Robust Speech Recognition in Noise Using Statistical Signal Mapping

by

Jean-Guy Dahan

B. Eng.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Department of Electrical Engineering
McGill University
Montréal, Canada
February, 1994

Abstract

This study aims to apply the Statistical Signal Mapping method to robust speech recognition. Using the method, a mapping function for transforming noisy speech observations to clean vectors can be found and used as a preprocessor for a recognition system. The statistical structure of the mapping function assumes N and M Gaussian random sources in the noisy and clean vector spaces respectively. Activities of the noisy sources are determined by a set of a-priori probabilities, and those of the clean vector sources are measured by a N-by-M correlation matrix between sources of each space. The function's parameters are derived from a training sequence of corresponding observations from each space, using the EM reestimation algorithm. The free variables associated with the method include the number of sources, number of features per observation vector and feature extraction type, each of which are tuned for best performance. Because there is no guarantee to converge to the global optimum in the algorithm, the initialization plays an important role, and VQ has been specialized to a certain extent for this application. The method is based on cepstral observations and is tailored for recognition systems using these as features. The optimization criterion approximates the optimal MAP estimation, and is compatible with that of the recognizers using continuous density Gaussian models. The approach is frame-based, but we attempt to use correlation across frames using a variety of schemes including multi-frame mapping and frame averaging. The function can also be specialized for voiced, unvoiced, and silent portions of speech when a detector of such type is used in conjunction. Since noisy spectra are mapped rather than filtered, any enhancement method can be applied to improve the feature extraction of the input noisy speech. We currently use a noise-independent method based on convolution of the power spectrum with a function of spectral lateral inhibition. This not only improves performance at fixed SNR but also reduces the dependence of the function to changes in test noise level. When trained on a clean speech database and tested with additive white Gaussian noise, our algorithm increases the recognition rate to within 5 % of that of the system trained and tested with noisy speech, and corresponds to an effective improvement of 17 dB for input speech of 10 dB SNR. Distortion between recovered (or mapped) noisy speech and actual clean speech has also shown an effective improvement of 14 dB, using the variance weighted cepstral distance measure. The algorithm requires only a fraction of the training needed to retrain the entire recognition system, and computations required at the testing stage

are minimal compared to those of the recognition system.

Sommaire

Cette étude vise à appliquer une méthode de transformation statistique de signal (Statistical Signal Mapping) à la reconnaisance de parole robuste. Une fonction de transformation pour relier les observations de parole bruitée aux vecteurs de parole sans bruits peut être trouvée et employée comme pré-processeur de système d'un système de reconnaisance de parole en utilisant cette méthode. La structure statistique de la fonction de transformation présuppose qu'il y a N et M sources aléatoires Gaussiennes dans les espaces de vecteurs bruité et propre respectivement. Les activités des sources bruitées sont déterminées par un ensemble de probabilités à-priori et celles des sources de vecteurs propres sont mesurées par une matrice de corrélation de N par M, entre les sources de chaque espace. Les paramètres de la fonction sont dérivés d'une séquence d'entraînement d'observations corespondantes de chaque espace, et emploie l'algorithme de réestimation EM (Expectation-Modification). Les différentes variables associées à la méthode incluent le nombre de sources, le nombre de traits par vecteur d'observation, et le genre de détermination de traits, chacune d'elles étant syntonisée pour une meilleure performance. Puisqu'il n'y a aucune assurance de convergence à une optimalité globale dans l'algorithme, l'initialisation joue un rôle important et la quantification vectorielle (VQ) a été particularisée à une certaine mesure dans cette application. La méthode est basée sur des observations cepstrales et est façonnée pour les systèmes de reconnaisance les employant comme traits. Le critère d'optimisation est une approximation de l'estimation optimale MAP (maximum a-posteriori probability), et donc est compatible avec les systèmes de reconnaissance utilisant des modèles Gaussiens de densité continue. L'approche est basée par trame, mais nous essayons d'utiliser les corrélations à travers les trames en utilisant une variété de méthodes incluant une transformation multi-trames et l'établissement de trames moyennes. La fonction peut aussi être particularisée pour les parties de parole vocalisée, non-vocalisée et silencieuse, quand un détecteur d'un tel type est employé de concert. Puisque les spectres bruyants sont cadrés plutôt que filtrés, toute méthode de réhaussement peut être appliquée pour améliorer la détermination de traits de la parole bruitée d'entrée. Actuellement, nous employons une méthode qui ne dépend pas du bruit basée sur la convolution du spectre de puissance avec une fonction d'inhibition latérale spectrale. Celle-ci améliore la performance à SNR fixe mais aussi reduit la dépendence de la fonction aux changements du niveau de bruit. Lorsqu'entraîné par une base de données de parole propre et

testé avec l'ajout de bruit blanc Gaussien, notre algorithme augmente le taux de reconnaissance à 5 % près du système entraîné et testé avec la parole bruitée. Ceci correspond à une amélioration effective de 17 dB pour une parole d'entrée à 10 dB SNR. La distortion entre la parole bruitée récupérée et la vraie parole claire a aussi démontré une amélioration effective de 14 dB, en employant la mesure de distance cepstrale pondérée. L'algorithme n'exige qu'une fraction de l'entraînement requis par le réentraînement du système de reconnaissance entier. Les calculs exigés au stage de testing sont minimals comparés à ceux du reconnaisseur.

Acknowledgements

I would like to thank my supervisor, Professor Douglas O'Shaughnessy for his guidance throughout my graduate studies. The help in choosing a research area, and subsequent motivation and feedback I received from Yan Ming Cheng of Bell Northern Research proved to be essential and very helpful. The research was conducted at l'Institut National de la Recherche Scientifique (INRS)- Télécommunications laboratories. All the facilities provided by INRS contributed greatly to the accomplishments of this work. Financial support provided by the Natural Science and Engineering Research Council (NSERC) was very much appreciated.

The thesis could not have been completed with out the support and motivation of my family. Special thanks go to my father for his constant encouragement to finish.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Person-Machine communication by voice has been a dream ever since the first computers were developed tens of years ago. The link is bi-directional and so includes speech synthesis (or text-to-speech conversion) and ASR, automatic speech recognition (or speech-to-text conversion). Since human hearing is quite adaptable, imperfections in speech synthesizers producing unnatural and rugged-sounding speech have been acceptable to many users. In contrast, limitations of computer "hearing" have made it difficult to recognize speech largely due to the myriad of variables involved, whether they be related to the speaker or the speaking environment.

Two main methods of speech recognition have been developed over the past 20 years: Hidden Markov Models (HMM) and Dynamic Time Warping (DTW). In both methods the recognition system must first be trained. Even for the simplest of systems up to one hour of speech may be necessary for the system to collect statistics and trends in order to form "templates" against which new speech can be compared during the actual recognition tests. The templates, or speech models the system has learned, also define what the system can be used for in recognition. If an unknown speaker attempts to use the system, performance will be close to nil since the models were designed specifically for the speaker from training. This is also the case when any other conditions from the training stage have changed. These can include level of ambient noise, any channel variations (microphone, telephone line) or presence of other sounds. Although such conditions pose no problem to humans, these can be extremely detrimental to ASR performance.

Some of these problems can be solved by retraining the speech models of the recognition system under the new conditions. This is undesirable because it is very costly in both time and effort. Alternatively, the speech models can be made more general, i.e., trained on a variety of speakers and under different environments. Unless we know recognition will be needed under these various conditions, this solution is less desirable since performance invariably decreases in this way.

Making speech recognition more robust to environment changes is the subject of this thesis. We specifically focus on robust speech recognition in noise, but we believe our method can equally be applied when other changes in recording environment are present, or even when new speakers are to use the recognition system, making it speaker adaptable.

A large part of recent recognition research has been involved with strictly clean speech, meaning speech recorded using high quality microphones and very low noise. Recognition rates under these conditions have reached beyond 95% but it is unlikely such systems could be presented in the marketplace. Offices, computer rooms, phone lines, and other real user environments rarely have the same conditions as those of speech labs. Furthermore, the recording environment can change very much. Whether the sound quality increases or decreases, recognition in a changing environment, different from that in training, is always detrimental to performance.

Noise has a large detrimental effect on the performance of recognition when training has been performed on clean speech, at times reducing recognition rates by an order of magnitude. While it is true that some information within the speech signal has been lost to noise, much of the blame lies in the fact that the noisy speech is distorted. Noisy speech is still quite intelligible to humans and most of the degraded recognition performance due to noise can be overcome by simply retraining the recognizer in the noisy environment. These ideas suggest that most if not all of the speech information is still within the noisy speech. Nevertheless, it is clear that the speech features cannot be extracted in the same way as with clean speech.

Speech enhancement systems have been widely used to remove noise present in speech. Many methods have enjoyed success in the sense that much of the noise component is removed along with its unpleasant nature and intelligibility is improved slightly. In general these have not been successful in improving recognition in noise

since they often change the properties of speech spectra, and it is already known that speech unfamiliar to the computer is poorly recognized regardless of its quality.

For this reason the specialized field of speech recognition in noise has appeared and many useful schemes have been developed. Whereas the signal processing in the human auditory system is mostly unknown, that of recognition systems is. In this light it should be easier to define an optimality criterion of enhancement for recognition. It is desirable to define an optimality criterion compatible with that of the recognition system.

Most recognition systems are based on a weighted cepstral distance measure, where the cepstrum is typically derived by a cosine transformation of the logarithm of energies in a filterbank, and is truncated to about 8-12 coefficients. Of the enhancement methods using similar criteria, several studies should be noted.

Erell and Weintraub [12] have developed a method of MMSE estimation of filterbank log energies, for application to noisy speech recognition. Based on learned statistics of noisy and clean speech, their criterion minimizes the mean log spectral distance or the error in filterbank log energies. This corresponds to minimizing the nonweighted, nontruncated cepstral distance, rather than the weighted, truncated one used by the recognizer. The difficulty lies in modelling the statistics of additive noise in the cepstral domain. By the nature of our formulation, this difficulty has been overcome.

Acero and Stern [1] have conducted studies in cepstral normalization for different microphones and noise levels in which an additive correction vector is applied to the cepstrum. Although their study has been fruitful, it has only addressed problems with low and roughly constant noise levels.

Other attempts to improve robustness of speech recognizers in noisy environments have been to estimate the uncorrupted signal from the noisy speech. Van Compernolle [24] has used spectral subtraction, but at the cost of losing some information when the input speech has only a little noise. Other studies have relied on modified distance metrics in the template matcher for noisy speech [19][22], but these approaches are difficult to use in statistically based modelling methodologies, particularly for metrics which are asymmetric. Semi-continuous HMMs with noise-adaptive models were introduced by Nadas et al [20] with success, but these require almost double the

computation of conventional HMMs.

The approach presented in this thesis relies on a spectral mapping from noisy to clean speech in the cepstral domain. In our opinion, it is a more comprehensive approach for the recovery of clean speech from noisy speech than any other known method.

We assume that there is an underlying correlation between clean and noisy speech and that there exists a mapping function (albeit non-linear) which can transform noisy speech observations to clean ones. We model the non-linear function by a parametric statistical mapping function and estimate the parameters of the function by training from a long sequence of vector pairs, the elements of each pair being noisy and clean realizations of the same speech. The approach was originally developed by Cheng, O'Shaughnessy, and Mermelstein [8][7][9], but in our study it has been specialized and applied for the first time to robust speech recognition.

Once the parameters of the mapping function are estimated, the function is applied to new noisy speech in order to estimate the corresponding clean speech. Recognition and other tests are then applied to the recovered speech. In contrast to the recognition system, the mapping function needs only a fraction of the amount of training, so it creates a good alternative to retraining the recognizer.

The statistical structure used to model the non-linear mapping function can be described as follows. We assume the input noisy speech vectors are completely generated by N random sources with Gaussian pdfs. The corresponding clean speech vectors are outputs of the mapping function and are assumed to be produced by M random sources with Gaussian pdfs. The activities of the input (noisy) sources are determined by a set of a priori probabilities, and those of the output (clean) sources are measured by a $N - by - M$ correlation matrix between the two sets of sources. An iterative estimation of the parameters of the statistical structure using the EM algorithm with the training data has been developed. Thus, given an input noisy vector, we can estimate the probability of any output clean vector. Furthermore, using any optimization criterion and the estimates of probabilities, we can obtain an estimate of the clean vector in the optimization sense. Because there is no guarantee to converge to the global optimum in the algorithm, its initialization plays an important role. Many facets of the method have been explored, including the number of

sources, number of features per observation vector, feature generation, organization of the feature space and initialization of the mapping parameters. Since the mapping function is trained on noisy speech of a specific SNR, we also study methods by which the same function can be useful over a wider range of SNRs.

1.1 Organization of the Thesis

The ultimate aim of this study is to develop a method by which a speech recognizer trained on clean speech can be used to recognize noisy speech. Chapter 2 reviews the mathematical theory behind statistical mapping. The development is general, making reference to robust speech recognition only to clarify its use. Issues of implementation of the mapping function for robust speech recognition are discussed in Chapter 3, where each of the variables associated with the method are optimized for best performance, these include the number of sources and features, the initialization method, and the recovery method. Since the approach is to estimate vectors frame-by-frame, in Chapter 4 we attempt to use information beyond the frame boundaries to improve recognition. We explored different ways in which features can incorporate contextual information, ways in which phonemic segmentation information can be used and finally a way in which the feature extraction can be improved.

The different optimizations are used together in Chapter 5. Realizing that the mapping function will be dependent on the noise-level at which it has been trained, we attempt to monitor the degree of this dependence. We investigate a noise independent mapping scheme, so that the method can be more robust to changes in noise-level. Another approach to varying noise, combined mapping functions, is also described.

The overall performance of the system is reviewed and discussed in Chapter 6, where we also present other possibilities for performance improvement not explored within this study. Other applications of the mapping technique are also suggested.

Chapter 2

Statistical Signal Mapping

2.1 Introduction

The previous chapter has introduced the area of speech recognition in noisy environments. In particular, motivation for a speech enhancement system using spectral mapping was presented. In this chapter, the general method will be mathematically developed.

We assume a non-linear function $y = f(x)$ exists which can map noisy speech vectors to their clean counterparts. A method by which this mapping function can be modelled and identification of the model parameters will given in this introduction.

The training algorithm, by which the parameters of the model are estimated, will be developed in the next section. The testing stage, in which the function is actually applied, will be presented in section 2.3. This will be followed by an overview of the computational requirements and limitations of the method and finally by a preliminary analysis of these procedures with synthetic data.

Since the problem of relating vectors from two spaces has many applications in speech processing, the mathematical development will be general. Specific application of the method to speech enhancement for recognition will be given in the following chapters.

A general mapping function maps vectors from one observation space $\mathcal{X}$, called the source space, to another space $\mathcal{Y}$, called the target space.

Let $\mathbf{x}$ be the random variable defined on the sample space $\mathcal{X}$ of all possible source vectors, and $\mathbf{y}$ be the random variable defined in $\mathcal{Y}$ of target vectors.

The sample vectors of $\mathbf{x}$ are in $K_{\mathcal{X}}$ -dimensional space $\mathcal{X}$ with corresponding vector sample vectors of $\mathbf{y}$ in a $K_{\mathcal{Y}}$ -dimensional target space $\mathcal{Y}$. We assume that the vectors $x \in \mathcal{X}$ are generated by N random sources, $\lambda_i, 1 \leq i \leq N$, and the vectors $y \in \mathcal{Y}$ are generated by M random sources, $\theta_j, 1 \leq j \leq M$.

With no loss of generality, successive realizations of $\mathbf{x}$ and $\mathbf{y}$ can be indexed by time, but are not necessarily time sequences. Assuming the random sources are Gaussian we can write

$$p(x_t|\lambda_i) = \frac{1}{(2\pi)^{Kx/2} det^{1/2}\Sigma_{\lambda_i}} \exp\{-\frac{1}{2}(x_t - \mu_{\lambda_i})^{\#}\Sigma_{\lambda_i}^{-1}(x_t - \mu_{\lambda_i})\} \tag{2.1}$$

and

$$p(y_t|\theta_j) = \frac{1}{(2\pi)^{K_{\mathcal{Y}}/2} det^{1/2}\Sigma_{\theta_j}} \exp\{-\frac{1}{2}(y_t - \mu_{\theta_j})^{\#}\Sigma_{\theta_j}^{-1}(y_t - \mu_{\theta_j})\}, \tag{2.2}$$

where μ_{λ_i} and Σ_{λ_i} are respectively the mean vector and covariance matrix for the source λ_i, μ_{θ_j} and Σ_{θ_j} are respectively the mean vector and covariance matrix for source θ_j, and $\#$ denotes vector transpose.

The choice of these distributions is optional within the framework of this method, but for consistency and convenience they have been chosen to be the same as that in our speech recognition system with single mixture continuous-density Gaussian observations. The Gaussian autoregressive assumption on the observations gives rise to alternate distributions $p(x_t|\lambda_i)$ and $p(y_t|\theta_j)$, and have been used effectively in the SSM method in [9].

Since the vectors will be generated by sources, and we wish to find a mapping between the vectors, we will try to map sources from one space to the other. The source cross-correlation probability, $\alpha_{ij} = p(\theta_j|\lambda_i)$, is the probability that source θ_j is active in $\mathcal{Y}$ given that source λ_i is active in $\mathcal{X}$. The non-linear mapping function $f(\cdot)$ can now be defined in terms of the three parameters: the source cross-correlation matrix $A = \{\alpha_{ij}\}$; the source space vector sources $\Lambda = \{\lambda_i\}$, and the target space vector sources $\Theta = \{\theta_j\}$. With an input vector x, we have

$$y = f(x, A, \Lambda, \Theta) \tag{2.3}$$

With this formulation, the problem of estimation of source statistics and correlations becomes the much simpler problem of parameter estimation. The parameters of the function can be estimated using a long sequence of training data containing corresponding vectors from each space. This is called the training stage and it is described in the following section. In the testing stage, presented in section 2.3, new vectors from the space $\mathcal{X}$ will be given, and the corresponding vectors from the space $\mathcal{Y}$ will be estimated by mapping. The estimates will be compared to the actual corresponding vectors so that the effectiveness of the mapping function can be measured.

2.2 Training method: Estimation of the mapping function parameters

Let $X = \{x_t\}_{t=1}^T \in \mathcal{X}$ denote the long training sequence from one space, with corresponding vectors $Y = \{y_t\}_{t=1}^T \in \mathcal{Y}$ from the target space. It is from these vector pairs that the mapping function must be trained. The likelihood function of the model parameters and the training sequence will be derived and the "optimal" model parameters will be found by maximization of this function with respect to these.

Consider the joint probability distribution function (pdf) for the training vectors and individual sources at time t,

$$\begin{aligned} p(x_t, y_t, \lambda_i, \theta_j) &= p(y_t|x_t, \lambda_i, \theta_j)p(x_t, \lambda_i, \theta_j) \\ &= p(y_t|x_t, \lambda_i, \theta_j)p(\theta_j|x_t, \lambda_i)p(x_t, \lambda_i) \\ &= p(y_t|x_t, \lambda_i, \theta_j)p(\theta_j|x_t, \lambda_i)p(x_t|\lambda_i)p(\lambda_i) \end{aligned} \tag{2.4}$$

Since y_t is assumed to be generated solely by the sources θ_j, and activation of each source θ_j is dependent only on activation of the sources λ_i, we can simplify eq. (2.4) as:

$$\begin{aligned} p(x_t, y_t, \lambda_i, \theta_j) &= p(y_t|\lambda_j)p(\theta_j|\lambda_i)p(x_t|\lambda_i)p(\lambda_i) \\ &= p(y_t|\lambda_j)\alpha_{ij}p(x_t|\lambda_i)p(\lambda_i). \end{aligned} \tag{2.5}$$

Given the training sequences from each space $(X, Y) = \{x_t, y_t | 1 \leq t \leq T\}$ (corresponding to noisy and clean speech vectors in the speech enhancement application),

and denoting the set of mapping parameters by $\Phi = \{A, \Lambda, \Theta\}$, the joint probability of the two sequences is

$$\begin{aligned} p(X,Y) &= p(\Phi, X, Y) \\ &= \prod_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{M} p(x_t, y_t, \lambda_i, \theta_j) \\ &= \sum_{k=1}^{(NM)^T} \prod_{t=1}^{T} p(x_t, y_t, s_k(t)) \\ &= \sum_{k=1}^{(NM)^T} p(\Phi, X, Y, S_k) \end{aligned} \tag{2.6}$$

where $s_k(t) = [\lambda_i, \theta_j], 1 \leq k \leq (NM)$ is the event of activation of sources λ_i and θ_j at time t, and $S_k, 1 \leq k \leq (NM)^T$is the set of events $s_k(t), 1 \leq k \leq (NM)$ at each time t.

We wish to estimate the set of parameters Φ, which maximize the likelihood of observing the joint sequence (X, Y). The likelihood function is $p(\Phi, X, Y)$, but for convenience we can maximize its log(). The "optimal" values for the set of parameters we seek occur at the maximum likelihood:

$$\Phi^* = \arg\max_{\Phi} \log p(\Phi, X, Y). \tag{2.7}$$

As in HMM parameter estimation there is no closed form formula which can solve eq. (2.7), but in the same way that the EM algorithm can estimate the parameters of an HMM model, it can be used to estimate the "optimal" parameters Φ^* of the mapping function.

The EM algorithm [10]

The EM algorithm was originally presented as a general method for computing iterative estimates of maximum likelihood. Given an initial estimate of source parameters, it can find better estimates by maximization of a likelihood function of data and source parameters. Since there are only a finite number of observations (T), the EM algorithm views the data (X, Y) as an incomplete set enabling satisfactory estimation. The procedure consists of two steps. The E-step computes the expectation of the likelihood function over the event paths (eq. 2.6) based on an initial estimate of the parameters $\Phi_0 = (A_0, \Lambda_0, \Theta_0)$. In the M-step, the expectation function is maximized by modification of the parameters. In practice, we do not use the likelihood function but rather an auxiliary function[2], as in HMM parameter estimation.

Here, the likelihood function to maximize is $\log p(\Phi, X, Y)$, and we assume an initial estimate $\Phi_0 = (A_0, \Lambda_0, \Theta_0)$ of the parameters has been made. Methods for making this initial guess are discussed in section 3.4.3. Given Φ_0, the auxiliary function is

$$\begin{aligned} Q(\Phi|\Phi_0) &= E(\log p(\Phi, X, Y, S_k)|\Phi_0) \\ &= \textstyle\sum_{k=1}^{(NM)^T} p(\Phi_0, X, Y, S_k) \log p(\Phi, X, Y, S_k) \\ &= \textstyle\sum_{k=1}^{(NM)^T} p(\Phi_0, X, Y, S_k) \sum_{t=1}^{T} \{\log p(\theta_j \in s_{k(t)}|\lambda_i \in s_{k(t)}) \\ &\qquad + \log p(x_t|\lambda_i \in s_{k(t)}) + \log p(y_t|\theta_j \in s_{k(t)}) + \log p(\lambda_i \in s_{k(t)})\} \end{aligned} \tag{2.8}$$

We can find the source correlation matrix α_{ij} using Lagrange optimization on $Q(\cdot)$ with the constraint

$$\sum_{j=1}^{M} \alpha_{ij} = 1 \quad . \tag{2.9}$$

The Lagrange function to maximize is

$$L(\Phi|\Phi_0, \beta) = Q(\Phi|\Phi_0) - \beta(\sum_{j=1}^{M} \alpha_{ij} - 1) \tag{2.10}$$

where β is the Lagrange multiplier. Taking the partial derivatives with respect to $\alpha_{ij} = p(\theta_j|\lambda_i)$, we obtain

$$\frac{\partial}{\partial \alpha_{ij}}(\cdot) = \frac{\sum_{k=1}^{(NM)^T} p(\Phi_0, X, Y, S_k) \sum_{t=1}^{T} \delta(s_k(t) = [\lambda_i, \theta_j])}{\alpha_{ij}} - \beta = 0 \tag{2.11}$$

where the truth function $\delta(\cdot)$ equals 1 for a true argument or otherwise 0. Solving for α_{ij} in (2.11) and substituting in (2.9), we obtain

$$\alpha_{ij} = \frac{C_{ij}}{\sum_{j=1}^{M} C_{ij}} \tag{2.12}$$

where

$$C_{ij} = \sum_{k=1}^{(NM)^T} p(\Phi_0, X, Y, S_k) c_{ij}(S_k) \tag{2.13}$$

is the expected count of the event $[\lambda_i, \theta_j]$ being seen over all t, and $c_{ij}(S_k)$ is the count of the event S_k at each t.

The summation over all $(NM)^T$ event paths can be split into a double summation over events and time,

$$\begin{aligned} C_{ij} &= \sum_{k=1}^{(NM)} \sum_{t=1}^{T} p(\Phi_0, x_t, y_t, s_{k(t)}) c_{ij}(s_{k(t)}) \\ &= \sum_{t=1}^{T} p(t : \lambda_i, \theta_j, X, Y) \\ &= \sum_{t=1}^{T} p(t : \lambda_i, \theta_j | X, Y) p(X, Y) \end{aligned} \tag{2.14}$$

An efficient form for α_{ij} can now be found,

$$\begin{aligned} \alpha_{ij} &= \frac{\sum_{t=1}^{T} p(t:\lambda_i,\theta_j|X,Y)p(X,Y)}{\sum_{m=1}^{M}\sum_{t=1}^{T} p(t:\lambda_i,\theta_m|X,Y)p(X,Y)} \\ &= \frac{\sum_{t=1}^{T} \frac{p(x_t,y_t,\lambda_i,\theta_j)}{p(x_t,y_t)}}{\sum_{t=1}^{T}\sum_{m=1}^{M} \frac{p(x_t,y_t,\lambda_i,\theta_m)}{p(x_t,y_t)}} \end{aligned} \tag{2.15}$$

where $p(X, Y)$ has been dropped since it is independent of i, j, t. Expressing $p(x_t, y_t)$ in terms of the known distribution $p(x_t, y_t, \lambda_i, \theta_j)$ we finally obtain the reestimation formula for the source correlation matrix,

$$\alpha_{ij} = \frac{\sum_{t=1}^{T} \frac{p(x_t,y_t,\lambda_i,\theta_j)}{\sum_{n=1}^{N}\sum_{m=1}^{M} p(x_t,y_t,\lambda_n,\theta_m)}}{\sum_{t=1}^{T} \frac{\sum_{m=1}^{M} p(x_t,y_t,\lambda_i,\theta_m)}{\sum_{n=1}^{N}\sum_{m=1}^{M} p(x_t,y_t,\lambda_n,\theta_m)}} \tag{2.16}$$

A reestimation equation for the a-priori source probabilities $p(\lambda_i)$ can also be derived by Lagrange optimization on the auxiliary function $Q(\Phi|\Phi_0)$, with the constraint $\sum_{i=1}^{N} p(\lambda_i) = 1$. We find that $p(\lambda_i)$ is the count of events $[\lambda_i]([\lambda_i, \theta_j] \forall j)$ divided by the total count of events $([\lambda_i, \theta_j] \forall i, j)$:

$$\begin{aligned} p(\lambda_i) &= \frac{\sum_{j=1}^{M} C_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij}} \\ &= \frac{1}{T}\sum_{t=1}^{T} p(t : \lambda_i | X, Y) \\ &= \frac{1}{T}\sum_{t=1}^{T} \frac{\sum_{m=1}^{M} p(x_t,y_t,\lambda_i,\theta_m)}{\sum_{n=1}^{N}\sum_{m=1}^{M} p(x_t,y_t,\lambda_i,\theta_m)} \end{aligned} \tag{2.17}$$

where $\sum_{i=1}^{N}\sum_{j=1}^{M} C_{ij} = T$ is the total count of all events $([\lambda_i, \theta_j] \forall\ i, j)$.

Finally, the reestimation formulas for the distributions $p(x_t|\lambda_i)$ and $p(y_t|\theta_j)$ can be found by Lagrange optimization with respect to μ_{λ_i}, μ_{θ_j}, Σ_{λ_i}, and Σ_{θ_j}. For continuous density Gaussian sources, the reestimation formulas are the same as those for Gaussian HMMs:

$$
\begin{aligned}
\mu'_{\lambda_i} &= \frac{\sum_{t=1}^{T} x_t p(\lambda_i|x_t,y_t)}{\sum_{t=1}^{T} p(\lambda_i|x_t,y_t)} \\
\Sigma'_{\lambda_i} &= \frac{\sum_{t=1}^{T} (x_t-\mu_{\lambda_i})^{\#}(x_t-\mu_{\lambda_i}) p(\lambda_i|x_t,y_t)}{\sum_{t=1}^{T} p(\lambda_i|x_t,y_t)} \\
\mu'_{\theta_j} &= \frac{\sum_{t=1}^{T} y_t p(\theta_j|x_t,y_t)}{\sum_{t=1}^{T} p(\theta_j|x_t,y_t)} \\
\Sigma'_{\theta_j} &= \frac{\sum_{t=1}^{T} (y_t-\mu_{\theta_j})^{\#}(x_t-\mu_{\theta_j}) p(\theta_j|x_t,y_t)}{\sum_{t=1}^{T} p(\theta_j|x_t,y_t)}
\end{aligned}
\tag{2.18}
$$

where

$$
p(\lambda_i|x_t,y_t) = \frac{\sum_{m=1}^{M} p(x_t,y_t,\lambda_i,\theta_m)}{\sum_{n=1}^{N}\sum_{m=1}^{M} p(x_t,y_t,\lambda_n,\theta_m)} \tag{2.19}
$$

and

$$
p(\theta_j|x_t,y_t) = \frac{\sum_{n=1}^{N} p(x_t,y_t,\lambda_n,\theta_j)}{\sum_{n=1}^{N}\sum_{m=1}^{M} p(x_t,y_t,\lambda_n,\theta_m)} \quad . \tag{2.20}
$$

For autoregressive Gaussian sources, other updating formulas for the autoregressive coefficients can be found in [9][8].

The training algorithm reestimates the parameters in the joint pdf $p(x_t, y_t, \lambda_i, \theta_j) = p(y_t|\theta_j)\alpha_{ij}p(x_t|\lambda_i)p(\lambda_i)$ by performing eqs. (2.16) - (2.18) iteratively. At each iteration the log likelihood of source and target vectors (X, Y), can be calculated:

$$
\log p(X,Y) = \log \prod_{t=1}^{T} p(x_t,y_t) = \sum_{t=1}^{T} \log \sum_{i=1}^{N}\sum_{j=1}^{M} p(x_t,y_t,\lambda_i,\theta_j) \quad . \tag{2.21}
$$

By the EM algorithm, this quantity is guaranteed to increase at each iteration. It is also proven [2][10] that the log likelihood, $\log p(X, Y)$, tends a local maximum at which point the mapping parameters converge to a critical point. In practice the training algorithm is stopped after a fixed number of iterations or when the change in log likelihood falls below a certain threshold.

It is important to stress that the critical point reached is only a local maximum, and depends on the initial choice of parameters Φ_0. The sensitivity of the mapping function to changes in the initial Φ_0 will be exposed in section 3.4.3. The following section will discuss how the mapping function is used to estimate output vectors by mapping the input vectors. Figure 2.1 shows the block diagram of the system which realizes the training procedure and testing procedures.

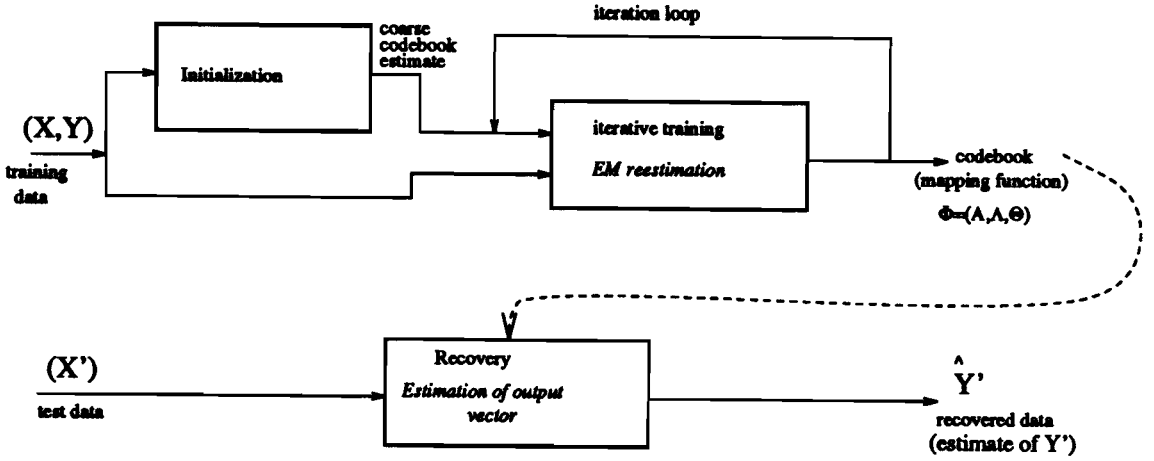

Figure 2.1: Block diagram of the Statistical Signal Mapping method. At the training stage, the mapping function parameters, $(A, \Lambda, \Theta) \equiv \Phi$, are estimated from corresponding vector training sequences, (X, Y). In the testing stage, new vectors, X' are input to the recovery block which uses the mapping function to estimate the corresponding vectors, $\hat{Y}'$, from the other space.

2.3 Testing: Estimation of the output vectors

The mapping function $f(\cdot)$ is used to translate vectors from one space to another. The previous section has outlined the method by which the parameters of the mapping function, $y = f(x, A, \Lambda, \Theta)$ can be found using a training sequence of corresponding vectors from each space. Once these are found, a new sequence of input vectors, $X' = \{x'_t\}_{t=1}^{T'}$, can be mapped to unknown output vectors $Y' = \{y'_t\}_{t=1}^{T'}$, as shown in Figure 2.1. The conditional pdf of the output vectors Y' given the input vectors X' is

$$p(Y'|X') = \frac{p(Y', X')}{p(X')} = \frac{\prod_{t=1}^{T'} p(x'_t, y'_t)}{\prod_{t=1}^{T'} p(x'_t)} = \prod_{t=1}^{T'} p(y'_t|x'_t) \tag{2.22}$$

where we have assumed that input and output vectors are independent in different time frames. In terms of the known distributions we can write (using eq. 2.5)

$$\begin{aligned} p(y'_t|x'_t) &= \frac{p(y'_t, x'_t)}{p(x'_t)} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} p(x'_t, y'_t, \lambda_i, \theta_j)}{p(x'_t)} \\ &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} p(y'_t|\theta_j)\alpha_{ij}p(x'_t|\lambda_i)p(\lambda_i)}{p(x'_t)} \end{aligned} \tag{2.23}$$

where $p(x'_t) = \sum_{i=1}^{N} p(x_t|\lambda_i)p(\lambda_i)$. Given an input vector x'_t, we can estimate the probability of any output vector, y'_t. Thus using any optimization criterion with these probabilities, we can obtain an estimate of the output vector in the optimization sense.

We know from statistical communication theory that the estimation technique which minimizes the probability of misclassification is the maximum-a-posteriori probability (MAP) estimator. The ultimate application here is pattern classification (speech recognition) but the MAP estimator would be:

$$\hat{y'_t} = \arg\max_{y_t} p(y_t|x'_t) \quad , \tag{2.24}$$

for which there is no closed form solution. Two alternatives have been applied here. Minimum mean square error estimation and a method we coin MAP-S, which is the MAP estimator applied to the sources.

2.3.1 Minimum Mean Square Error (MMSE) Estimation of the output vector

Let $\hat{Y'} = \{\hat{y'_t}\}_{t=1}^{T'}$ denote the output vectors estimated by using the mapping function function with X' as input. The MMSE estimate $\hat{y'_t}$ is given by the expected value over the a-posteriori probability,

$$\hat{y'_t} = \int_{y_t} y_t dy_t p(y_t|x'_t) \tag{2.25}$$

substituting (2.23) in (2.25) we have

$$\begin{aligned} \hat{y'_t} &= \tfrac{1}{p(x'_t)} \textstyle\int_{y_t} y_t dy_t \sum_{i=1}^{N} \sum_{j=1}^{M} p(y_t|\theta_j)\alpha_{ij}p(x'_t|\lambda_i)p(\lambda_i) \\ &= \tfrac{1}{p(x'_t)} \textstyle\sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_{ij}p(x'_t|\lambda_i)p(\lambda_i) \int_{y_t} y_t dy_t p(y_t|\theta_j) \\ &= \tfrac{1}{p(x'_t)} \textstyle\sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_{ij}p(x'_t|\lambda_i)p(\lambda_i)\mu_{\theta_j} \end{aligned} \tag{2.26}$$

where $p(x'_t) = \sum_{i=1}^{N} p(x'_t|\lambda_i)p(\lambda_i)$. It is interesting to note here that the source θ_j does not necessarily have to be Gaussian. Appearance of the quantity μ_{θ_j} is merely a consequence of taking the mean value from the source θ_j. Equation (2.26) can be intuitively explained as follows: the probability of activating source θ_j given the input vector is

$$
\begin{aligned}
p(\theta_j|x_t') &= \frac{p(x_t'|\theta_j)p(\theta_j)}{p(x_t')} \\
&= \sum_{i=1}^{N} \frac{p(x_t'|\lambda_i)p(\lambda_i|\theta_j)p(\theta_j)}{p(x_t')} \\
&= \frac{1}{p(x_t')} \sum_{i=1}^{N} \alpha_{ij} p(x_t'|\lambda_i)p(\lambda_i)
\end{aligned}
\tag{2.27}
$$

since $\alpha_{ij} = p(\lambda_i|\theta_j)p(\lambda_i)/p(\theta_j)$. Now, eq. (2.26) can be rewritten as

$$
\hat{y_t'} = \sum_{j=1}^{M} \mu_{\theta_j} p(\theta_j|x_t') \tag{2.28}
$$

where it is now evident that $\hat{y_t'}$ is effectively a weighted average of all the source means, where each source mean is weighted by the posterior probability of the source.

To speed up the recovery process, equation 2.26 is simplified to a single summation. We define $\mu_{y|\lambda_i} = \sum_{j=1}^{M} \alpha_{ij}\mu_{\theta_j}$, and can write

$$
\hat{y_t'} = \frac{1}{p(x_t')} \sum_{i=1}^{N} p(x_t'|\lambda_i)p(\lambda_i)\mu_{y|\lambda_i} \quad . \tag{2.29}
$$

In this way, $\mu_{y|\lambda_i}$ can be precomputed for all i, speeding up recovery which must be performed in real time.

2.3.2 MAP Estimation on the target sources: MAP-S

Rather than take an average of all source means weighted by their posterior probability, we could define the output vector as that mean whose source has the maximum a-posteriori probability:

$$
\hat{y_t'} = \mu_{\theta_{j^*}} \quad \text{where} \quad j^* = \arg\max_j p(\theta_j|x_t') \quad . \tag{2.30}
$$

This yields a much closer approximation to the MAP estimator. Since we only require the maximum value of $p(\theta_j|x_t)$, simplifications in computation can be made here also.

These methods are applied in the following chapter in recovery of clean speech vectors from noisy observations. The following section provides a computational analysis of the method and in the final section a preliminary study of the method is made using synthetic data.

2.4 Computational Aspects

The reestimation process of eqs. (2.16) - (2.18) involves calculating the basic quantity $p(x_t, y_t, \lambda_i, \theta_j)$ for each t, i, and j. By the nature of the calculation, this quantity can be very small. Since all the reestimation formulas involve ratios of this quantity, for each vector pair (x_t, y_t) we have scaled it by a constant to avoid underflows in the computation:

$$p(x_t, y_t, \lambda_i, \theta_j) \rightarrow \frac{p(x_t, y_t, \lambda_i, \theta_j)}{\max_i p(x_t|\lambda_i) \max_j p(y_t|\theta_j)} \quad . \tag{2.31}$$

Careful examination of the reestimation formulas reveals that they remain unchanged even though the scaling is a function of t.

A similar scaling is performed at the testing stage. Eq. (2.26) involves the division $p(x'_t|\lambda_i)/\sum_{i=1}^{N} p(x_t|\lambda_i)p(\lambda_i)$. To avoid underflows the quantity $p(x'_t|\lambda_i)$ is also scaled:

$$p(x'_t|\lambda_i) \rightarrow \frac{p(x'_t|\lambda_i)}{\max_i p(x'_t|\lambda_i)} \quad . \tag{2.32}$$

There are other ways in which the limitations of the computer come into play. The quantity $p(x_t, y_t, \lambda_i, \theta_j) = p(y_t|\lambda_j)\alpha_{ij}p(x_t|\lambda_i)p(\lambda_i)$ represents the joint probability of observing the vector pair (x_t, y_t) and the sources (λ_i, θ_j). Since the sources are distributed all over the clean and noisy vector spaces, the resulting joint probability is most often very low. The dynamic range of this quantity (after the above scaling) has been monitored and we have observed values ranging from about $O(10^0)$ to $O(10^{-30})$. The reestimation formulas all involve $p(x_t, y_t, \lambda_i, \theta_j)$ within sums over one or both sets of sources: $\sum_{i=1}^{N}$ and/or $\sum_{j=1}^{M}$. Addition of numbers with such a dynamic range is clearly inefficient.

Performing the sums over the top K values can alter a summation significantly if all values are roughly equal. Instead, by limiting $A = \{\alpha_{ij}\}$ to a smaller dynamic range, the dynamic range of $p(x_t, y_t, \lambda_i, \theta_j)$ is also reduced. We currently use the range $O(1)$ to $O(0.01/NMT)$ so that even the sum $\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}$ remains largely unaffected in precision. This has brought over 75% savings in computation with no reduction in performance. Further reductions are also possible through processing and reduction of the database, but these have not been investigated.

At the training stage the amount of computations required can be calculated as

follows. Where N and M are the number of Gaussian sources in the model, and T is the amount of training data, reestimation of $\{\alpha_{ij}\}$ and $\{p(\lambda_i)\}$ require $O(T \times N \times M)$ additions each so that, for all $i = 1 \ldots N$ and $j = 1 \ldots M$, computations increase as $O(T \times N^2 \times M^2)$. For the noisy (or clean) source means and covariance matrices the order of operations required is $O(T \times N^2 \times M)$ (or $O(T \times N \times M^2)$). If the number of iterations of EM reestimation is I, then the number of operations grows as $O(I \times T \times N^2 \times M^2)$. Since these are all vector operations, we must take into account the dimensions of the noisy and clean vector spaces, $K_{\mathcal{X}}$ and $K_{\mathcal{Y}}$ respectively. When diagonal covariance matrices are used

$$N.of.operations \propto I \times T \times N^2 \times M^2 \times (K_{\mathcal{X}} + K_{\mathcal{Y}})) \tag{2.33}$$

If a full covariance matrix is used, the power of $(K_{\mathcal{X}} + K_{\mathcal{Y}})$ is raised to 3.

It is much more critical to examine the computations required at the testing stage. For estimation of a single output vector, MMSE estimation requires $O(N \times (K_{\mathcal{X}} + K_{\mathcal{Y}}))$ operations, as per eq. 2.29 for diagonal covariance matrices. Used for speech enhancement, this is quite a reasonable amount that can be performed in real time.

2.5 Application of the method to Synthetic Data

In order to test the training algorithm and understand its behaviour, a set of artificial data with known statistical structure can be created. Two sets of data, from a different set of sources and in a different space will be created. From the set of corresponding observations in the two spaces, the training algorithm will attempt to discover the underlying statistical structure in each data set and their relationship to each other. This will serve as a guideline by which we move on to the real application of finding the underlying statistical structural relationship between clean and noisy realizations of speech in Chapter 3.

Creation of the Synthetic Data

First, N Gaussian sources, $\lambda_i, i = 1..N$, with arbitrary mean vectors, μ_{λ_i}, and covariance matrices $\Sigma_{\lambda_i}$are created, and each source is also assigned a prior probability, $p(\lambda_i)$. M Gaussian sources, $\theta_j, j = 1..M$ with arbitrary means μ_{θ_j}, and covariance matrices Σ_{θ_j}, are also created. Using these sources a training set of input vectors

$X = \{x_t\}_{t=1}^{T}$ is generated. At each time instant, a source $\lambda_{\hat{i}}$ is chosen from the $p(\lambda_i)$ distribution. The pdf for a realization of x_t is then the standard Gaussian $\mathcal{N}(\mu_{\lambda_{\hat{i}}}, \Sigma_{\lambda_{\hat{i}}})$ as in eq. (2.1).

A correlation matrix $\alpha_{ij} = p(\theta_j|\lambda_i)$, has also been created with arbitrary values, with the constraint $\sum_{j=1}^{M} \alpha_{ij} = 1$. After the $\lambda_{\hat{i}}$ input vector source is chosen, an output vector source $\theta_{\hat{j}}$ is chosen from the distribution $p(\theta_j|\lambda_{\hat{i}})$. This defines the Gaussian pdf $\mathcal{N}(\mu_{\theta_{\hat{j}}}, \Sigma_{\theta_{\hat{j}}})$ for the realization of y_t as in eq. (2.2).

This procedure is repeated at every time instant, for $t = 1 \ldots T = 5000$, so that a long sequence of vector pairs is created. To emphasize that the spaces can be different, the dimension of the input vectors, x_t is set to $K_X = 2$ and the dimension of the output vectors, y_t, is set to $K_Y = 3$.

The covariance matrices Σ_{λ_i} (and Σ_{θ_j}) used in the generation process, as well as those modelled in the training algorithm are diagonal, enabling each element of x_t (and y_t) to be generated independently. Table 2.1 summarizes the values used in the experiment.

Initialization of the Training Algorithm for Synthetic Data

In the training algorithm, the simplest initialization method has been used to give an initial estimate of the mapping parameter. The input source distribution is initially flat, i.e., $p(\lambda_i) = \frac{1}{N}, 1 \leq i \leq N$, as is the correlation matrix, i.e., $\alpha_{ij} = p(\theta_j|\lambda_i) = \frac{1}{M}, 1 \leq j \leq M, 1 \leq i \leq N$. The vector means, μ_{λ_i} (or μ_{θ_j}), are initialized using the standard VQ algorithm initialization method (section 3.4.3) in which N (or M) cluster means are found to minimize the distortion between the vector tokens x_t (or y_t) and their respective cluster means.

The initial estimates for the covariance matrices are the same for all sources, λ_i (or θ_j) and is equal to the global variance of the vectors x_t (or y_t), $1 \leq t \leq T$.

Results

Figure 2.2 shows the increasing log likelihood at each iteration in the training algorithm. After about 15 iterations the statistical structure of input and output vectors is discovered with very good accuracy. Table 2.1 summarizes the values obtained for each of the parameters in the mapping function.

The estimates for the means of the Gaussian sources achieve about 99% accuracy, and the other statistical parameters, α_{ij} and $p(\lambda_i)$, are accurate to about 95%. It is

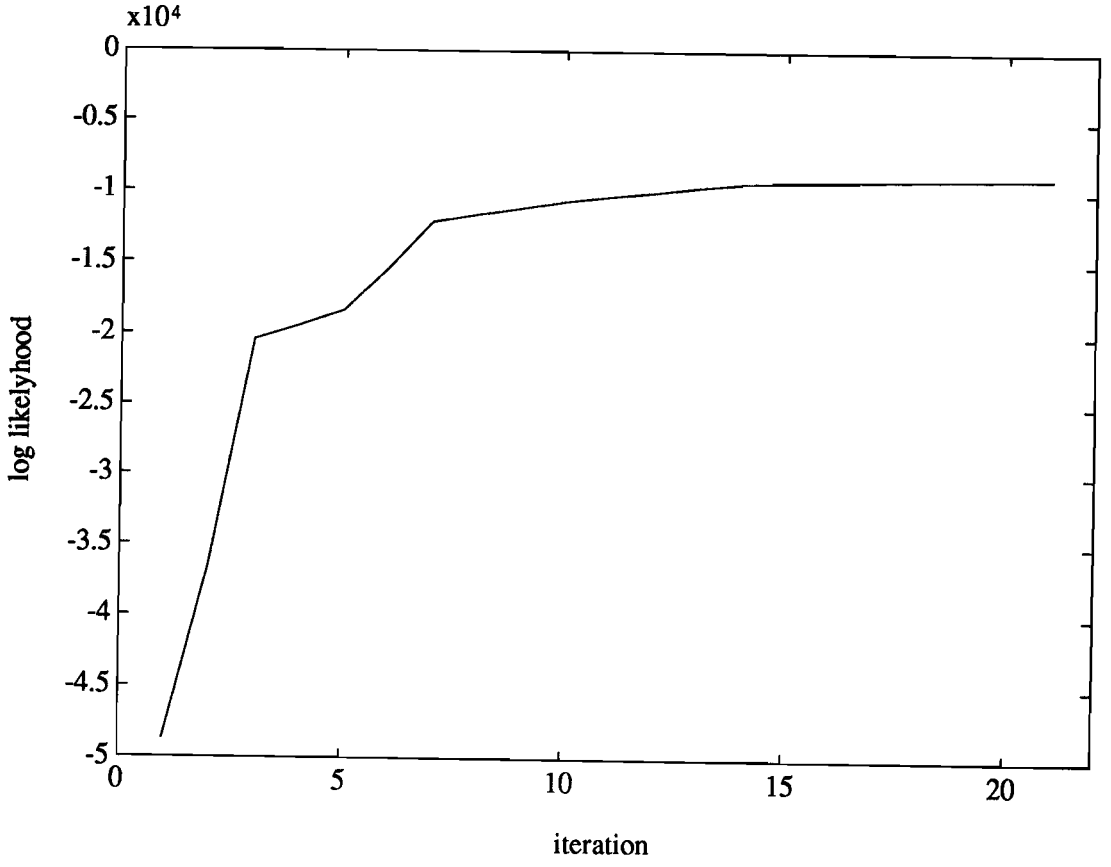

Figure 2.2: Log likelihood of the joint sequence versus iteration number. At each iteration, an increasing likelihood indicates better modelling of the training data.

Parameters	Actual Values	15^{th} iteration estimates	Accuracy
$\mu_{\lambda_1}, \sigma_{\lambda_1}$	$[-2 \quad -2], [.2 \quad .25]$	$[-1.98 \quad -1.96], [0.202 \quad 0.241]$	
$\mu_{\lambda_2}, \sigma_{\lambda_2}$	$[-2 \quad -1], [.25 \quad .25]$	$[-2.03 \quad -1.15], [0.245 \quad 0.281]$	98.5%
$\mu_{\lambda_3}, \sigma_{\lambda_3}$	$[-1 \quad -1], [.25 \quad .20]$	$[-1.01 \quad -1.00], [0.257 \quad 0.197]$	
$\mu_{\lambda_4}, \sigma_{\lambda_4}$	$[-1 \quad -2], [.20 \quad .33]$	$[-1.00 \quad -2.01], [0.199 \quad 0.329]$	
$\mu_{\theta_1}, \sigma_{\theta_1}$	$[5 \quad 1 \quad 3], [.20 \quad .25 \quad .20]$	$[5.00 \quad 1.00 \quad 2.99], [0.195 \quad 0.265 \quad 0.197]$	
$\mu_{\theta_2}, \sigma_{\theta_2}$	$[2 \quad 1 \quad 2], [.20 \quad .20 \quad .20]$	$[2.00 \quad 1.00 \quad 2.00], [0.195 \quad 0.201 \quad 0.207]$	98.9%
$\mu_{\theta_3}, \sigma_{\theta_3}$	$[1 \quad 0 \quad 1], [.25 \quad .25 \quad .25]$	$[1.00 \quad 0.01 \quad 1.00], [0.245 \quad 0.253 \quad 0.251]$	
	$0.5 \quad 0.3 \quad 0.2$	$0.518 \quad 0.315 \quad 0.168$	
α_{ij}	$0.5 \quad 0.2 \quad 0.3$	$0.503 \quad 0.168 \quad 0.328$	94.0%
	$0.1 \quad 0.1 \quad 0.8$	$0.109 \quad 0.103 \quad 0.788$	
	$0.1 \quad 0.8 \quad 0.1$	$0.098 \quad 0.806 \quad 0.096$	
$p_(\lambda_i)$	$[0.2 \quad 0.2 \quad 0.3 \quad 0.3]$	$[0.189 \quad 0.208 \quad 0.299 \quad 0.304]$	97.0%

Table 2.1: Estimation of parameters of the statistical structure for synthetic data using the Statistical Mapping method. Note estimation accuracy in the right column.

interesting to note that although the log likelihood of the joint sequences increases at each iteration, the accuracy of some of the statistical parameters does not necessarily increase monotonically. There is a general trend towards increasing accuracy but some reestimations provide less accurate parameters. Parameter accuracy is evidently sensitive to the termination point. Nevertheless, the results clearly show the effectiveness of the training algorithm.

2.6 Summary

The method of Statistical Mapping has been introduced. It addresses the problem of identifying a non-linear function which relates vectors in one space $\mathcal{X}$ with vectors from another space $\mathcal{Y}$. We assume the non-linear function can be modelled by a statistical structure able to map the vectors from one space to the other. The estimation of source statistics for each space is formulated into a parameter estimation problem. An efficient method for estimating the parameters of the mapping function from a long sequence of training data of corresponding vector pairs was developed using the EM algorithm. Given an initial estimate of the parameters, the iterative training tends to reach a critical point at which locally optimal estimates of the parameters are found. The behaviour of the training algorithm has been investigated using a set of artificial data with known statistical structure. Using this method, the parameters of this structure have been effectively estimated. This is what we have called the training stage.

In the testing stage, the mapping function is defined and using the optimization criterion new vectors from one space can be mapped in order to find the corresponding vectors from the other space.

There are many possible applications for this method. The following chapters investigate its use for robust speech recognition in noise. Vectors from corresponding clean and noisy speech are used to train the mapping function in order to find a underlying statistical structural relationship between clean and noisy speech. When new noisy vectors are presented to the speech recognizer, the clean vectors corresponding to these can be estimated by mapping. By the use of the mapping pre-processor, it

is hoped improvements can be brought to the recognition of noisy speech when the recognizer has already been trained on clean speech.

Chapter 3

Application to speech enhancement for recognition

3.1 Introduction

The previous chapter has presented the development of the method of statistical mapping. The statistical structure of computer generated data was effectively discovered using the method. This chapter focuses on the use of the method on real speech. As described in the introduction, the objective is to improve the recognition of noisy speech, when a recognizer has already been trained on clean speech.

In the majority of speech recognition systems, while the recognition rate for clean speech may be satisfactory, the rate decreases dramatically when the input speech is noisy. This is due to differing noise levels between the training and testing stages of the recognizer. Because the training stage of the recognizer is very long and requires large amounts of speech, it is inefficient or often impossible to retrain the recognizer under the new noisy conditions.

The proposed method estimates clean speech vectors by mapping noisy speech observations. Thus, the two spaces we try to correlate with the mapping function are the space of clean speech observations and that of noisy speech observations. A mapping function which maps noisy speech to clean speech is trained using a long sequence of corresponding observations from each space. When new noisy speech is presented to a speech recognizer, its corresponding clean speech can be estimated

using the mapping function.

The advantage over retraining is that this method requires much less speech than that needed to retrain the recognizer. The additional computations that this scheme requires at recognition time are minimal compared to the recognition computations and are of a similar nature to that of a recognizer so that they can be performed on the same specialized computer if necessary.

This method can work effectively in any observation space that contains distinguishable features. LPC-coefficients have been used in [8] as a domain in which to map low bandwidth speech to high frequencies. Filterbank energies with a similar mapping scheme have been used for noisy speech recognition in [12]. Since our application is a front-end speech enhancer for a speech recognition system, and we wish to have a compatible optimality criterion, we have chosen the same space as that of the recognizer: cepstral observations. Descriptions of the generation of the cepstral observation vectors are given in the following section. Section 3.3 overviews the recognition tests and other performance measures. Implementation issues at each stage (initialization, training, and recovery & testing) are presented in section 3.4.

3.2 Speech Database: Generation of features

The speech used in the experiments is the same as those used to train the recognition system: the novel "White Fang" read by a male speaker. The sampling rate for the speech was 16000 samples/s, and each speech frame was 30 ms long. The frame advance time was 10 ms, so there are typically 10-20 observations per phoneme. Speech frames are Fourier transformed and passed through a Bark-spaced[1] bank of 25 filters. The filter-bank log energies are computed, and a cosine transform of the filter-bank log energies is taken to generate the cepstral coefficients. The feature vectors for the clean observation space are

$$x_t = [\ c_{1,t}\ \ c_{2,t}\ \ \ldots\ \ c_{7,t}\ \ d_{0,t}\ \ d_{1,t}\ \ \ldots\ \ d_{7,t}\ ] \tag{3.1}$$

where $c_{n,t}$ is the n^{th} cepstral coefficient at time t, and

$$d_{n,t} = c_{n,t+2} - c_{n,t-2} \tag{3.2}$$

[1]The Bark scale, detailed in §4.4, weights low frequencies more than high, as in human hearing.

are delta cepstral coefficients.

3.2.1 Noisy speech

Noisy speech observations were generated in the same way as the clean observation vectors, but from noisy speech. The noisy speech was created by adding computer-generated white Gaussian noise to the clean speech. Noise energy (E_n) was set by calculating the clean signal energy (E_s) over approximately 6 minutes of speech, and using the conversion formula $E_n = E_s \times 10^{-SNR_{dB}/10}$. The signal level within each 6 minute file does vary somewhat, but experiments across different signal-to-noise ratios (SNR) have also been conducted.

Typically, the mapping function was trained using 2-3 minutes of clean speech and corresponding noisy speech. The effect of the amount of training data was investigated in section 3.4.4. Testing was performed on 6 minutes of speech.

Use of white Gaussian noise as a corrupting factor is arbitrary. By its nature, the method could be applied to any type of stationary noise, even of multiplicative type.

3.3 Performance Measures and Testing Platform

After training is completed and a suitable mapping function is found, new noisy vectors X' are presented to the mapping function which estimates the corresponding clean vectors, $\hat{Y}'$. In the following sections two tests are presented as a means to monitor the performance of the mapping function. The first is a phonemic recognition test, in which we compare the recognition rate for recovered noisy speech, $\hat{Y}' = f(X')$, with that of the unprocessed noisy speech X'. The second is a distortion measure by which we examine the gains in distortion brought about by the same processing.

3.3.1 The "INRS TOY" recognition system

The recognition system used in the experiments is a continuous speech phonetic recognizer with 38 phonemes including silence. The training and testing files are phonetically segmented and the segmentation is automatic, but fixed for our test purposes. The recognizer uses HMM models for each phoneme, each model having five states

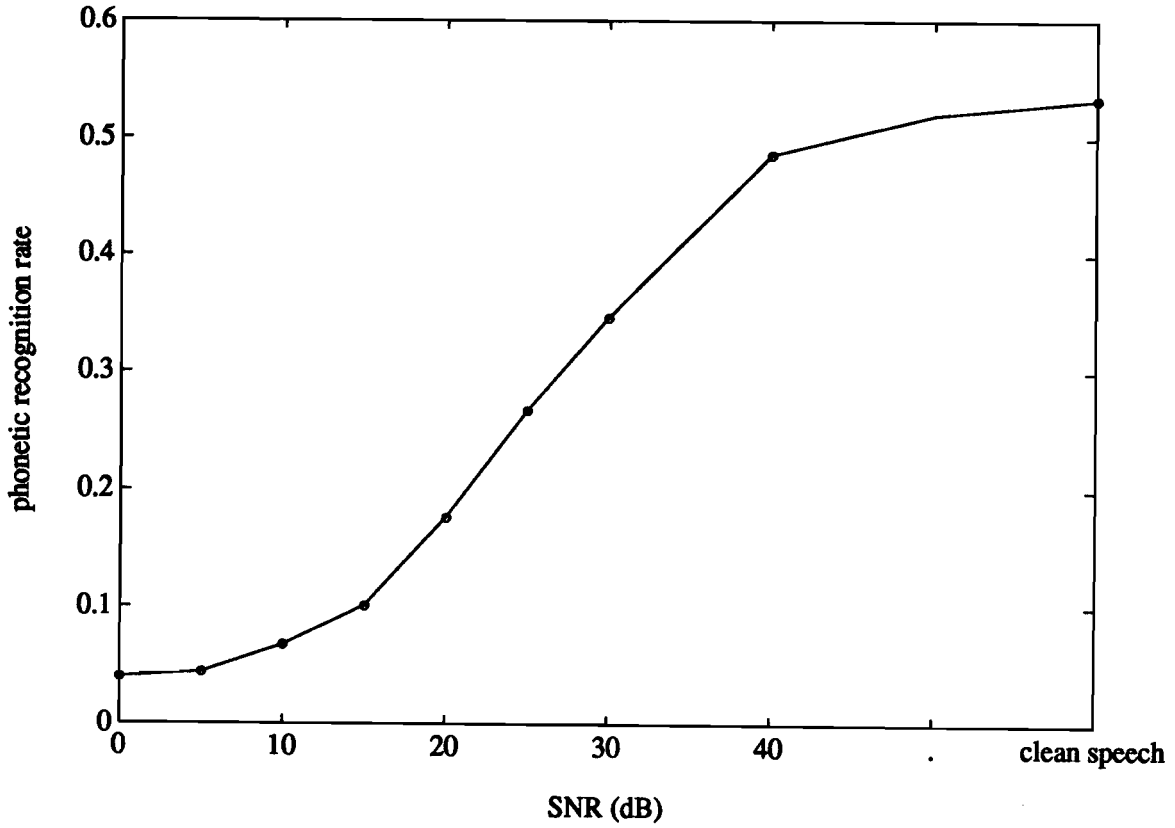

Figure 3.1: Phonetic recognition rate as a function of the SNR of the input speech, for a recognizer trained with clean speech.

and being limited to 11 transitions. Continuous density Gaussian models are used to model the observations within each state. For simplicity and ease of computation, diagonal covariance matrices are used. The search method used is the block Viterbi search as presented in [3] with block length 200 frames and block advance 100 frames. Training is performed on about 60 minutes of speech. Baseline recognition rates for this system are as follows: For a recognizer trained and tested on clean speech the recognition rate is 53%, but tested on noisy (10 dB SNR) speech it is 6.8%. If the recognizer is both trained and tested on the noisy speech, the performance is 35%. Baseline recognition rates at other noise levels are presented in Figure 3.1. Using the proposed method, performance of the recognizer trained on clean-speech and tested on noisy speech comes close to that of the recognizer trained on noisy speech.

3.3.2 The Variance Weighted Cepstral Distortion Measure

The sequence of estimated output vectors, $\hat{Y}'$, can also be compared directly to the actual vectors Y' to examine the effectiveness of the recovery. The variance weighted cepstral distortion measure[23] has been widely used for this purpose. Since we used observation vectors composed of 7 cepstral and 8 delta cepstral coefficients (c_0 excluded), we use the distortion measure,

$$d(u,v) = (u-v)^{\#}\Sigma^{-1}(u-v) = \sum_{n=1}^{7}(c_{u,n}-c_{v,n})^2/\Sigma_{nn} + \sum_{n=0}^{7}(d_{u,n}-d_{v,n})^2/\Sigma_{nn} \quad (3.3)$$

where $\Sigma = \{\Sigma_{nm}\}$ is the diagonal covariance matrix for the clean vectors over the entire training set. For two corresponding sets of data, $U = \{u_t\}_{t=1}^{T}$ and $V = \{v_t\}_{t=1}^{T}$, the distortion $d(U,V)$ is an average over time:

$$d(U,V) = \frac{1}{T}\sum_{t=1}^{T} d(u_t, v_t) \quad . \quad (3.4)$$

In this way, the distortion $d(\hat{Y}', Y')$, between the estimated output vectors and the actual clean speech, can be compared to the distortion $d(X', Y')$, between the unprocessed noisy speech and the actual clean speech. Gains in distortion can be translated to gains in SNR using Fig. 3.2 which plots distortion between clean and noisy speech at different noise levels.

3.4 Implementation Issues

Experiments using the statistical mapping method with iterative EM reestimation have been performed. We have examined the factors associated with the method at each stage of the system. The choice of the number of sources to model the clean and noisy speech vectors: N and M, are presented in the next section. Despite the fact that most recognition systems accept a fixed number of features, the number of features to be mapped can also be tuned as described further on.

At the initialization stage we have experimented with four different techniques leading to different performances and revealing a large sensitivity of the mapping function to its initialization.

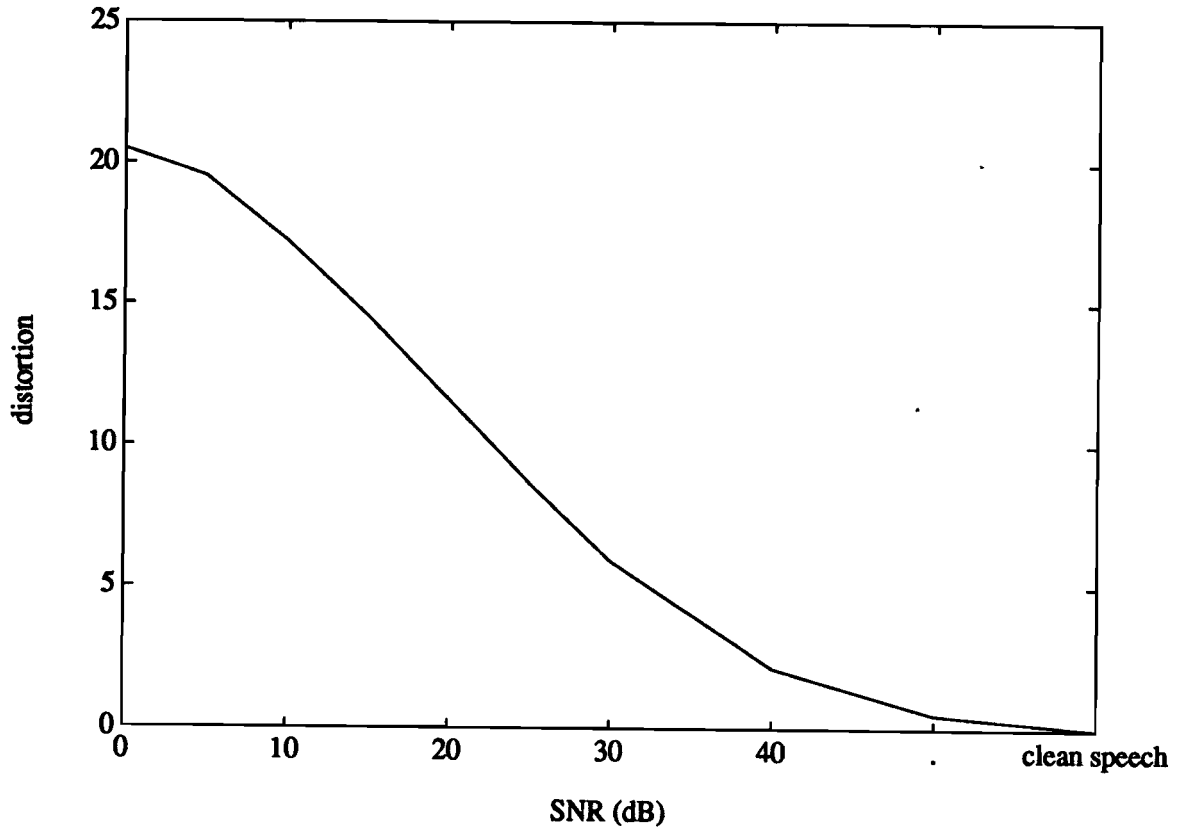

Figure 3.2: Distortion between clean speech and noisy speech at different SNRs, as measured by the variance weighted cepstral distortion measure.

In the iterative training stage the important variables are training size and number of iterations, described in section 3.4.4. At this stage the log likelihood of the joint sequence of training observations $\log p(X, Y)$ can be monitored as a function of each of the variables mentioned. The higher the likelihood, the better the model, $\Phi = (A, \Lambda, \Theta)$, for the training set (X, Y).

At the recovery stage, once the statistical model accounting for (X, Y) is found, the mapping function is applied to new noisy speech, X', corresponding to the clean speech, Y'. The output of the mapping function is an estimate of Y', and is labeled $\hat{Y}' = f(X')$. The type of estimation (MMSE or other) used at the recovery (section 2.3.1) is the final variable that can be tuned.

After the sequence of output vectors has been estimated, recognition tests (we use the "INRS TOY" recognizer, section 3.3.1), can be executed. The recognition rate of the processed noisy speech, $\hat{Y}' = f(X')$, is compared with the rates for unprocessed noisy speech, X', at different noise levels. Figure 3.3.1 in section 3.3.1 plots the recognition rate of the "INRS TOY" recognizer against the signal-to-noise ratio (SNR) of the input speech. The resulting savings in distortion have also been monitored using the variance weighted cepstral distance measure.

3.4.1 Number of Sources

The number of sources in the statistical model is among the most important variables in the model. N Gaussian sources model the noisy speech and M Gaussian sources model the clean speech. It is obvious that a larger number of sources increases the model accuracy, but computational costs at the training stage are $O(N^2M^2)$, and at the recovery stage $O(N)$. Figures 3.3 and 3.4 show the effect of varying N at constant M, and of varying M at constant N. We use $T = 12000$ training tokens or 2 minutes of speech, and perform 10 training iterations after conventional VQ initializations. The fact that there is very little difference in two plots (dashed and dotted) is an indication that the numbers of sources in the clean and noisy spaces should be increased together for best performance, as in the solid plot ($N = M$). In Fig. 3.4 we can also observe earlier asymptotic behaviour when only one of the spaces is assigned more sources. Experiments have been conducted where the product $N \times M$ is held constant - holding computational load and memory requirements fixed.

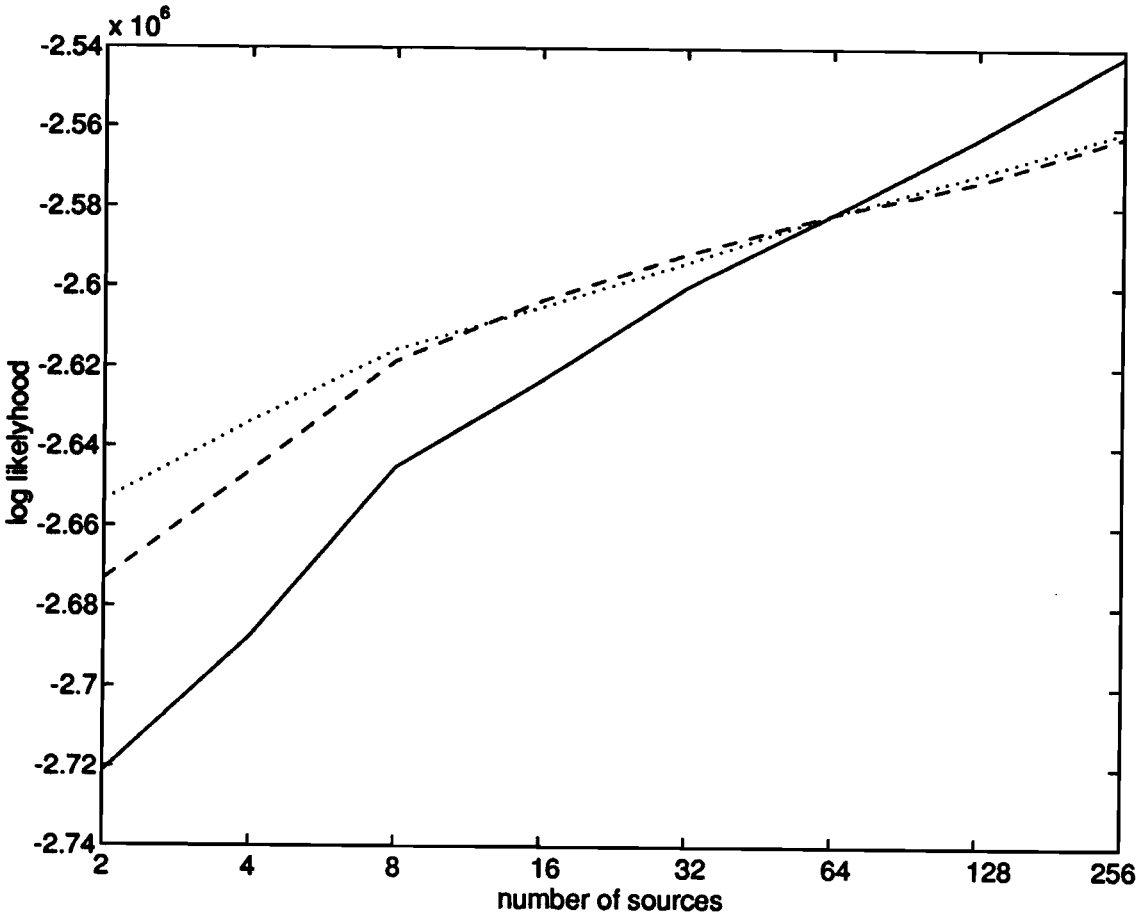

Figure 3.3: Log likelihood at termination for increasing number of sources, M at $N = 64$ (dotted), N at $M = 64$ (dashed), and $N = M$ (solid). N and M are respectively the number of noisy and clean sources.

The best results were observed when $N = M$, as in Fig. 3.4 where $N = M = 128$ outperforms $N = 64, M = 256$ and $N = 256, M = 64$. In the remainder of the experiments of this report we have in fact chosen $N = M$.

Recognition rates of the recovered clean speech have also been measured, and observed to increase with very little asymptotic behaviour up to $N = M = 256$ sources, as shown in Fig. 3.5. We have not performed simulations beyond this point, since the computational load is too heavy.

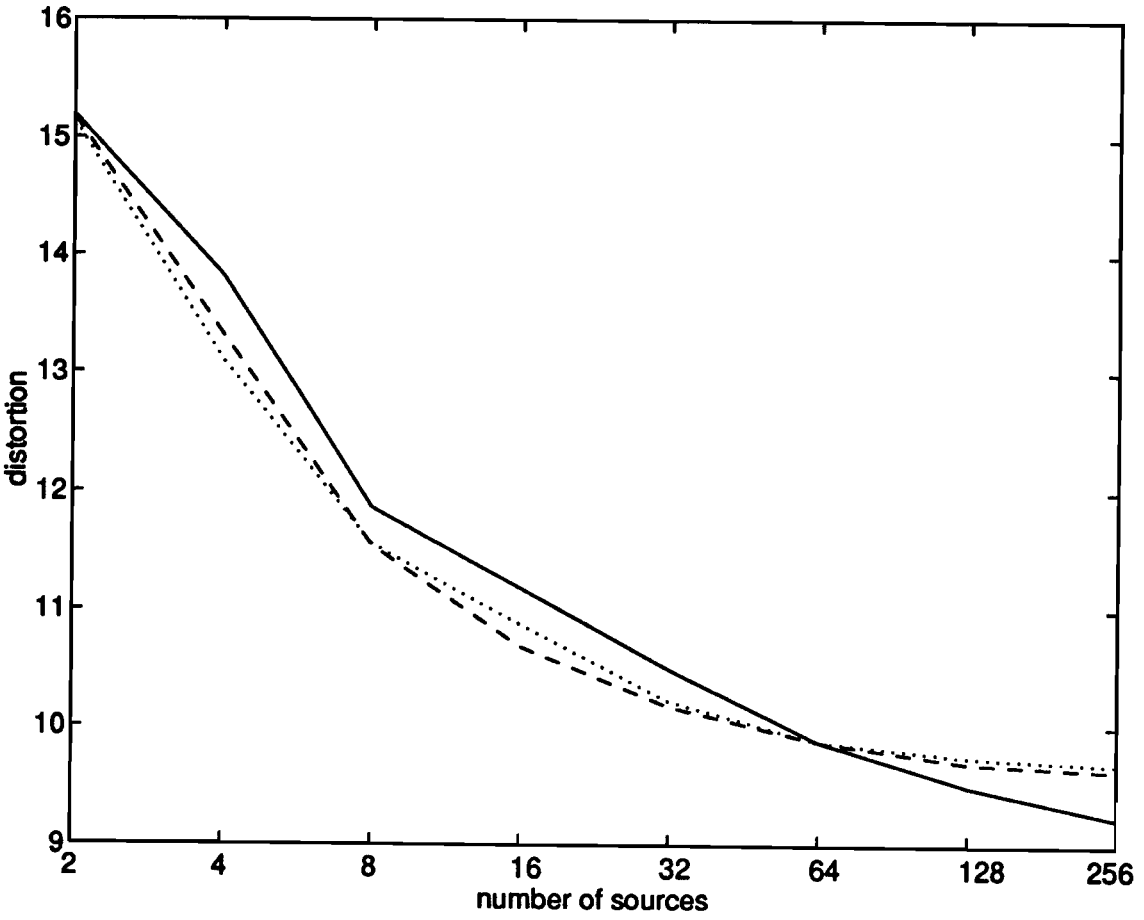

Figure 3.4: Distortion at recovery for increasing number of sources, M at $N = 64$ (dotted), N at $M = 64$ (dashed), and $N = M$ (solid). N and M are respectively the number of noisy and clean sources.

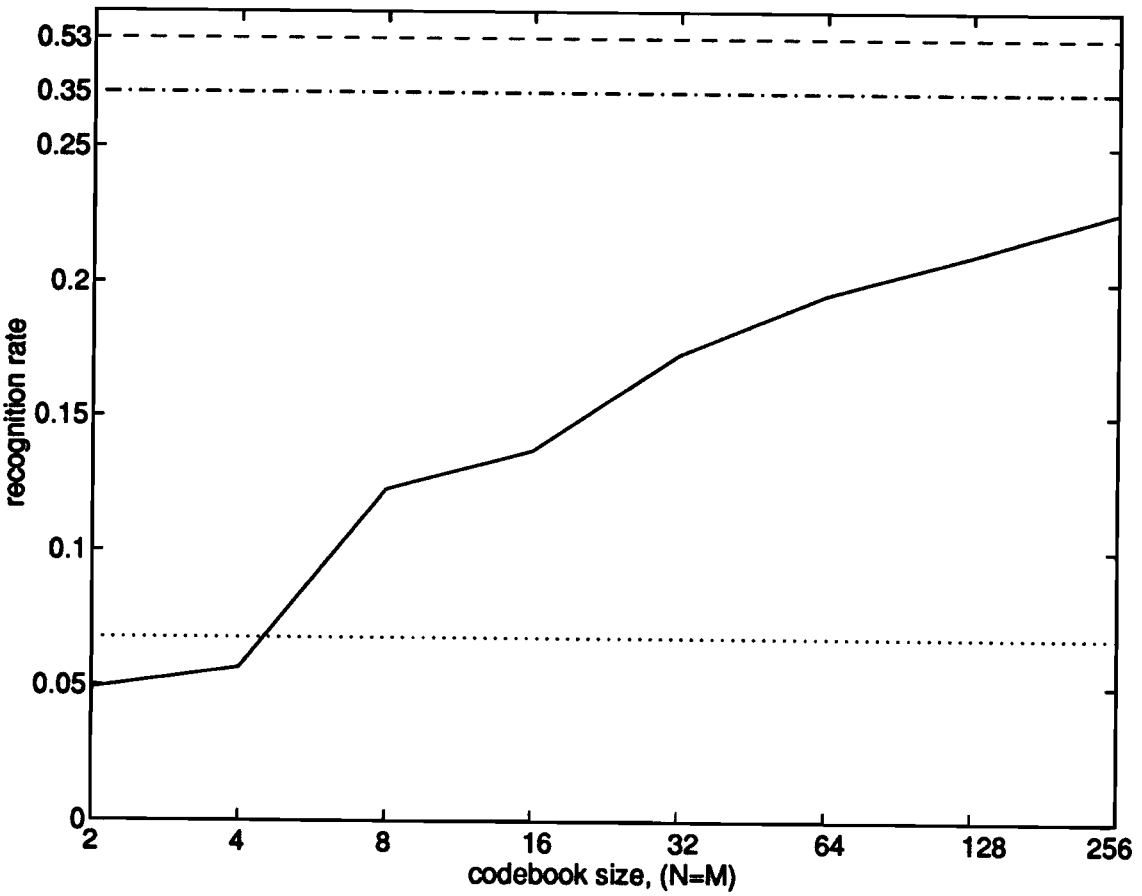

Figure 3.5: Recognition rate of processed noisy (10 dB SNR) speech recovered using the mapping function for increasing number of noisy speech (N) and clean speech (M) model sources, at $N = M$. The baseline rate for 10 dB SNR noisy speech (dotted), clean speech (dashed) and noisy speech on a noisy speech trained recognizer (dot-dashed) are also shown.

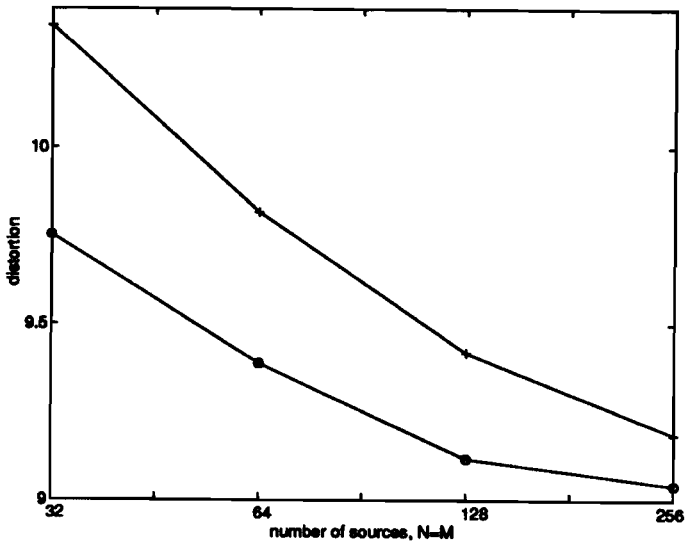

Figure 3.6: Distortion reductions at different numbers of sources when 8 (o) rather than 15 (+) features are modeled by the mapping function.

3.4.2 Number of Features

The features of the "INRS TOY" recognizer are 7 cepstral coefficients ($c_1 - c_7$) and 8 delta cepstral coefficients ($d_0 - d_7$), determined entirely from the cepstrum. The mapping function can be trained on the full 15-dimensional feature vectors or alternatively, on the first 8 features ($c_1 - c_7, d_0$) alone. In the latter case, the missing features are derived after recovery is completed, using $d_{n,t} = c_{n,t+2} - c_{n,t-2}$ for $n = 1 \ldots 7$.

Figures 3.6 and 3.7 compare resulting distortion and recognition rates for the two methods. Derivation of delta cepstral coefficients after the mapping is clearly the better choice, over the whole range of number of sources. Distortion is reduced by up to 6%, pushing the recognition rate up 4 %.

The results are not surprising. For noisy speech, the cepstral coefficients are much less accurate than for clean speech, giving rise to delta cepstral coefficients with double the uncertainty, and containing very little information. Since the mapping function attempts to model all the information it is given, much of the statistical capacity of the mapping is wasted on ($d_1 - d_7$). Although derivation of ($d_1 - d_7$) after mapping may produce equally uncertain values, the parameters ($c_1 - c_7, d_0$) are estimated more accurately since a larger amount of statistics models them. Further reduction

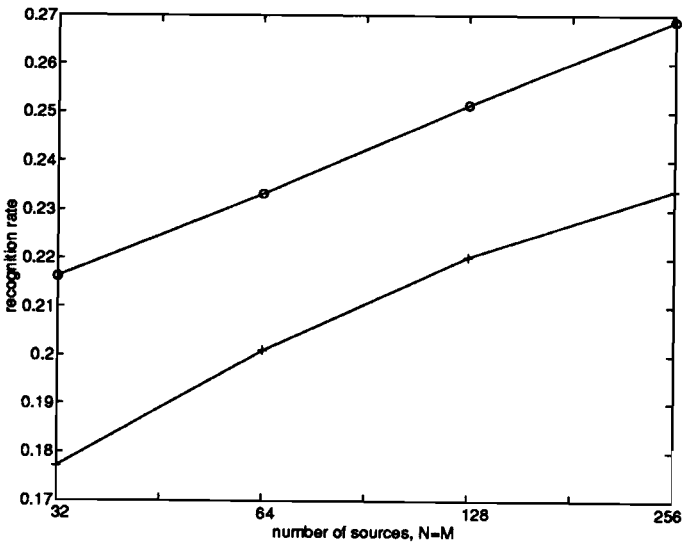

Figure 3.7: Improvements in recognition rate when 8 (o) rather than 15 (+) features are modeled by the mapping function.

feature vector dimension may even be more beneficial. The following three sections examine the effect of other free variables at the initialization, training and recovery stages of the system.

3.4.3 Initialization Techniques

The solution to the mapping function estimation problem from the previous chapter involved first estimating an initial set of parameters for the mapping, followed by iterative reestimation from this starting point. This section outlines the different initialization methods for the training algorithm. It is important to note that these methods are applied only in the training algorithm, not in the recovery or mapping of noisy vectors to clean. Thus computational complexity in one method or another is not critical, since all the training is performed on "presupplied" data.

The EM iterative algorithm guarantees convergence only to a locally optimal solution, so good initialization is essential to finding a global optimum. For this reason, it is instructive to examine different initialization methods. The sensitivity of the algorithm to different initializations as far as performance has also been examined.

We found that the most sensitive parameters are the source means and covari-

ance matrices. For the source correlation matrix and the a-priori source probabilities we have chosen uniform distributions, $p(\theta_j|\lambda_i) = 1/M$, and $p(\lambda_i) = 1/N$, for $i = 1 \ldots N$, $j = 1 \ldots M$, as in section 2.5.

For the source means and covariance matrices, the simple bootstrap initialization, vector quantization (VQ), modified VQ, and a joint VQ initialization have been examined and are presented in the following sections.

Bootstrap initialization

The bootstrap (BS) initialization technique is the simplest technique of all, and is only tested as a reference with which other initializations will be compared. For the sources $\lambda_i, 1 \leq i \leq N$, N vectors are chosen at random from the training set $x_t, t = 1..T$, to become the source means: μ_{λ_i}. The source covariance matrices Σ_{λ_i} are diagonal, and are initially set all equal to the covariance of the entire training set: $E(x_t^{\#} x_t) - E(x_t^T)E(x_t)$. The same procedure is used for the sources $\theta_j, 1 \leq j \leq M$.

A preliminary study of this simple initialization scheme has been made using the same synthetic data as in section 2.5. The results are not expanded here but it suffices to say that the bootstrap method is very ineffective as compared to the standard VQ detailed in the next section. Since the choice of means is random, there is high chance these choices are closer to a local maximum likelihood rather than a global one. For example, if two means are chosen that are closest to the same source mean, then both the means will merge towards modelling the same source. This inevitably leaves other regions of the space unrepresented, and the mapping function would not map these regions. Such an examination is very difficult when real speech data is used, but there is nothing to indicate that this type of behaviour cannot occur.

Figure 3.9 compares the increasing log likelihood of the iterative training algorithm after BS initialization with that after other initializations, described in the following sections.

Standard VQ initialization using the LBG method

The LBG VQ method was first introduced in [17] and [4] as a method to quantize LPC vectors with minimum distortion in order to transmit them in a speech coder. It has since been used extensively outside speech coding, as a method to "optimally"

represent a large set of data vectors by a small number of symbols. Its application here is to divide each set of cepstral vector samples, noisy and clean, into N and M clusters respectively. The noisy (or clean) clusters are represented by centroids $\mu_{\lambda_i}, 1 \leq i \leq N$ (or $\mu_{\theta_j}, 1 \leq j \leq M$), computed as the mean of all vectors within the cluster.

Because the training algorithm is sensitive to the initialization procedure, the standard VQ procedure is outlined in the following steps to differentiate it from the extended VQ procedure of the next section.

It is necessary to first define the distortion function to be used as a measure of dissimilarity between a sample training vector x_t and the centroid vectors μ_{λ_i}. We currently use the variance weighted Euclidean distance since it closely resembles the metric used in many speech recognizers:

$$d(x_t, \mu_{\lambda_i}) = (x_t - \mu_{\lambda_i})^{\#} \Sigma^{-1} (x_t - \mu_{\lambda_i}) \tag{3.5}$$

where Σ is the diagonal covariance matrix of the entire clean training set. The VQ algorithm attempts to find centroids $\mu_{\lambda_i}, 1 \leq i \leq N$ such that the overall distortion

$$\sum_{t=1}^{T} \min_i d(x_t, \mu_{\lambda_i}) \tag{3.6}$$

between the x_t and the centroid of the cluster it belongs to is minimized.
The flow of the algorithm is given below:

1. *The centroid μ of the entire training data is computed.*

2. *Each centroid (initially only one) is split in two by perturbing its components: The centroid μ splits into $\mu(1+\epsilon)$ and $\mu(1-\epsilon)$, where ϵ is a small but arbitrary factor ≈ 0.02.*

3. *The training data are reclustered around the closest new centroid, using the variance weighted Euclidean distance (eq. 3.5).*

4. *The new centroid of each cluster of data is determined. Steps 3 and 4 have the effect of redistributing the data such that the total distortion is reduced. These*

steps are repeated until the reduction falls below a certain threshold; we use 1%.

5. Go back to step 2 until the desired codebook size is reached.

At the end of the procedure, $\{\mu_{\lambda_i}\}_{i=1}^{N}$ are known and are used in the initialization of the iterative training algorithm. The source covariance matrices $\{\Sigma_{\lambda_i}\}_{i=1}^{N}$ are then computed as the variance of all $x_t \in \lambda_i$ around the centroids μ_{λ_i}. The process is repeated for the clean training vectors, $\{y_t\}_{t=1}^{T}$, to produce M centroids $\{\mu_{\theta_j}\}_{j=1}^{M}$, and covariance matrices $\{\Sigma_{\theta_j}\}_{j=1}^{M}$. Comparison of this method with the BS initialization is shown in Figure 3.9. The log likelihood of the joint sequence, $\log p(X, Y)$, is much higher for VQ than for BS, which implies better modelling of the training data. The following section introduces a third initialization technique, which extends VQ with small modifications.

Variations of the LBG method

The fact that VQ initialization outperforms the BS type is one indication of the sensitivity of the training algorithm to its initialization. Two modifications have been brought to the VQ algorithm which further improve the modelling capability of the mapping function.

Centroid Splitting

In the original VQ algorithm, a centroid μ, was split by defining two new points: $\mu(1+\epsilon)$ and $\mu(1-\epsilon)$. This method has proven to be satisfactory but it is not difficult to show that for very symmetric data successive splitting in this way will create parallel cylindrical shaped regions. This can be visualized by the following example.

Let $x_t = (x_{1,t}, x_{2,t}), 1 \leq t \leq T$ be the 2-dimensional vector training set such that $x_{1,t}$ is uniformly distributed in [0,16] and $x_{2,t}$ uniform between [-10,10]. The first centroid will be placed at (8,0) and when split the two clusters means will be (4,0) and (12,0) for centroids. Successive splitting will create more centroids on the $x_{2,t} = 0$ axis, defining parallel cylinders of clusters, which very inefficiently represent the data points. This is due to the same "splitting direction" taken at each iteration of VQ.

Codebook Size	File I	File II	File III
128	5.847	5.634	5.744
128*	5.837	5.623	5.711
256	5.027	4.856	4.941
256*	5.006	4.827	4.930

Table 3.1: Comparison of VQ distortions with (*) and without random direction splitting of centroids.

Of course, exactly symmetric data is not common, and it is likely the centroids will drift off the splitting axis. Nevertheless, this phenomenon is what has motivated an alternative centroid splitting method.

Three techniques were tried. In the first, successive splitting of centroids is always performed in a direction perpendicular to the previous ones. The second technique has tried to split the centroid in the direction away from the overall average of the sample vectors. Since the two methods produced about equal results, a third and simpler technique in which splitting direction is random was tried and found to be not worse than the other techniques. Table 3.1 summarizes the results obtained for codebooks of size 128 and 256. The improvements from this method are small but they are consistent over several different data sets.

Cluster Splitting [14]

Another variation on standard VQ has been made with respect to cluster reorganization after a full space partion has been made. It has been observed that in the VQ initialization algorithm that the size of the clusters can vary greatly. Specifically, the size of the largest cluster can be as many as four times larger than the smallest one. In terms of distortion, the larger cluster will usually contribute a larger distortion than the smaller one. However, it seems intuitive that to minimize overall distortion it would be preferable for each cluster to contribute roughly equal distortions. For this reason the VQ algorithm has been extended in the following way.

Once the training sequence has been separated into N clusters, the cluster with the largest overall distortion is split into two. The sum of distortions of each new

cluster is invariably lower than that of the original one, but we now have $N+1$ clusters, or one too many. This is resolved by merging the two nearest clusters in the new set of $N+1$. Let s be the index of the cluster which is split into s' and s'', and let the two clusters m' and m'' denote those that have merged into cluster m. The reduction in distortion from splitting is:

$$\Delta_{split} = \sum_{x_t \in C_s} d(x_t, \mu_s) - \sum_{x_t \in C_{s'}} d(x_t, \mu_{s'}) - \sum_{x_t \in C_{s''}} d(x_t, \mu_{s''}) \tag{3.7}$$

and the increase in distortion from merging is:

$$\Delta_{merge} = \sum_{x_t \in C_m} d(x_t, \mu_s) - \sum_{x_t \in C_{m'}} d(x_t, \mu_{m'}) - \sum_{x_t \in C_{m''}} d(x_t, \mu_{m''}) \ . \tag{3.8}$$

The procedure represents a savings in distortion if $\Delta_{split} > \Delta_{merge}$, which is generally the case, since the worst case scenario would be to merge back those clusters which have just been generated by splitting. The procedure can be repeated any number of times but its benefit reduces with repetitions. This has been verified experimentally, results of which are shown in Figure 3.8. The total VQ distortion after conventional VQ into 256 clusters is shown at left (iteration 0). Successive iterations of the "split-and-merge" procedure reduce the total distortion significantly. The total reduction after about 15 iterations is about 3%, reaching the distortion of a codebook of size 324 when this procedure is not applied.

Although both techniques improve the partitioning of the vector space, their combined effect has not presented further improvements, which was expected since both techniques have the common aim of reducing the incidence of large clusters. In the remainder of this thesis we shall refer to this initialization technique as VQ^*.

Figure 3.9 demonstrates the effect of each of the initialization techniques (BS, VQ, VQ^*, and joint VQ) on the training procedure. The log likelihood of the joint sequence, $\log p(X,Y)$, differs greatly for each initialization scheme which is an indication that a specialized initialization may be necessary. The effects each of the initialization methods has at the recovery and testing stage are shown in Table 3.2.

Joint Initialization in two vector spaces

Up to now, the same initialization methods were used for both the clean and noisy training sets, but these were performed independently of each other. In this section we

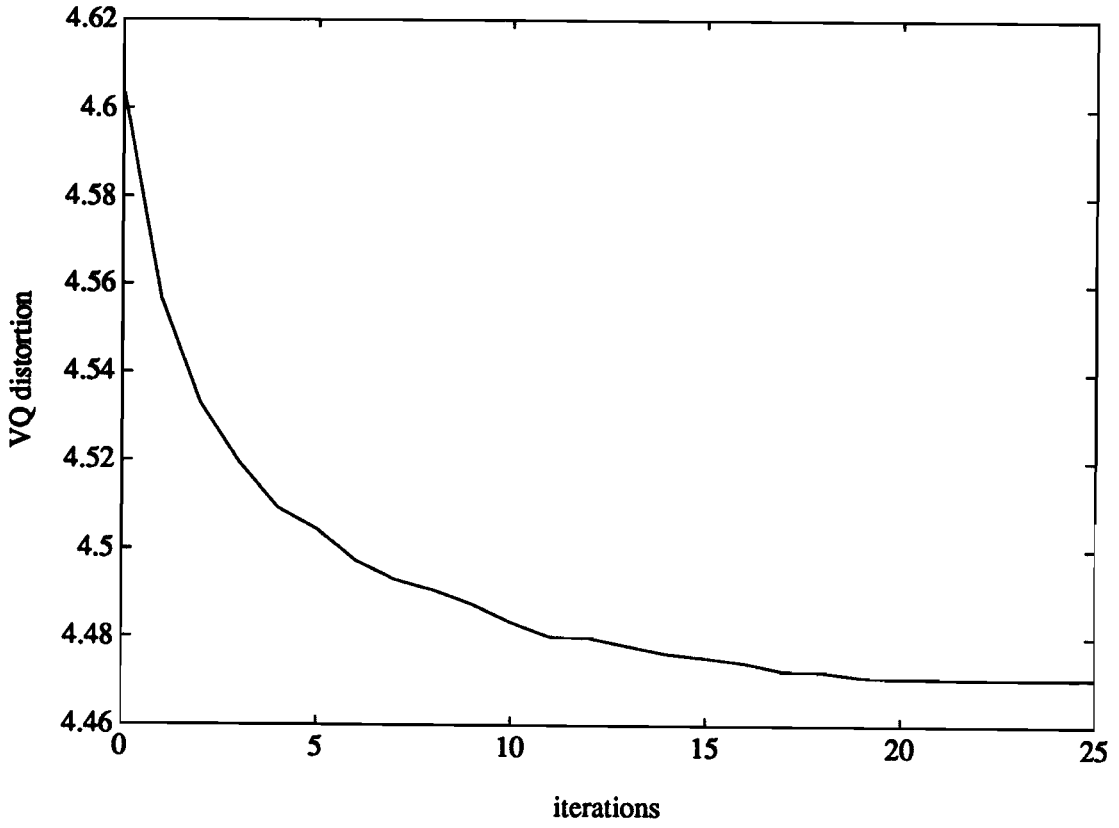

Figure 3.8: Total VQ distortion as reduced by successive iterations of the "split-and-merge" technique.

Initialization Type	Log Likelihood	distortion	recognition rate
BS	-2.585060e+06	9.96	19.0
VQ	-2.581678e+06	9.87	19.5
VQ^*	-2.580600e+06	9.82	20.1
JVQ	-2.581418e+06	9.79	21.2

Table 3.2: Summary of recovery results for different initialization schemes. Bootstrap (BS) initialization performs worst, followed by VQ, modified VQ (VQ^*) and joint VQ. In general a higher likelihood at termination of the training algorithm leads to lower distortion and higher recognition rates for the recovered speech.

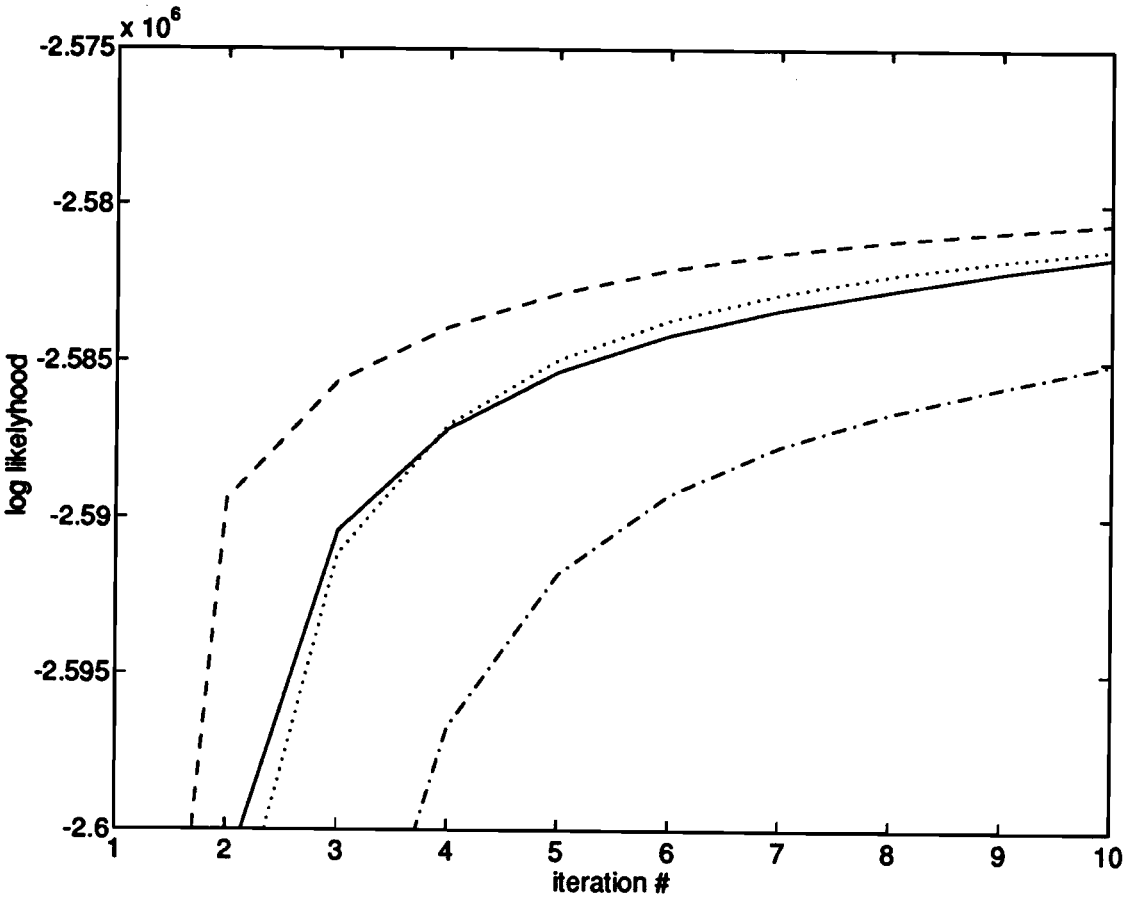

Figure 3.9: Initial and increasing log likelihoods for the four different initialization schemes: bootstrap (dot-dashed), vector quantization (solid), modified VQ (dashed), and joint VQ (dotted). (Number of sources are $N = M = 64$, 15 features, 12000 training tokens.)

examine the possibility of using information regarding the correspondence of vectors between one space and another.

Observation vectors in the noisy space are very distorted. In recognition tests we have observed many errors whose nature suggests that similar sounding phonemes get lumped into the same region of the acoustic space by standard VQ initialization. The problem seems to originate from the addition of noise and not from phoneme similarity because the phonemes remain in distinct acoustic regions in the clean observation space. Since clean versions of the noisy observations are also in the training set, we have attempted to use the clean vector observations, $\{x_t\}_{t=1}^{T}$, to partition both the clean and noisy vector spaces, or $\mathcal{Y}$ and $\mathcal{X}$ respectively. In other words, the index set of clean vector tokens in region i:

$$I(X, \mu_{\lambda_i}) = \{t : i = \arg\min_{i'} d(x_t, \mu_{\lambda_{i'}})\} \tag{3.9}$$

is used to calculate the noisy means and variances as well:

$$\mu_{\theta_j} = \frac{1}{||I(X, \mu_{\lambda_j})||} \sum_{I(X, \mu_{\lambda_j})} y_t \quad , \tag{3.10}$$

where $||\cdot||$ indicates cardinality. The noisy covariance matrices Σ_{θ_j} are similarly defined with $(y_t - \mu_{\theta_j})^{\#}(y_t - \mu_{\theta_j})$ in the summation. Of course, this can only be done when the same number of sources model the clean and noisy vector spaces ($N = M$), but experimental results have shown this choice to be optimal.

For clean speech the partioning of the observation space is not changed but the partioning of the noisy observation space changes significantly. The overall VQ distortion for the noisy observation space using this technique is significantly higher than with conventional VQ, implying that many of the regions defined overlap one another. Nevertheless, we have observed reductions in distortion and improvements in recognition rate using this technique. Figure 3.9 shows the performance of this method compared with the other initializations, where we observe slightly better modelling by the joint VQ algorithm compared with regular VQ. Table 3.4.3 summarizes the results after recovery of the noisy test speech, where it is clear that joint VQ outperforms the others. Further experiments using the joint VQ initialization on top of the VQ modifications are given in section 3.4.5.

The reason for improved performance is not yet fully understood but two important effects of the joint initialization have been noted.

The effect this method has on the noisy source means is that it tends to cluster the noisy training tokens by their reliable (clean) features rather than by the noise they contain. The addition of noise has a detrimental effect on VQ in that many different parts of the acoustic space are merged toward the center where the cepstral observations represent the flat spectrum of white noise. This is reflected in the fact that VQ distortion is much lower for noisy speech than clean. When clean tokens are used to partition the noisy space, VQ distortion is increased indicating that many regions now overlap. The overlap is undesired but the source means now lie at the center of acoustic regions corresponding to distinct phones.

The effect the joint initialization has on the source correlation matrix is slight. Each source in the noisy space maps to a different degree to each source in the clean space where the rows of the $\alpha_{ij} = p(\theta_j|\lambda_i)$ matrix indicate to what degree source θ_j maps to λ_i. A diagonal correlation matrix with a single non-zero element in each row would indicate a "one-to-one" mapping. The degree to which the matrix is diagonal could be measured by the average maximum value in each row of the matrix. From this test we have observed average maxima up to 15% larger in matrices reestimated from joint initialization. This indicates that in a sense, the matrix is partially "diagonalized" by the joint initialization, rendering the mapping more "one-to-one". The reason why this improves performance is not yet fully understood, but this question is discussed further in section 3.4.5.

Conclusions

Large differences in performance due to different initialization techniques have been found, demonstrating the sensitivity of the mapping function to the initialization. The simplicity of Bootstrap initialization is evident in its poor results. VQ initialization is better but leaves room for improvement as demonstrated by the modified VQ. Joint VQ seems to perform quite well requiring half the computation of VQ. The modifications to VQ are equally applicable in the joint VQ case and experiments to this end are conducted further on.

The different results suggest that a specialized initialization technique may be

necessary to fully exploit the possibilities of the mapping function. One possibility may be use the available phonetic segmentation (from recognition training) in the initialization in order to somehow guide the mapping function. Although this may not have any effect on reducing the distortion, it may ultimately improve recognition.

It is important to note again that although these techniques may increase computational requirements in the training stage, they in no way lengthen the recovery stage, at which time recognition tests or other processing is performed. Any added computational requirements are unimportant.

3.4.4 The Iterative Algorithm

At the training stage two variables can be tuned to optimize the performance of the mapping function: training size and number of iteration of EM reestimation.

Training Size

Since all regions of the acoustic spaces must be well represented by the mapping function, a good coverage implies a larger training size, which is easily handled since computations grow only linearly with training size. Nevertheless, the non-linear mapping function does have some linear properties which could reduce the amount of training required. If a pair of sources in the noisy space maps one-to-one with a pair of sources in the clean space, then by the nature of the correlational statistical structure and MMSE estimation, it follows that any vector in between the former two sources will map to a vector in between the latter two. This would suggest that the space is also covered in between sources, and a smaller amount of training data is necessary. Fixing the number of sources ($N = M = 64$) and features ($K_{\mathcal{X}} = K_{\mathcal{Y}} = 8$), we have experimented with training sizes varying from 1 minute to 10 minutes of speech, corresponding to $T = 6000$ to $T = 60000$ training tokens for each space (clean and noisy). Figure 3.10 shows the log likelihoods of the joint sequences. Since these depend on the number of tokens, T, we plot the normalized log likelihood: $\frac{1}{T}\sum_{t=1}^{T} \log p(x_t, y_t)$.

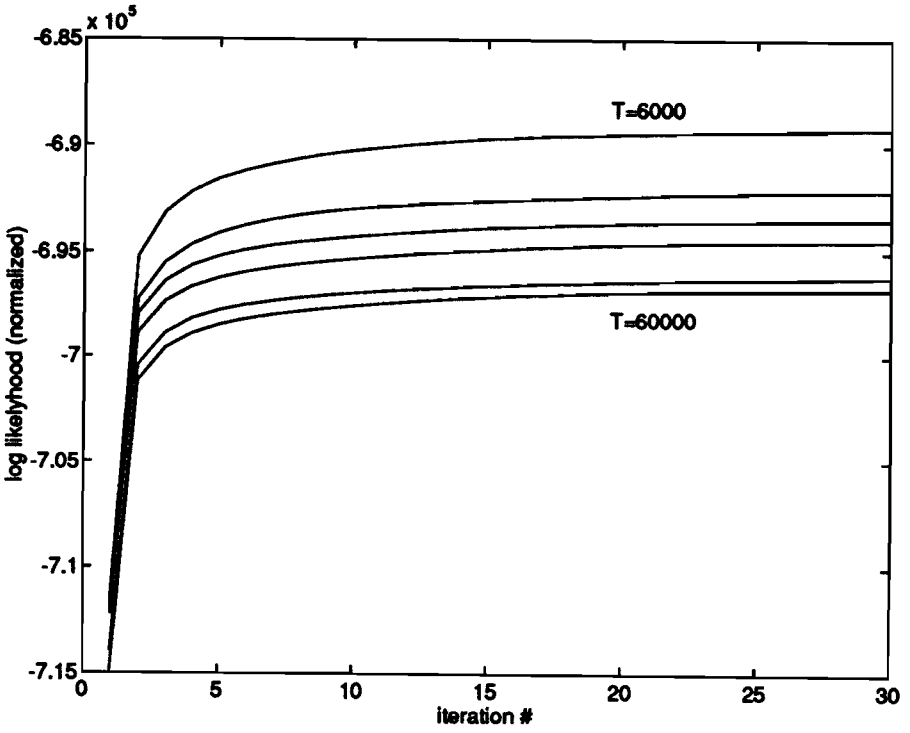

Figure 3.10: Log likelihoods at recovery when amount of training speech is increased and number of iterations is increased. Each plot corresponds to a different training size but all plots seem to reach maxima at about 15 iterations of EM reestimation.

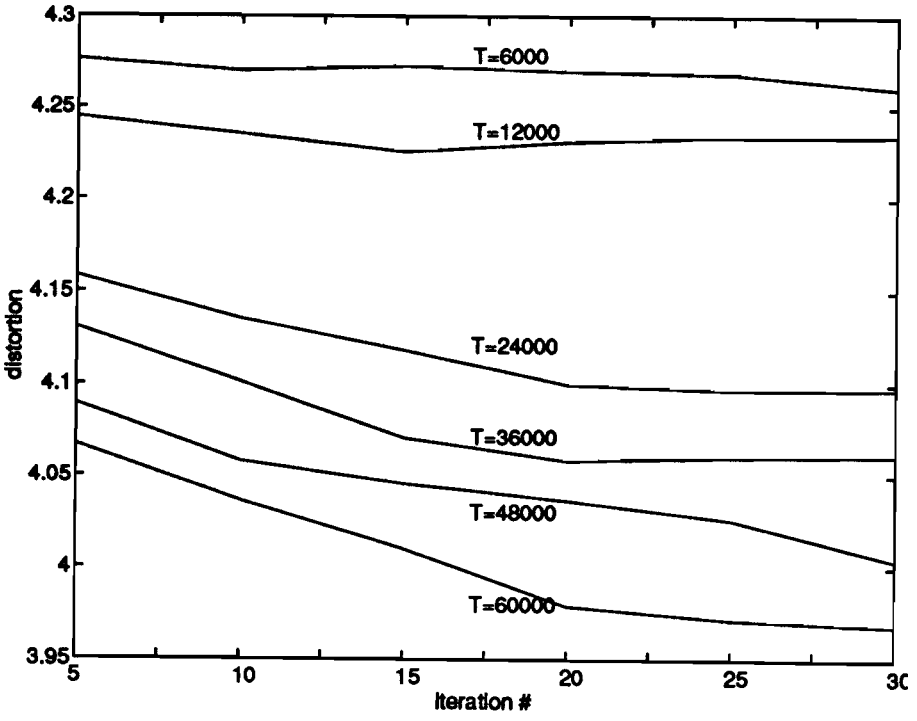

Figure 3.11: Resulting distortion at recovery amount of training speech is increased, and number of iterations is increased. For small training sets (< 24000 tokens), speech recovery is not improved with more iterations, where improvements are still possible with large (> 24000 tokens) training sizes. Note: 1 minute of speech corresponds to 6000 tokens.

Number of iterations

Figure 3.10 shows log likelihoods increasing at each iteration in the training algorithm for different sizes' training sets. After 10-15 iterations we observe asymptotic behaviour in all training sets indicating that the log likelihood of the joint probability $p(X, Y)$ has reached a maximum and the mapping function is "optimized". But, for a larger amount of training data, better performance from the mapping function can be obtained by performing more iterations, as shown in Fig. 3.11. This would suggest that a much finer test for termination of the training algorithm may be necessary.

Indeed, when using a test for asymptotic behaviour which sets a slope threshold, we cannot justify using the log likelihood more than the direct likelihood.

We have not yet discovered a suitable test, but the number of iterations necessary to achieve maximum performance, seems to depend mostly on the number of training tokens, and marginally on the number of sources and features. For the experiments conducted in this thesis, about 5 iterations of training have been used for each minute of training speech. For the amount of training, most of the noisy speech in properly recovered using 5-6 minutes of training, but better results could be obtained when up to 10 minutes of speech are used.

3.4.5 Mapping Recovery: Estimation of the Output Vector

The only degree of freedom at the recovery stage is the type of estimation used at the recovery stage. We currently use MMSE estimation but in section 2.3.1 an alternate estimator was presented in which the mean of the source with highest a-posteriori probability (MAP-S) could be chosen as output vector. Table 3.3 compares the performance of this recovery rule with MMSE estimation for fixed number of sources ($N = M = 64$), training size $T = 12000$ and 10 iterations of EM reestimation. Resulting distortion between the estimated vectors and the actual clean vectors is much higher but recognition tests result in slightly improved performance.

One of the reasons why MAP-S performs slightly better in recognition is that the MAP rule by its nature is more suited for classification tasks. MMSE estimation performs better in minimizing distortion is because distortion is effectively a weighted square error.

For each source in the noisy space, the source correlation matrix computes which

	distortion	recognition rate
MAP-S	12.57	24.3 %
MMSE	9.41	23.2 %

Table 3.3: Comparison of two estimation techniques at the recovery stage. MMSE estimation computes a weighted average of all source means, while MAP-S outputs the mean from source with highest a-posteriori probability. The number of sources used here is $N = M = 64$ with VQ^* initialization but similar results have been obtained with a varying number of sources (32,128).

sources in the clean space should be activated. Part of each quantity in this matrix is probablistic and part behaves as a coefficient in a linear transformation. Use of the MMSE estimator is well suited to the latter property of the matrix because it forms a weighted sum (or linear combination) of means from every clean source to find the output clean vector. The MAP-S estimator is more suited the the probabilistic nature of the source correlation matrix since it seeks the clean source with the highest posterior probability. In fact, presence of any linear transformation component may have an adverse effect in the use of MAP-S estimation.

One way this property of the source correlation matrix can be removed is in the initialization of the source means. When VQ is used independently in the two spaces, the means in the clean space are completely independent of those in the noisy space. This obviously leads to a mapping from the noisy source mean to clean ones which looks like a linear transformation. The joint initialization technique (§3.4.3) has the effect of removing this aspect of the A matrix.

We have performed experiments using both joint VQ^* at initialization and MAP-S estimation at recovery. For comparison, the same experiment has been performed with VQ^* initialization and/or MMSE recovery. Figures 3.12 and 3.13 show the resulting distortion and recognition rates for test speech with a varying number of sources and 8 features. Training was performed using 12000 training tokens for 10 iterations of model reestimation.

Recognition results are slightly better for MAP-S when the initialization is not joint. When initialization is joint, MMSE estimation improves (as in the 15 features'

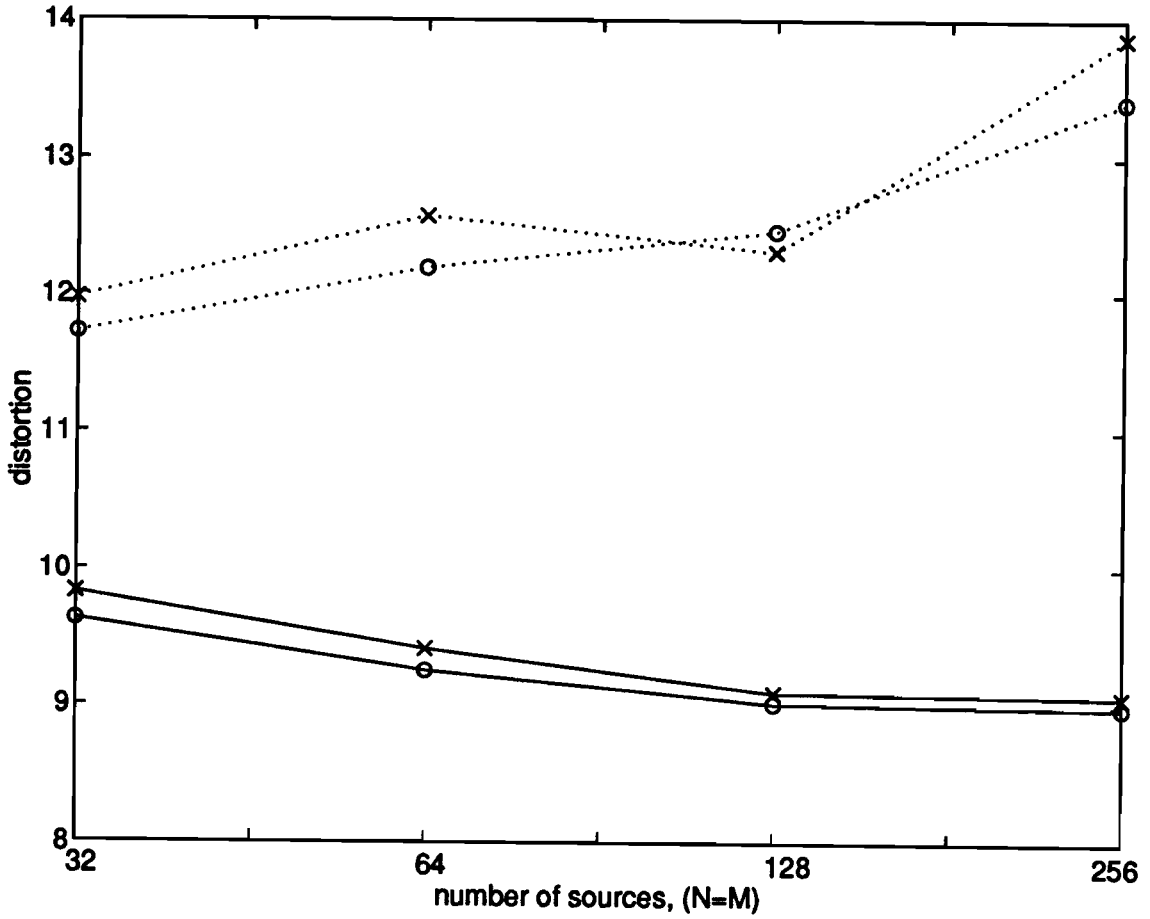

Figure 3.12: Distortion of recovered speech when different output vector estimation methods are used, for varying number of sources. MMSE estimation (solid) outputs a weighted sum of the source means with a-posteriori source probabilities for weights. MAP-S estimation (dotted) outputs the source mean with highest a-posteriori probability. For each estimation type, joint VQ^* (o) and VQ^* (x) initializations are shown.

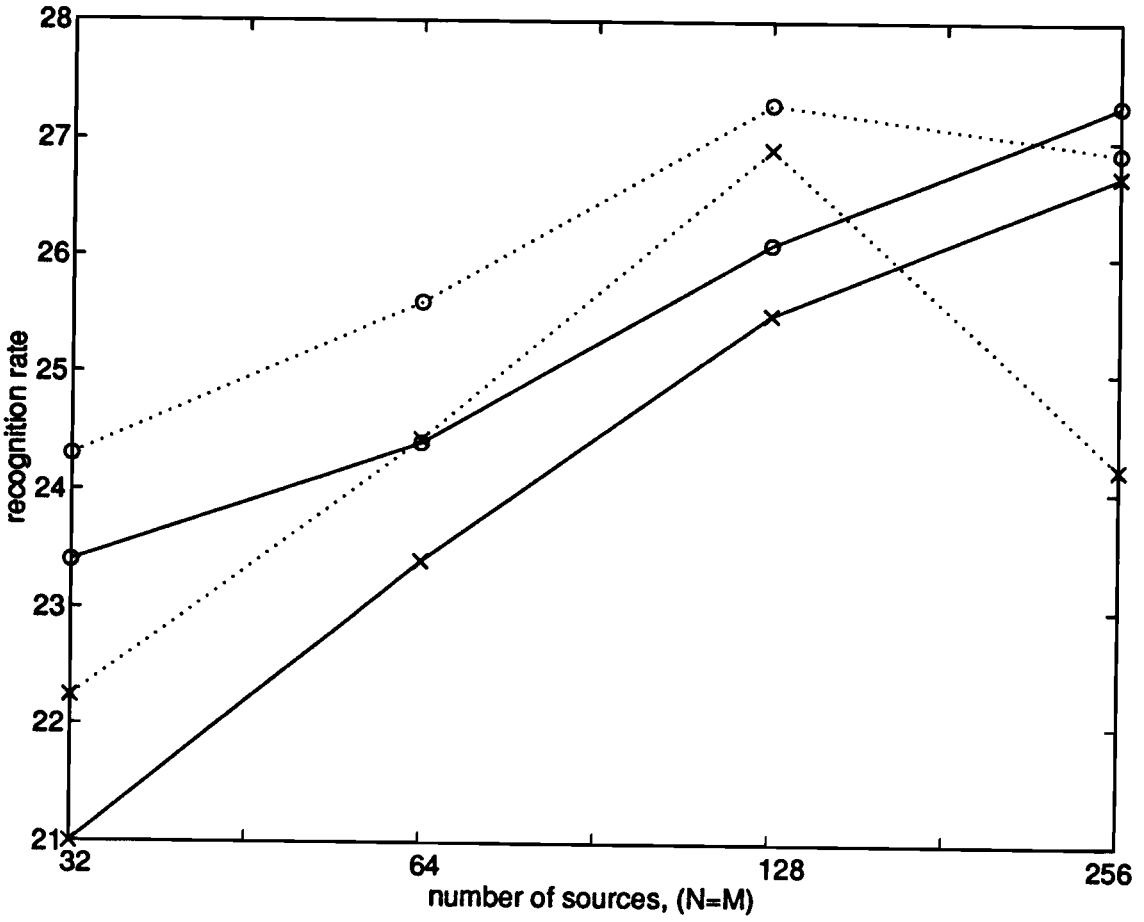

Figure 3.13: Recognition rate of recovered speech when different output vector estimation methods are used, for varying number of sources. MMSE (solid) and MAP-S (dotted) estimation are shown for joint VQ^* (o) and VQ^* (x) initializations.

case §3.4.3) and MAP-S estimation improves on that result even further. Difficulties do arise from using MAP-S when large numbers of sources are used ($N = M = 256$), requiring perhaps another approximation to MAP estimation for further gains by increasing the number of sources.

Distortion results are also as expected. In both MAP-S and MMSE estimation, joint initialization presents a small reduction in distortion compared in independent VQ, but MAP-S estimation compares much worse than MMSE estimation with both initialization types.

Joint initialization creates clusters from corresponding tokens in the noisy and clean spaces and MAP-S estimation attempts to find the one most probable source a recovery. Although the mapping function is a complicated structure within, in the end it looks like a one-to-one mapping. This may suggest that a much simpler nearest neighbour (NN) type mapping as done by Juang and Rabiner [16] may perform just as well. In that study, VQ is used to define regions in the noisy and clean cepstral space from a long training sequence. When a noisy vector is to be recovered, it's nearest neighbour (one of the regional means) is found and the corresponding mean from the clean space is outputted. This is a truly one-to-one mapping. In our study, when a noisy observation x_t is presented, $p(\lambda_i|x_t)$ is computed for each source λ_i. Then, $p(\lambda_i|x_t)$ is transformed by the source correlation matrix into: $p(\theta_j|x_t) = \sum_{i=1}^{N} p(\theta_j|\lambda_i)p(\lambda_i|x_t)$. Although the source correlation matrix $p(\theta_j|\lambda_i)$ has relatively large diagonal elements, $\arg\max_i p(\lambda_i|x_t) \neq \arg\max_j p(\theta_j|x_t)$, so the transformation is not one-to-one. We found that in more than 15 % of the transformations, λ_i leads to θ_j with $j \neq i$. The increased complexity of this method is reflected in it's improved results: whereas [16] reports an 8.5 dB effective distortion gain for 14 dB SNR speech, our methodology results a 14 dB gain at 10 dB SNR.

Other evidence in recognition tests have supported the use of MAP-S. When the recognizer ranks the probabilities of each phoneme, MAP-S estimation results in the correct phoneme appearing in the top ranks consistently more often than in MMSE estimation. We currently use a phoneme recognition system, but in word recognition lexical constraints sometimes select phonemes of lower ranks, so it is possible that relative performance with MAP-S will be even better. Further research with a more accurate recognizer may be necessary to draw proper conclusions.

The fact that a MAP oriented estimation method can be used at all and performs better than MMSE estimation is an important advantage of this mapping technique over other methods for robust speech recognition. Erell and Weintraub [12] used MMSE estimation of the filter bank log energies and Ephraim [11] used MMSE estimation of DFT coefficients. In recognition tests these criteria do not necessarily lead to improved performance. We have not seen any method which uses MAP criteria to improve noisy speech recognition.

3.5 Conclusions

The statistical signal mapping method has been successfully applied to improve recognition of speech in a noisy environment. The process has been broken down to three stages, initialization, iterative training, and recovery. The first two generate a mapping function from training data, and the last uses the function to map noisy test speech to in order to generate "clean" speech.

We have attempted to optimize each stage of the statistical mapping system within the limits of computational feasibility. In the overall system we have found that the most statistics can be gathered by using a large number sources with an equal number in the noisy and clean spaces. We have also found that statistical requirements needed for modelling larger number of features are greater, and thus for a fixed number of sources, the mapping function can model fewer features better.

We have tried many initialization procedures and have found best results with a modified VQ initialization which tends to reduce the incidence of large clusters. In addition, we found that partitioning of the noisy space using the results of vector quantization in the clean vector space performs better than when conventional VQ is used in the noisy space.

In the training stage we found that six minutes of speech are often enough to improve the recognition rate to a respectable level. Using up to 10 minutes of training speech, reductions in distortion can still be gained, pushing recognition rates even higher. The computational cost incurred is large not only due to a larger training base but because more training iterations are required.

At the recovery stage, the type of estimation to be used depends on the application.

We have found two viable alternatives. MMSE estimates the output vectors as the weighted average of the clean source means where the weights are the a-posteriori probabilities of each source. This has been found to minimize the average distortion in the recovered speech. Alternatively, MAP-S estimation selects the output vector as the source mean from that source with the highest a-posteriori probability. MAP-S estimation has been found to maximize the recognition rate the most when the joint type initialization is used.

Using the reference figures 3.1 and 3.2, when all the optimizations are used together we obtain recognition results similar to unprocessed speech of 25 dB SNR when using input speech of 10 dB SNR. Distortion is also reduced to the level of 24 dB unprocessed speech.

The following chapter extends the SSM method, attempting to use information beyond individual cepstral observations in each frame to perform the mapping.

Chapter 4

Extensions of the SSM method

4.1 Introduction

The previous chapter presented the statistical mapping function as applied to robust speech recognition in noise. The effects of each of the free variables (number of sources, training size, initialization scheme, number of features, etc ...) were examined for "optimal" performance of the mapping function. In this chapter modifications to the mapping scheme will be introduced and performance tests will be described. The modifications attempt to enhance the mapping function by using additional information, other than the cepstra of individual speech frames. The first section attempts to use contextual information at the inter-frame level of the speech vectors. In the second section we use the available phonetic segmentation information of the training speech. Finally, in the last section, information in the noisy speech not extracted by cepstral analysis is used.

4.2 Contextual Information Modelling

One of the major disadvantages of this method is that it employs frame by frame estimation. It is well known that, in the presence of noise, processing speech over a larger context is beneficial. In speaker adaptation, Huang [15] has developed a normalization neural network which maps a frame with its left and right contexts (one frame each) to generate one normalized frame. Erell and Weintraub [12] have

explored the use of HMMs for filter bank energy estimation, replacing the single-frame Bayes' rule estimation with forward-backward estimation of a frame probability.

In our study we have explored two methods of incorporating contextual information into the mapping function. The first is an extension of the usual method in that the mapping function is designed to map several frames of noisy speech (cf. a single frame) to a frame of clean speech. In the second scheme, several frames of noisy speech vectors are first normalized to a single vector which is then mapped to a clean speech frame. A full description of the methods with their results are presented in the following sections.

4.2.1 Multi-frame to single frame mapping

One way to include contextual information is to view cepstral observations of several frames of speech as a single speech vector. A noisy speech vector with its context (3, 5 or 7 frames together) are mapped to a single clean speech frame. The input noisy vectors are of higher dimension, $(3 * K_X)-$, $(5 * K_X)-$, or $(7 * K_X)-$ dimensional depending on the size of the desired context. The size of the output clean vector is the usual K_Y-dimensional.

The use of contextual information in this sense is that the mapping function is designed to model trajectories rather than single observations. Although the additive noise is stationary, its effect on speech can disturb the observation trajectory to a large extent. Only by observing several frames of noisy speech at a time can the underlying trajectory be discovered, as shown in Figure 4.1 where the phoneme transition /e/z/ on the (c_1, c_2) plane is shown for clean and noisy speech.

In the training and testing portions of this experiment, input vectors are made up of one frame's noisy speech vector with that of the W previous and W following frames, for $W = 1$ and 2. For the output vectors, the usual one clean speech frame is used. Thus, we map x'_t to y_t where

$$x'_t = x_{t-W} \ldots x_t \ldots x_{t+W} \tag{4.1}$$

in both the training and recovery algorithms. Computations at the training and testing stages increase significantly, but this increase would be marginal if a vector processor implements the algorithm.

Sources	Window size	Distortion	Recognition Rate
32^2	1	10.49	17.3
	3	11.16	16.9
	5	12.90	12.7
64^2	1	9.87	19.5
	3	10.66	19.4
	5	12.35	17.9
128^2	1	9.50	21.0
	3	10.33	21.4
	5	12.23	19.5

Table 4.1: Summary of results for trajectory mapping. Window size ($= 2W + 1$) indicates the number of noisy speech frames to be mapped to a single clean speech frame. Relative performance (sizes 3,5 *cf.* no processing size 1) improves when larger numbers of sources are used, but a full mapping of all trajectories would require too large a number of sources.

Results are summarized in Table 4.1 where it is observed that performance is not improved. This may be due to many factors. First, in the original mathematical formulation we have explicitly assumed that neighbouring frames are independent of each other. This contradicts the concept of trajectory mapping. The diagonal covariance assumption has also been stretched. We have assumed that the vector elements, or more specifically the cepstral and delta-cepstral coefficients within a speech vector, are independent. However, the covariance matrix of the *augmented* input vector containing several frames has off-diagonal elements involving the same coefficient from different frames. Setting these elements to zero values can have a significant impact on performance. Further work may be necessary to develop a mathematically sound approach to trajectory mapping.

Another reason that performance may not be as high as expected could be that a larger number of sources may be necessary to model the augmented acoustic space. While 32 to 128 sources may be enough to model 15 dimensions, it is unreasonable to assume that the same number of sources can model 45- and 75- dimensional vec-

tors. The number of vector trajectories encountered in normal speech far exceeds the number of possible vectors. Full modelling of all trajectories would require a number of sources beyond reasonable computation limits. This problem has not been solved in a computationally cost-effective way.

Frame Averaging

The second method is simpler and involves merely averaging the cepstral observations of noisy speech frames before training and testing. Instead of mapping noisy speech frames, $x_t, 1 \leq t \leq T$ directly, we use a weighted average with neighbouring frames:

$$x_t^* = \sum_{s=-W}^{+W} x_{t+s} \times w_s \tag{4.2}$$

where $w_s = w_{-s}, -W \leq s \leq W$, $\sum_{s=-W}^{W} w_s = 1$, make a weighted average of the center frame with its neighbouring frames. The resulting noisy observation vectors are much more easily modelled because they form smoother trajectories, as shown in Fig. 4.1 for the case W=1.

Experiments with $W = 1, W = 2$ and $W = 3$ have been performed with a triangular set of weights. We use $w_s = w_0 - \frac{|s|}{W+1} w_0$ where w_0 is chosen to satisfy $\sum_{s=-W}^{W} w_s = 1$. Figures 4.2 and 4.3 show the performance of this scheme as well as that of the mapping function without frame averaging. For window size 3 (or $W = 1$) and a codebook size of $N = M = 256$, sources the distortion at recovery is reduced from 9.23 to 9.06 but recognition rate improved at most by 1%.

The benefit of frame smoothing is very slight but consistent, especially when $W = 1$. This may be due to the fact that averaging reduces resolution of vectors making them easier to model. Beyond $W = 1$ resolution is reduced too much and the benefit of mapping is lost. Compared with $W = 0$ (or no window processing), fewer statistics of the mapping function of spent modelling the rugged nature of the noisy spectra. Figure 4.1 shows the actual effect of smoothing out the irregular noisy cepstra. Noisy information is dropped but actual speech based on contextual and other reliable features remains (see Fig. 4.1) and is subsequently better modelled by the mapping function.

It is important to note that in the original derivation of the training algorithm (chapter 2), contextual information was not modelled but rather a simple frame based

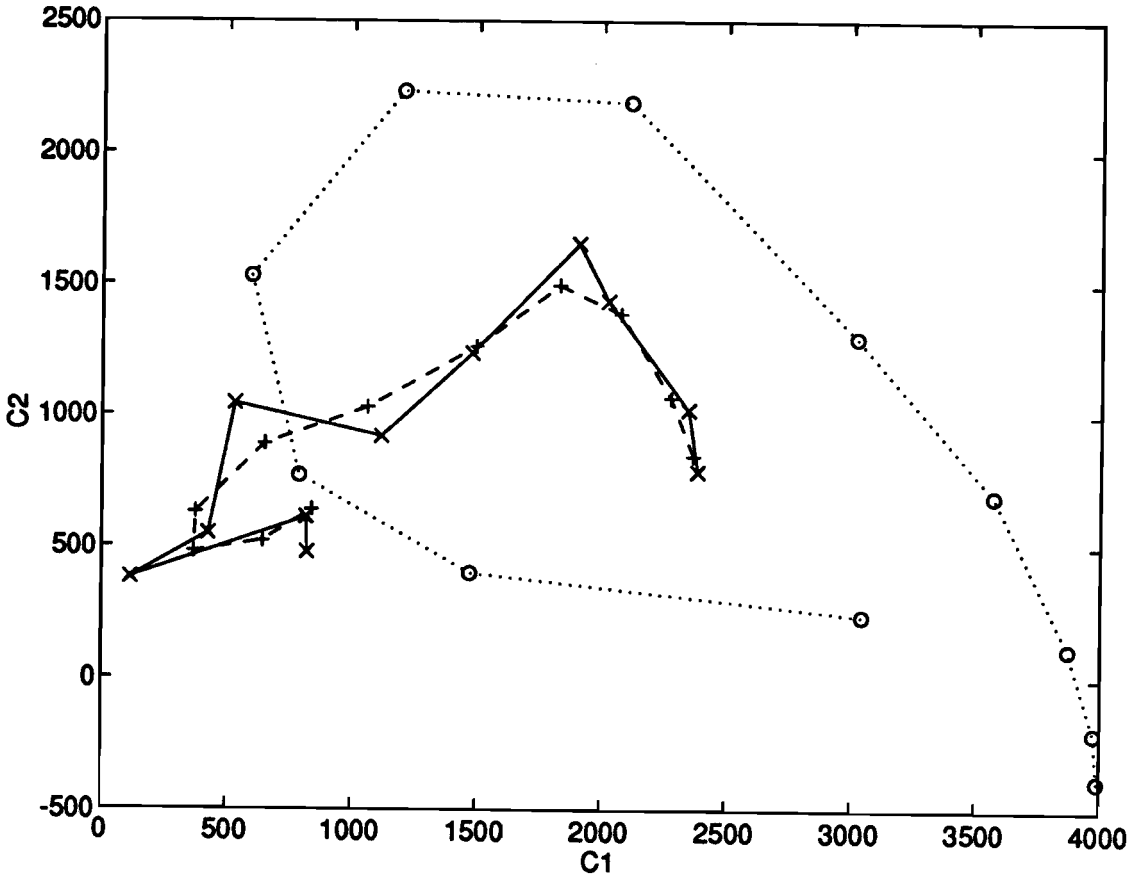

Figure 4.1: Trajectories of parameters (C1,C2) for clean (o, dotted) and noisy (x, solid) realizations of the phoneme transition (/e/z/). The underlying trajectory for clean speech is fairly smooth. For noisy speech a contextual view with several frames may be necessary to accurately map noisy vectors or points to clean counterparts. By averaging the noisy observation vectors over 3 frames, the trajectory (+, dashed) of the noisy speech smoothes out. (SNR=10 dB)

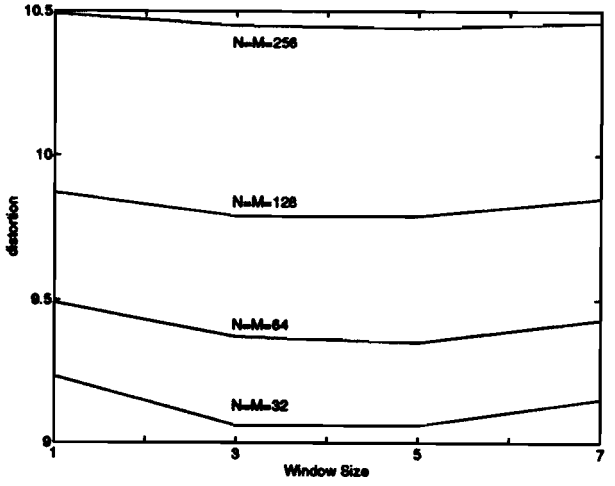

Figure 4.2: Reduction of distortion when different numbers of noisy speech frame vectors are averaged before mapping. In all cases of codebook size the distortion is reduced by averaging with the previous and next frames (Window Size 3).

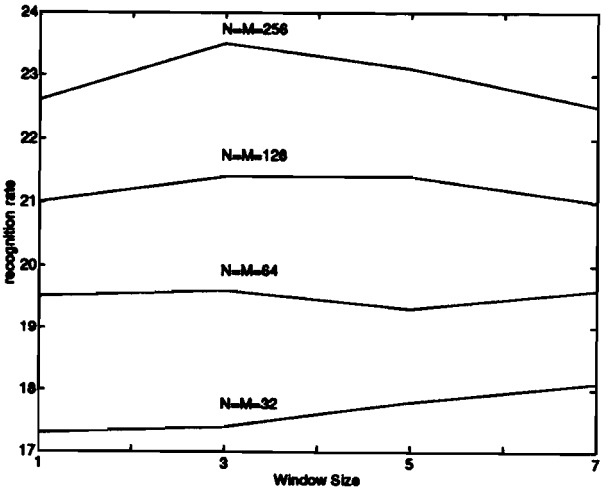

Figure 4.3: Improvements in recognition rate when different numbers of noisy speech frame vectors are averaged before mapping. At window size 3 or averaging only the immediately adjacent frames, recognition rate is improved for all codebook sizes.

approach was used. The fact that application frame averaging is beneficial to the mapping function is one indication that further gains can be made with a mathematically sound contextual information modelling. A natural choice would be to use Hidden Markov models, where each source could be identified with a state of a HMM, and a joint transition probability matrix in the clean and noisy vector spaces could be estimated and used to add context.

4.3 Class conditioned multiple codebook generation

Up to now, we have assumed that the entire cepstral space can be modelled by a number Gaussian sources. The statistics for all the sources are generated together from all the training data. This implicitly assumes that the cepstral space is roughly uniform and that all speech can be modelled together. In actual fact the spectra for different types of phonemes can be very different, and so can be the necessary spectral modelling. It is known for example that the space of LPC parameters models voiced speech much better that unvoiced speech. This section examines the possibility of using different codebooks (or mapping functions) for different types of phonemes.

One of the most natural ways to divide the phonetic space is through voicing. The use of voicing detectors in LPC codecs is widespread and can be used here as a preprocessor to separate voiced and unvoiced speech so that they can be modelled separately. Along the same lines, a silence detector can be designed so that we can split the overall acoustic cepstral space into 3 regions: voiced, unvoiced and silence. Each phoneme type can now be modelled by different mapping functions. Figure 4.4 shows a block diagram of the training and testing algorithms for this scheme.

We have chosen to model voiced (V), unvoiced (UV), and silent (S) speech with 128, 64, and, 8 sources respectively, roughly proportional to the number of different phones in each group. The number of sources for noisy speech has been set equal to the number of sources of clean speech and the VQ algorithm has been used for initialization of the mapping function.

The results of this experiment compare equally with the recognition results of the single codebook design. Table 4.2 shows the achieved distortions and recognition

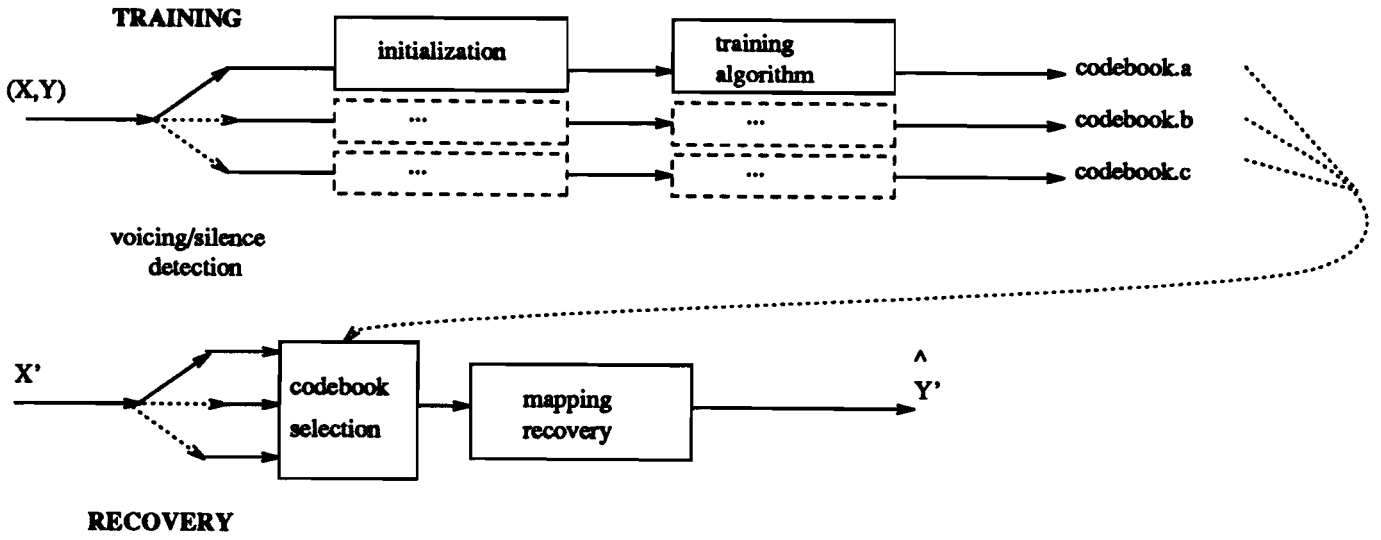

Figure 4.4: Block diagram of alternative system in which three regions (eg., voiced, unvoiced, silence regions) of the acoustic space are modelled separately. A mapping function for each region is trained using observations from each respective region. At the testing stage, one of the mapping functions is triggered by detecting from which region new test vectors come.

rates for the split codebooks ($N \times M = \{128 \times 128(V), 64 \times 64(UV), 8 \times 8(SIL)\}$, and $\{64 \times 64(V), 32 \times 32(UV), 8 \times 8(SIL)\}$) and the single codebook designs: sizes $N = M = 128$ and 256.

Splitting the codebook improves performance slightly but not significantly enough to justify the added complexity of the system. The voicing and silence detectors modelled here are assumed to be perfect, but in the testing stage actual implementation of these would be imperfect. One way to realize this system would be to model the detectors with probabilities: $P(voiced), P(unvoiced)$, and $P(silence)$. For each of the three codebooks the output vector could be found, and a weighted sum using the above probabilities could be used as the output vector. The performance of such a system could only be worse than that with perfect detectors. Since the added performance is only marginal the method has not been explored further.

It has also been found that such splitting of the codebook is done automatically to a very large extent. Examination of the source correlation matrix $\alpha_{ij} = p(\theta_j|\lambda_i)$ has revealed that few of the elements in a single row are non-zero. This shows that the

Conditions	distortion	recognition rates			
		voiced	unvoiced	silence	overall
clean speech	0	53	52	80	53
noisy, unprocessed	25.5	4.0	16.8	0	6.8
$N = M = 256$ mapped noisy	9.23	25.2	5.1	98.2	22.6
$N = M = 128$ mapped noisy	9.49	21.0	4.7	99.0	21.0
$64^2 \times 32^2 \times 4^2$	9.44	-	-	-	-
$128^2 \times 64^2 \times 8^2$	9.34	23	8	99.1	21.8

Table 4.2: Recognition results for split codebook mapping scheme. Compared with conventional codebooks of sizes 128^2 and 256^2, the split codebook does not perform better. Recognition of unvoiced and unprocessed speech is unusually high due to noise resembling certain phonemes.

Gaussian vector sources operate largely independently, modelling only small regions of the acoustic space. Splitting the acoustic space "manually" does not improve performance significantly. Furthermore, by the nature of the reestimation formula for α_{ij}, independent sources remain independent since new estimates are always proportional to old: $\alpha'_{ij} \propto \alpha_{ij}$, ie., matrix zeros remain zero (see eq. 2.16).

4.4 Noise resistant feature extraction for mapping

One of the properties of the mapping function formulation is that the noisy and clean speech vectors come from entirely different feature spaces. Thus, essentially any features can be used in the noisy speech vectors since these are not directly fed into the recognizer. These are merely used as input to the mapping function, which then outputs an optimal clean vector from the space in which the recognizer has been trained.

Noisy speech power spectra often contain significant formant information, but this information is overshadowed by the effect of the noise. In the extraction of the feature

vector (cepstral parameters) from a speech frame, the features may contain as much information about the noise as about the speech. It may be necessary to specialize the feature extraction for the noisy environment. We have applied an alternative feature extraction technique to the noisy speech involving convolution of the Fourier transform with a function of lateral spectral inhibition (FLSI) on a Bark frequency scale[5][21].

Lateral inhibition (LI) is a common phenomenon involved in sensory perception of biological systems. Of interest here is the spatial (or spectral) lateral inhibition in human hearing, which is one of the reasons speech intelligibility in humans is largely unaffected by a noisy background. When the speech and noise (white) are stationary and uncorrelated, then convolution of the noisy speech power spectrum with the function of lateral inhibition effectively removes the white or flat component of the noisy power spectrum while sharpening its peaks. In our study, we model the LI function by a piecewise linear function illustrated in Figure 4.5. The function is defined on the Bark frequency scale which reflects the fact that discriminability of frequencies by the human auditory system decreases logarithmically as frequency increases. A piecewise function for the Bark-frequency relation is used:

$$B(f) = \begin{cases} 0.001 \times f & 0 \leq f \leq 500Hz \\ 0.007 \times f - 1.5 & 500 \leq f < 1220Hz \\ 6 \times \ln(f) - 32.6 & 1220 \leq f \leq \infty \end{cases} . \tag{4.3}$$

The LI function, as shown in Figure 4.5, has six parameters: B_L, B_c, B_r, and P_L, P_c, P_r. We have set these according to experimental results in [5], which have yielded the best results for speech enhancement: $B_L = B_r = B_c = 1$ Bark, $P_c = 1, P_L = -0.6, P_r = -0.4$.

Figure 4.6 shows a block diagram of the complete feature extraction system. First, the harmonics associated with voiced speech are removed via cepstral processing, smoothing the spectrum. In the diagram $w_1(k)$ and $w_2(k)$ represent Hamming windows, the first applied to speech frames and the second to cut off cepstra at 10 ms. Since lateral inhibition is very sensitive to spectral peaks, some undesirable peaks from the noise component of the power spectrum may be enhanced. To counter this effect, we take a weighted average of the power spectrum over 5 speech frames to remove the irregular peaks due to the noise signal. Formant peaks which usually last

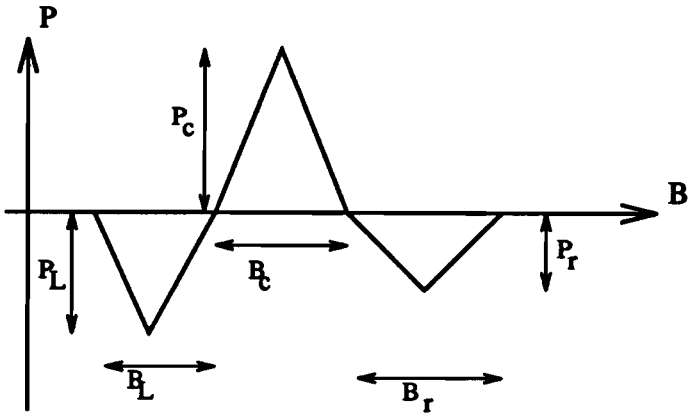

Figure 4.5: Modeling of the function of lateral spectral inhibition (FLSI).

longer are retained. For a power spectrum of the signal at time t, $P_x^{(t)}(\omega)$, we use a weighted spectral average

$$\bar{P}_x^{(t)}(\omega) = \sum_{s=-M}^{M} W_s P_x^{(t-s)}(\omega) \quad . \tag{4.4}$$

In practice, we set $M = 2$ and we have empirically defined the weights $[W_{-2}, W_{-1}, W_0, W_{+1}, W_{+2}] = [0.08, 0.26, 0.32, 0.26, 0.08]$. The averaged power spectrum is then convolved with the LI function, and we use the resulting sequence to filter the original power spectrum in the adaptive filtering block. Filter bank energies are computed, and after cosine transformation the cepstral coefficients obtained make up the feature vector. In comparison, conventional feature extraction would be made only of the left portion of the figure, without the two delays and adaptive filtering.

The effect the convolution has on spectra is shown in Figure 4.7. Cepstral processing is performed over a filter bank of 25 filters, to which the cosine transform is applied. Formants in the clean (solid) spectrum are much more evident than in the noisy (dashed) spectrum, but some extraction through LI processing is possible. The LI processed spectrum (dotted) effectively extracts the formant structure at filters 1, 4, 11, 14, 17 and 21. It is important to note that the aim here is not make the noisy spectrum look like the clean one; LI processing merely extracts features not evident in the noisy spectrum. Transformation into a "clean-looking" spectrum is performed at a later stage by the mapping function.

LI processing is applied only to the noisy speech for feature extraction. These

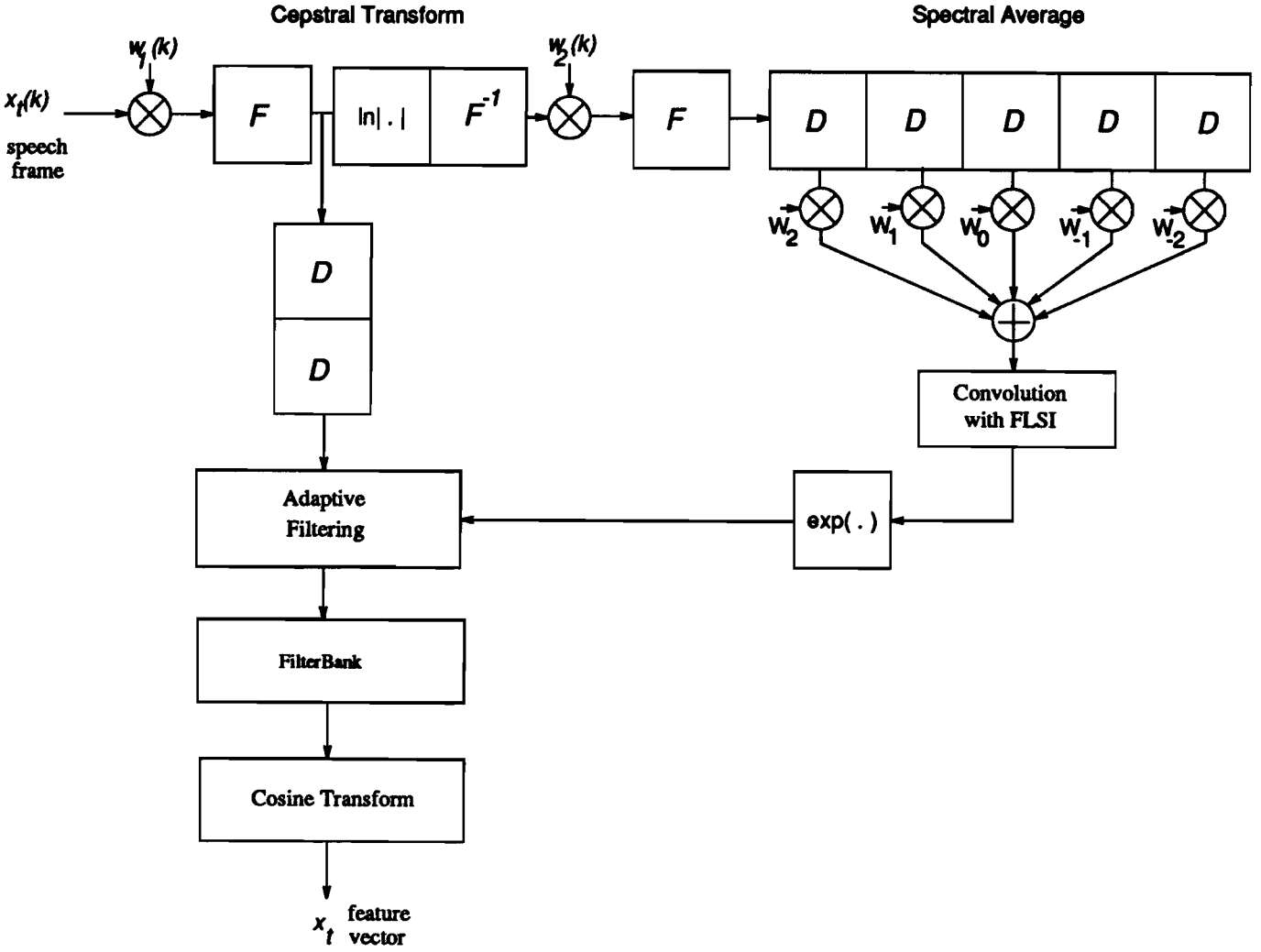

Figure 4.6: Block diagram of the lateral inhibition processing within the feature extraction module for the noisy speech. $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent direct and inverse Fourier transforms, respectively. $\mathcal{D}$ represents a one frame delay.

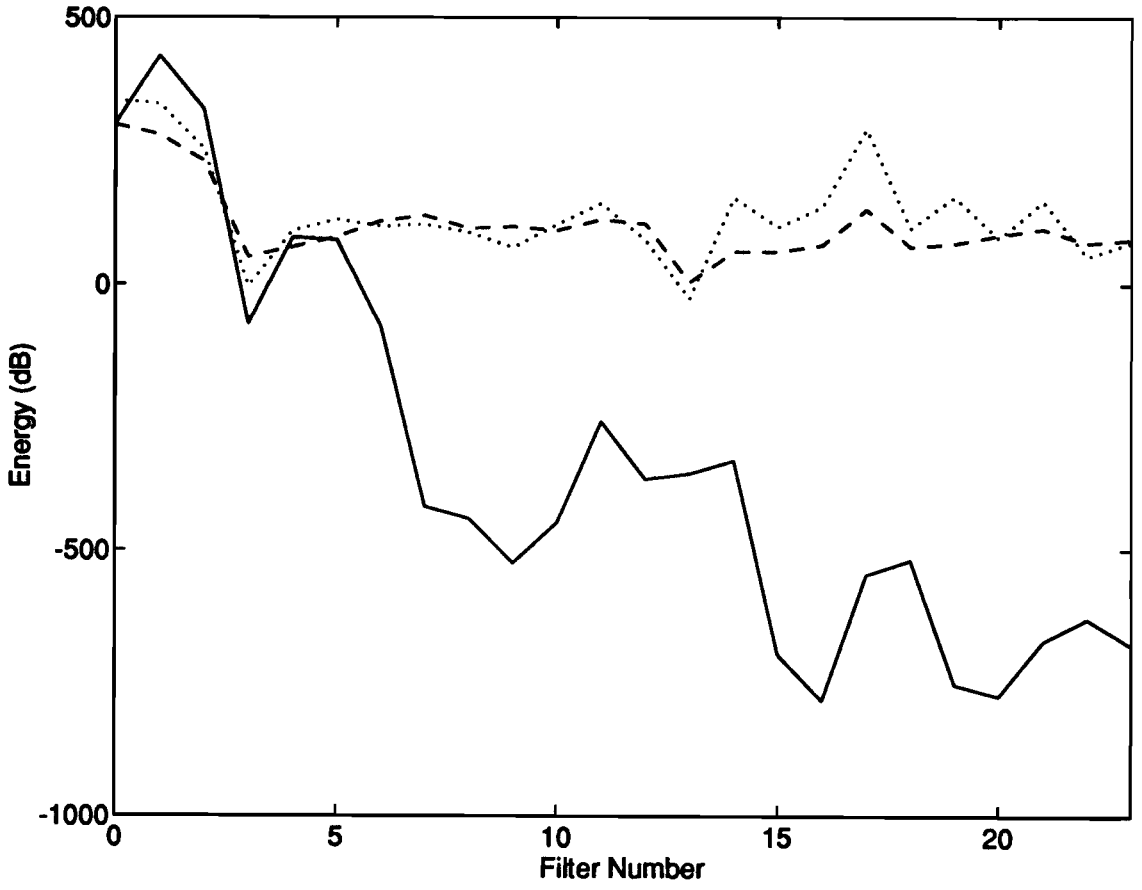

Figure 4.7: Comparison of clean and noisy spectra, by filter bank energies used to calculate the cepstrum. The clean spectrum (solid) shows good formant peaks but addition of noise (10dB SNR) removed much to the structure (dashed). Processing through lateral inhibition extracts much of the formant structure (dotted) for later mapping to a clean spectrum.

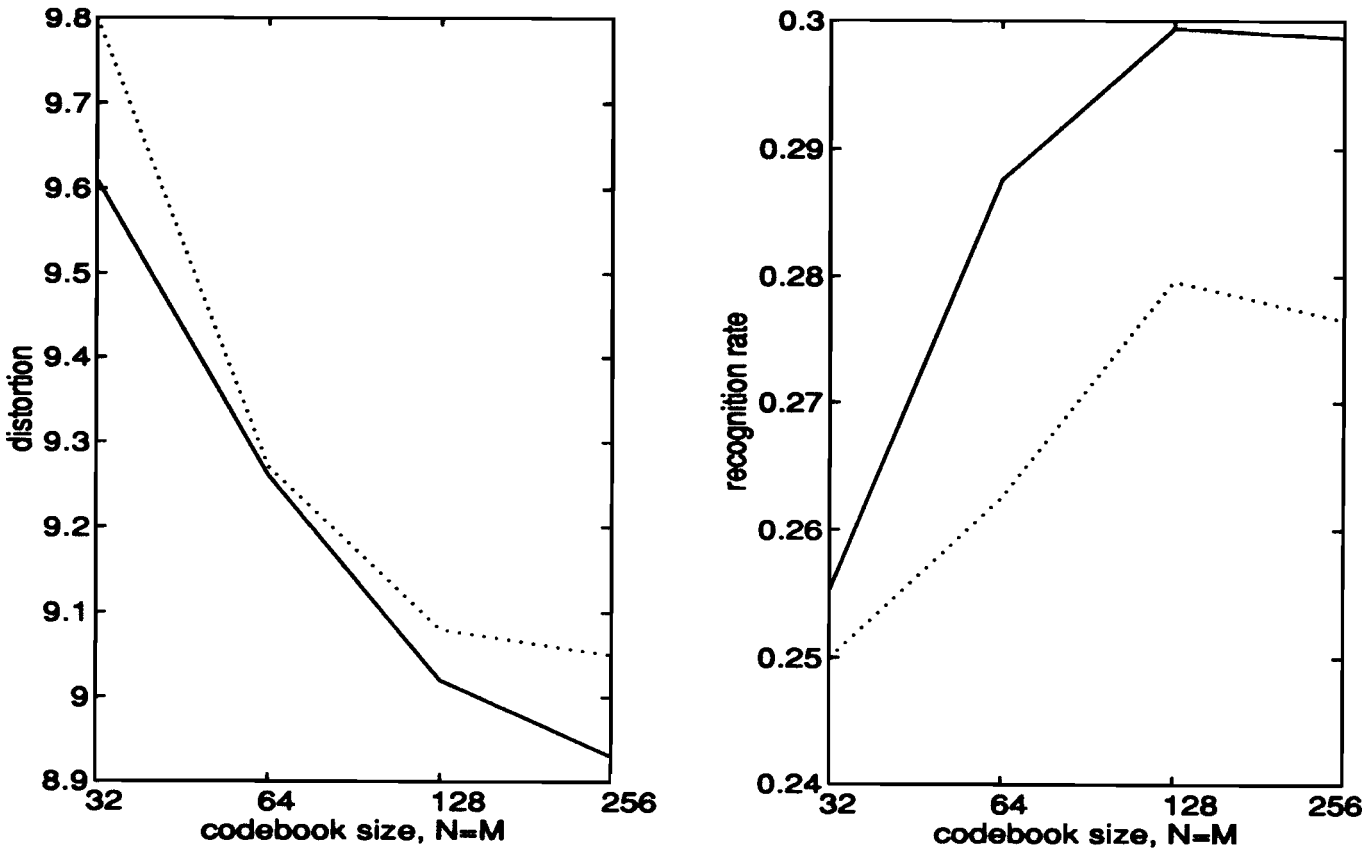

Figure 4.8: Comparison of performance with (solid) and without (dotted) lateral inhibition processing. For all codebook sizes distortion of recovered speech is reduced and recognition rate is improved. Note: MMSE estimation is used towards distortion plot and MAP-S in recognition plot.

new features are used as noisy observation vectors, $\{x_t\}$. With the conventional clean speech feature vectors, these are used to train the mapping function. At the recovery stage, LI is applied again to generate noisy speech feature vectors, to be mapped via the trained function into clean vectors. Figure 4.8 summarizes the results obtained. For mapping 8 features with the JVQ^* mapping function initialization and 6 minutes training, distortion has been reduced and recognition rates have improved over nearly all codebook sizes.

Computational costs incurred using this method are high. Whereas conventional cepstrum generation required one Fourier transform and one cosine transform, the

LI processing requires an additional two Fourier transforms, a convolution, and filtering. Unless specialized components are used, the advances using this technique are not worthwhile. Nevertheless, other benefits of this processing in robust speech recognition will be explored in the following chapter.

4.5 Summary

Three extensions to the mapping function that would improve feature extraction have been proposed in this chapter.

The first attempted to use features spanning several frames to reduce the impact of the noise component. This was done by either (a) augmenting the feature dimension to include cepstral observations from several frames or (b) averaging these over a window of frames while keeping the feature dimension the same. We have found better performance in the latter method primarily due to the limited statistical capacity of the mapping function which cannot model large numbers of features with reasonable numbers of sources and training.

The second method attempted the use of available and effective techniques for dividing the acoustic space into voiced, unvoiced, and silence regions so that different and non-interfering mapping functions could be generated from and applied in each region. We have found that, to a large extent, this is done automatically by the source cross-correlation matrix which determines independence in different regions of the cepstral observation space. Improvements from this method were negligible if any.

In the third method we applied an advanced feature extraction technique based on lateral inhibition, as in human hearing. Noise suppression performed by a convolution of the power spectrum with a spectral lateral inhibition function before feature extraction was observed to improve both distortion and recognition rates of recovered noisy test speech. From Figure 3.1, our best recognition result from application of this method are the same as that of unprocessed speech at 27 dB, implying an effective improvement of 17 dB. Similarly, our best effective improvement in distortion (using Figure 3.2) is about 14 dB.

While the mapping function effectively recovers much clean features from noisy

speech, it's performance depends very much on the training environment, especially the noise level at which it was trained. The degree of that dependence and methods to reduce it are presented in a feasibility analysis in the next chapter.

Chapter 5

Feasability in Varying Noise Environment

The previous chapters have presented the statistical mapping function for mapping noisy speech observations to clean observations. The method is useful when a recognition system trained on clean speech must be adapted as best as possible to a noisy environment without retraining. In this chapter, we will show how this system is useful even when retraining the recognizer is possible but cannot improve system performance due to a changing noise environment.

Since the mapping function is trained on noisy and clean speech and the noisy speech has a fixed SNR, the function generated will invariably be SNR-dependent, meaning that noisy speech with an SNR other than that in training will perform poorly compared to speech with the training SNR. The degree of dependence can have a large effect on system performance when the testing environment has a changing noise level, which is often the case in real world applications.

Two ways to reduce the SNR dependence of the mapping function are presented.

The first, presented in the following section, is to train the mapping function on noise-level independent features.

A second way is to generate many mapping functions at different noise levels and to lump all the functions in to a combined one. The degree to which each mapping function in the combined one is activated could be set by a noise-level estimator. Of course this is a difficult task but we will show that improvements are still possible

even with a most rudimentary estimator. This is presented in section 5.2.

5.1 Noise-Independent Statistical Spectral Mapping

A method of noise-independent feature extraction for speech enhancement has been developed by Cheng, O'Shaughnessy, and Kabal [7]. As in the previous chapter, it involves convolution of the power spectrum with a function of spectral lateral inhibition. When trained using these features, the SNR dependence of the mapping function should be reduced.

We have first monitored the SNR dependence of the mapping function trained with regular features. Three mapping functions have been trained at SNR's 10, 15, and 20 dB, each of size 64 by 64 sources, and mapping 8 features. Training size was 60000 tokens (10 min.) and 20 iterations of EM reestimation were performed. These were tested on speech ranging in SNR from 8 to ∞ dB or clean speech. For 10 dB test speech, recognition performance of recovered speech using a 10 dB trained function was 26.2 % correct. For 15 and 20 dB functions performance was reduced by 2 and 11% respectively. For clean test speech, results were very poor, falling below the 10% recognition mark for all three functions. Complete results are shown in Figure 5.1 where for each function (different training SNR's), we plot (dotted) the recognition rate versus test SNR.

The same experiments were conducted using mapping functions trained with noise-independent features. For these, lateral inhibition processing was applied to noisy speech before cepstral processing. Three mapping functions were trained at SNR's 10, 15 and 20 dB. At the testing stage, LI processing is applied again to all input speech followed by cepstral processing and mapping to clean speech. Figure 5.1 compares the results obtained with and without LI processing. As expected from the results of chapter 4, the entire plot for LI processed speech (solid) is higher indicating better performance at fixed SNR. We also see that the LI processing plots are less peaked, indicating that the system performance is maintained over a wider range of SNR's. Although the LI processing does not remove the SNR dependence completely, the results remain better across different testing SNR's.

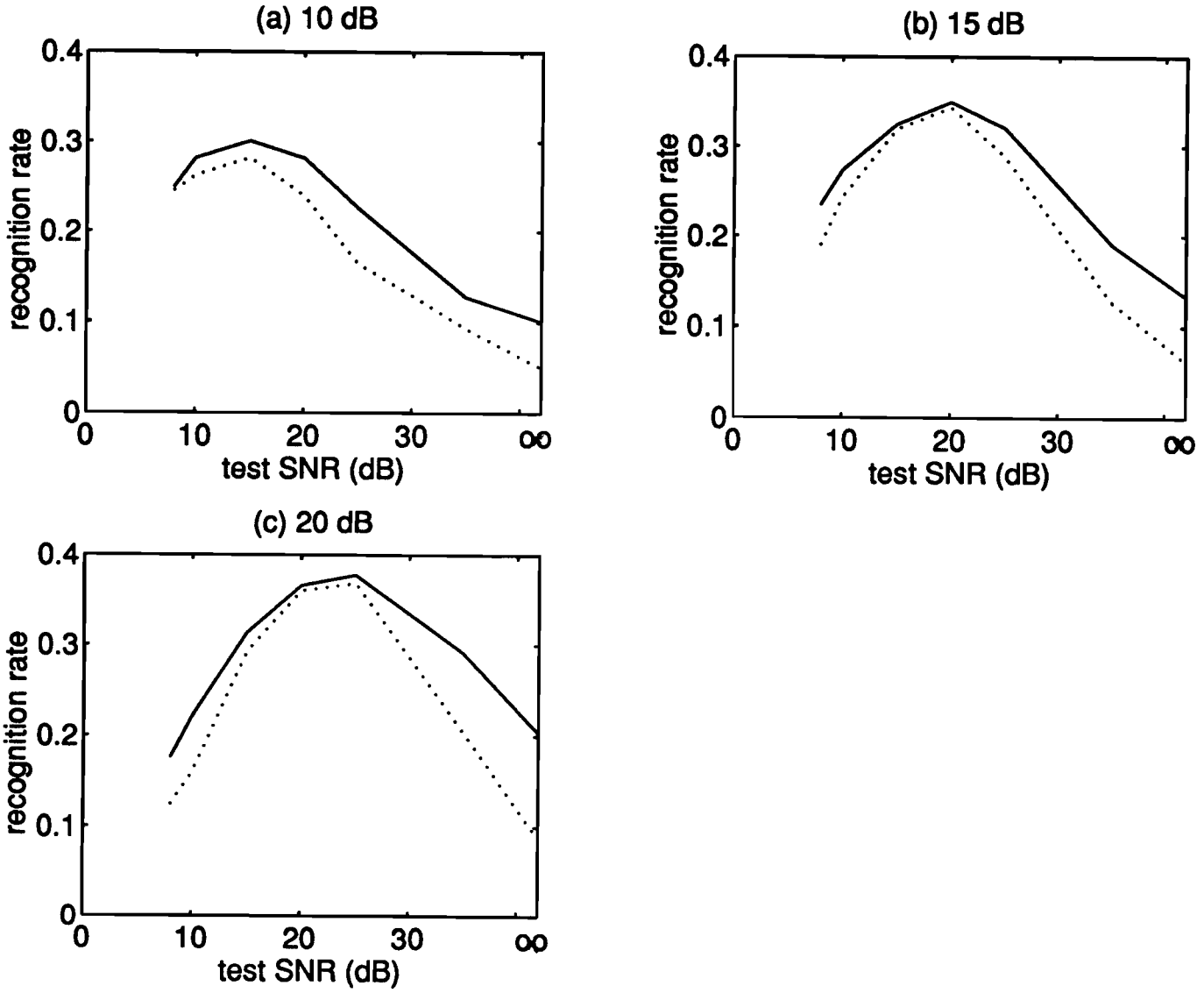

Figure 5.1: Recognition rate versus testing SNR for LI processed speech (solid) and unprocessed speech (dotted), and for functions trained at (a) 10 dB, (b) 15 dB, and (c) 20 dB. Flatter and wider plots indicate less dependence of the function to noise-level at testing.

5.2 Lumped codebooks

Lateral inhibition processing effectively reduces the noise level dependence of the mapping function, but unless a specialized computer is used, the computational load associated with it is high. One alternative is to estimate the noise level and activate the appropriate mapping function, but this would imply full knowledge of the noise level. Since it is difficult to estimate the noise level, it may be more realistic to attach probabilities to each noise level at which a mapping function has been trained. This would indicate partial knowledge of the test SNR, which is realistic. Using these probabilities and a combined mapping function the output vector can be estimated.

Let $\Phi_k = \{\Lambda_k, \Theta_k, A_k\}, 1 \leq k \leq K$, be the set of mapping functions each having been trained at noise levels $SNR_k, 1 \leq k \leq K$, respectively. With the SNR estimator giving probabilities, $p(SNR_k)$, the combined mapping function is denoted $\Phi = \{\Lambda, \Theta, A\}$ and is defined in the following way:

- The sources Λ have means μ_λ and diagonal covariance matrices Σ_λ, defined by concatenating the means and variances of Λ_k from functions $\Phi_k, 1 \leq k \leq K$, respectively.

- The sources Θ have means μ_θ and diagonal covariance matrices Σ_θ, defined by concatenating the means and variances of Θ_k from functions $\Phi_k, 1 \leq k \leq K$, respectively.

- The a-priori source probabilities, $p(\lambda_i)$ are defined by concatenating the source probabilities $p(\lambda_i^{(k)})$ weighted by the SNR_k probability, $p(SNR_k)$. Since $\sum_{k=1}^{K} p(SNR_k) = 1$, and for each function $\sum_i p(\lambda_i^{(k)}) = 1$, we have

$$\begin{aligned} \textstyle\sum_i p(\lambda_i) &= p(SNR_1) \textstyle\sum_i p(\lambda_i^{(1)}) + p(SNR_2) \sum_i p(\lambda_i^{(2)}) + \ldots + p(SNR_K) \sum_i p(\lambda_i^{(k)}) \\ &= p(SNR_1) \cdot 1 + p(SNR_2) \cdot 1 + \ldots + p(SNR_K) \cdot 1 = 1. \end{aligned} \tag{5.1}$$

- The source correlation matrix, A is the concatenation of matrices $A^{(k)}, k = 1 \ldots K$,

$$A = \begin{pmatrix} A^{(1)} & 0 & \dots & 0 \\ 0 & A^{(2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A^{(K)} \end{pmatrix} . \tag{5.2}$$

The end performance of the system will depend on on how well the test SNR has been estimated. If the SNR estimator is perfect, it will associate a probability of 1 to the correct SNR, and will trigger only the mapping function with that training SNR. In the worst case, and for equally spaced SNR_k, the SNR estimator will output equal probabilities for all SNR_k. Any SNR estimator will perform somewhere in between that of a perfect detector and the worst case detector. Since the performance with a perfect estimator or fixed SNR is known from the previous section, we have conducted experiments with the equal probabilities type detector.

The three mapping functions (10, 15, and 20 dB) were combined in the manner described above, with the SNR probabilities $p(10dB) = p(15dB) = p(20dB) = 1/3$. Testing was performed on speech at 10, 15 and 20 dB SNR's. The results are summarized in table 5.1. The first column displays results when a single mapping function trained at 15 dB is used, i.e., no SNR detector. The second column indicates results when the mapping function used has the same training SNR as at testing, i.e., a perfect SNR detector. Finally, the third column shows results with a combined mapping function using equal probabilities for each noise-level. As expected the combined mapping function compares favourably with the single function 15 dB case, except when testing at 15 dB where the latter is specialized.

The results show that gains can be made even if a rudimentary SNR detector, estimating the SNR to within +/- 5 dB, is used. If the detector is improved, performance can be pushed even further up to the point where the noise-level specialized function is used all the time. Computations arising from this method are much less than with LI processing, requiring K^2 times the computational load when K mapping functions are combined, but for good SNR detection K could be as low as 2. The computations from the SNR detection are minimal in relation.

In these experiments, the mapping functions have been combined in a very simple way. Concatenation of sources in the clean space could be replaced by a more ad-

test SNR (dB)	single map (15 dB)		single map (test SNR)		combined (10,15,20 dB)	
	rec. rate	dist.	rec. rate	dist.	rec. rate	dist.
10	24.3	11.98	26.2	11.92	26.0	11.92
15	32.0	10.33	32.0	10.33	31.4	10.36
20	34.4	9.80	36.0	9.03	35.1	9.40

Table 5.1: Comparison of performance for combined mapping function (10,15,20 dB) with a single mapping function (15 dB) and a single mapping function whose training SNR is the same as testing. The combined mapping function compares favourably with the single 15 dB codebook case except at the 15 dB test SNR for which the latter is specialized. The single function whose training and testing have equal SNR performs best but would require a perfect SNR detector under a changing noise environment.

vanced technique since the clean sources from each mapping function model the same space. In this case many fewer computations would be needed at recovery time.

Although this method can be applied independently of the LI processing of the previous section, further improvements from applying both methods together have not been observed.

In both cases, the usability of the system in a clean test environment could be improved with controlled addition of noise prior to processing, as has been used by Van Compernolle [24].

The following chapter reviews and concludes this thesis. Other possible methods for improving application of the function are also suggested for future research.

Chapter 6

Summary and Conclusions

A general framework of statistical signal mapping (SSM) has been applied to robust speech recognition in noise. We have assumed that a non-linear function exists which can map noisy speech observations to their clean speech equivalents. The function has been modelled as a statistical mapping whose parameters are estimated by a long sequence of vector pairs where the elements correspond to observations in each respective space, noisy and clean. With the ultimate aim of improving the recognition rate for noisy speech we have applied and specialized the SSM method.

Since our recognition system uses cepstral observations and continuous density Gaussian sources, we have chosen these for our mapping study as well. In this way, the optimality criterion for the mapping function is equivalent to that in the recognizer, unlike many other studies. The noisy (and clean) observations were assumed to be generated by an ensemble of Gaussian sources defined over the noisy (and clean) speech vector space. Parameters of the mapping function are the means and diagonal covariance matrices of each source, prior probabilities of each source and a source correlation matrix whose elements define the activation probability of a source in the clean space given one in the noisy space.

Estimation of the parameters in the mapping function is iterative, each iteration being an application of the EM reestimation algorithm which tends to maximize the joint likelihood of the training sequence of vector pairs and the model. We found that convergence of the mapping parameters to a global optimum is sensitive to the initialization of the iterative algorithm. For the initialization stage many

techniques were tried, the best of which involves vector quantization with a "split-and-merge" scheme which reduces the incidence of large clusters. We have also seen that partitioning of observations in the noisy space should be based on the clean speech observations. We believe that these two improvements (over conventional VQ in each space) are an indication that a specialized initialization technique could be found. One using the available phonetic segmentation information may be useful, especially if this information is also used in the iterative process.

The variables in the model were tuned for best performance while limiting computation. In general, we have have seen monotonic improvement with increasing numbers of sources and have observed best results when an equal number of sources is used to model noisy and clean speech. Also, since the statistical capacity of the mapping function is limited, we have observed better results when fewer features per observation vector are used in the modelling. Thus, enhanced delta cepstral observations are better to be derived from the enhanced cepstrum rather than by mapping from the noisy delta cepstrum.

We found that 5-6 minutes of training data with 10-15 iterations of EM reestimation to be adequate for close to optimal performance. We have not yet discovered a good stopping criterion for the number of iterations since performance sometimes improves beyond the point where the log likelihood levels off.

When the mapping function is derived, it is applied to new noisy speech in order to estimate clean speech vectors which are then used as input to the recognizer. Given a noisy observation, the mapping function can assign a probability to any clean vector. Thus, using any optimization criterion, the corresponding clean vector can be found. From statistical communication theory, the estimator which minimizes the number of classification errors is the maximum-a-posteriori probability (MAP) estimator. Since there is no closed form solution for this estimator, we have relied on two others: MMSE and MAP-S. In MMSE estimation, the output vector is taken as the sum of clean source means where each mean is weighted by the posterior probability of the source given the noisy vector. In MAP-S estimation, the mean from the source with the highest a-posteriori probability is chosen. Recognition performance is consistently better when MAP-S estimation is used, especially when initialization of the noisy source means and covariance matrices is performed using the clean training tokens.

This is not surprising since MAP-S comes much closer to approximating the MAP estimator than does MMSE. Whereas other studies have used MMSE estimation of various functions of the log-DFT [12][13][11], we have not yet seen enhancement methods both using MAP and applied in the cepstral domain- both of which are needed for compatibility with continuous density Gaussian HMM recognition systems.

Beyond basic use of the mapping function we have attempted to improve its performance by using other techniques including contextual modelling. Since noisy speech observations are often erratic, we have tried mapping several observations from consecutive noisy frames (trajectories) to single clean speech vectors. This has not brought improvements, which we believe is due to the fact that the mathematical development is single-frame based. Improvements have been observed when the noisy frames are averaged with their left and right contexts but these have been small and the use of this technique may harm performance in low noise environments.

Since cepstra of voiced and unvoiced speech are very different we have also tried mapping these types of speech separately since they may interfere with each other within the mapping function. Separate mapping functions were trained for voiced, unvoiced and silent portions of speech, and were used to recover the different types of speech separately. These have not improved performance but we have learned that this is largely done automatically by the mapping function since there are many zeros in the source correlation matrix. Nevertheless, this type of splitting can enable the use of different types of models (rather than Gaussian and cepstral all around) for different types of speech when a separate detector is used to determine the type.

Since the noisy speech is mapped rather than filtered, any speech enhancement technique can be used as a pre-processor before mapping. An improved feature extraction technique based conceptually on auditory evidence has been used. The method involves convolution of the power spectrum with a function of spectral lateral inhibition. With this technique we have observed improvements of 2 % in recognition rate and reductions in distortion.

One advantage to using the mapping function over retraining the recognizer is that much less training is required. Where about an hour of speech is required to train even the simplest of recognizers, training the mapping function requires less than 10 minutes of speech. If retraining the entire recognizer is impossible, use of the

mapping function creates a good alternative.

Since the mapping function is trained at a certain SNR, it is SNR dependent. This means that application of the function to noisy speech of SNR different from that in training will perform poorly, even if the test speech has less noise. We have addressed this problem and have found two viable solutions. The first is to train the mapping function on noise-independent features. Using the improved feature extraction technique as above, we have effectively widened the range over which the mapping can be applied. The second way this can be done is to lump mapping functions having been trained at different SNRs. The combined function has some SNR detection capability from the feature extraction stage so the proper function can often be triggered automatically. When coupled with a specialized but simple SNR detector, the usable SNR range can effectively be widened. The SNR dependence has not been removed completely but we believe further research along these lines can reduce the dependence further.

The goal of this study was to improve the speech recognition rate for noisy speech perhaps beyond the performance of the noisy speech trained recognizer. While clean speech is recognized at 54 %, and the noisy speech recognition rate for 10 dB SNR is 7 %, our best result with application of this method yields a rate of 30 %. Since the performance of the recognizer trained and tested on noisy speech is 35 % correct we have fallen short of our goal. Nevertheless, the recognition rate improvement from 6 to 30 % represents an effective 17 dB gain in SNR. Reduction in distortion due to this processing also corresponds to effective improvement of 14 dB. These are respectable results similar to those reported in the literature [16][19][24][20][12]. We believe that further potential of the method could be shown using a superior recognition system. Experiments based on other speakers and with environmental noise may also be necessary to draw further conclusions.

The SSM method has a unique advantage over other techniques in its ability to be optimized and employed with the same criteria as most any speech recognition system. Some of the remaining areas where improvements could still be made are mentioned below.

6.1 Future Directions

One of the disadvantages of the mapping technique lies in the Gaussian source modelling. The Gaussian vector sources view all the coefficients of the observation vector as equally important, since they are always weighted by their variance. It is known however, that the higher order coefficients usually contain less information than the lower order ones - especially when they originate from noisy speech. Some research effort may be necessary to incorporate liftering of cepstra in the models. Of course, it is not clear that this will lead to improved recognition performance since the optimization criterion for the mapping function would now be different from that of the recognizer, which uses non-liftered cepstra.

We also believe improvements can be made with regards to the distribution of sources in the clean and noisy vector spaces. The performance improvement in going from 32 to 64 and 128 sources is not as significant as might be expected, which may be due to irregular sizes (or covariance matrix elements) of the sources. At initialization we have observed that reducing the incidence of large clusters by a "split-and-merge" technique has led to improved performance in the mapping function. Executing this type of redistribution may be useful during the iterative stage also, but an increasing log likelihood could not be guaranteed in this case due to the deviation from straightforward application of the EM algorithm.

With regards to the estimation of the output vector using the mapping function, of the two methods found, MAP-S outperforms MMSE in recognition tests. Since MAP-S outputs vectors selected from the finite set of M clean source means, and M is of the order of the number of phonemes, this creates many "runs" of the same output vector for up to 7-8 frames. When delta cepstral coefficients are derived from this data, many frames have zeros for those features. With the aim of recovering useful delta cepstral coefficients, further research could be directed towards finding an alternative to MAP-S for approximating the optimal MAP estimation. Alternatively, a specialized smoothing method could be applied to remove the "runs" from the output sequence. Both these methods may also overcome difficulties in using MAP-S when large numbers of sources are used.

The SSM method, as developed in [8] and [9] and described here, estimate a single frame at a time. Some studies [12],[11],[20] in speech enhancement and robust speech

recognition have removed this limitation by incorporation of Markov models in their estimation methods with successful results. Such an approach may also prove useful in our method as well.

The SSM method has already been used in speech enhancement and recovery of wideband speech from narrowband speech, but it can potentially be applied to speaker and microphone adaptation as well.

References

[1] A. Acero and R. M. Stern, "Cepstral Normalization for Robust Speech Recognition" *Proceedings, Speech Processing in Adverse Environments,* pp. 89-92, 1992.

[2] L.E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov processes," *Inequalities 3:* pp. 1-8, 1972.

[3] G. Boulianne, P. Kenny, M. Lennig, D. O'Shaughnessy and P. Mermelstein, "HMM Training on Unconstrained Speech For Large Vocabulary Continuous Speech Recognition," *Proceedings, Int. Conf. on Spoken Language Processing,* pp. 229-232, 1992.

[4] A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel, "Speech Coding Based Upon Vector Quantization," *IEEE Trans. on Acoustics, Speech, Signal Processing,* vol. ASSP-28, no. 5, pp. 562-574, 1980.

[5] Y. M. Cheng and D. O'Shaughnessy, "Speech Enhancement Based Conceptually on Auditory Evidence," *IEEE Trans. on Acoustics, Speech, Signal Processing,* vol. ASSP-39, no. 1, pp. 1943-1954, 1991.

[6] Y. M. Cheng and D. O'Shaughnessy, "Speech Enhancement Based Conceptually on Auditory Evidence," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 961-964, 1991.

[7] Y. M. Cheng, D. O'Shaughnessy and P. Kabal, "Speech Enhancement Using a Statistically Derived Filter Mapping," *Proceedings, Int. Conf. on Spoken Language Processing,* pp. 515-518, 1992.

[8] Y. M. Cheng, D. O'Shaughnessy and P. Mermelstein, "Statistical Signal Mapping: A General Tool for Speech Signal Processing," *Proceedings, Workshop on Statistical Signal and Array Processing,* pp. 436-439, 1992.

[9] Y. M. Cheng, D. O'Shaughnessy and P. Mermelstein, "Statistical Recovery of Wideband Speech from Narrowband Speech," *Proceedings, Int. Conf. on Spoken Language Processing,* pp. 1577-1580, 1992.

[10] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Ann. of the Royal Stat. Society,* pp. 1-38, 1977.

[11] Y. Ephraim, "A minimum mean square error approach to speech enhancement," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 829-832, 1988.

[12] A. Erell and M. Weintraub, "Filter Energy Estimation Using Mixture and Markov Models for Recognition of Noisy Speech," *IEEE Trans. on Speech and Audio Processing,* vol. SAP-1, no. 1, pp. 68-76, 1993.

[13] A. Erell and M. Weintraub, "Energy Conditioned Spectral Estimation for Recognition of Noisy Speech," *IEEE Trans. on Speech and Audio Processing,* vol. SAP-1, no. 1, pp. 84-89, 1993.

[14] H. Garudadri, personal communication.

[15] X. Huang, "Speaker Normalization for Speech Recognition," *Proc. Int. Conf. IEEE Acoust., Speech, Signal Processing,* pp. 2368-2371, 1987.

[16] B.-H. Juang and L. R. Rabiner, "Signal Restoration by Spectral Mapping," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 2368-2371, 1987.

[17] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Communications,* vol. COM-28, no. 1, pp. 84-95, 1980.

[18] F-H. Liu, A. Acero, and R. M. Stern, "Efficient Joint Compenstation of Speech For the Effects of Additive Noise and Linear Filtering," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 257-260, 1992.

[19] D. Mansour and B.H. Juang, "A family of distortion measures based upon projection operators for robust speech recognition," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 36-39, 1988.

[20] A. Nádas, D. Nahamoo, and M. A. Picheny, "Speech Recognition Using Noise-Adaptive Prototypes," *IEEE Trans. on Acoutics, Speech, and Signal Processing,* vol. ASSP-37, no. 10, pp. 1495-1503, 1989.

[21] S. A. Shamma, "Speech Processing in the Auditory System II: Lateral Inhibition and the central processing evoked activity in the auditory nerve," *J. Acoust. Soc. Amer.,* vol. 76, pp. 1622-1632, 1985.

[22] F. Soong and M. M. Sondhi, "A frequency-weighted Itakura spectal distortion measure and its application to speech recognition in noise," *IEEE Trans. on Acoutics, Speech, and Signal Processing,* vol. ASSP-36, no. 1, pp. 41-48, 1989.

[23] Y. Tohkura, "A Weighted Cepstral Distance Measure for Speech Recognition," *IEEE Trans. on Acoutics, Speech, and Signal Processing,* vol. ASSP-35, no. 10, pp. 1414-1422, 1987.

[24] D. Van Compernolle, "Noise adaptation in a hidden Markov model speech recognition system," *Comp. Speech & Language,* vol. 3, pp. 151-167, 1989.