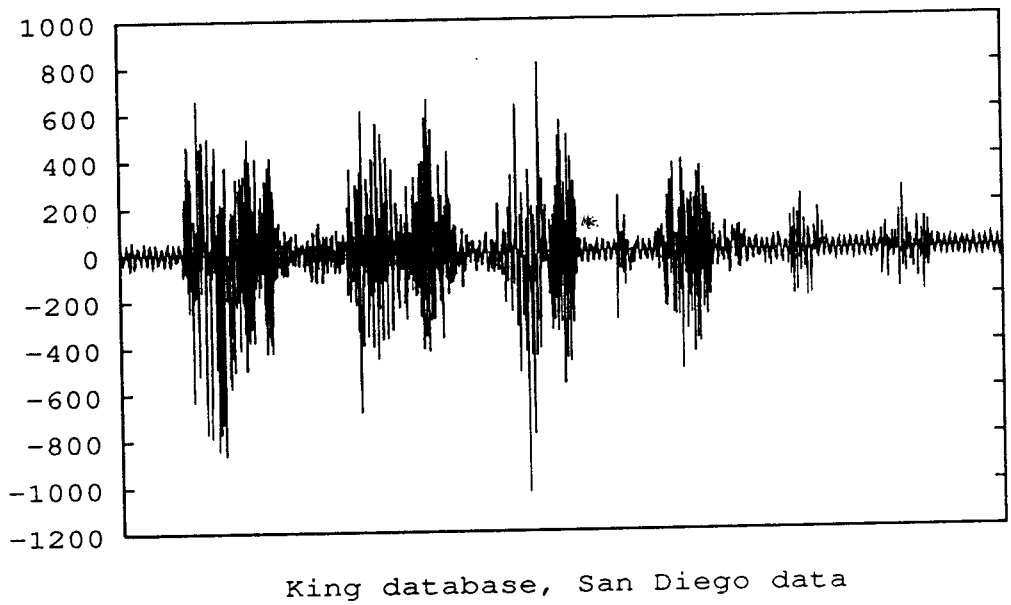


Figure 10.1: Typical waveform of San Diego data

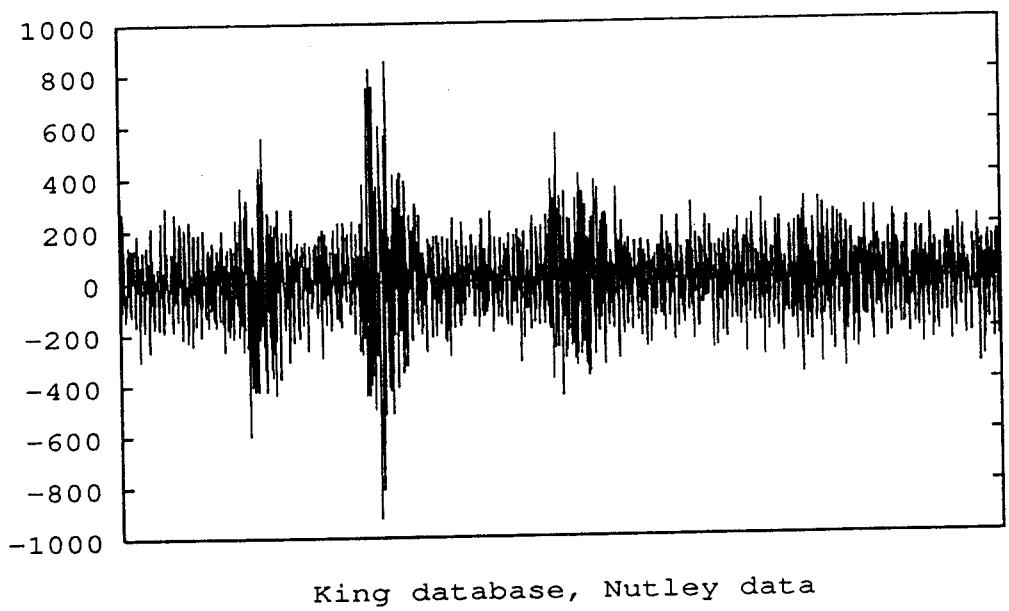


Figure 10.2: Typical waveform of Nutley data

Baseline			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	81.73%	35%	58.82%
Average-rank	2.01923	5.58	3.87745

Bandpass Liftering			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	85.58%	47%	66.67%
Average-rank	1.68269	4.11	2.88725

RASTA			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	91.35%	50%	71.08%
Average-rank	1.10577	3.51	2.29902

Bandpass Liftering & RASTA			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	94.23%	61%	77.94%
Average-rank	1.07692	2.72	1.89706

Table 10.1: Identification results: Within the great divide

10.2 Speaker Identification

Table 10.1 shows the results "within the great divide", table 10.2 shows the results "across the great divide". As the robustness processing techniques were added, the performance improved significantly.

- Within the great-divide: train on sessions 1, 2, and 3, test on sessions 4 and 5; train on sessions 6, 7, and 8, test on sessions 9 and 10.

- Across the great-divide: train on sessions 1, 2, and 3, test on sessions 9 and 10; train on sessions 6, 7, and 8, test on sessions 4 and 5.

Table 10.3 shows the comparison of our results with [11], for 16 San Diego speakers, trained on sessions 1, 2, and 3, tested on sessions 9 and 10.

Baseline			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	7.69%	36%	19.61%
Average-rank	10.0385	6.49	11.4608

Bandpass Liftering			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	36.54%	46%	36.76%
Average-rank	4.86538	4.86	6.37745

RASTA			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	42.31%	53%	43.63%
Average-rank	4.56731	3.35	6.23039

Bandpass Liftering & RASTA			
	San Diego (26)	Nutley (25)	All (51)
ID-rate	77.88%	65%	58.82%
Average-rank	1.86538	2.24	3.48529

Table 10.2: Identification results: Across the great divide

	[11]	BPL & RASTA
ID-rate	75%	87.5%
Average-rank	1.56	1.12

Table 10.3: Comparison between the best published result and BPL & RASTA

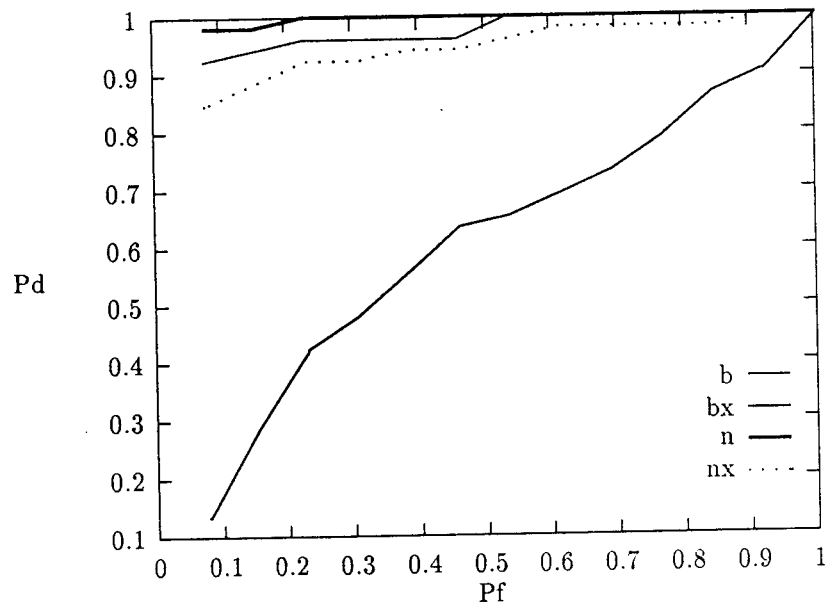
10.3 Speaker Verification

We performed "open set speaker verification" on the "King" database. Half of the speakers were registered speakers, the other half impostors. All registered speakers claimed their true identities; these attempts were used to compute detection rate. All impostors claimed all registered speaker identities; these attempts were used to compute the false alarm rate. There are more impostor attempts than true speaker attempts. For example, in San Diego data, 26 speakers, 13 are registered speakers and the other 13 are impostors. For each session of data, there are 13 true speaker attempts and 13 (impostors) * 13 (registered speaker identities) = 169 impostor attempts. As in the speaker identification case, sessions 1, 2, and 3 were used to train division 1 models, sessions 6, 7, and 8 were used to train division 2 models. sessions 4, 5, 9, and 10 were used for tests. The results were averaged on four tests:

- Within the great divide: Train on sessions 1, 2, and 3; test on sessions 4 and 5. Train on sessions 6, 7, and 8, test on sessions 9 and 10.
- Across the great divide: Train on sessions 1, 2, and 3; test on sessions 9 and 10. Train on sessions 6, 7, and 8, test on sessions 4 and 5.

Figure 10.3, 10.4, and 10.5 show the ROC plots for San Diego, Nutley and all 51 speakers experiments respectively. There are four curves in each figure:

- b: baseline, within the great divide.
- n: normalized (bandpass liftering & RASTA), within the great divide.
- bx: baseline, across the great divide.



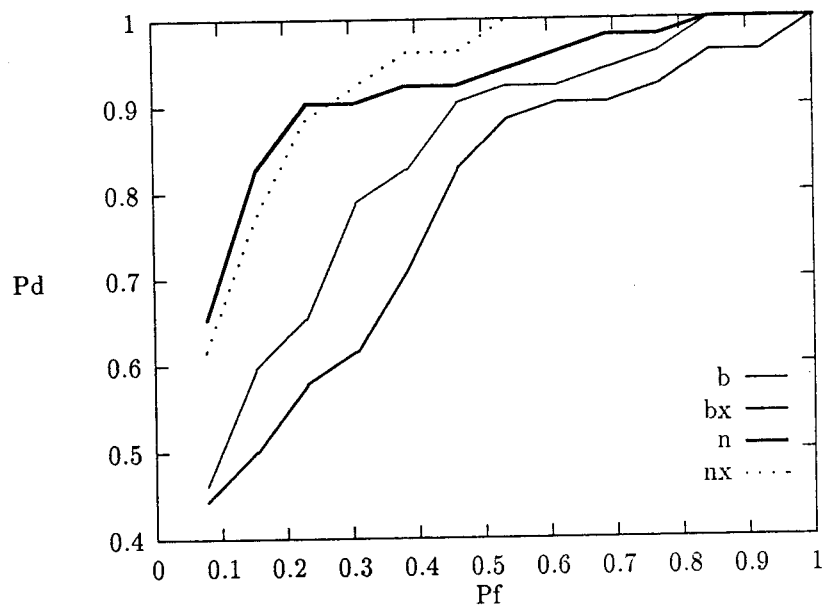
- Within the great divide, equal error rate: 8% vs. 4%.
- Across the great divide, equal error rate: 42% vs. 13%.

Figure 10.3: ROC for 26 San Diego speakers

- nx: normalized (bandpass liftering & RASTA), across the great divide.

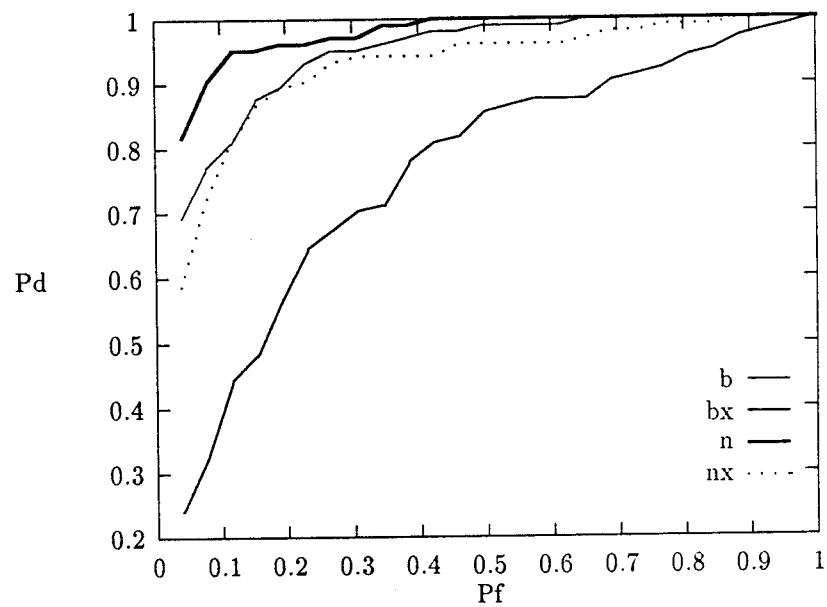
10.4 Discussion and Conclusions

Tables 10.1 and 10.2 clearly show the independent contributions of bandpass liftering and RASTA to performance improvement. Bandpass liftering deemphasizes the highly variant and noisy cepstral coefficients and is a static correction. RASTA smoothes all of the cepstral coefficients by a bandpass filtering operation thereby attempting to remove the effects of the channel and the transducer. In this sense, the spoken material is "self-normalized," providing robustness. Thus, by combining the static (bandpass liftering) and dynamic (RASTA) techniques,



- Within the great divide, equal error rate: 27% vs. 16%.
- Across the great divide, equal error rate: 35% vs. 18%.

Figure 10.4: ROC for 25 Nutley speakers



- Within the great divide, equal error rate: 15% vs. 9%.
- Across the great divide, equal error rate: 30% vs. 15%.

Figure 10.5: ROC for combined 51 speakers

we obtain the benefits of both techniques. Note that improvements are dramatic when the testing is across the great-divide. An interesting observation is that we get these improvements without using any specific noise-removal technique, such as spectral subtraction used in [17]. We have verified the consistency of these results on an independent database, called the continuous speech recognition (CSR) database. CSR consists of simultaneous recording of speech material from subjects using two different types of microphones. Hence CSR provides an excellent means of not only establishing consistency of our results, but also to develop insight into why these techniques work. This is accomplished by examining scattergrams of the various cepstral coefficients for recording with the two different microphones with and without robustness processing. As shown in the scattergrams in Section 8.3, the cepstral coefficients of mic-1 vs. mic-2 are more aligned with $x = y$ after RASTA filtering. All of the above discussions hold for speaker verification as well.

We have identified, by systematic investigation, a combination of techniques that provides a very robust performance. However, additional work is needed to improve the performance across the great divide to be at the level of performance within the great divide. Also, additional investigations planned with experimentation on a highly challenging corpus, the Switchboard [13], will shed further light.

Chapter 11

Speaker Verification

11.1 Introduction

Speaker verification is more useful than speaker identification in real world applications. The difference between them is:

- Speaker identification: For test speaker x, it is known that this speaker is one of the m "registered speakers" (for these speakers we have trained models). The task is to decide which one of these m "known" speakers the test speaker x is.
- Speaker verification: For test speaker x who claims identity i (i is one of the m "registered speakers"), the task is a binary decision – accept or reject. Depending on whether speaker x is one of the m registered speakers or not, the problem is further classified into:
  - Close set speaker verification: Speaker x is one of the m registered speakers.
  - Open set speaker verification: Speaker x is not one of the m registered speakers.

While speaker identification is much easier than open set speaker verification (which is most useful in applications), we found that it is beneficial to study an easier problem before the challenging one because it is easier to interpret the results and gain insights. Besides, we found a simple method to convert a speaker identification system into a speaker verification system, so the improvements achieved in speaker identification system can be readily translated into speaker verification systems.

In the following sections we will introduce the traditional speaker verification algorithms, and then compare our new approach with it.

11.2 Speaker Verification

Because speaker verification is a binary decision – accept or reject, it is only natural to base this decision on a "threshold". When speaker "x" claims identity "i", we compute the score of the input utterance with respect to speaker model "i". If the score is better than the threshold - accept; otherwise - reject, Figure 11.1.

The threshold value is decided by many training instances. In Figure 11.2, the *xaxis* is threshold, the *yaxis* is error rate; the decreasing curves represent true speaker rejection rate, increasing curves the impostor acceptance rate. This plot is the result of 1250 true speaker attempts, and 1250 impostor attempts. The score computed for each attempt is the "distance" between the input utterance and the claimed identity model, the lower the better. If the score is lower than the threshold - accept; otherwise - reject. As the threshold increases, it becomes easier to be accepted; therefore the true speaker rejection rate decreases and the impostor acceptance rate increases. A low true speaker rejection rate

and a low impostor acceptance rate are desired; but as we change the threshold value, trade-off is being made between these two error rates – one decreases, while the other increases. Usually people choose the threshold value at "equal error" point (where the two curves cross) as the operating threshold. At this threshold, true speaker rejection rate and impostor acceptance rate are the same.

The major problem associated with using an absolute threshold value for different input utterances is "normalization". The length and content of the input utterance may vary. For longer utterances, the scores (distances) will become larger; for speech contents of vowels, the scores will become smaller because of better match with the models (compared with consonants - consonant models are more difficult to train and match). The length problem is easy to solve: we can normalize the scores according to time, which can be measured precisely. The speech content problem is not easy to normalize; it is essentially a "speech recognition" problem, which by itself is no less difficult than speaker verification.

Because of this normalization problem, we developed a new algorithm with "self-normalization" property.

11.3 Results and Motivation

11.3.1 Database

The database used for the speaker verification experiment is the "Total-Voice database". It has 50 speakers, 25 males and 25 females. Only male data are used in this experiment. There is a different ID number (10 digits) for each speaker. Each speaker called in 25 times from different places, using different telephone handsets. In each call, the speaker gives six utterances; three of them

are true speaker attempts (the speaker speaks his (or her) own ID number), the other three are impostor attempts (the speaker tries to impose each one of the 25 speakers in each call (male imposes male, female imposes female only)). We call the three utterances t1, t2, and t3 ("t" stands for token).

There are 25 (speakers) * 25 (true speaker attempts) * 3 (repetitions) = 1875 true speaker utterances, and also 1875 impostor utterances. (Actually in the 25 impostor attempts from each speaker, there is one attempt that should be considered as true speaker attempt because the speaker speaks his own ID number.)

This is a very difficult database because it is telephone speech, and from different handsets. Also we have to make a decision from a 10-digit utterance, which is very short.

11.3.2 Experiments

This is a free-text speaker verification:

- Training: Use 25 t1 impostor utterances from each speaker to train the speaker model. Because these 25 utterances include all the 25 different ID's, the training material covers all the phonetic events.

- Test: Use t2 and t3 true speaker utterances to compute the true speaker rejection rate, that amounts to 25 (speakers) * 25 (true speaker attempts) * 2 (t2 and t3) = 1250 attempts. Use t2 and t3 impostor utterances (also 1250 attempts) to compute the impostor acceptance rate.

We tried four speech front ends: 1) baseline, 2) bandpass liftering, 3) RASTA filtering, and 4) BPL + RASTA. Figure 11.2 shows the error rate plot. Figure 11.3 shows the zoom in of the equal error points area. The lower the equal

error point, the better the performance. Although bandpass liftering improves the performance, we were surprised to see that RASTA, which performs so well in our speaker identification experiments, actually "decreases" the speaker verification performance!

There must be a reason to explain this, and we found the answer lies in the difference between "absolute threshold vs. relative comparison".

11.4 Relative Ranking

We can inspect the scores of both true speaker attempts and impostor attempts before and after RASTA filtering to see what has changed. Table 11.1 shows the sorted scores before (column 2) and after (column 4) RASTA filtering, column 1 and 3 shows whether the score is a true speaker score (1) or an impostor score (0). As shown in the table, the scores improve in general (become smaller), but the impostor scores improve "more" than the true speaker scores (0.4 vs. 0.06).

In the absolute threshold approach, if true speaker scores and impostor scores improve as much, the performance stays the same; if true speaker scores improve more than impostor scores, the performance becomes better; but when impostor scores improve more than true speaker scores, the performance become worse. The reason why impostor scores improve more is quite straight forward: because the impostor scores are originally bad and larger, thus leave more room for improvement.

Why in the speaker identification experiment this uneven improvement does not cause trouble and RASTA filtering still improves identification performance? It is because in the speaker identification experiment the scores are not compared with an absolute threshold, but compared with "peer scores", i.e. the score

Baseline		RASTA	
1	0.16695	0	0.10563
1	0.17544	1	0.10651
1	0.17919	1	0.10663
1	0.17954	0	0.10765
1	0.18245	1	0.10843
1	0.18278	1	0.11057
1	0.18442	1	0.11245
1	0.18517	1	0.11400
1	0.18564	0	0.11421
1	0.18642	0	0.11449
1	0.18696	1	0.11455
1	0.18751	1	0.11618
1	0.19016	1	0.11855
1	0.19059	1	0.11893
1	0.19091	1	0.11895
...			
0	0.70576	0	0.37748
0	0.70801	0	0.37754
0	0.70802	0	0.38021
0	0.71076	0	0.38051
0	0.71548	0	0.38224
0	0.72282	0	0.38972
0	0.72528	0	0.39246
0	0.72976	0	0.39389
0	0.74229	0	0.39543
0	0.75965	0	0.39691
0	0.76707	0	0.39892
0	0.76728	0	0.41661
0	0.78016	0	0.42243
0	0.78134	0	0.42771
0	0.87750	0	0.44899

Table 11.1: Sorted scores before and after RASTA filtering

with respect to speaker model "i" is compared with scores with respect to other speaker models. This way, the peer scores are all dependent on input utterance, they all adapt "together", we don't have to worry about the normalization problem. It is "self-normalized".

Following this idea, we can change the verification algorithm from comparing with an absolute threshold into comparing with peer scores, as shown in Figure 11.1. Instead of computing the score of claimed identity model only, we compute scores with respect to all the registered models, then compute the rank of the score of the claimed identity. If the rank is better than the threshold rank - accept; otherwise - reject.

Another way to interpret this relative ranking approach is speaker "discrimination". Like the computation of the discriminant distance matrix mentioned in Section 2.2, it is a statistic of all the data, then the information can be used to help in the discrimination decision. Relative ranking can be considered as a "discrete discriminant".

11.5 Comparison

Instead of error rate plots as in Figure 11.2 and 11.3, we use ROC (Receiver Operating Characteristics, Prob(detection) vs. Prob(false alarm)) plot to compare the absolute threshold and relative ranking approaches. Figure 11.4 is the ROC of Figure 11.2. Figure 11.5 is the ROC using relative ranking approach. In Figure 11.4, the RASTA curve is much worse than the baseline curve; however, in Figure 11.5, most of the RASTA curve is better than baseline curve (only at very restrictive threshold levels RASTA performs a little worse). The bandpass liftering + RASTA front end, relative ranking approach curve achieves the best

performance of all curves - 2.5% equal error rate.

Figure 11.6 to Figure 11.9 compare the ROC curves of relative ranking vs. absolute threshold under four different front ends; equal error rates are listed with figures, too. The cross point of ROC curve and the line $x = 1 - y$ is the equal error point. In all four situations, the improvement from absolute threshold to relative ranking is consistent and quite substantial.

Although the performance is for free-text speaker verification, it is even better than some fixed-text speaker verification systems. (equal error rate 2.5% vs. 4%)

11.6 Discussion

There are two points needed to be addressed when using this relative ranking approach instead of the absolute threshold approach:

- The choice of operating point: When using the absolute threshold approach, the threshold can be chosen from an infinite set of numbers of continuous real values; however, in the relative ranking approach, when there are 25 registered speakers, there are only 25 possibilities of threshold rank. This limitation implies that the operating point may not be allocated with high resolution, e.g. the choice between 1% true speaker rejection, 10% impostor acceptance or 10% true speaker rejection, 1% impostor acceptance. Although 5% equal error rate point can be interpolated between these two choices, it does not exist. This limitation can be solved by increasing the number of registered speakers.

- Phonetic coverage in the speaker model: Good phonetic coverage in the training material for each speaker is required, while it is not required in fixed-text speaker verification training. Because the input test utterance is compared with "all" the speaker models, if the models do not cover the acoustic space, then the "speech factor" will interfere with the speaker verification. Speech features are more salient than speaker features (that is why "speaker-independent" speech recognition is possible), this speech factor will often hurt the speaker matching badly.

  We performed the following three experiments on the total voice database to prove this point:

  1. Train on true speaker utterances, test on impostor utterances: Table 11.2.

	Baseline	BPL	RASTA	BPL+ RASTA
ID-rate	84.8%	86.8%	67.2%	78.8%
average-rank	1.444	1.284	2.024	1.512

Table 11.2: Train on true speaker utterances, test on impostor utterances

  2. Train on true speaker utterances, test on true speaker utterances: Table 11.3.

	Baseline	BPL	RASTA	BPL + RASTA
ID-rate	98.4%	99.6%	99.6%	100%
average-rank	1.08	1.02	1.004	1

Table 11.3: Train on true speaker utterances, test on true speaker utterances

  3. Train on impostor utterances, test on impostor utterances: Table 11.4.

  When trained on true speaker utterance, the training material only contains each speaker's ID number, which means insufficient phonetic cover-

	Baseline	BPL	RASTA	BPL + RASTA
ID-rate	93.2%	98%	92.8%	98.4%
average-rank	1.148	1.052	1.152	1.028

Table 11.4: Train on impostor utterances, test on impostor utterances

age.

In experiment 1, RASTA decreases ID-rate. That is because speech features are more salient than speaker features, RASTA helps speech recognition more than speaker identification. The test impostor speech will be matched with the model of the person being imposed, because they have the "same speech".

In experiment 2, we use true speaker utterances as test utterances. In this way, the correct match has same speech materials in both training and test data; therefore the confusing factor from speech was removed. As shown, RASTA improves the ID-rate and average-rank.

In experiment 3, we used impostor utterances for training, which means more ID numbers and better phonetic coverage. (we used 10 impostor sentences, with some overlapping ID numbers, in average about 7 ID numbers were covered in these 10 sentences. Compared with 25 speaker ID's, this is still not very good coverage.) As shown in experiment 3, with better phonetic coverage in training material, RASTA does not decrease performance as much as in experiment 1 (0.4% vs. 17.6%, this decrease, 0.4%, is statistically insignificant); and BPL + RASTA achieves the best performance.

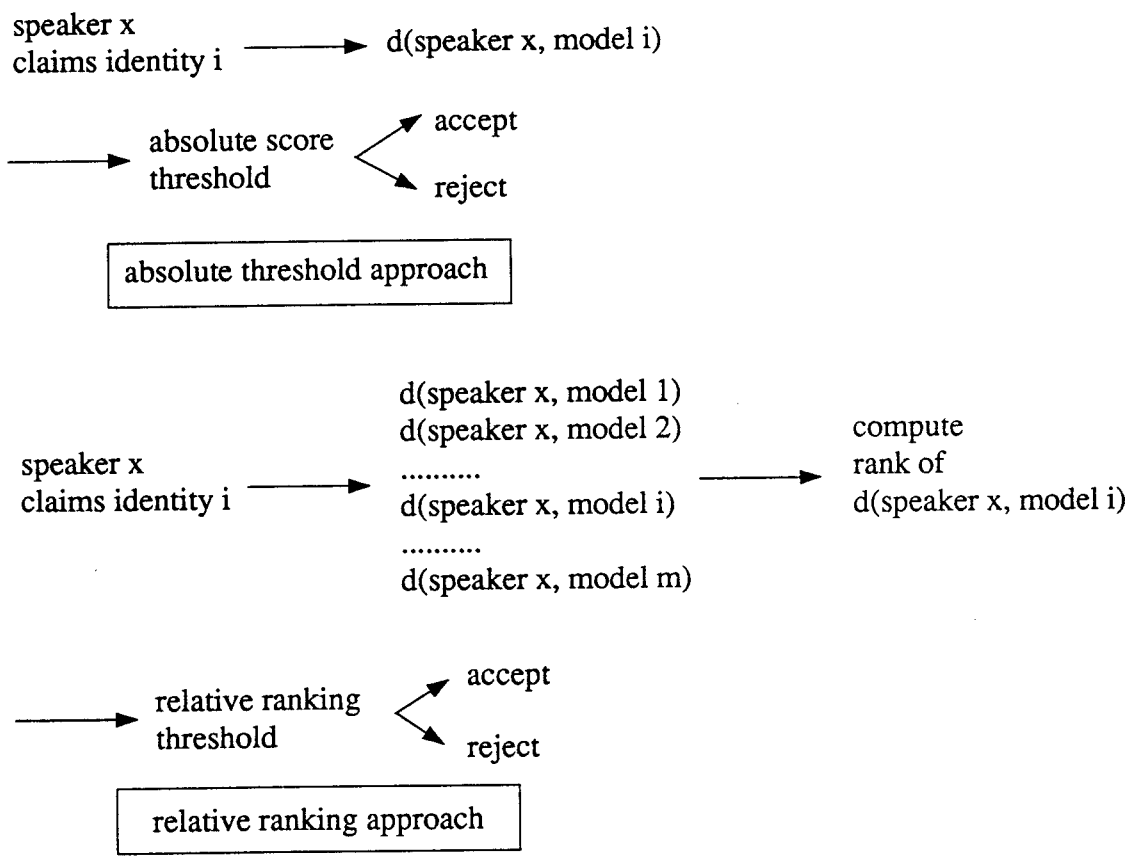


Figure 11.1: Absolute score threshold vs. relative ranking threshold

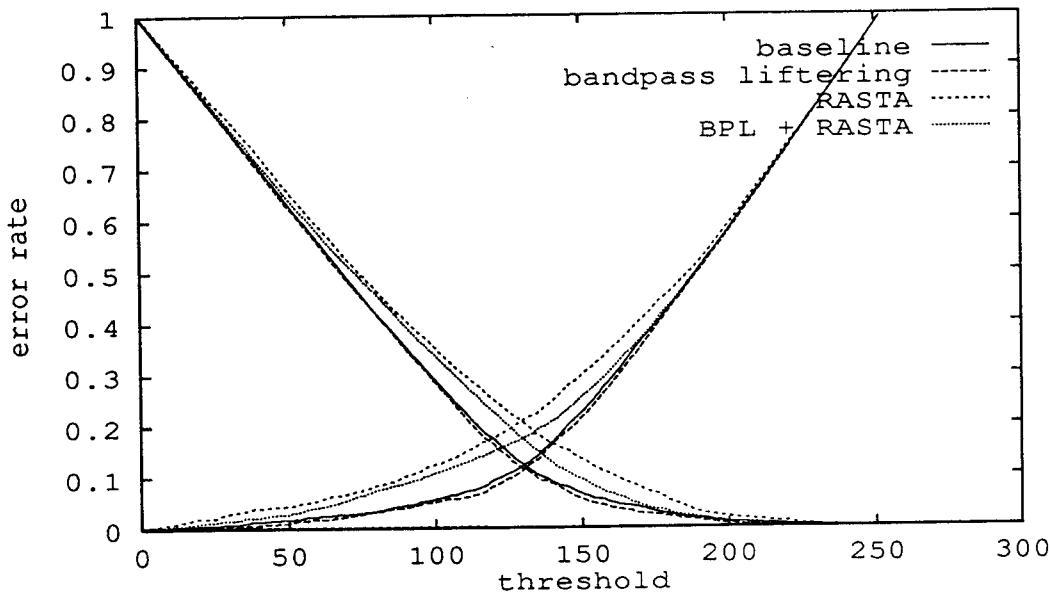


Figure 11.2: True speaker rejection and impostor acceptance vs. threshold, using absolute threshold value to decide reject / accept

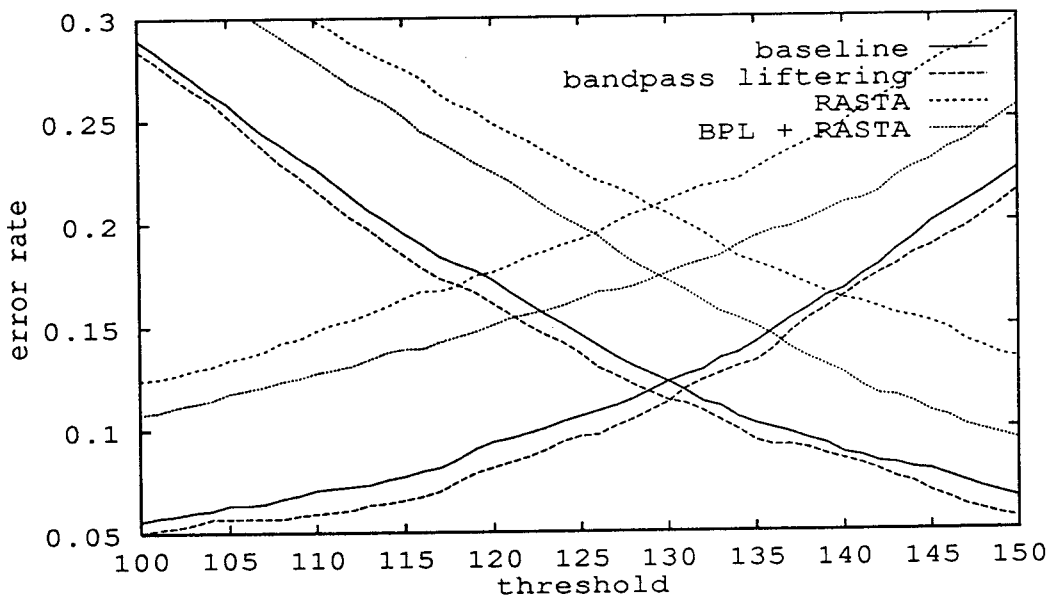


Figure 11.3: Zoom in of Figure 11.2

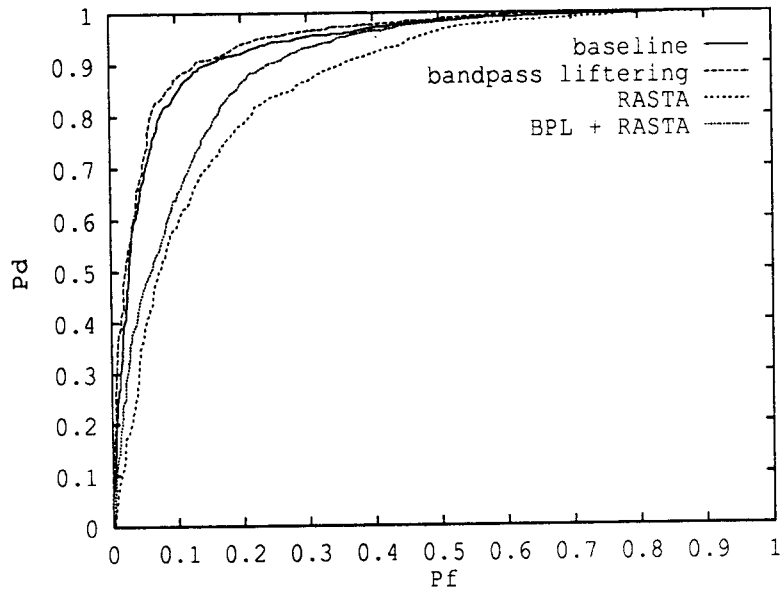


Figure 11.4: ROC plot of Figure 11.2, Prob(detection) vs. Prob(false alarm)

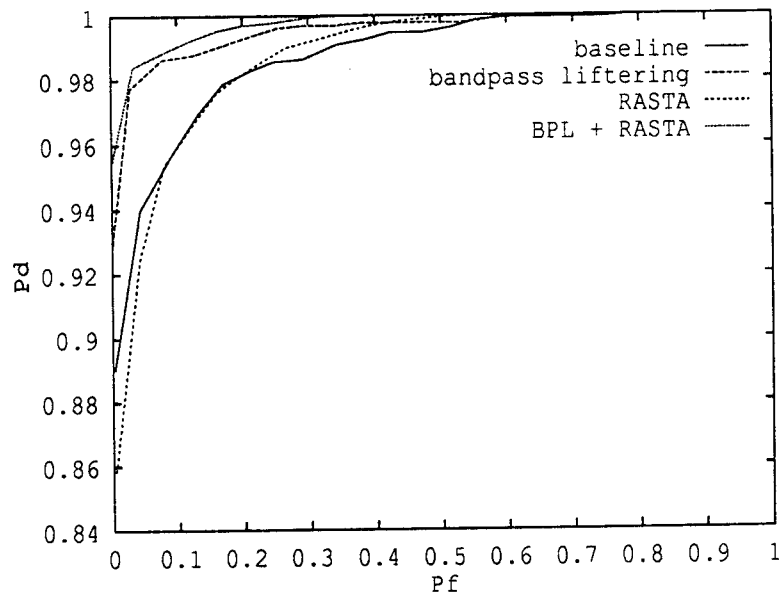


Figure 11.5: ROC plot using "relative ranking" approach

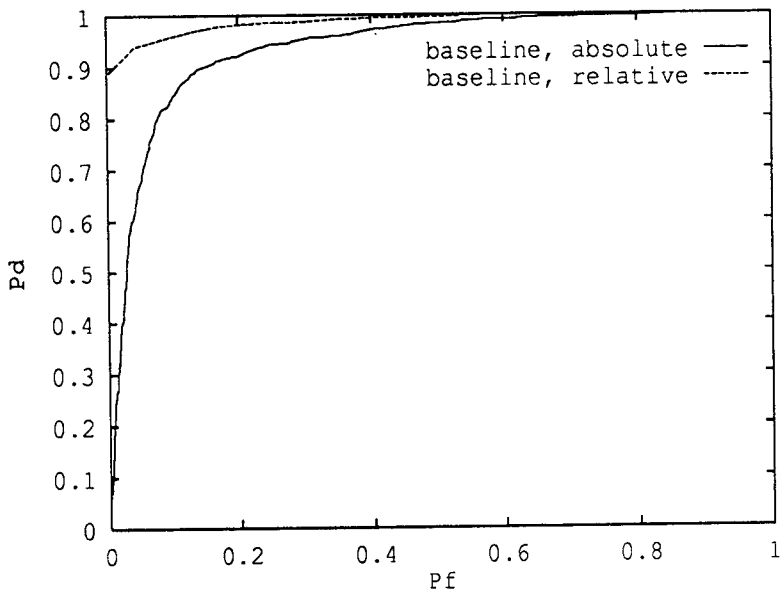


Figure 11.6: baseline, equal error rate 13% vs. 5.5%

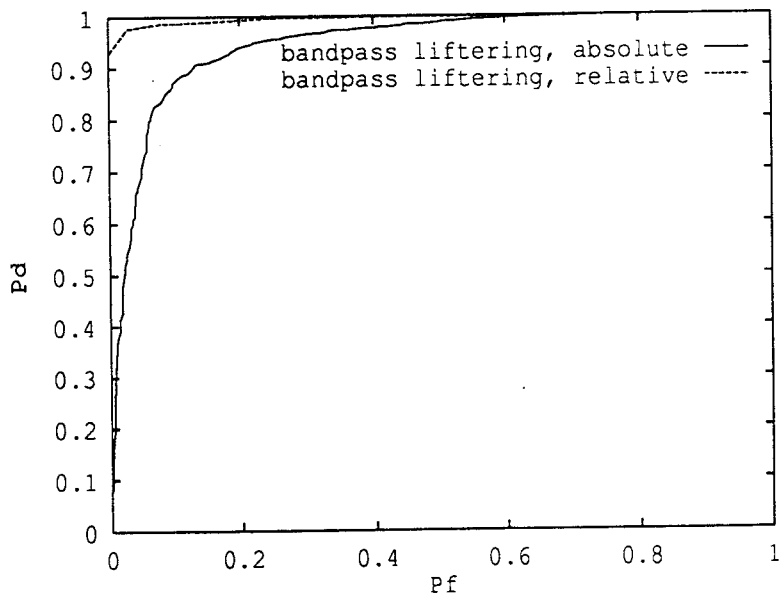


Figure 11.7: bandpass liftering, equal error rate 12% vs. 2.5%

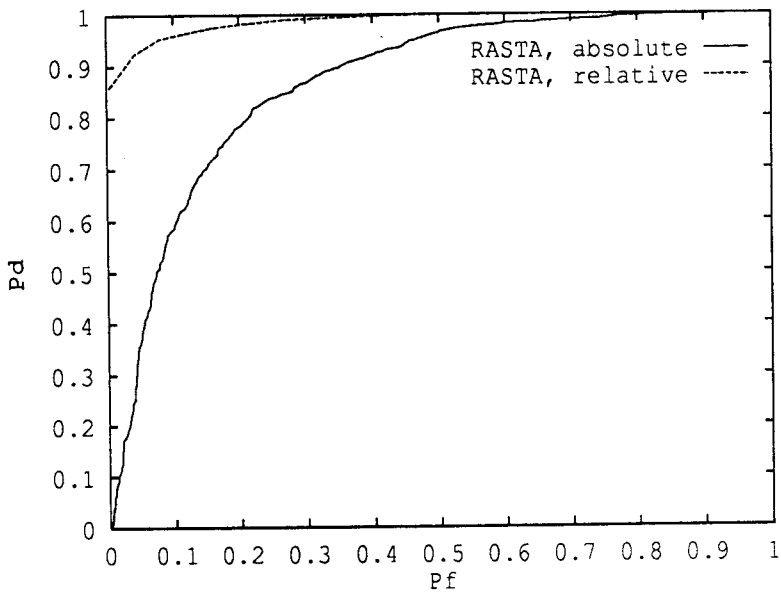


Figure 11.8: RASTA, equal error rate 21% vs. 6%

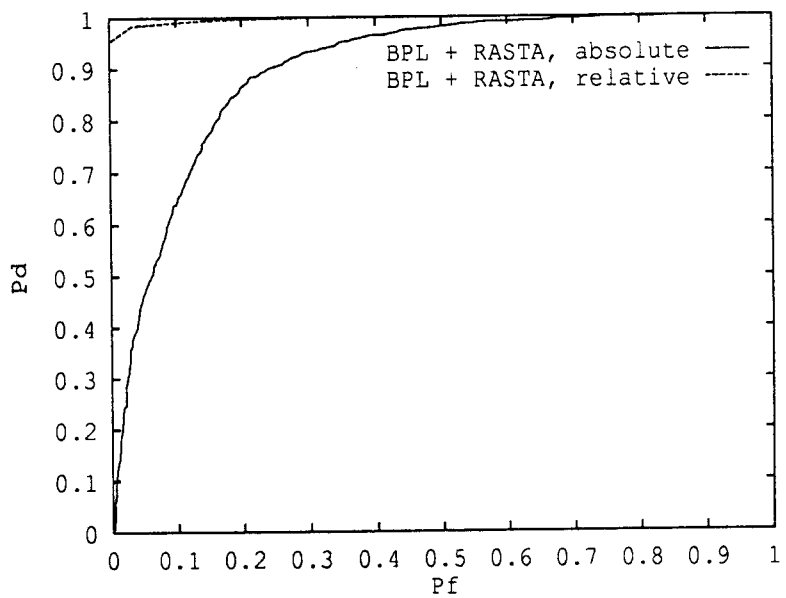


Figure 11.9: BPL + RASTA, equal error rate 18% vs. 2.5%

Appendix A

Linear Predictive Analysis Frond End

A.1 LPC Front End

Assuming an all-pole, order p speech production model, the system transfer function is:

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{A.1.1}$$

For the above model, the speech samples $s(n)$ are related to excitation samples $u(n)$ by a simple difference equation:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \tag{A.1.2}$$

The coefficients a_k can be estimated from speech data. We form a linear predictor with prediction coefficients α_k defined as a system whose output is:

$$\hat{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \tag{A.1.3}$$

The prediction error is:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k) \tag{A.1.4}$$

We select the coefficients a_k so as to minimize the average prediction error:

$$E_n = \sum_m e_n^2(m) \tag{A.1.5}$$

$$= \sum_m (s_n(m) - \hat{s}_n(m))^2 \tag{A.1.6}$$

$$= \sum_m (s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k))^2 \tag{A.1.7}$$

where $s_n(m) = s(m+n)$, a segment of speech samples of length m starting from time n. We can find the values of α_k that minimize E_n by setting $\partial E_n / \partial \alpha_i = 0, i = 1, 2, \ldots, p$, therefore getting the equations:

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \alpha_k \sum_m s_n(m-i)s_n(m-k),\ 1 \le i \le p \tag{A.1.8}$$

Assume

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k) \tag{A.1.9}$$

The above equation can be rewritten into:

$$\phi_n(i,0) = \sum_{k=1}^{p} \alpha_k \phi_n(i,k),\ 1 \le i \le p \tag{A.1.10}$$

α_k can be computed by solving the above linear equation set. The minimum mean-squared prediction error can be shown to be:

$$E_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \alpha_k \sum_m s_n(m)s_n(m-k) \tag{A.1.11}$$

$$= \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k) \tag{A.1.12}$$

In our implementation, the window length is 30 ms, the frame step is 20 ms (thus there is a 10 ms overlap). Each speech frame is first pre-emphasized, $y(n) = x(n) - 0.97 * x(n-1)$, then Hamming window is applied.

A.2 Durbin's Algorithm

Assume the autocorrelation function is wide-sense stationary:

$$\phi_n(i,k) = R_n(|i-k|),\ i = 1,2,\ldots,p,\ k = 0,1,\ldots,p \tag{A.2.13}$$

To determine the prediction coefficients α_k we must solve:

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(p-1) \\ R(1) & R(0) & R(1) & \cdots & R(p-2) \\ R(2) & R(1) & R(0) & \cdots & R(p-3) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R(p-1) & R(p-2) & R(p-3) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \cdots \\ \cdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \cdots \\ \cdots \\ R(p) \end{bmatrix} \tag{A.2.14}$$

The following is Durbin's algorithm to solve α_k (we remove the subscript n from R_n, since this algorithm is applied to a set of autocorrelation coefficients $R_n(i), i = 0,1,\ldots,p$ at a particular time n):

$$E^{(0)} = R(0) \tag{A.2.15}$$

$$k_i = (R(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R(i-j))/E^{(i-1)},\ 1 \le i \le p \tag{A.2.16}$$

$$\alpha_i^{(i)} = k_i \tag{A.2.17}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)},\ 1 \le j \le i-1 \tag{A.2.18}$$

$$E^{(i)} = (1-k_i^2)E^{(i-1)} \tag{A.2.19}$$

These steps are carried out iteratively for $i = 1,2,\ldots,p$ and the final solution is:

$$\alpha_j = \alpha_j^{(p)},\ 1 \le j \le p \tag{A.2.20}$$

The model gain G can be computed as (from the minimum mean-squared prediction error):

$$G^2 = R(0) - \sum_{k=1}^{p} \alpha_k R(k) \tag{A.2.21}$$

A.3 Cepstral Coefficients

We can now compute the cepstral coefficients c_n from α_k. The definition of cepstral coefficients (refer to Figure A.1, $(c_i, i = 0, \ldots, \infty)$) is

$$\ln \frac{G}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}} = \sum_{n=0}^{\infty} c_n z^{-n} \tag{A.3.22}$$

$$\ln(G) - \ln(1 - \sum_{k=1}^{p} \alpha_k z^{-k}) = c_0 + \sum_{n=1}^{\infty} c_n z^{-n} \tag{A.3.23}$$

Now we have

$$c_0 = \ln(G) \tag{A.3.24}$$

To derive the other $c_n, n = 1, \ldots, \infty$, we can differentiate (A.3.23) with respect to z,

$$\frac{1}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}} \sum_{k=1}^{p} k\alpha_k z^{-k-1} = \sum_{n=1}^{\infty} n c_n z^{-n-1} \tag{A.3.25}$$

$$\sum_{k=1}^{p} k\alpha_k z^{-k-1} = (1 - \sum_{k=1}^{p} \alpha_k z^{-k}) \sum_{n=1}^{\infty} n c_n z^{-n-1} \tag{A.3.26}$$

$$\sum_{k=1}^{p} k\alpha_k z^{-k} = (1 - \sum_{k=1}^{p} \alpha_k z^{-k}) \sum_{n=1}^{\infty} n c_n z^{-n} \tag{A.3.27}$$

$$\sum_{k=1}^{p} k\alpha_k z^{-k} = \sum_{n=1}^{\infty} n c_n z^{-n} - \sum_{k=1}^{p} \sum_{n=1}^{\infty} n\alpha_k c_n z^{-(n+k)} \tag{A.3.28}$$

Now let's compare the coefficients of z^{-m} of both sides of the equation. Let $n + k = m$, and using the substitution: $n = m - k$,

$$m\alpha_m z^{-m} = mc_m z^{-m} - \sum_{k=1}^{m-1}(m-k)\alpha_k c_{m-k} z^{-m} \tag{A.3.29}$$

Thus for $1 \leq m \leq p$,

$$c_m = \alpha_m + \sum_{k=1}^{m-1}(\frac{m-k}{m})\alpha_k c_{m-k} \tag{A.3.30}$$

And for $p < m$,

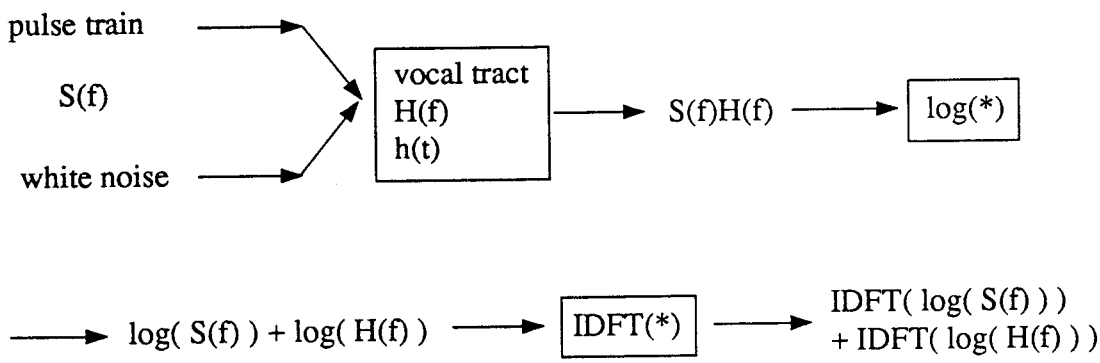
$$c_m = \sum_{k=1}^{p}(\frac{m-k}{m})\alpha_k c_{m-k} \tag{A.3.31}$$

Now we can rearrange (A.3.24), (A.3.30) and (A.3.31) and obtain a recursive formula to compute the cepstral coefficients $(c_n, n = 0, \ldots, \infty)$ from LPC coefficients $\alpha_k, k = 1, \ldots, p$.

$$c_0 = \ln(G) \tag{A.3.32}$$

$$c_n = \alpha_n + \sum_{k=1}^{n-1}(\frac{k}{n})c_k\alpha_{n-k}, \text{for } 1 \leq n \leq p \tag{A.3.33}$$

$$c_n = \sum_{k=1}^{p}(1 - \frac{k}{n})c_{n-k}\alpha_k, \text{for } n > p \tag{A.3.34}$$



Usually the dimension of cepstral coefficients are less than 20,
not long enough to capture the pulse train repetition of S(f);
therefore, cepstrum represents the "impulse response" of vocal tract, h(t).
However, h(t) = IDFT(log(H(f))) instead of IDFT(H(f)).

Figure A.1: Cepstrum analysis

Appendix B

Frequency Warping

B.1 Digital Warping of Spectra

In a variety of applications, it is useful to transform a sequence to a new sequence whose Fourier transform is equal to the Fourier transform of the original sequence on a distorted (warped) frequency scale. For instance, because of different sensitivities of human perception towards different frequency bands, we can give different weights to these bands according to physiological or experimental analysis. Bilinear transform is used for this frequency warping purpose. Let's focus on any mapping $\hat{z} = m(z)$ which maps the unit circle in the z plane to the unit circle in the $\hat{z}$ plane. Letting Ω be angular frequency in the z plane and $\hat{\Omega}$ be angular frequency in the $\hat{z}$ plane, we want $\hat{z} = m(z)$ to satisfy

$$e^{j\hat{\Omega}} = m(e^{j\Omega}) \tag{B.1.1}$$

or

$$\hat{\Omega} = \theta(\Omega) \tag{B.1.2}$$

where $m[e^{j\Omega}] = e^{j\theta\Omega}$. Therefore, if the spectrum of g_k is to be a warped version of the spectrum of f_n, the transform $\hat{z} = m(z)$ must have an all-pass characteristic,

like

$$\hat{z} = m(z) = \prod_k \frac{1 - a_k z^{-1}}{z^{-1} - a_k^*} \tag{B.1.3}$$

where a_k^* is the complex conjugate of a_k.

In order that $m(z)$ be invertible and in the form already given, the coefficient a_k must be such that the interval $-\pi < \Omega \leq \pi$ maps one-to-one to the interval $-\pi < \hat{\Omega} \leq \pi$. A necessary (but not sufficient) condition for this to be true is that the number of zeros minus the number of poles of $m(z)$ which lie inside the unit circle be plus or minus unity. A useful choice of $m(z)$ is a first-order all-pass of the form

$$\hat{z} = m(z) = \frac{1 - az^{-1}}{z^{-1} - a^*} \tag{B.1.4}$$

where $0 <\mid a \mid< 1$. For real values of parameter a, the mapping between the frequency variables $\hat{\Omega}$ and Ω is given by

$$\hat{\Omega} = \theta(\Omega) = \tan^{-1}[\frac{(1 - a^2)\sin(\Omega)}{(1 + a^2)\cos(\Omega) - 2a}] \tag{B.1.5}$$

An alternative form is

$$\hat{\Omega} = \theta(\Omega) = \Omega + 2\tan^{-1}[\frac{a\sin(\Omega)}{1 - a\cos(\Omega)}] \tag{B.1.6}$$

If the parameter a is picked to be real and between 0 and 1, the effect on the spectrum of f_n will be to sample with higher resolution at low frequencies and with lower resolution at higher frequencies. If instead, a is negative between 0 and -1, the effect is reversed, that is, the spectrum of f_n is evaluated with greater resolution at high frequencies than at low frequencies. By letting the parameter a assume complex values, the point of maximal resolution can be placed at any desired frequency.

Usually people use positive a, which emphasize lower frequencies. For in-

stance, SPHINX use 0.6. However, that is for "speech recognition" purpose, in which lower frequencies (major formant frequencies) are more important. In "speaker identification", higher frequencies may be more important.

B.2 Digital Warping of Spectra by Time Domain Filtering

Cepstrum is a time domain signal, there exists a way to do frequency warping by time domain filtering. If we perform the following algorithm on a time series, its frequency spectrum will be warped with a warping coefficient a. Since we use cepstral coefficients as our feature vectors, being able to do frequency warping by time domain filtering is much more efficient.

$$\tilde{g}_{0,n} = a[\tilde{g}_{0,n-1} - 0] + f_{-n} \tag{B.2.7}$$

$$\tilde{g}_{1,n} = a[\tilde{g}_{1,n-1} - 0] + \tilde{g}_{0,n-1} \tag{B.2.8}$$

$$\tilde{g}_{k,n} = a[\tilde{g}_{k,n-1} - \tilde{g}_{k-1,n}] + \tilde{g}_{k-1,n-1}, k = 2, 3, \cdots \tag{B.2.9}$$

In Figure B.1, we show the frequency warping with different a. In Figure B.2, we show the spectra of a signal before and after frequency warping with $a = 0.6$. The original spectrum has four equally-spaced, equal-magnitude components; after the warping, lower frequency components acquire higher resolution and their magnitudes are amplified more than higher frequency components.

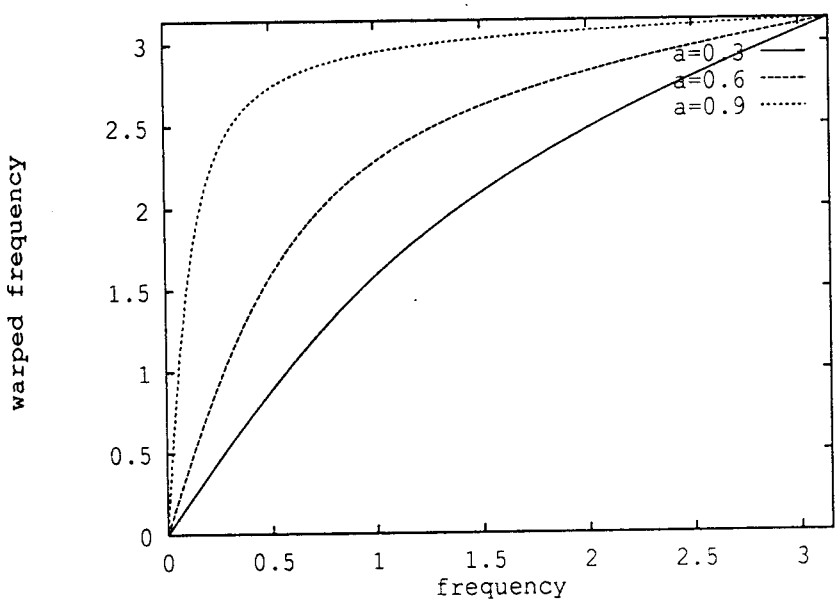


Figure B.1: Frequency vs. warped frequency

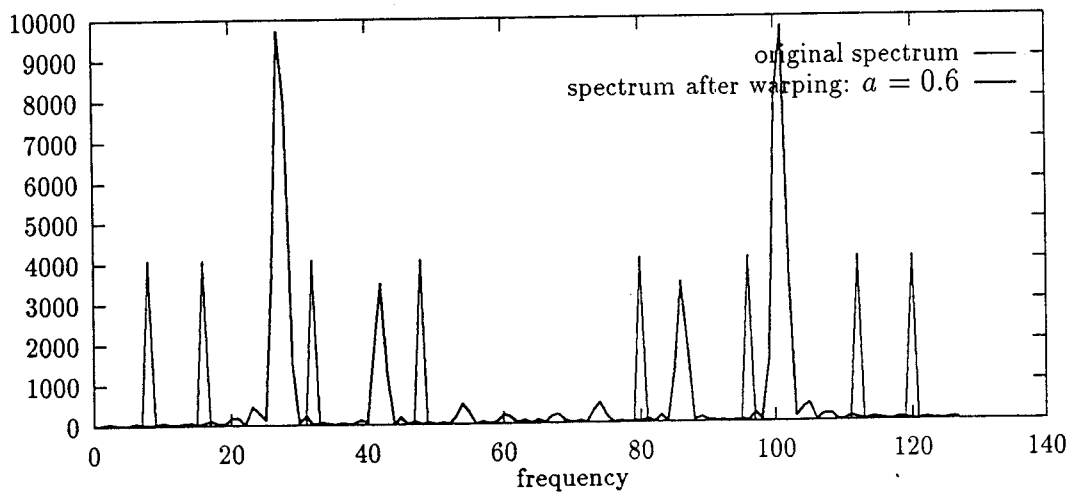


Figure B.2: Spectrum before and after warping

Appendix C

Phonotactic Grammar For Phoneme Recognition

C.1 Introduction

The ambition of this project is to perform speaker recognition on free text. So the accuracy of speech recognition directly affects the performance of speaker recognition. In our previous experiments, the results show that if we use **null grammar** to do the speech recognition, there will be a lot of illogical phone streams got recognized. For example, t, s, p, t ... and we know that these phone sequences can not possibly be from normal English speeches. So under the assumption that the test utterances are real speeches, we can use linguistic knowledge to constrain the grammar instead of using a null grammar. And after using the grammar described below, we do get better speech recognition results. People may want to argue that speech recognition is not necessary for speaker identification, because human can recognize speaker without knowing the real text. However, the statistics shows that the variances between different phones from the same speaker are larger than the variances between the same phones from different speakers. That proves speech recognition front end sure will help in speaker identification task. And it is well known that text-dependent speaker

identification performs far better than text-independent speaker identification. So it makes sense to convert text-independent problem into text-dependent. This observation also proves the reason why we can do **speaker-independent** speech recognition.

C.1.1 general_phonotactic

start(<S>).

<S> —> BACKGROUND <S>.

<S> —> word <S>.

<S> —> "".

start(word).

word —> syllable restofword.

restofword —> syllable restofword.

restofword —> suffix.

restofword —> "".

C.1.2 background

start(BACKGROUND).

BACKGROUND —> $SILENCE.

BACKGROUND —> $INHALATION.

BACKGROUND —> $EXHALATION.

BACKGROUND —> <NULL>.

C.1.3 syllable

start(syllable).

syllable —> onset syll_1.

syllable —> nucleus syll_3.

syll_1 —> nucleus syll_3.

syll_3 —> coda.

syll_3 —> "".

C.1.4 onset

start(onset).

onset —> onset_consonant.

onset —> s s_cluster.

onset —> pbf pbf_cluster.

onset —> tdth tdth_cluster.

onset —> kg kg_cluster.

onset —> sh r.

start(s_cluster).

s_cluster —> p pbf_cluster.

s_cluster —> t tdth_cluster.

s_cluster —> k kg_cluster.

s_cluster —> m.

s_cluster —> n.

s_cluster —> l.

s_cluster —> w.

s_cluster —> f.

start(pbf).

pbf —> p.

pbf —> b.

pbf —> f.

start(pbf_cluster).

pbf_cluster —> l.

pbf_cluster —> r.

start(tdth).

tdth —> t.

tdth —> d.

tdth —> th.

start(tdth_cluster).

tdth_cluster —> r.

tdth_cluster —> w.

start(kg).

kg —> k.

kg —> g.

start(kg_cluster).

kg_cluster —> r.

kg_cluster —> l.

kg_cluster —> w.

C.1.5 nucleus

start(nucleus).

nucleus —> vocalic_nucleus.

nucleus —> syll_nasal.

start(vocalic_nucleus).

vocalic_nucleus —> y uw.

vocalic_nucleus —> aa.

vocalic_nucleus —> ae.

vocalic_nucleus —> ah.

vocalic_nucleus —> ao.

vocalic_nucleus —> aw.

vocalic_nucleus —> ax.

vocalic_nucleus —> axr.

vocalic_nucleus —> ay.

vocalic_nucleus —> eh.

vocalic_nucleus —> ih.

vocalic_nucleus —> ix.

vocalic_nucleus —> iy.

vocalic_nucleus —> ow.

vocalic_nucleus —> oy.

vocalic_nucleus —> uh.

vocalic_nucleus —> uw.

vocalic_nucleus —> er.

vocalic_nucleus —> ey.

vocalic_nucleus —> el.

start(syll_nasal).

syll_nasal —> em.

syll_nasal —> en.

C.1.6 coda

start(coda).

coda —> obstruent.

coda —> n.

coda —> m.

coda —> ng.

coda —> nasal_cluster.

coda —> liquid.

coda —> liquid obstruent.

coda —> liquid m.

coda —> liquid n.

coda —> r l.

coda —> s p.

coda —> s t.

coda —> s k.

coda —> sh t.

coda —> k s.

start(nasal_cluster).

nasal_cluster —> m p.

nasal_cluster —> m f.

nasal_cluster —> n ch.

nasal_cluster —> n jh.

nasal_cluster —> ng k.

start(liquid).

liquid —> l.

liquid —> r.

C.1.7 suffix

start(suffix).

suffix —> t.

suffix —> d.

suffix —> s.

suffix —> z.

suffix —> th.

suffix —> s t.

suffix —> t s.

suffix —> d z.

suffix —> th s.

C.1.8 consonants

start(onset_consonant).

onset_consonant —> obstruent.

onset_consonant —> onset_sonorant.

start(obstruent).

obstruent —> b.

obstruent —> ch.

obstruent —> d.

obstruent —> dh.

obstruent —> f.

obstruent —> g.

obstruent —> jh.

obstruent —> k.

obstruent —> p.

obstruent —> s.

obstruent —> sh.

obstruent —> t.

obstruent —> th.

obstruent —> v.

obstruent —> z.

obstruent —> zh.

start(onset_sonorant).

onset_sonorant —> m.

onset_sonorant —> n.

onset_sonorant —> l.

onset_sonorant —> r.

onset_sonorant —> w.

onset_sonorant —> wh.

onset_sonorant —> y.

onset_sonorant —> hh.

onset_sonorant —> dx.

Bibliography

[1] Alejandro Acero. *Acoustical and Environmental Robustness in Automatic Speech Recognition.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1990.

[2] Alejandro Acero and Richard M. Stern. Environmental robustness in automatic speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, 1990.

[3] Alejandro Acero and Richard M. Stern. Robust speech recognition by normalization of the acoustic space. In *International Conference on Acoustics, Speech, and Signal Processing*, 1991.

[4] Gerome R. Bellegarda and David Nahamoo. Tied mixture continuous parameter models for large vocabulary isolated speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, 1989.

[5] Steven F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-27(2), April 1979.

[6] Leo Breiman, Jerome H. Friedman, Richard E. Olsen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth Statistics and Probability Series, 1984.

[7] William H. Equitz. A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-37(10), Oct. 1989.

[8] S. Furui. Research on individuality features in speech waves and automatic speaker recognition techniques. *Speech Communication 5*, 1986.

[9] Saul B. Gelfand, C. S. Ravishankar, and Edward J. Delp. An iterative growing and pruning algorithm for calssification tree design. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-13(2), Feb. 1991.

[10] Allen Gersho. On the structure of vector quantizers. *IEEE Transactions on Information Theory*, IT-28(2), March 1982.

[11] Herbert Gish. Robust discrimination in automatic speaker identification. In *International Conference on Acoustics, Speech, and Signal Processing*, 1990.

[12] Herbert Gish, Michael Krasner, William Russell, and Jare3d Wolf. Methods and experiments for text-independent speaker recognition over telephone channels. In *International Conference on Acoustics, Speech, and Signal Processing*, 1986.

[13] John J. Godfrey, Edward C. Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[14] Yifan Gong and Jean-Paul Haton. Text-independent speaker recognition by trajectory space comparison. In *International Conference on Acoustics, Speech, and Signal Processing*, 1990.

[15] A. L. Higgins and L. G. Bahler. Text-independent speaker verification by discriminant count. In *International Conference on Acoustics, Speech, and Signal Processing*, 1991.

[16] Biing-Hwang Juang, Lawrence R. Rabiner, and Jay G. Wilpon. On the use of bandpass liftering in speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(7), July 1987.

[17] Yu-Hung Kao, P. K. Rajasekaran, and John S. Baras. Free-text speaker identification over long distance telephone channel using hypothesized phonetic segmentation. In *International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[18] Yoseph Linde, Andres Buzo, and Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1), Jan. 1980.

[19] T. Matsui and S. Furui. A text-independent speaker recognition method robust against utterance variation. In *International Conference on Acoustics, Speech, and Signal Processing*, 1991.

[20] D. A. Reynolds and R. C. Rose. An integrated speech-background model for robust speaker identification. In *International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[21] R. C. Rose, J. A. Fitzmaurice, E. M. Hofstetter, and D. A. Reynolds. Robust speaker identification in noisy environment using noise adaptive speaker models. In *International Conference on Acoustics, Speech, and Signal Processing*, 1991.

[22] R. C. Rose and D. A. Reynolds. Text-independent speaker identification using automatic acoustic segmentation. In *International Conference on Acoustics, Speech, and Signal Processing*, 1990.

[23] Marvin R. Sambur. Selection of acoustic features for speaker identification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23(2), April 1975.

[24] M. Savic and J. Sorensen. Text-independent speaker recognition based on phonetic segmentation. In *Speech Research Symposium X*, 1990.

[25] F. K. Soong, A. E. Rosenberg, L. R. Rabiner, and B. H. Juang. A vector quantization approach to speaker recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, 1985.

[26] Belle L. Tseng, Frank K. Soong, and Aaron E. Rosenberg. Continuous probabilistic acoustic map for speaker recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[27] B. Wheatley. Robust automatic time alignment of orthographic transcriptions with unconstrained speech. In *International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[28] B. Wheatley and J. Picone. Voice across america: Toward robust speaker-independent speech recognition for telecommunications applications. *Digital Signal Processing*, April 1991.