Classification of noise relative to speech

Marte Haavik Vadla

Master of Science in Electronics
Submission date: July 2018
Supervisor: Tor Andre Myrvoll, IES

Norwegian University of Science and Technology
Department of Electronic Systems

Problem Description

It is desirable to locate a speaker in a room and have the microphones focus on this speaker. The problem that will be explored in this thesis is to implement source localization based on the inputs from the 64-channel microphone array, and thereafter find an appropriate method for classifying speech relative to noise. A classification method will be chosen to be examined in detail. Model adaption will be implemented as a way of improving the system performance, with the aim to adapt the parameters of the speech model to better match the input data.

Preface

The following work is produced by Marte H. Vadla as my master thesis, as part of my Master of Science degree in Electronics Engineering, with the specialization of Signal Processing and Communication, at the Norwegian University of Science and Technology (NTNU). The thesis was formulated and submitted at the Institute of Electronic Systems, in collaboration with SINTEF Digital and Armoni AS. The master thesis was formulated and carried out throughout the spring semester of 2018.

Trondheim, July 1, 2018

Marte Haavik Vadla

Acknowledgment

I would like to thank my supervisor Tor Andre Myrvoll at NTNU, my co-supervisor Tor Arne Reinen at SINTEF Digital and the people at Armoni AS for all their help and support with the thesis.

MHV

Summary

The implementation of the source localization in MATLAB showed satisfactory results, where the system was able to determine from which directions sound sources were coming from. However, the system only found the strongest sound sources without taking into account which type of sound it was. Due to this, it was necessary to implement sound classification to distinguish between speech and non-speech.

The methods that were used in this thesis were Mel-frequency cepstral coefficients for feature extraction, and Gaussian mixture models for classification. Two systems were developed for classification, each with a speech model with 512 mixtures and a noise model. One noise model had 2048 mixtures and the other noise model had one mixture. Based on the log likelihood of each model output, the performance was measured. The system performance showed to be poor for inputs of one frame of 25ms, with an equal error rate above 30% for both systems. The more frames that were used as inputs, the higher the accuracy was and the more robust the threshold value became. Both systems reached 100% accuracy when the input sequences had a length of up to several seconds. Silence was classified as noise, which was optimal as it was undesirable for the system to focus on silence.

The system performance was expected to be improved from model adaption. If only the speech model was adapted, the system with the 1-mixture noise model showed very positive improvements with an average of 8.6% decrease in the error rate, reaching 5.6% error rate at 500ms inputs. If the noise model was adapted too, the system was expected to be perform even better. However, it was not possible to obtain an adaption matrix for this noise model as the mean values of the model was too small. The system with the 2048-mixture noise model showed a small improvement when adapting the speech model, and the system was further slightly improved by adapting both the speech and noise model. However, the improvement was not as big as expected.

The difference in performance between the two noise models were not significant enough to choose one over the other. In general, if more data, especially keyboard noise, were obtained for training, adaption and testing, it is likely that the system performance would show clear improvements. There were a lack of adaption data, in particular for the noise model adaption, resulting in an adaption matrix that was not entirely optimal.

The system reaching 5.6% error rate at 500ms input is working well for offline use, but improvements would need to be made if either one of the systems were to be used in real-time. The results for short input sequences are currently not adequate as the error rate is too high for providing good accuracy and thereby a good user experience.

Sammendrag

Implementasjonen av kildelokalisering i MATLAB resulterte i tilfredsstillende resultater, hvor systemet klarte å bestemme hvilken retning de ulike lydkildene kom fra. Systemet fant imidlertid kun de sterkeste lydkildene i rommet, uten å ta hensyn til hvilken type lyd det var. På grunn av dette var det nødvendig å implementere klassifikasjon for å skille mellom tale og ikke-tale.

Metodene som ble brukt i denne avhandlingen var Mel-frekvens cepstralkoeffisienter til funksjonsekstraksjon, og Gaussiske blandingsmodeller til klassifisering. To ulike systemer ble utviklet for klassifikasjon, hvor hvert system inneholdt én talemodell med 512 blandingsfordelinger og én universell bakgrunnsmodell for støy. Den ene støymodellen brukte 2048 blandingsfordelinger, mens den andre brukte én. Ytelsen til systemene ble målt basert på log-sannsynligheten til utgangen av hver modell. Ytelsen til systemene viste seg å være dårlig når inngangen kun inneholdt én ramme på 25ms, hvor feilraten var på over 30%. Dess flere rammer som ble brukt på inngangen, dess høyere ble nøyaktigheten og dess mer robust ble verdien på terskelen. Begge systemene nådde 100% nøyaktighet når inngangen inneholdt opptil flere sekunder med data. Stillhet ble klassifisert som støy, noe som var optimalt ettersom det var uønsket at systemene skulle fokusere på stillhet.

Systemenes ytelse var forventet å forbedre seg ved å innføre modelltilpasning. Systemet med kun én blandingsfordeling i støymodellen viste positive resultater når talemodellen ble tilpasset, med en gjennomsnittlig reduksjon på 8.6% i feilraten og en minimal feilrate på 5.6% ved 500ms på inngangen. Det viste seg å ikke være mulig å adaptere støymodellen til dette systemet ettersom middelverdiene til modellen var for små. For systemet med en støymodell med 2048 blandingsfordelinger viste tilpasningen av talemodellen en liten forbedring av systemet, mens tilpasning av både tale- og støymodell resulterte i et enda litt bedre system. Til tross for dette var ikke forbedringene like store som forventet.

Forskjellen i ytelse mellom de to støymodellene var ikke stor nok til å velge en over den andre. Hvis mer data, spesielt støy fra tastatur, ble innhentet til trening, tilpasning og testing, er det sannsynlig at systemenes ytelse ville blitt tydelig forbedret. Det var mangel på tilpasningsdata, spesielt for støymodell, noe som resulterte i at tilpasningsmatrisen ikke ble helt optimal. Systemet som nådde 5.6% feilrate ved 500ms på inngangen vil fungere godt til frakoblet bruk, men forbedringer i systemenes ytelse måtte blitt oppnådd før de kunne blitt brukt i sanntid. Resultatene for korte inngangssekvenser er ikke tilstrekkelig gode nok slik systemene er nå ettersom feilraten er for høy til å gi god nøyaktighet og en god brukeropplevelse.

Contents

List of Figures

List of Tables

Acronyms

AGC Automatic Gain Control
AIC Akaike Information Criterion
ANN Artificial Neural Network
CMN Cepstral Mean Normalization
DAT Digital Audio Tape
DET Detection Error Tradeoff
DFT Discrete Fourier Transform
DTW Dynamic Time Warping
EER Equal Error Rate
EM Expectation Maximization
FA False Acceptance
FAR False Acceptance Rate
FFT Fast Fourier Transform
FR False Rejection
FRR False Rejection Rate
GMM Gaussian Mixture Model
HMM Hidden Markov Model
LRT Likelihood Ratio Test
MAP Maximum A Posteriori
MFCC Mel-Frequency Cepstral Coefficients
MLLR Maximum Likelihood Liner Regression
RL Relative Likelihood
SNR Signal-to-Noise Ratio
STFT Short-Time Fourier Transform
UBM Universal Background Model
XML Extensible Markup Language

Chapter 1

Introduction

1.1 Background and Motivation

This thesis is focused on the UniSound 64-channel microphone array. In various contexts, it is beneficial with microphone systems that automatically align with the speaker. The easiest way to solve this is to focus on the strongest source. However, this may have unwanted consequences if noise dominates in certain periods and is amplified as a result of this. It would be beneficial for a microphone system to be able to neglect unwanted noises, such as air conditioning noise or noise from keyboards.

As for today, the 64 channels in the microphone array are processed and comes out as one channel. However, the processing of the 64 channels is volume-based, meaning that the array focuses its attention to where the strongest energy source is located without any concern to what makes the sound, whether it being speech or non-speech. As the microphone array studied in this thesis is primarily made for offices and lecture rooms, this is a big concern as there is often background noises occurring in these rooms, such as clinking of coffee cups or clicking on a keyboard.

If the microphone system would be able to locate the strongest sound sources in the room, and thereafter choosing to focus on one of them if it recognizes it to be speech, it would improve the user experience of the system significantly.

1.2 Scope and Focus

Trying to recognize speech in noisy environments, as well as tracking an object in a room based on sound, are problems that have been researched for many years and that are still hot topics today. The problem of this thesis can be divided into *source localization* and *speech recognition*. Even though the problems are connected, it has been necessary to choose which one should be in focus. The main focus has therefor been chosen to be on the speech recognition problem.

Many different approaches have been proposed for recognition of speech in noisy environments, such as Hidden Markov Models (HMM), Dynamic Time Warping (DTW), Artificial Neural Networks (ANN) and Gaussian Mixture Models (GMM) [1]. DTW, ANNs and GMMs are used for classification, based on feature extraction from other methods, such as Mel-Frequency Cepstral Coefficient (MFCCs) or Short-Time Fourier Transform (STFT). This thesis is not so comprehensive that it allows several different methods to be examined. Due to this, the thesis will be focusing on one of the most popular methods for feature extraction and for classification, namely MFCCs and GMMs. MATLAB will be used as the main tool for this thesis, both for source localization and classification.

1.3 Outline

The thesis is structured as explained below, where basic information about the problem and methods is explained before going into the implementation of the source localization and classification, and then ending with results, discussion and a conclusion.

Chapter 2. The Microphone Array: The thesis will start out with an introduction of the microphone array to form a basis of the problem which is faced in this thesis.

Chapter 3. Theoretical Background: This chapter forms a basis of the theory that is useful to know in order to understand the focus of the thesis. It contains explanations to methods and terms that are explored throughout the work of this thesis.

Chapter 4. Database: In this chapter, there will be a description of the recordings that are obtained from the microphone array, as well as external databases that have been used both for speech and for noise. The chapter will explain the details of how the recordings have been obtained.

Chapter 5. Implementation: This chapter contains a description of the tools and methods that have been used to implement a proposed solution to the problem of the thesis.

Chapter 6. Experiments and Results: Contains the results that have been obtained after implementation. The results will be explained and discussed in detail together with plots and tables in this chapter.

Chapter 7. Conclusion: This chapter draws a conclusion based on what has been found and discussed previously in the thesis.

Chapter 8. Future Work: The chapter suggests work that can be done in the future, as a continuation of the work that has been provided in this thesis.

Chapter 2

The Microphone Array

The microphone array that is used for this thesis has dimensions 60x60cm. The area of the array containing microphones has dimensions 52.5x52.5cm. The dimension of the array is the same as the regular size of a ceiling tile. The idea is that a ceiling tile can be switched with a microphone array and that way not taking up any space in the room. There are 64 microphones distributed across the array in a specific pattern, as displayed in figure 2.1, where each blue dot represents a microphone element.

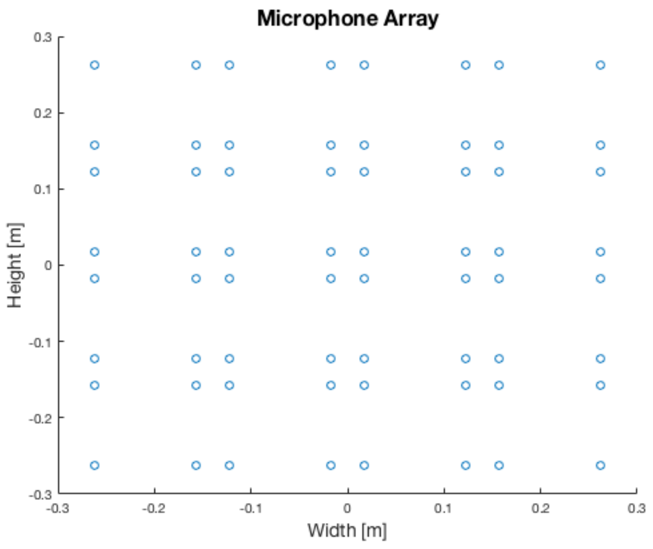


Figure 2.1: An illustration of the positioning of the microphones on the array

As can be seen in the figure, the microphone array has non-uniformly distributed elements. However, there is a clear pattern to the distribution of the elements, where the horizontal and vertical spacing between two adjacent elements can have two different values. The largest horizontal or vertical spacing between two adjacent elements in the array is 10.5cm, while the smallest corresponding spacing is 3.5cm.

The array sends the audio information from the 64 microphones over Ethernet, and passes it through different types of audio processing steps before it reaches the loudspeaker or recording device as a one-channel output. Figure 2.2 gives a detailed explanation of how the array operates.

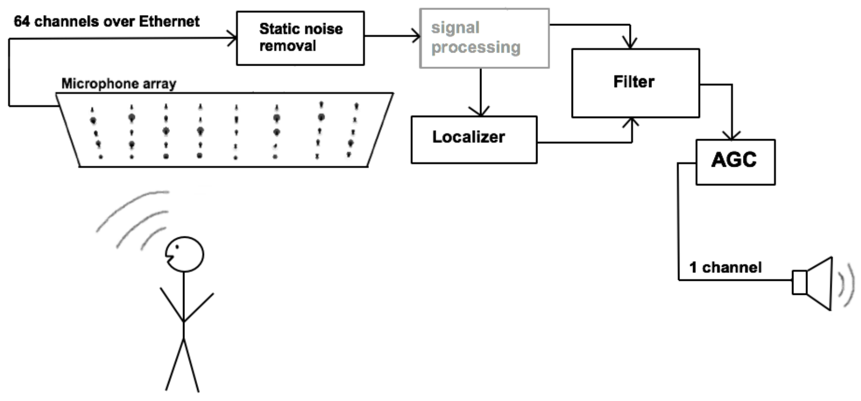
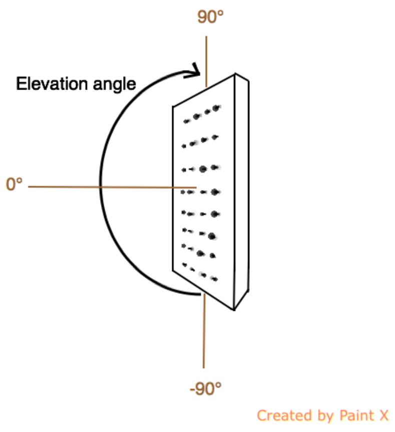


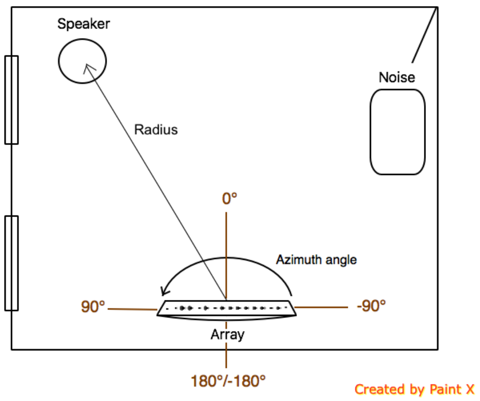
Figure 2.2: Display of how the microphone array works

The 64 channels will first pass a static noise removal filter, which, as the name implies, has the objective to remove static noise. This thesis will be focusing on the *signal processing*-part of the block scheme, where there can hopefully be added useful techniques for focusing the system on speech instead of noise. After this step, the data reaches the localizer and the filter. Here, beamforming techniques are used together with information about the predicted location of the sound source to concatenate the 64 channels into one. The single channel audio is then sent to the Automatic Gain Control (AGC), which provides a controlled and adjusted signal amplitude at its output. From there, the audio can be played through a speaker.

A way of describing the coordinates of the array is by the spherical coordinate system. These coordinates use the elevation angle, azimuth angle and radius to explain where in a room an object is located. This coordinate system will be used further in this thesis when it comes to positions in a room. Figure 2.3 illustrates how this coordinate system is related to the microphone array.



(a) Elevation angle, seen from side.



(b) Azimuth angle and radius, from above.

Figure 2.3: Spherical coordinates of the microphone array

From the input of the different microphones of the array, it should be possible to locate a sound sources in the room by the spherical coordinate system. In the figures above, the microphone array is place in the side so that is looks straight at the speaker and noise sources, instead of being placed in the ceiling. The microphone array is not yet on the market, as it is currently under further development.

Chapter 3

Theoretical Background

The way that a microphone array is structured can be used for multiple causes. The most basic problems that can be solved with a microphone array are to find the number and location of energy-radiating sources, enhancement of the signal-to-noise ratio (SNR), as well as tracking of moving sources. This chapter will provide useful theory to understand the implementation and results of the thesis, such as a theoretical background of source localization, creation of speech models and universal background models, as well as model adaption.

3.1 Source Localization

The first step in the thesis' problem is to locate the sound sources in the room. By identifying the sound sources, it is further possible to do a classification to determine which sources are speech and which are non-speech. In order to detect the positions of the sounds sources, it is necessary to have some theoretical background on the subject which will be explained in the next sections.

3.1.1 Beamforming

Array processing is the processing of the information that is obtained by the microphones of an array. It is an overall term that includes several estimation techniques. Beamforming techniques are spectral-based estimation techniques of array processing [2]. The microphone array in this thesis uses a delay-sum beamforming technique.

Beamforming techniques are algorithms for determining the complex sensor weights w_n(f), which are used to implement a desired shaping and steering of the array directivity pattern. The simplest of all beamforming techniques is known as delay-sum beamforming. With this technique, the time domain sensor inputs are delayed by τ_n seconds before they are summed to provide a single array output [3]. An illustration of this technique is shown in figure 3.1.

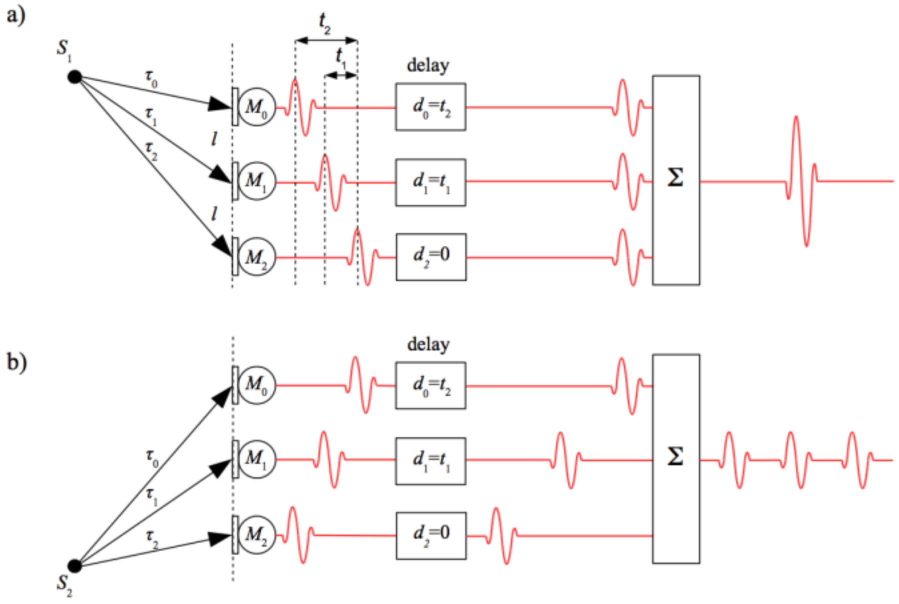


Figure 3.1: Illustration of the delay-sum beamforming method [4]

As figure 3.1 illustrates, delay-sum can be used to merge several audio signals into one enhanced audio signal. The technique can also be used to merge the audio signals into one, consisting of each of the original audio signals in a sequence. The delay-sum is a fixed beamforming technique, meaning that the parameters of the array are fixed during operation. The technique should be used for a broadside array configuration and is operating under optimal noise conditions when there is incoherent noise [5]. A source that is placed perpendicular to the aperture axis, is referred to as a broadside source. Incoherent noise is known to be spatially white, and in the case of microphone arrays, electrical noise in the microphones is mostly randomly distributed. The noise can therefore be considered to be a source of incoherent noise.

3.1.2 Grating Lobes

When the separation between each microphone element in an array is too large, there can occur grating lobes. Grating lobes are known as unintended beams of radiation. The occurrence of these lobes can be eliminated in uniformly spaced arrays by decreasing the separation between the elements. For non-uniform arrays, like the one in this thesis, it can be more challenging to predict grating lobes. Equation 3.1 shows a common formula used to calculate the maximum allowed spacing between elements, in order to eliminate grating lobes [6].

$$\Delta = \lambda/2 \tag{3.1}$$

The formula cannot guarantee that all grating lobes will be eliminated, but it provides a good approach. The wavelength λ equals the speed of light divided by the wave frequency. An illustration of typical grating lobes can be seen in figure 3.2. Depending on the spacing between the elements, as well as the relative phase between them, the grating lobes will move in or out of the visible region [7].

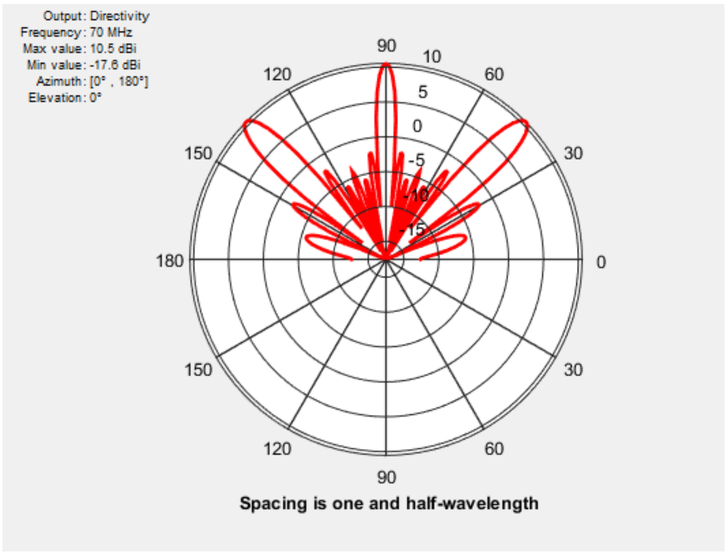


Figure 3.2: Visible grating lobes at 1.5 wavelength spacing [6]

In the figure, there are two large grating lobes at 45°and 135°azimuth. If the spacing between the microphones for this example would be decreased, these grating lobes would also be reduced and eventually disappear.

3.2 Speech Model

The next step after source localization is to classify the sound sources in a room as either speech or non-speech. To make a microphone focus on the speaker can be done by detection of speech. There are several ways that this can be done. However, the methods that are discussed in this section are among the most common ones in the area of speech recognition. The human vocal system is only able to produce a limited amount of phonemes. A well-trained speech recognition system would therefore only need to recognize each of these phonemes [1].

3.2.1 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCC) are coefficients that make up a representation of the short-term power spectrum of a sound. The spectral features are commonly used in speech applications due to its robustness [8]. Extracting the MFCCs from a speech signal is a common method in speech recognition. The MFCCs are then used as features in a speech recognition system. This feature extraction method was first mentioned in the 1970s, and it has been further developed since then [9]. MFCC mimics the logarithmic perception of loudness and pitch of the human auditory system. It also tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics [9]. The procedure to obtain the MFCCs is depicted in figure 3.3.

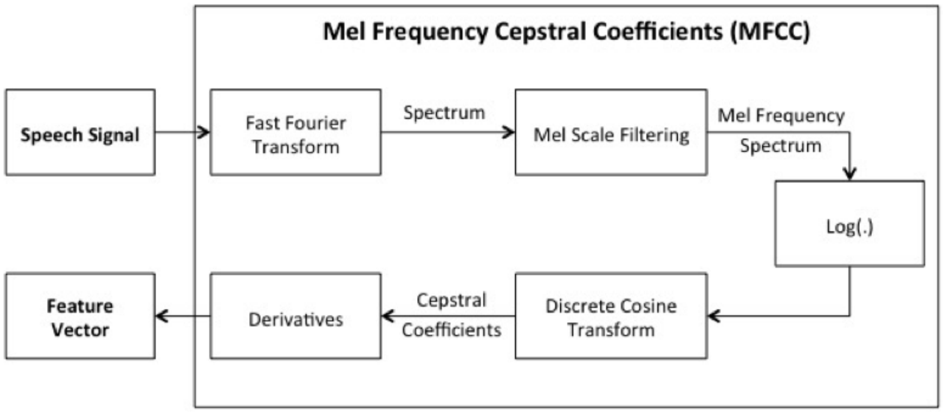


Figure 3.3: Block diagram of the MFCC algorithm [9].

At first, the audio signal is split into frames using Hamming windows, before a Fast

Fourier Transform (FFT) is performed on each frame. A frame is typically around 25ms long, as this length will represent one sound in regards to speech [1]. The FFT is a version of the Discrete Fourier Transform (DFT). MFCC feature vectors are calculated for each frame of speech because speech can be considered short-term stationary [8]. The formula for calculating the FFT of the speech signal is explained in equation 3.2, where N represents the number of samples inside a speech frame [10].

$$X[k] = \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-2i(\pi kn/N)}, \qquad k = 0, ..., N-1 \tag{3.2}$$

After the FFT is performed, triangular overlapping windows are used to map the powers of the spectrum onto Mel scale. The spectrum is filtered with different band-pass filters and the power of each frequency band is computed. The number and shape of the filter (rectangular, triangular etc.), as well as the center frequency, can be varied for performing this step [9]. Figure 3.4 illustrates a typical filterbank of 25 triangular band-pass filters used to compute the Mel-frequency spectrum. Typically, the window size is around 25ms and the overlap between successive window is 10ms.

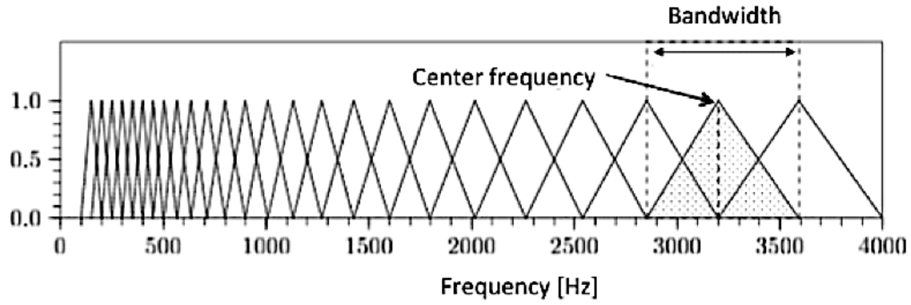


Figure 3.4: Filterbank with 25 triangular band-pass filters [9]

After the Mel-frequency spectrum have been obtained, the log of the powers of each of the Mel frequencies is taken. The reason for this is because the MFCC aims to mimic human hearing and humans perceive loudness on a logarithmic scale [9]. Thereafter, the Discrete Cosine Transform (DCT) of the resulting powers is conducted. The formula for calculating the DCT of the powers is explained in equation 3.3, where N denotes the length of the input vector [11].

$$X[k] = \frac{2}{N} \sum_{n=0}^{N-1} x[n] \cos\left[\frac{(2n+1)k\pi}{2N}\right] \qquad k = 1, ..., N-1 \tag{3.3}$$

At last, the MFCCs are found as the amplitudes of the resulting spectrum. The MFCCs that are extracted from the audio signal can be used to train a Gaussian mixture model, which is explained in detail in subsection 3.2.3.

3.2.2 Cepstral Mean Normalization

It would be beneficial being able to remove any effects caused by the channel. Cepstral Mean Normalization (CMN) is a technique that is used for this purpose. Given an input signal x[n] and a channel impulse response h[n], the recorded signal, y[n], can be considered a linear convolution of the two, such as

$$y[n] = x[n] \star h[n]. \tag{3.4}$$

The next step in the CMN technique is taking the FFT of equation 3.4. This leads to the important convolution-multiplication equivalence [12]. Equation 3.5 denotes the resulting signal.

$$Y(f) = X(f) \cdot H(f) \tag{3.5}$$

The next step after this, is to perform the logarithm of the equation. In the cepstral domain, multiplication becomes addition, which is a very important property [12]. The expression then becomes

$$\log[Y(f)] = \log[X(f) \cdot H(f)] \tag{3.6}$$

$$\Rightarrow Y[q] = X[q] + H[q]. \tag{3.7}$$

When taking the difference between the original signal and the average over all samples, the resulting signal does not contain the channel interference. The resulting signal is denoted as D_i, where the mean subtraction is performed on the input signal. Equation 3.8 explains the steps for obtaining this result [12] [13].

$$\frac{1}{N}\sum_i Y_i[q] = H[q] + \frac{1}{N}\sum_i X_i[q] \tag{3.8a}$$

$$D_i[q] = Y_i[q] - \frac{1}{N}\sum Y_i[q] \tag{3.8b}$$

$$D_i[q] = H[q] + X_i[q] - \big(H[q] + \frac{1}{N}\sum_i X_i[q]\big) \tag{3.8c}$$

$$D_i[q] = X_i[q] - \frac{1}{N}\sum_i X_i[q] \tag{3.8d}$$

As the result in equation 3.8d implies, the advantage of obtaining the cepstral feature vector is that it is possible to remove the channel interference.

3.2.3 Gaussian Mixture Model

A Gaussian mixture model is a probabilistic model for representing normally distributed subgroups within an overall group. The method is known as an unsupervised learning method, as it does not require knowledge about which subgroup a data point belongs to. That way, the model is able to learn the subgroups automatically. GMMs are used for multiple purposes, including feature extraction from speech data to be used in speech recognition systems, and object tracking of multiple objects in a video sequence [14]. In a video sequence, the number of mixture components and their means are used to predict the location of an object at each frame.

Mixture models are used for multimodal data, i.e. data that have more than one peak in its distribution. Many distributions are unimodal, i.e. they only have one peak, and is modeled by a Gaussian distribution. One way to look at multimodal data is therefore to think of the data as multiple unimodal Gaussian distributions. GMMs maintain several of the computational and theoretical advantages of Gaussian models. This way, they are highly practical for modeling very large datasets efficiently [14].

A GMM is parameterized by the weights of the mixture component, as well as the means and variances/covariances of the component. The multivariate representation of the mixture model is explained in equation 3.9. Here, K represents the number of components, and the i^{th} component has mean vector $\vec{\mu}_i$ and covariance matrix $\vec{\Sigma}_i$. The weight of the i^{th} component is represented as ϕ_i.

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x}|\vec{\mu}_i, \Sigma_i) \tag{3.9a}$$

$$\mathcal{N}(\vec{x}|\vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\phi)^K|\Sigma_i|}} \exp\Big(-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)\Big) \tag{3.9b}$$

The mixture model has a constraint regarding the component weights. The total probability distribution needs to be normalized to 1, as explained in equation 3.10.

$$\sum_{i=1}^{K} \phi_i = 1 \tag{3.10}$$

When the number of components K is known, the Expectation Maximization (EM) algorithm is the most common technique for estimating the parameters of the GMM [14]. This technique is explained further in the following subsection.

3.2.4 Expectation Maximization algorithm

The EM algorithm is an iterative, numerical technique for estimating maximum likelihood. The maximum likelihood of the data strictly increases by each iteration, implying that it will always move towards a local maximum or a saddle point [14]. The results will give an implication of what the optimal number of parameters is for the GMM.

The procedure of the EM algorithm for mixture models consist of two steps: the Expectation (E) step and the Maximization (M) step. The algorithm starts with random initialization, before it moves on to the E step. The E step assigns points to the nearest cluster center, given the model parameters ϕ_i, μ_i and Σ [15]. The M step maximizes the expectations calculated in the E step with respect to the model parameters, i.e. it sets the cluster centers to the mean [14]. The steps repeats in an iterative manner until the algorithm converges and provides a maximum likelihood estimate. An illustration explaining the steps of the algorithm is shown in figure 3.5.

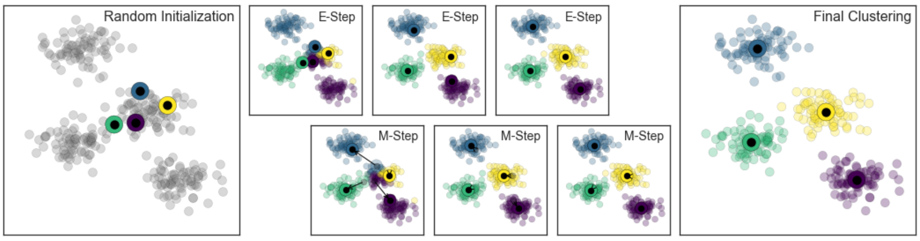


Figure 3.5: Illustration of the steps of the EM algorithm [15].

As can be seen from the figure, the EM algorithm for GMMs starts with an initialization where the model parameters are assigned values based on the data. Thereafter, the model iterates over the Expectation and Maximization steps until the estimates of the parameters converge.

It is possible to work with the algorithm even if the number of components, K, is unknown. Then, different values for K will be tested and the model with the best trade-off between number of components and fit will be chosen [14]. The steps of the algorithm can be expressed mathematically. The mathematical expressions for the E and M step for the multivariate case are rather complex. Hence, they are not included in this subsection but can be found in [14].

3.2.5 Akaike Information Criterion

The Akaike Information Criterion (AIC) is a criterion that is used to simplify model selection. The AIC value is calculated for each model, with the same data set. The model with the lowest AIC value, is considered the best model. The value of AIC is dependent on the data y, which leads to uncertainty in regards to model selection [16]. Equation 3.11 presents the mathematical expression for the AIC.

$$AIC = 2k - 2\ln\left(\mathscr{L}(\hat{\theta}|y)\right) \tag{3.11}$$

In the formula, k denotes the number of estimated parameters, $\mathscr{L}$ denotes the likelihood function, while $\hat{\theta}$ denotes the maximum likelihood estimate and y is the empirical data. By using this formula, it is possible to choose a reasonable value for the number of components to use when fitting a GMM to empirical data. However, the results are not necessarily unambiguous. Therefore, the relative likelihood criterion can be used, which is explained further in the next subsection.

3.2.6 Relative Likelihood

The Relative Likelihood (RL) criterion can be used to choose the number of components when fitting a GMM to data. The formula for the RL criterion is relying on the AIC values, and is denoted as

$$RL_i = \exp(\frac{\text{AIC}_{min} - \text{AIC}_i)}{2}) \tag{3.12}$$

where RL_i denotes the relative likelihood for model i. It can be said that the relative likelihood for model i is proportional to the likelihood that model i is able to minimize the information loss [17]. The closer RL_i is to zero, the lower the probability is that model i can minimize the information loss. However, the closer RL_i is to 1, the higher the probability is that model i can minimize the information loss. If a model provides relative likelihood close to 1, the model should not be left out [17].

3.3 Universal Background Model

It may help to identify non-speech elements, to exclude it as speech. Using MFCCs is found to perform well also for detection of non-speech [1]. In order to detect non-speech, it is necessary to have a very large library of sounds, on the contrary to speech recognition. The complexity of non-speech sound recognition techniques increase because of the extremely large non-speech corpus that is needed [1]. However, in the scenario of a meeting room, it is possible to focus on certain sounds that occur very frequently. Sounds like the clink of coffee cups, air conditioning or keyboard clicking are likely to occur in a typical meeting room. Because of this, a possibility is to detect these commonly occurring sounds as non-speech, and thus simplifying and improving the speech detection in the room. It is important to use a computationally fast method for detection in this scenario, due to the sudden occurrences of these types of sounds.

A Universal Background Model (UBM) is a model that represents general feature characteristics. This model can be used for comparison with a model with specific feature characteristics. This section is mainly based on findings in [18]. A typical example of where a UBM is used is within speaker verification. Here, a model is trained for a specific person with specific feature characteristics, and a UBM is trained with speech samples from various speakers. A Likelihood Ratio Test (LRT) can be used to make a decision based on the score between the UBM and the speaker-specific model, where a speaker is then either accepted or rejected. The UBM can also be used in the case of separating speech from noise or non-speech, as seen in the following subsection.

3.3.1 Likelihood Ratio Test

The LRT is closely related to the RL criterion from subsection 3.2.6. The LRT is more complex, as it requires nested models, whereas the RL does not. Despite the complexity, they both have the same goal: evaluating which model provides the best likelihood for the input data [18].

The LRT can be used to determine whether a sound should be classified as speech or non-speech. This is done by forming two hypotheses, H_0 and H_1, based on a sound sample X:

$$\begin{aligned} H_0&: \text{ X is speech} \\ H_1&: \text{ X is non-speech} \end{aligned}$$

In general, the test is said to verify the null hypothesis, which in this case will be to verify that the sample is speech. This test is the optimum test to choose which of the two hypotheses is most likely, and is given by equation 3.13, where T denotes a threshold.

$$\frac{p(X|H_0)}{p(X|H_1)} = \begin{cases} \geq T, \text{ Accept } H_0 \\ \leq T, \text{ Reject } H_0 \end{cases} \tag{3.13}$$

Writing the test in a different form, equation 3.14 is obtained. By taking the logarithm of the probability, the division is replaced by subtraction. By summing over all samples, a single value is obtained, which can be compared to a threshold.

$$\text{LLR} = \sum_i \log(p(X_i|H_0)) - \sum_i \log(p(X_i|H_1)) \tag{3.14}$$

The threshold T should be optimized for best performance, depending on how vulnerable the scenario of wrong classification is. Section 3.4 goes deeper into selection of threshold and possible consequences.

3.3.2 Alternative hypothesis modeling

The model for H_0 is clearly defined and can be trained with speech samples denoted as P. The alternative hypothesis H_1, modeled by $\lambda_{\overline{P}}$, is not as clearly defined, as it might need to represent all possible alternatives of non-speech. From the area of speaker recognition, there are mainly two ways of defining the alternative hypothesis H_1. The first approach is to use a set of different models to represent the entire space of H_1. The alternative hypothesis can be represented as in equation 3.15, from a set of N background noise models $\{\lambda_1, \dots, \lambda_N\}$.

$$p(X|\lambda_{\overline{P}}) = \mathscr{F}\big(p(X|\lambda_1), \dots, p(X|\lambda_N)\big) \tag{3.15}$$

The function $\mathscr{F}()$ can be an averaging or maximum function of the likelihood values of the background noise set. Each model would represent individual background noises, such as having one model for keyboard noise, one for clinking of chinaware, one for door slamming etc. This can end up being a very complex way of modeling the alternative hypothesis if speech should be recognized from all types of noise. In the scenario of classifying speech from a small, specific number of noise sources, this can be a good representation of H_1.

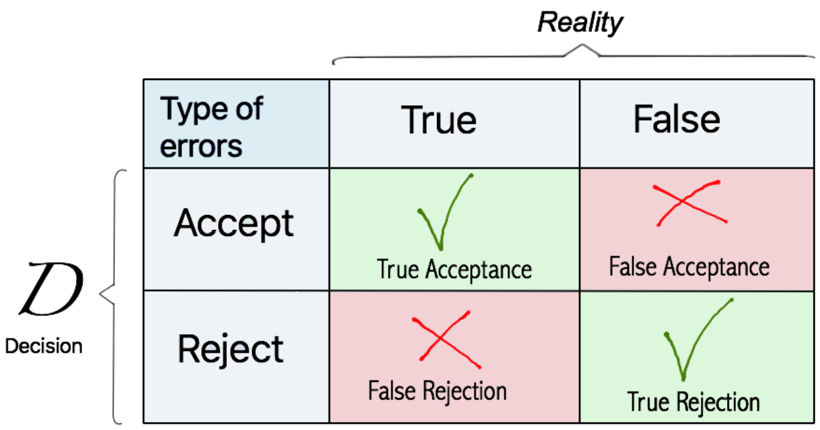
The second approach of defining the alternative hypothesis is to collect noise from multiple sources in order to form a collection of noise sources that can be used to train a single model. The single model is often referred to as a world model, a general model or a universal background model. This method creates a model that is supposed to represents all types of background noise. It is therefore important to chose noise sources with big variations in order to get a widespread representation. One way to represent this model is by a GMM with large variance. By choosing this approach, the model can be used for all cases, whereas with the first approach it is necessary to adapt the model for each scenario where it should be used.

3.4 Threshold

When performing classification, there are certain factors that need to be considered. These factors include choosing a suitable threshold for the system, as well as a level of significance, in order to reduce the amount of errors.

3.4.1 Choosing a threshold

When performing classification on a set of samples, there are two types of errors that can occur: *False Acceptance* (FA) and *False Rejection* (FR). False rejection means that what in reality is true, is classified as false, while false acceptance means that what in reality is false, is classified as true. This is illustrated in the scheme in figure 3.6.



Type of errors	True	False
Accept	✓ True Acceptance	✗ False Acceptance
Reject	✗ False Rejection	✓ True Rejection

Reality
D
Decision

Figure 3.6: Scheme of the four possible results of classification

This problem can be illustrated by many scenarios, such as an impostor trying to get authorization from a system, or a voice recognition system trying to detect speech. Based on the last scenario, the illustration in figure 3.7 explains how the threshold can bring out both types of errors.

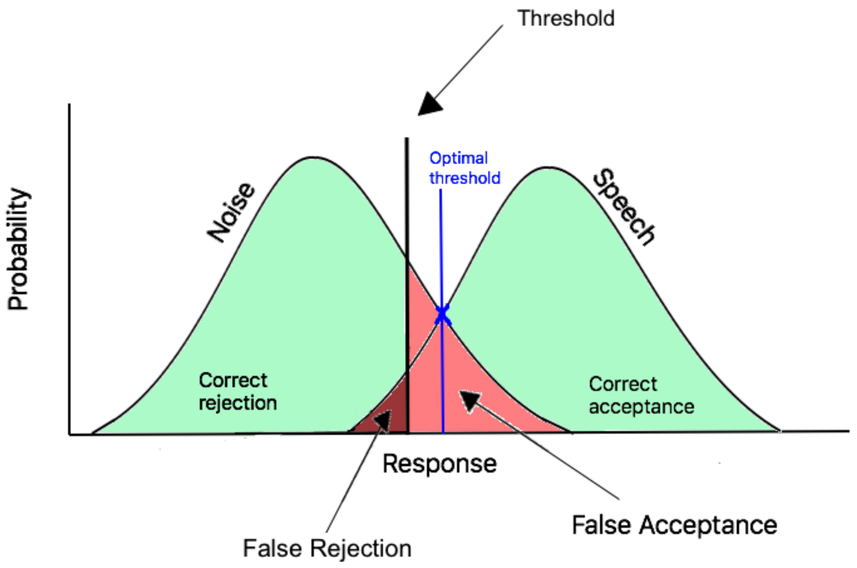


Figure 3.7: Threshold creating errors when classifying speech

Based on this example, the problem of choosing the threshold ends up being a question of how sensitive it is if either one of the errors occur. If the scenario is detection of speech, it may not be too big of a problem if FA occurs. However, if the scenario is that an impostor is trying to get authorization to a security system, it may be very dangerous if FA occurs. It is common to denote the errors by False Rejection Rate (FRR) and False Acceptance Rate (FAR), which is the rate in which the errors occur. Equal Error Rate (EER) is known as the error rate which equates to the point at which the FAR and FRR cross, namely a compromise between FAR and FRR [19]. This is often chosen as the optimal threshold, as seen in the figure above.

3.4.2 Statistical significance

This subsection is based on what is found in [13], where it is stated that statistical significance is important when dealing with the confidence of statistical inference, such as knowing if the estimation of a parameter can be accepted with confidence. In pattern recognition, such as recognition of speech, significance testing is highly important for deciding whether the difference between two classifiers is real. It is highly important to compare the performance of the classifiers, and by that choosing the optimal classifier. One common approach is to test two classifiers on the same test samples and determine whether one classifier is better than the other. The difference in performance between the two classifiers needs to be statistically significant in order to choose one over the other.

Significance testing is one of the most important methods of statistical inference. Similar to the LRT, significance testing consists of a null-hypothesis, H_0, and an alternative hypothesis, H_1. Also here, the aim is to either accept or reject the null-hypothesis. The objective is to find the probability of the test rejecting H_0. The upper bound α is known as the level of significance and specifies the the upper bound for false rejection. The greater α is, the more likely the test is to reject H_0. In other words, the aim is to have α as low as possible. Once a hypothesis is rejected by a test, the decision can be considered 'correct' with (1-α) confidence. The most common value for α is 0.05. It is then said that a test is carried out with 0.05 level of significance, or 0.95 level of confidence.

3.5 Training, validation and testing

In the case of creating a speech model, a database made up of speech samples is essential. The database can be divided into training, test and validation sets for processing. The training set should be significantly larger than both the test set and the validation set. Figure 3.8 illustrates the division of a dataset.

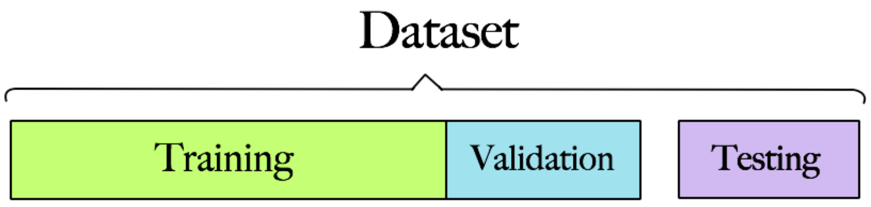


Figure 3.8: How the dataset is divided into 3 individual sets

The training set contains a set of input data used for learning, where the aim is to fit the parameters of the speech model. Based on this input data, the speech model should be trained to recognize speech. For a general speech model, it is important that the training set contains a broad variation of speech samples, with variations in regards to age, gender, voice and dialect of the different speakers.

The validation set is a set of input data used for adjusting of the parameters of the speech model [1]. The model that performs best with the validation set is usually the one that is chosen as the final one. The test set on the other hand, is not used for creating or tuning the speech model. The test set is a set of input data used for testing the performance of the speech model. The test set does not provide any expected output, but it rather shows how accurate the model is at classification.

3.6 Cross validation

In order to make sure that a model has got most of the patterns from the input data correct, performing cross validation is a common approach. For the model to be stable, it should be low on bias and variance. The model might be underfitting or overfitting the input data. According to [20], overfitting is when the system learns features or correlations that are specific to the training data, making the test data inapplicable. The training data is then modeled too well, resulting in poor performance for the system, as the model is not generalized for different types of test data. Underfitting is the opposite, where the system is too generalized, resulting in poor performance as system has not learned the important features of the training data. The aim of cross validation is to give an indication of how well the trained model will generalize to unknown data [13].

In [13], the **Holdout Method**, or **H Method**, is explained as using some of the data samples for training and the rest for testing. Based on error estimation, it is possible to known how the model is performing on unknown data or the validation set. This is the most basic approach of cross validation and it is easy to implement. However, the approach is known to suffer from issues of high variance as the results are very dependent on which data that gets included in the validation set.

Another approach that is described in [13] is the **V Fold cross validation**. This cross validation approach involves dividing the dataset into V equal parts, or subsets. Thereafter, the holdout method is repeated V times, where each time, one of the subsets is used as the validation set, while the other subsets are used for training. The error estimation is then averaged over all the V trials. That way, all the data appears in the validation set once and in the training set V-1 times. By using this approach, bias is significantly reduced as the training set is larger, and the variance is reduced since all the data is at some point used in the validation set. The idea behind this approach is to have as much training data as possible, as there can never be enough, and removing parts of the data for validation can cause underfitting.

3.7 Model adaption using MLLR

The idea behind model adaption is to adapt the parameters of a model to better match the input data. There are different approaches to do so, such as Maximum A Posteriori (MAP) adaption of HMM or GMM parameters, and Maximum Likelihood Linear Regression (MLLR) of GMM parameters [21]. In this section, the focus will be on model adaption using MLLR of GMM parameters.

MLLR uses linear transformation of Gaussian model parameters to adapt to test data. This is done by calculating a transformation of a given model's parameters to maximize the observations' likelihood [22]. New, adapted parameters, such as mean vectors μ, are calculated by applying MLLR. There are different approaches to the implementation. One approach is based on [21], where the mean vectors are given by

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_m &= \mathbf{A}\mu_m + \mathbf{b} && (3.16) \\ &= \mathbf{W}\tilde{\boldsymbol{\mu}}_m && (3.17) \end{aligned}$$

where $\mathbf{A}$ is a matrix, $\mathbf{b}$ is a bias vector, $\mathbf{W}$ denotes the transformation matrix, m denotes the mixture component and $\tilde{\boldsymbol{\mu}}$ is given by

$$\tilde{\boldsymbol{\mu}} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_p \\ 1 \end{bmatrix}$$

According to [21], given an adoption utterance $\mathbf{O}$ and a given model $\mathbf{\Lambda}$, the maximum likelihood estimate of the transformation is given by

$$\hat{W} = \underset{\mathbf{W}}{\arg\max}\, p(\mathbf{O}|\mathbf{W}, \mathbf{\Lambda}). \tag{3.18}$$

One option is to make the transformation matrix diagonal, in order to have less parameters to estimate. A drawback of having diagonal transformation matrices is that the resulting transformation is limited when it comes to adaption capabilities [22].

Another option is to make the transformation matrix block diagonal. Most practical systems are based on diagonal covariance matrices, which will turn into a block diagonal matrix [21], denoted as

$$\begin{bmatrix} \mathbf{G}_1 & 0 & \dots & 0 & 0 \\ 0 & \mathbf{G}_2 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \mathbf{G}_{p-1} & 0 \\ 0 & 0 & \dots & 0 & \mathbf{G}_p \end{bmatrix}$$

where G_i is denotes as

$$\mathbf{G}_i = \sum_{m=1}^{M} \overline{\gamma}_m \sigma_m^{-2}(i) \tilde{\boldsymbol{\mu}}_m \tilde{\boldsymbol{\mu}}_m^T \tag{3.19}$$

and $\overline{\boldsymbol{\gamma_m}}$, $\gamma_{m,n}$ and $\overline{\boldsymbol{o}}_m$ are denoted as

$$\overline{\boldsymbol{\gamma_m}} = \sum_{n=1}^{N} \gamma_{m,n} \tag{3.20}$$

$$\gamma_{m,n} = \frac{\mathcal{N}(x_n;\mu_m,\Sigma_m)c_m}{\sum_{l=1}^{M}\mathcal{N}(x_n;\mu_l,\Sigma_l)c_l} \tag{3.21}$$

$$\overline{\boldsymbol{o}}_m = \sum_{n=1}^{N} \gamma_{m,n}\boldsymbol{o}_n. \tag{3.22}$$

In the equations above, $\sigma_m(i)$ is the *i*th diagonal element of the covariance matrix of mixture component *m*, while $\overline{\boldsymbol{o}}_m(i)$ is the *i*th element of the cumulative observation vector corresponding to mixture *m*. The observation vector normally includes three partitions: MFCC features, and their first and second derivative [22]. Due to this, the block diagonal transform normally consists of three corresponding G_i. If it is decided to use full transformation matrices instead, the computational cost and statistics needed for estimation are maximal.

By using the block diagonal transform given above, it is possible to use equation 3.23 to solve *p* sets of *p+1* equations in *p+1* unknowns, as

$$\mathbf{G}_i\overline{\mathbf{w}}_i = \mathbf{z}_i. \tag{3.23}$$

In the equation above, $\overline{\boldsymbol{w}}_i$ represents the *i*th row of the transformation matrix, **W**, while $\mathbf{z}_i$ is given as

$$\mathbf{z}_i = \sum_{m=1}^{M} \sigma_m^{-2}(i)\overline{\mathbf{o}}_m(i)\tilde{\boldsymbol{\mu}}_m. \tag{3.24}$$

Based on the system of equations given above, it is possible to implement an iterative algorithm in order to find a suitable adaption matrix **W**.

Chapter 4

Database

The complete database that has been used in this thesis consists of different types of recordings of both speech and noise, as listed below:

- Recordings of speech, noise and silence from the UniSound microphone array
- Speech database of Norwegian language [23]
- Database of non-speech sounds [24]

The methods that were tested in this thesis are based on the available data. Even though it in some cases could be beneficial to have other types of data than the sources listed in this chapter, it has been necessary to use what was available, as it would be time-consuming to record a large dataset of sound samples.

4.1 Array Recordings

The microphone array database contained six recordings from the microphone array, each with a length of approximately two minutes. Both the processed single-channel output, as well as the raw unprocessed 64 channels were saved to the database. All of the recordings were obtained with the UniSound microphone array with a sampling rate of 24410Hz and 16 bits per sample.

The audio tests that were obtained found place in a room with a size of 14m^2. The room was covered in thick, sound absorbing blankets on all four walls, as well as in the ceiling. The array was placed on a chair along one of the walls, with the microphone elements on the microphone array facing the wall of the opposite side of the room. The person talking was placed in the corner on the opposite side of the room from the microphone array, while three different noise sources were placed in a specific location away from both the array and the speaker. An illustration of the setup in the room can be seen in figure 4.1.

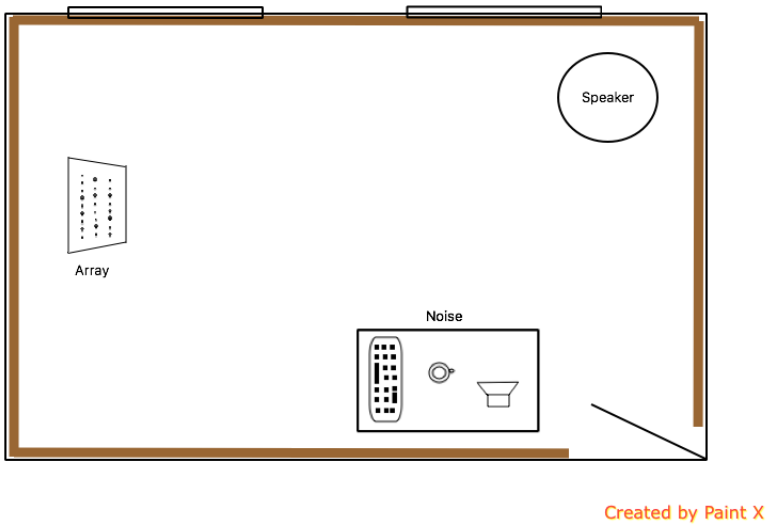


Figure 4.1: An illustration of the test setup, seen from above.

The brown colored lines in the figure illustrate the sound absorbing blankets. There were two windows and one door in the room. The microphone array was in an upright position, facing the wall of the other side of the room. As there are endless amounts of different noise sources that could be present in a meeting room, it was convenient to choose a few typical noise sources to focus on. The noise sources that were chosen were the following:

- Keyboard noise
- Noise from coffee cup
- Fan noise (recording played from smartphone)

These noise sources were chosen both because they are commonly occurring in a meeting room, but also because they are easy to reproduce in test recordings. There were three different speakers in the recordings, as listed in table 4.1.

Table 4.1: Speakers in the array recordings

Gender	Dialect/Region	Age group [years]	Length [min]
Female	Rogaland	20-30	4
Male	Trøndelag	20-30	4
Male	Oslo	20-30	4

From the table above, it is seen that all three speaker are in the same age group. However, they all have different dialects and both genders are represented, which are important factors in building a speech database. In total, there exists almost 4 minutes of speech of each person through the microphone array. Ideally, there should be more different speakers and a bigger variety in age among them. However, this was not possible to obtain at the time.

4.2 Norwegian Speech Database

In addition to the self-produced recordings, a speech bank from Nasjonalbiblioteket [23] was used. This is an acoustic-phonetic speech bank of Norwegian speech. The module that was used for this thesis was one with manuscript-read speech from 240 different speakers, all with Norwegian as their mother tongue. The speech samples are in general less than 10 seconds, as they are one sentence utterances. In total, there are 4800 speech samples in the corpus, whereof 2159 of them are unique sentences. There are variations in age, gender and dialect between the samples, and the corpus is divided into a training set and a test set. The training and test set are independent of each other, with no overlap of speakers. Each speaker reads 20 sentences, whereof 3 are used for calibration, 5 are unique and 12 are read by 3 different speaker. Two different microphones were used to record the utterances:

- Headset microphone: Sennheiser HS 2-5-1
- Multi-Pattern Dual Diaphragm Microphone: Shure KSM44

All of the recordings should have a sampling rate of 48kHz with 16 bit per sample. However, some of them were recorded with a sampling rate of 44.1kHz by mistake. Those are neglected in this thesis. The producers of the speech bank used a greedy search algorithm to find a suitable amount of sentences from a large corpus. The algorithm was set to include a large amount of different acoustic incidents, to get large variability between the recordings.

4.3 Non-Speech Database

The database of non-speech sounds, found in [24], consists of 105 types of sound sources. These recordings were obtained with a standard single microphone in an anechoic room, and are therefore known as dry source recordings since they are free of influences. The recordings are divided into three different categories:

- Collision sound source
- Action sound source
- Characteristic sound source

Collision sound sources are classified as sources caused by one-time collision of an object, such as dropping a hard object on the floor or clinking of china. Action sound sources are classified as a sounds that cannot be clearly specified by itself on the basis of the sound, but the sound source presents a characteristic tone. Examples of sounds like these can be clapping of hands or sawing. Characteristic sound sources are the sound source whose tone characteristically expresses the type of sound source, such as sounds from musical instruments or electronic sounds from toys or phones. All of the recordings have a sampling rate of 48kHz and 16 bit per sample and the database includes 9700 noise samples. The recordings of the sound sources were done with the following equipment:

- A standard microphone: B&K 4134
- A microphone amplifier filter: B&K 2636
- A DAT recorder: SONY DTC-77ES

Only parts of the database were used in this thesis, namely the ones that were more likely to occur in a meeting room. Table 4.2 displays the different sound sources that were chosen from the database, the average length of those files and the number of files that were used. In total, 904 non-speech audio samples were used.

The sound sources that were chosen, and the amount that was chosen, was related to the probability of occurrence in a meeting room. As an example, there are 133 audio files of chinaware while there are only 31 audio files of a bell, as clinking of chinaware is more likely to occur than ringing of a bell. The dataset is structured to include a wide variety of non-speech sound sources, but still with the focus on the most likely sound sources that could occur in a meeting room. One drawback is that the non-speech database in [24] does not contain clicking of a keyboard. As this is one of the sounds that is in focus in this thesis, it can be a source of error that this sound is not available nor included in the chosen database.

Table 4.2: Sound sources from the non-speech database [24]

Type of Sound	*Description*	*Average length*	*Amount*
Collision			
Chinaware	Using a wooden stick or a spoon to beat chinaware that is placed on a sound absorbing board	767ms	133
Coffee can	Beating a handheld metallic coffee can with a metal stick	468ms	41
Cola can	Beating a handheld metallic soda can with metal stick	543ms	41
Cup	Using a wooden stick or a spoon to beat a glass cup that is placed on a sound absorbing board	1796ms	82
Glass bottle	Using a wooden stick to beat a glass bottle that is placed on a sound absorbing board	567ms	41
Action			
Multiple claps	Multiple individual clap of hands	758ms	97
Single clap	A single clap of hand	430ms	97
Characteristic			
Bell	Ringing a suspended small bell by pulling the cord	1075ms	31
Clock	Ringing of a bell-alarm clock	500ms	41
Coin	Dropping a coin on a wooden board	1833ms	41
Paper Crumble	Crushing copy paper by hand	1796ms	71
Paper Tearing	Paper being teared apart by hand	1616ms	65
Phone1	Ringing of a home telephone	723ms	41
Phone2	Beep of a cellular phone	1798ms	41
Stapler	Stapling copy papers with a stapler	736ms	41

Chapter 5

Implementation

This chapter will provide explanations of how the work of this thesis has been implemented. The chapter will go through the implementation step by step, where each section and subsection builds upon the previous one. The software that has been used will be mentioned in the appropriate sections. The chapter will start off with the implementation of the source localization before it goes over to the implementation of the speech model.

5.1 Source Localization

For the implementation of source localization with the 64-element microphone array, MATLAB and its toolbox *Phased Array System Toolbox* was used [25]. The Phased Array System Toolbox uses the spherical coordinate system for direction. Figure 5.1 illustrates the setup that was used when obtaining recordings from the microphone array. As seen in the figure, azimuth represents the y-axis, elevation represents the z-axis and the radius represents the x-axis in this scenario. The square box illustrates the microphone array.

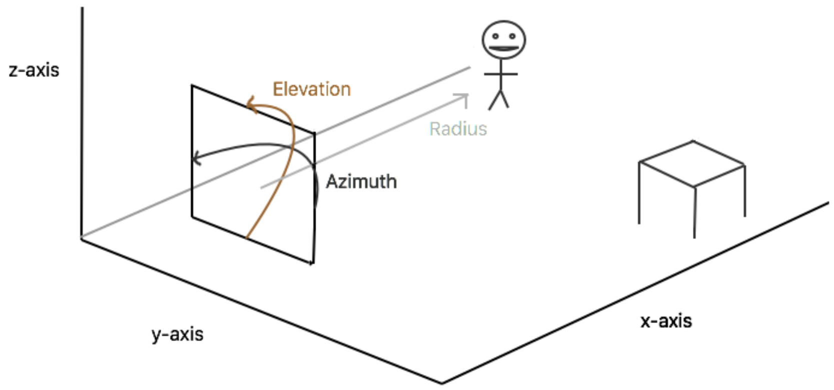


Figure 5.1: Illustration of the recording setup together with coordinates

The implementation started off by creating an object for the microphone element. This object, together with the positions of the microphones and the view angle in spherical coordinates, was then used to create the microphone array object, as encoded in the example below:

```
Mic = phased.OmnidirectionalMicrophoneElement('FrequencyRange', [0,fs/2]);

array = phased.ConformalArray('Element', Mic, 'ElementPosition', positions, ...
    'ElementNormal', [normal_az;  normal_el]);
```

Further, a beamformer object was created based on the array object, the sampling frequency, propagation speed and a chosen incident angle. The MATLAB script was sweeping through azimuth and elevation angles to cover many incident angles from the space in front of the microphone array. Each incident angle was used to create a beamformer object, which again was used on the original signal *ys*, as encoded in the example below:

```
InnVinkel = [azi(az+1); azi(el)];

beamformer = phased.TimeDelayBeamformer('SensorArray', array, ...
    'SampleRate', fs, 'PropagationSpeed', c, ...
    'Direction', Innvinkel);
Y = beamformer(ys);
```

The original signal *ys* represents a few seconds of a recording. That way, it could be possible to find whether those seconds contained speech, noise or both, and see if MATLAB was able to determine where the sound source was located.

5.2 Speech Model

Among several possible approaches for building a speech model, the results from [1] gave reason for implementing feature extraction by MFCCs and a GMM as a classifier, as these methods are found to perform well in speech and sound recognition. The implementation of the speech model that was used to recognize speech was done in MATLAB with help from the *Statistics and Machine Learning Toolbox* [26]. This section explains step by step how the model was implemented, where each step has its own subsection.

5.2.1 Training, validation and test set

The first step of implementing the speech model was to divide the samples of the speech database into a training, validation and test set. As the speech samples were already marked as training or test samples, the only set that needed to be created was the validation set. This set was created with a certain amount of training samples and an amount of test samples. Figure 5.2 shows the number of speech samples that was chosen for each set and how it was divided by gender, with 195 speech samples in total. Since the training set should be significantly larger than the two other sets, having a training set that was approximately three times larger than the other sets seemed reasonable.

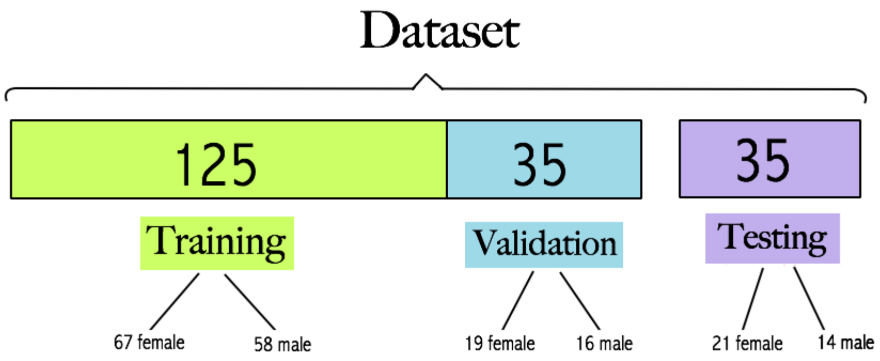


Figure 5.2: Number of speech samples in each set

Out of the 195 speech samples, 107 of them had female speakers and 88 had male speakers. Due to this, it was not possible to have equally many female and male speech samples in the sets. However, the difference between male and female speakers were attempted to be kept as little as possible.

5.2.2 Removal of silence

The Norwegian speech database contained an Extensible Markup Language (XML) file containing information about each file in the database. This included information about the time when the spoken sentence started and ended in each audio file. By using the Falkena's MATLAB function for converting XML-files into MATLAB structures [27], it was possible to access the data in the XML-file and use it to remove the silence at the beginning and end of every audio file. This way, the speech files used for training would contain exclusively speech and no silence. The silence between words were neglected as it was very brief and insignificant. Due to this procedure, the audio files were ready to be used for training a speech model, without training the model to recognize silence as speech.

5.2.3 Resampling

Resampling is the process of changing the sampling rate of an audio file. An audio file can be downsampled to reduce the sampling rate, or upsampled to increase it. Resampling is the general term for either one of these processes. The audio files from the microphone array had a sampling rate of 24410Hz, while the audio files from the speech and noise databases had a sampling rate of 48kHz. It is not possible to obtain information that is not available, as would be the case if upsampling the array recordings. However, it is possible to downsample the sound samples from the speech database without losing information or quality. The MATLAB function *resample()* was used instead of *downsample()*, when downsampling the samples, because *resample()* applies an antialiasing FIR lowpass filter to the signal and compensates for the delay introduced by the filter [28].

The audio samples from the speech database were downsampled to match the audio files from the microphone array. By resampling each audio file from 48kHz to 24410Hz, resampling with a factor of 4800/2441 seems like an clear choice. However, it is possible to simplify this to a factor of 2, since 4800/2441 = 1.9664. The difference between downsampling with a factor of 2 versus a factor of 1.9664 is insignificant and is therefore highly unlikely to affect the results in any way. This equals resampling the data at 0.5 times the original sampling rate.

5.2.4 Extraction of MFCCs

After the speech samples were downsampled to 24kHz, it was possible to use them for extraction of information. By using MATLAB code from [29], it was possible to extract the mel-frequency cepstral coefficients from each speech sample. The cepstral coefficients were extracted from each 25ms of the speech samples, which is an appropriate frame size as it normally contains one sound. The code returned 13 coefficients, namely $c_0,...,c_{12}$, where c_0 contained the information about the energy in the frame. After the cepstra had been extracted from each speech sample, cepstral mean normalization was performed on the cepstra. The cepstral coefficients of interest, namely $c_1,...c_{12}$, were saved in a matrix to be used in the next step.

5.2.5 Fitting data to a model

When the cepstral coefficients had been extracted from the speech samples, they were used to form a speech model. The function *fitgmdist()* from the *Statistical and Machine Learning Toolbox* was used to fit a Gaussian mixture model to data. By feeding the cepstral coefficients to the function, together with the number of components, a regularization value and a given termination tolerance value, it created a GMM, as implemented in the example below.

```
options = statset('TolFun', 1e-4);
GMModel = fitgmdist(cepstra, 512, 'CovarianceType', 'diagonal', ...
    'RegularizationValue', 0.001*min(var(cepstra)), 'Options', options);
```

The function *fitgmdist()* uses the EM algorithm to fit GMMs to data. When the object representing the GMM had been created, the function *pdf()* could be used to form a probability density function from the model and an input vector. This can be written in MATLAB as below, which corresponds to $p(X;\theta)$.

```
P = pdf(GMModel, Vcepstra);
```

In the example above, the object *GMModel* represents θ and the MFCCs from the validation set, *Vcepstra*, represents the input vector X. The last step in implementing the speech model was to determine the number of components to be used in the model.

5.2.6 Model selection

There are several different criteria that can be used for GMM model selection. One of the most popular is the AIC which can measure the goodness of fit of the GMM. As this is a common method to use, it was also used in the implementation of this speech model. The AIC was provided as a property of the fitted GMM object from using the function *fitgmdist()*.

Several different values were tested for the amount of mixtures of the GMM when fitting it to the data. It was desirable to get the lowest possible value for AIC to have the best model. However, the value of the AIC kept decreasing as the number of mixtures was increased. Due to this, it was necessary to choose a number of components that provided a low value for the AIC, but that was still withing certain limits, as it would be too computationally complex to have a number of components near infinity.

In [22], where the topic is speech enhancement and noise-robust speaker verification, 512 components are used when fitting a GMM to data. Due to this, it seemed reasonable to choose 512 components for the problem in this thesis, too. The model selection in [22] is not based on the AIC, but rather on the LRT.

5.3 Universal Background Model

The implementation of the UBM followed the same procedure as the implementation of the speech model, where feature extraction was done with MFCCs and a GMM was fitted to data. This approach is found to be a good method also for building a noise model, according to [1]. The implemented UBM represented a general noise model, based on the noise samples from table 4.2. The step by step implementation is explained in the following subsections.

5.3.1 Training, validation and test Set

The first step in creating the UBM was to divide the dataset of noise into a training set, a validation set and test set. Figure 5.3 illustrates the distribution of the noise samples in the different sets, in total 904 samples. As with the speech model, the sets were also here divided so that the training set was approximately three times larger than the other two sets.

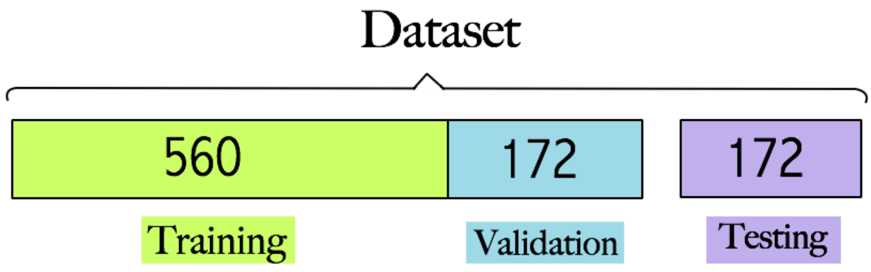


Figure 5.3: Distribution of noise files in the dataset

The dataset consisted of 15 different noise sources. There was an even distribution of the sound sources between the three sets. A drawback with the dataset was that it did not contain keyboard noise, which can be one the most common disturbing sounds in a meeting room, and also one of the sound sources that had been chosen as a focus for this thesis. The keyboard noise recorded by the array could therefore be a possible source of error.

5.3.2 Preprocessing of the Data

After the files had been distributed into three sets, they were preprocessed before the noise model was developed. The two steps of the preprocessing followed the block diagram in figure 5.4.

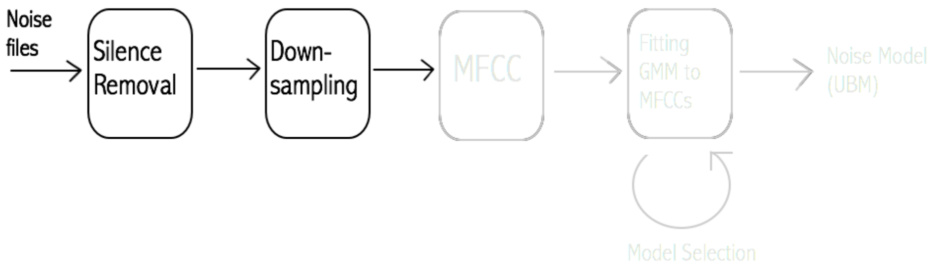


Figure 5.4: Block diagram highlighting the two preprocessing steps

The noise database did not contain information that could help remove the silence from the recordings, like the speech database did. The audio files contained little silence and it was therefore not as big of an issue as for the speech samples. However, the areas of the noise samples that had an amplitude below 10^{-4} were cut away to make sure the samples contained as little silence as possible. This cutaway value was chosen by inspecting several of the noise samples to find a suitable threshold between noise and silence.

As with the speech model, the audio samples from the noise database were downsampled to match the audio files from the microphone array. The audio files were downsampled with a factor of 2 for simplification, equal to 0.5 times the original sampling rate, resulting in a sampling rate of 24kHz.

5.3.3 Implementation of the Noise Model

The procedure of creating the noise model, or the UBM, was the same as with the speech model, where MATLAB's *Statistics and Machine Learning Toolbox* was used. The noise model was implemented according to the block diagram in figure 5.5.

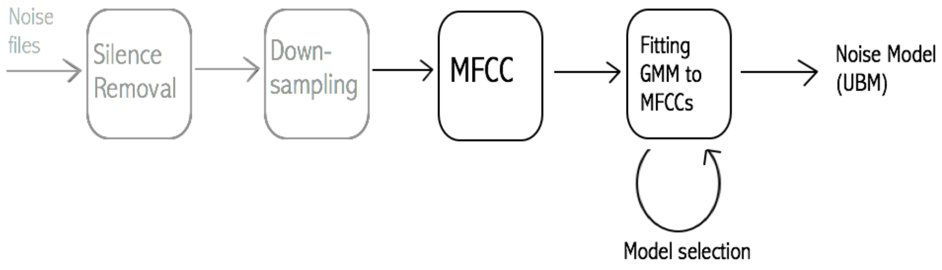


Figure 5.5: Block diagram highlighting the implementation steps of the noise model

The noise model was based on having a pool of different non-speech sound sources and training a GMM on that, rather than training a single model for each noise source. The reason for choosing this approach was due to complexity and convenience, as it was more time consuming, and required more data, to build a separate noise model for each sound.

The model was created by extracting the MFCCs from each 25ms-frame of each noise file, with 10ms step between successive windows. The function *fitgmdist()* was used to fit a GMM to the MFCCs from the training set. Two different noise models were chosen to be explored further, one GMM with 2048 mixtures and one GMM with one mixture. The reason for choosing these two models is discussed further in chapter 6.2.

5.4 Model Adaption using MLLR

Model adaption was the last problem to be implemented. It was implemented in order to adapt the parameters of the models to better match the input data, and in that way improve the overall system performance. The model adaption was conducted based on the equations in section 3.7. Figure 5.6 displays the iterative algorithm resulting from the equations which is used for obtaining the adaption matrix **W**.

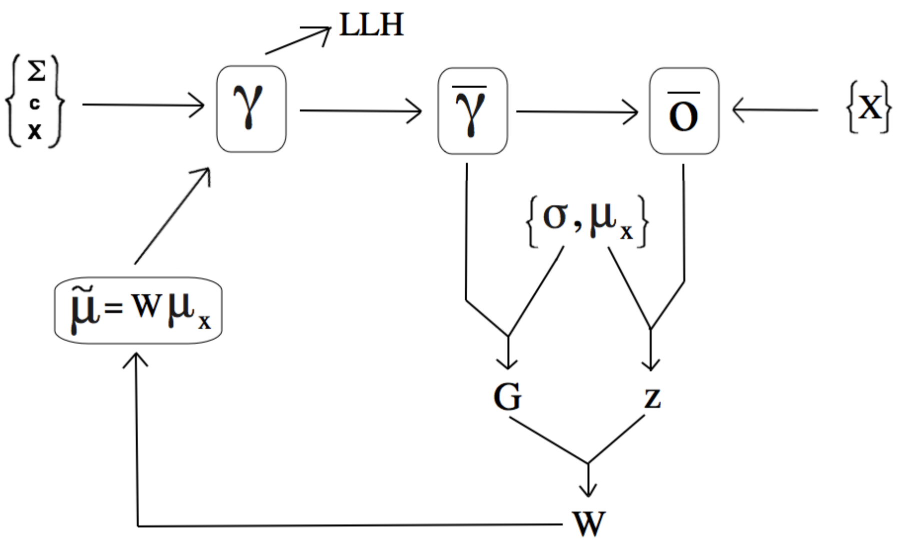


Figure 5.6: Diagram showing the iterative model adaption algorithm

In the algorithm, μ_x, Σ, σ and c are produced by either the speech model or the noise model, where μ_x is a vector of mean values with a 1 added to the end of the vector. X represents the cepstra that is produced by the speech or noise samples from external databases, while the adaption matrix W was initialized to W_{init} = [I; 0] for the first iteration.

The optimal adaption matrix was obtained by iterating through the algorithm until the maximum log likelihood was reached. The log likelihood was found by

$$\text{LLH} = \sum_{n=1}^{N} \log(\sum_{l=1}^{M} \mathcal{N}(x;\mu_l,\Sigma_l) * c_l)_n. \tag{5.1}$$

The adaption matrix W that followed from the iteration with the highest log likelihood was used to create the adapted model. The adapted model was implemented in MATLAB based on the *W* matrix and the original non-adapted model. An example where an adapted speech model is produced is described below, where *GMModel* represents the non-adapted speech model.

```
load('GMModel.mat'); load('W.mat');
muT = [GMModel.mu, ones(length(GMModel.mu),1)];

Adapted_SpeechModel = gmdistribution((W*muT)', GMModel.Sigma, ...
    GMModel.ComponentProportion);
```

The overall system performance is expected to be improved from adapting the models, compared to the non-adapted models. The EER is expected to be lowered when testing with speech and non-speech samples from the microphone array recordings. However, this is highly dependent on whether there is enough adaption data to obtain a suitable adaption matrix *W*.

Chapter 6

Experiments and Results

This chapter will present the different experiments that were conducted for this thesis, and the results that followed. In the beginning of this chapter, there is a section about source localization done with Matlab. Thereafter, the consecutive problem of classifying whether the detected sound source was speech or non-speech arises. The classification of the dominant sound source from the recordings have been chosen as the main focus, and will therefore be explored in more detail than source localization. It is necessary to take into account that the amount of data from the microphone array was limited. Due to this, the tests performed in this thesis can be used for indications, but more data would need to be obtained before making general statements about the methods.

6.1 Source Localization with MATLAB

In order to locate the sound source in the room, beamforming techniques from MATLAB was implemented. It was possible to plot the output to see where in the room the strongest sound source was located. The output was a sum of the energy in each direction. Several different recordings were tested to see how well it worked. The results are somewhat subjective as they are showing the direction of where the sound source is coming from, but not the exact position. Due to this, the results cannot be classified as strictly *correct* or *incorrect*, but rather *acceptable* or *not acceptable*. As an example, the plot in figure 6.1 shows the results from extracting and plotting the energy of 25ms of an array recording. This recording contains both speech and fan noise, and the axes are organized as in figure 5.1 from the previous section, where the x-axis now represents the energy value.

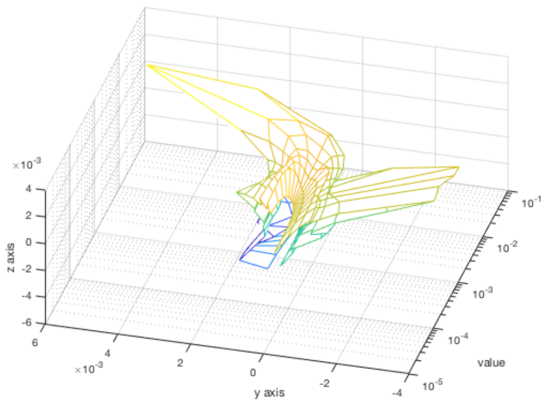
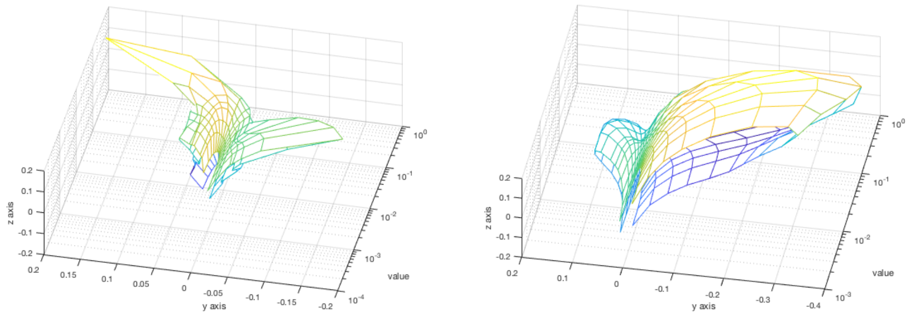


Figure 6.1: Plot of 25ms of speech and fan noise

From the figure, it can be seen that there are more energy to the left of the plot, namely at the positive values of the y-axis. By comparing to figure 5.1 it is found that this is the area where the speaker is located. The right side, where the y-axis is negative, is the side, or direction, where the noise sources are located. Due to this, it can be concluded that the microphone array is able to located two sources in the room, for this case being the speaker and the fan, whereas the speaker is the strongest sound source.

To explore this further, two different scenarios are plotted in figure 6.2. One of the scenarios includes a person speaking, while the other scenario includes noise from a coffee cup.



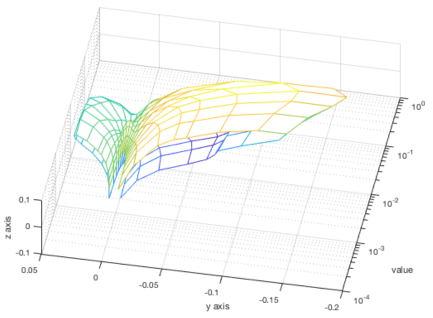
(a) Pure speech **(b)** Noise from coffee cup

Figure 6.2: Source localization of speech and coffee cup

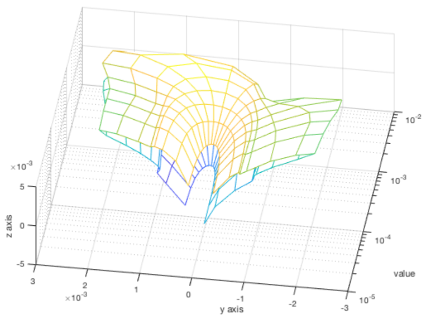
The person speaking and the noise sources were always placed apart, in order to distinguish them. The resulting plots were produced from 300ms of a recording in order to display the results properly. In some cases, 25ms was not enough to show a clear distinction, which is why 300ms were chosen in the upcoming plots.

The source localization in both scenarios of figure 6.2 clearly shows in which direction of the room the sound was coming from. Figure 6.2a detects the speaker as the strongest sound source, but it also detects some energy around the area of the noise sources. In all cases, there was a person standing next to the noise sources. Even though the recordings sometimes appear to be of only the speaker, the microphones might still detect some energy from the other person breathing or moving. This would affect the source localization, even though this person is "quiet".

Both figure 6.2a and figure 6.2b shows that the system is able to detect the strongest sound source, as well as some background noise from other areas. Having only one source makes the source localization a simple problem. However, when more than one source is present, the task might be more difficult. Another scenario that is therefore worth looking into is when both speech and noise occurs at the same time. In figure 6.3, two different plots are presented with speech and two noise sources. The plot in 6.3a represents the output when a coffee cup is making noise during the time a person is speaking. The plot in 6.3b represents a person speaking while someone is pressing a keyboard.



(a) Coffee cup and speech



(b) Keyboard and speech

Figure 6.3: Source localization with noise during speech

It can be seen from the plots that both the noise source and the speaker is perceived by the microphone array. However, the strongest source is shown to be the one with highest amplitude. The sound from a coffee cup hitting a table can be quite loud and draw attention away from the speaker, as seen in figure 6.3a. A keyboard, on the other hand, does not provide very much energy, which makes the speech the strongest source. The keyboard noise in figure 6.3b clearly confuses the source localization, as the localization is not as distinct as when only speech is present. Speech is nevertheless a stronger sound source than keyboard noise, which is also the conclusion from the figure.

The results of the implementation shows to be satisfactory, where the sound sources in the room are detected. There is still room for improvement, as the plots could be more specific in regards to exact location of the sound source. It becomes clear from the figures that when there are multiple sound sources in a room, it can become hard to choose which one to focus on. The source localization does not tell us anything about the type of sound that is detected, only the amount of energy that is detected for each incident angle. To build upon that, it would be useful to know which sound source is detected. The next sections will therefore explore the results of speech and noise classification. By combining source localization and speech/noise classification, a robust speaker detection system could be built.

6.2 Classification

The classifications step was the main focus of this thesis, where the aim was to distinguish between speech and noise in recordings done by the microphone array. This section will provide reasons for decisions that have been made, explore the experiments that was done, show results that were obtained and discuss the outcomes.

6.2.1 Choosing mixtures for the UBM

The noise model that was used in the classification system was a universal background model. This model should have a wide variance in order to be a general model that would pick up a wide variety of noise. It was therefore important that the model was not overfitting by learning only the specific noise sources used for training. Two models were explored, one that used one mixture and another that used 2048 mixtures.

Using a model with only one mixture would exclude the issue of overfitting and make the model very general. However, by training the model with 2048 mixture, a wider variance could be obtained and the data could potentially be fitted better to the model. The amount of 2048 mixtures were chosen based on findings in [18]. Figure 6.4 shows a comparison of the different noise models together with the speech model that was trained with 512 mixtures.

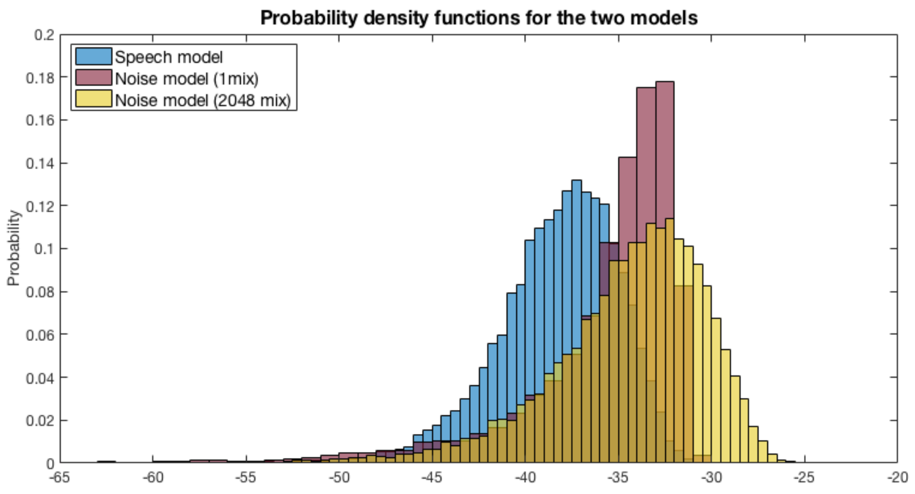
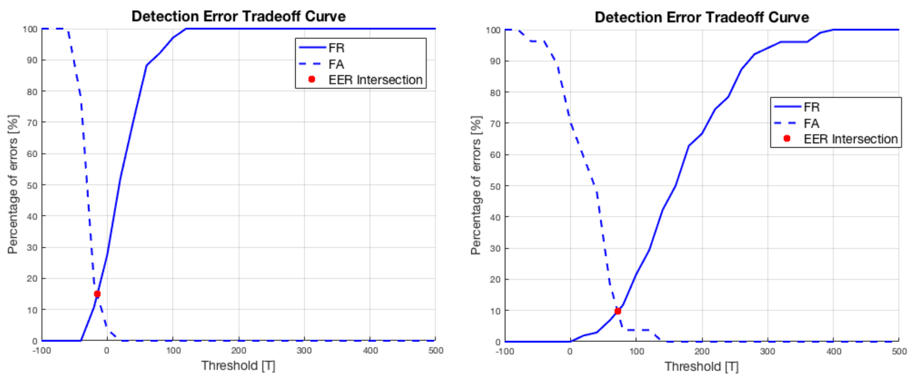


Figure 6.4: Comparison of the models

Figure 6.4 shows that using a UBM with 2048 mixtures, a greater variance is obtained compared to using one mixture. The noise models are shifted from the speech model, indicating that they are able to classify different input with little overlap. The two different UBMs were explored further in regards to performance. When feeding the models input from the microphone array of noise and speech, with length of 500ms, the results in figure 6.5 were obtained. The Detection Error Tradeoff (DET) curve compares the FFR and FAR and finds the EER intersection between the two rates, as well as the optimal threshold.



(a) GMM-UBM with 1 mixture **(b)** GMM-UBM with 2048 mixtures

Figure 6.5: Comparison of two different systems tested with 500ms input

It becomes clear from the DET curves in figure 6.5 that using a UBM with 2048 mixtures results in a more robust system. The reason for this is that the threshold decision is more sensitive for one mixture than for 2048. A slight shift in the threshold value for the 1-mixture UBM can result in large errors, either as FR or FA. A slight shift in the threshold value for the 2048-mixture UBM will also result in larger error rates, but not as large as for the 1-mixture UBM.

Overall, the UBM with 2048 mixtures shows to have a larger variance and to be more robust when it comes to the threshold decision, compared to the UBM with 1 mixture. However, as there are limited amounts of data, using a UBM with 1 mixture might result in better performance for the system, also because it eliminates the possibilities of overfitting. Due to this, both the UBMs were used for testing in this thesis and the results of both systems are compared later in this chapter.

6.2.2 Testing the system

The testing was done with sound samples from the array recordings as well as from the speech and non-speech databases. Table 6.1 explains the details of the test sets that were used. The table explains the amount of files in each test set and the length of each file in the test set. Each test set is numbered to easier explain which test set is used later in this chapter.

Table 6.1: Overview of the test sets that were used

Test Set No.	Sound Source	Source	Length [ms]	Amount
1	Speech	Array	500	102
2	Speech	Array	1500-5500	29
3	Noise	Array	500	27
4	Silence	Array	500-700	15
5	Speech	speech database	500	35
6	Noise	non-speech database	500	172

The test sets of the microphone array were made up of parts of the recordings, where only speech, noise or silence were present. There are two test sets of speech from the microphone array, namely test set 1 and test set 2. Test set 1 contains speech samples with a length of 500ms, while test set 2 contains longer speech samples with lengths between 1.5 and 5.5 seconds.

As it was known whether each sample was speech, noise or silence prior to the testing, it was possible to find the FAR and FRR when running the samples through the models. Each sample was tested as an input for both the noise model and the speech model, and the classification was done based on the difference in log likelihood between the two probabilities. If the difference in log likelihood exceeded the threshold *T*, the sample was classified as speech, otherwise it was classified as noise, according to the Likelihood Ratio Test in section 3.3.1. Table 6.2 shows the results that were obtained from the classification of speech and noise samples from the microphone array. In the table, *T* describes the threshold value at the EER intersection. Test set 1 was used for speech input, while test set 3 were used for noise input.

Table 6.2: Comparison of threshold values and equal error rates

Frames	**Length [ms]**	EER_1 [%]	EER_{2048} [%]	T_1	T_{2048}
1	25	32.4	43.2	-3	10
2	50	27.3	33.5	-4	9
3	75	31.4	29.7	-3	10
5	125	30.0	21.9	-2	17
8	200	26.6	16.2	-3	28
10	250	23.3	17.3	-5	35
15	375	20.2	14.9	-8	56
20	500	14.9	11.0	-15	65

It can be seen from the table that the threshold values for the 1-mixture UBM are rather stable over different amounts of frames, compared to the threshold values for the 2048-mixture UBM. The threshold is the point where FA and FR meets at the EER. As the threshold value is a value that can be chosen when a final, well functioning system is obtained, this chapter will not focus its attention towards finding an optimal threshold value, but rather focus on the EER.

The EER is high when only one frame of 25ms is used for classification, for both the UBMs. It is also seen that as more frames are used as input for the classification system, the lower the error rate becomes. The 1-mixture UBM performs better in regards to EER for one and two frame inputs, while the 2048-mixture UBM performs better for higher amounts of input frames. This is illustrated in figure 6.6.

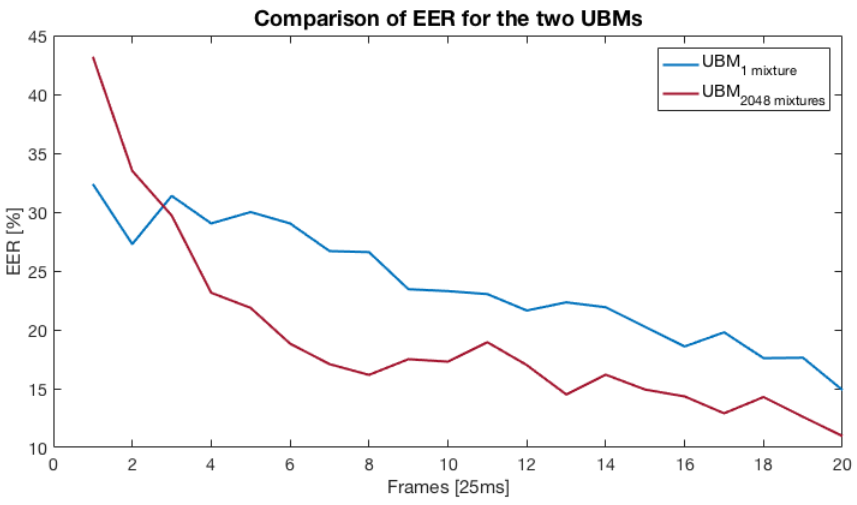


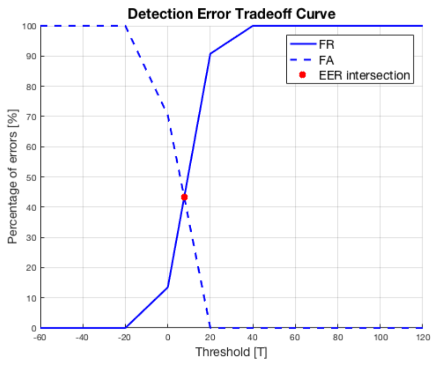
Figure 6.6: Comparison of the EER over frame input for the two systems

Figure 6.6, as well as table 6.2, show that the EER decreases more slowly when the number of frames become larger, for the system with the 2048-mixture UBM. Even though the system with the 2048-mixture UBM results in higher EER for one and two frame inputs, the EER is lower for all other amounts of input frames. The question of which model that should be used for classification relies on whether the system should work online or offline.

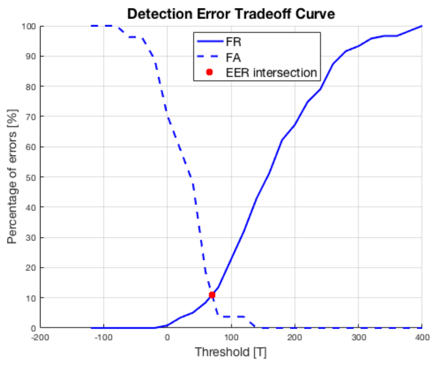
If the system is working online, namely in real-time, it is desirable to use few frame inputs in order to have a fast-responding system. The system should be able to classify speech as soon as someone starts to speak, thus, it should be able to make a decision based on a few frames only. This will naturally lead to a higher EER. By using many frames for online classification, the system will react more slowly to changes and thereby give a poor user experience. A balance between number of input frames and acceptable EER would then need to be chosen. If only one input frame were to be used, the system with the 1-mixture UBM should be chosen as this provides more than 10% lower EER than the system with the 2048-mixture UBM. Despite this, having a system with over 30% EER, like both the current systems have, is not a robust system and would not work well in real life.

If the system is working offline on the other hand, it is possible to use larger amounts of input frames. That way, the system has more data to rely on when making a classification decision, and the error rate will naturally drop. This is also seen in table 6.2, where the EER drops below 15% for both systems when 20 input frames are used, namely an input of 500ms. The length of the input is dependent on which types of noise source that are expected to be found.

From table 4.2 it is seen that clinking of chinaware has an average length of about 500ms, which indicates that using 500ms inputs should be enough to detect these types of sudden noises in an offline system. The noise from air conditioning or from clicking a keyboard can last from long periods of time, indicating that having input of a few seconds could be used to classify these noise sources. By comparing the DET curves for different amount of input frames for the system with the 2048-mixture UBM, figure 6.7 is obtained.



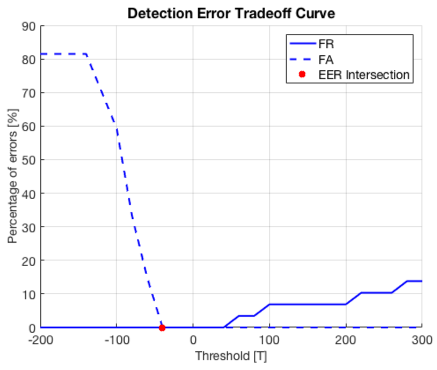
(a) 1 frame



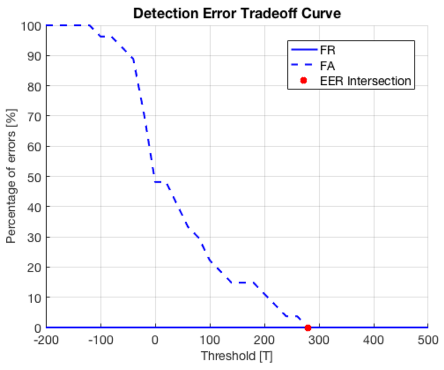
(b) 20 frames

Figure 6.7: Classification with different amounts of frames

The system is more robust the more frames that are used as input in the classification process. The more frames that are used, the more stable the threshold decision is. As an example, if the threshold is moved 10 values higher than what is optimal, it will lead to 83% false rejections when only one frame is used as input, by an increase of 43%. If the same shift is done for the scenario where 20 frames are used as input, the EER will only increase from 11% to 13% false rejection. This gives reason for inspecting the results when longer inputs are used for the classification system. In figure 6.8, the long speech samples from test set 2 and the noise samples from test set 3, namely samples from the microphone array, are used as input.



(a) System with UBM with 1 mixture



(b) System with UBM with 2048 mixtures

Figure 6.8: DET curves for long speech input sequences

When up to several seconds of a recording are used for classification, it can be found that the EER becomes zero for both systems. The system is able to distinguish between noise and speech with high confidence and with a wide range of optimal threshold values. However, the system is working slowly as it needs to record multiple seconds before making a decision. In other words, it may be used in an offline system, but not in a real-time system. The system might still not work optimally offline, as sudden noises might not be detected if they only take up a few frames of the entire input.

Another aspect of classification is to look at silence. The speech model should not have learned that silence is the same as speech, as the microphone array should not focus on silence. When running test set 4 of silence samples through the speech and noise model, table 6.3 was found.

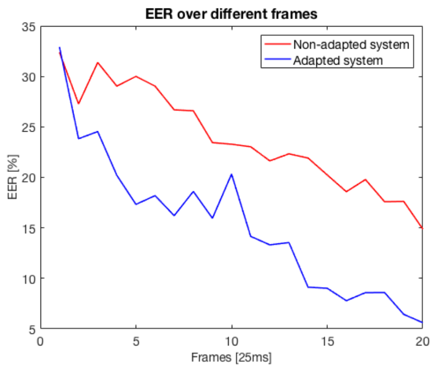
Table 6.3: Classification of silence

Number of frames	Length [ms]	Speech	Noise
1	25	0%	100%
10	250	0%	100%
20	500	0%	100%

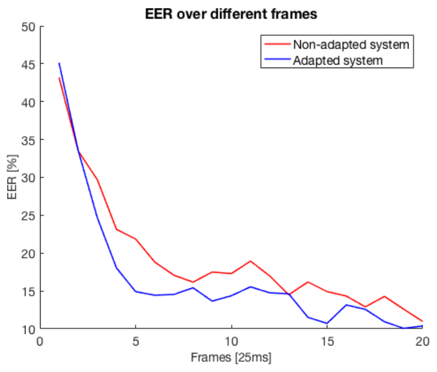
Both UBMs were used in the classification of silence, and they both gave the same results. Table 6.3 clearly shows that all samples of silence are classified as noise. This is optimal and as hoped for, as it is undesirable that the system focuses on silence when it is supposed to focus on speech.

6.3 Model Adaption

The goal of implementing model adaption was to improve system performance. It is crucial for the speech input to be perceived as well as possible, and it was therefore expected that using model adaption on the speech model would improve the overall system performance. Less adaption data were available for adaption of the noise model. However, the results where both models are adapted is found towards the end of this section. By implementing model adaption on the speech model, namely finding an optimal matrix W that could adapt the mean values of the mixture model, the results in figure 6.9 were found. Test set 1 with 102 speech samples from the microphone array were used to obtain the optimal W.



(a) Systems with UBM with 1 mixture



(b) Systems with UBM with 2048 mixtures

Figure 6.9: Comparison of non-adapted and adapted systems with array input

Figure 6.9a clearly show an improvement for the system with the 1-mixture UBM. The adapted system provides better results for every amount of input frames larger than one. For an input of only one frame, the non-adapted system provides an EER that is 0.5% lower than for the adapted system. Neglecting this point, the adapted system has an EER between 3.4% and 12.8% lower than the non-adapted system. By averaging over all the frames, the adapted system show an overall 8.6% decrease in EER, which is a very positive improvement. The results of the model adaption for this system can therefore be considered satisfactory.

The system with the 2048-mixture UBM, on the other hand, does not show as clear improvement. The adapted system can be interpreted as overall slightly better than the non-adapted system. For inputs of one frame, the non-adapted system provides better results than the adapted system, also for this system, with a difference in EER of 2%.

The poor improvement of the adapted system is likely due to the lack of data. Since the number of mixtures is high for the UBM in this system, it is likely that there is not enough data to make a good adaption for this system. An adaption of the speech model should be able to improve the system performance more than what is seen in figure 6.9b.

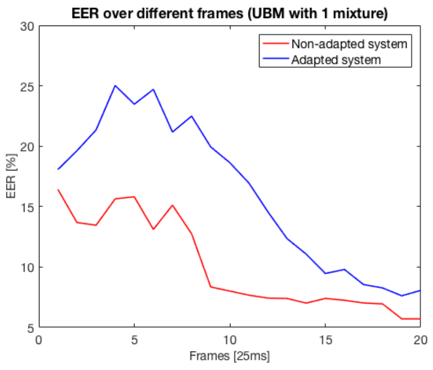
By comparing the different systems in regards to performance, it is found that there is not one single system that is optimal, but rather that it depends on the amount of input frames. Table 6.4 gives an overview of which system provides the lowest EER, and is thereby the optimal system, depending on how many frames are used as input.

Table 6.4: Overview of which system provides lowest EER

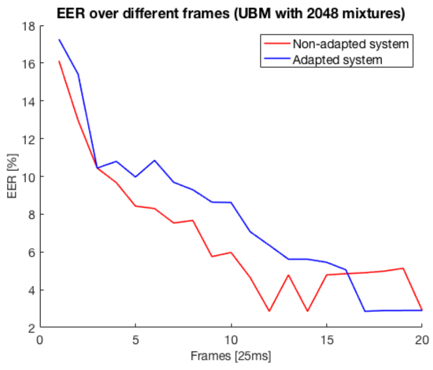
Input frames	Optimal System	EER
1	Non-adapted system w/UBM$_1$	32.4%
2-3	Adapted system w/UBM$_1$	23.8% - 24.5%
4-10	Adapted system w/UBM$_{2048}$	14.4% - 18.0%
11-20	Adapted system w/UBM$_1$	5.6% - 14.2%

The adapted system with the 1-mixture UBM turns out to be the best choice for very few inputs as well as larger inputs, which is seen from the results in figure 6.9 and table 6.4. When 20 frames are used as input, the adapted system reaches the minimum EER of 5.6%, meaning that the system classifies speech with high confidence. However, for inputs of 4-10 frames, the adapted system with the 2048-mixture UBM turns out to be the better choice. Altogether, there is not one system that is overall significantly better than the other. It is likely that having more data would result in clearer results in regards to finding an optimal system for all amounts of input.

Another approach for exploring how the adapted system is performing, is to test it on speech and noise from test sets 5 and 6, namely on input that is not from the microphone array. This should result in poorer performance for the adapted system since it is adapted to the microphone array. The results can be seen in figure 6.10.



(a) Systems with UBM with 1 mixture



(b) Systems with UBM with 2048 mixtures

Figure 6.10: Comparison of non-adapted and adapted systems with non-array input.

It becomes clear from the two figures above that the overall system performance is deteriorated due to the adaption of the speech model, regardless of which noise model is used in the system. This was as expected, and these results are therefore satisfactory. Similar to figure 6.9, it can also here be seen that an adaption of the system results in clearer differences in the system with the 1-mixture UBM than the system with the 2048-mixture UBM, which is likely due to the small amounts of data.

Taking a step further, the approach of adapting both the speech model and the noise model for the system was studied. The 27 noise files from test set 3 were used to find the optimal adaption matrix *W* for the adaption of the noise model with 2048 mixtures. The results of the adapted systems with the 2048-mixture UBM are presented in figure 6.11.

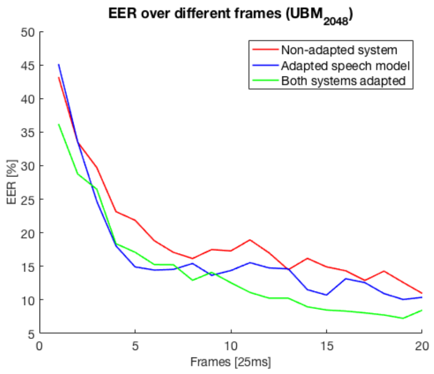
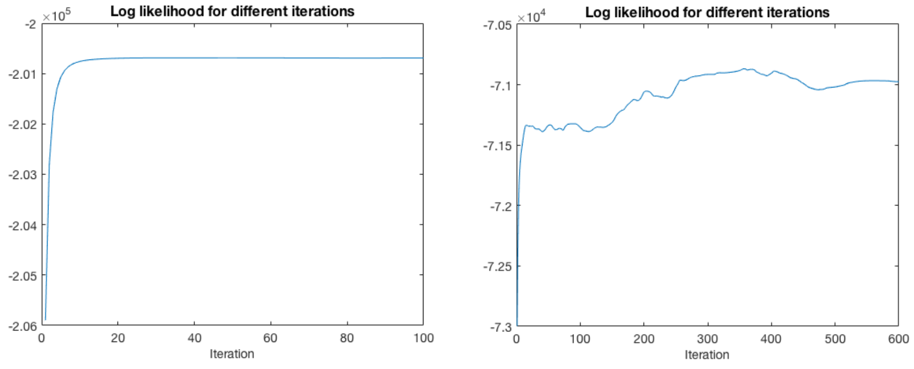


Figure 6.11: Comparison of systems with 2048-mixture UBM

From figure 6.11, it is seen that the EER drops by almost 10% for one frame inputs and there is a slight improvement at inputs of 10 frames or higher, for the system where both models are adapted. All in all, the system where both models are adapted is slightly better than the two other systems.

When trying to find an appropriate adaption matrix for the 1-mixture UBM, it was not possible to find the adaption matrix from the equation $G \cdot W = Z$ since G was not of full rank and thereby not invertible. This was caused by the very small mean values of the noise model, which was in the order of 1e-16 or lower. This led to the matrix G containing values between 1e-35 and 1e-16 which equals machine precision of zero in practice.

By inspecting the log likelihood values that were obtained in the search of optimal adaption matrices, clear differences were found. These results can be an indication that an optimal adaption matrix might not have been found for the noise model adaption. The log likelihood values are displayed in figure 6.12.



(a) Log likelihood for speech model adaption **(b)** Log likelihood for noise model adaption

Figure 6.12: Comparison of log likelihood values for adaption

Figure 6.12b clearly shows that the log likelihood values of the noise model adaption of the 2048-mixture UBM is alternating, while the values for the speech model adaption in figure 6.12a increases until a certain point before they very slowly decrease. The graph of the log likelihood values of the speech model adaption is smooth, as oppose to the graph for the noise model adaption. This indicated that an optimal adaption matrix is found for the speech model adaption, while there is still some uncertainty to whether the absolute optimal adaption matrix is found for the noise model adaption. This is likely due to the amounts of data that were used for adaption, as the adaption of the speech model used cepstra from 102 speech samples, while the adaption of the noise model only used cepstra from 27 noise samples.

If more data were available, adapting both models could potentially result in better overall performance. However, as the current amount of data, and especially amount of noise data, for adaption was very limited, adapting both models did not provide a very significant improvement, as somewhat expected. If more noise data were available for model adaption, a more optimal adaption matrix W could be found. That way, the parameters of the noise model could be adapted to better match the input data.

From subsection 3.4.2 in the theoretical background chapter, it is mentioned that the difference in performance between two classifiers needs to be statistically significant in order to choose one over the other. It would be preferable to see how the system with the 1-mixture UBM would perform if the noise model was adapted. Despite this, based on this statement, as well as table 6.4, it is not reasonable to choose one of the noise models over the other, as there is not one systems that is clearly better than the other.

Chapter 7

Conclusion

When implementing source localization in MATLAB, the results turned out to be satisfactory. The strongest sound source, as well as the less strong sound source, were detected by the system. However, the system only found the source with the highest amplitude without any consideration of what the source was. This is the reason classification had to be implemented.

The methods that were used in this thesis were Mel-frequency cepstral coefficients for feature extraction, and Gaussian Mixture Model for classification. A GMM for speech classification was trained with 512 mixtures, while two GMMs for noise classification were trained: one with 2048 mixtures and another with one mixture. The noise models were trained to be universal background models with large variance in order to pick up a wide variety of noise sources.

The performance of the classification systems were dependent on the length of the input sample. With only one frame of 25ms, the performance of the systems were poor, with EER above 30%. However, both systems were able to recognize speech better, the longer the input sequence was. Inputs of length 500ms resulted in EER below 15% for both systems, while inputs of up to several seconds reached EER of 0% for both systems. Longer inputs could be used in an offline system, where the input length should be chosen dependent on which noise sources are expected. A real-time system should make computationally fast decisions in order to switch between speakers rapidly and not focus on noise, and by that only use a few frames as input. A real-time system should therefore be much more robust for very short input sequences than what is found in this thesis.

The system with the 1-mixture UBM showed best results from input sequences of one and two frames. However, from input sequences greater than two frames, the system with the 2048-mixture UBM showed the best results. As more frames were used as input, not only did the EER decrease, but the threshold value also became more robust. A slight shift in the threshold value had smaller impact on the error rate, the more frames were used as input. When classifying samples of silence, they were found to be classified as noise for both systems, regardless of the input length. This was optimal, as it was undesirable that the systems classified silence as speech and thereby focused on silence.

Model adaption was implemented in order to adapt the parameters of the models to better match the input data. When only the speech model was adapted, the system with the 1-mixture UBM showed great improvement, with an average of 8.6% decrease in EER, which was a very positive improvement. However, there was no improvement for inputs of one frame, only on inputs of more than one frame. The system with the 2048-mixture UBM showed a slight improvement when the speech model was adapted. When feeding the systems input that was not from the microphone array, the adapted systems showed poorer performance than the non-adapted systems. This was expected as the adapted systems were adapted to the microphone array.

Adapting both the speech and noise model resulted in a slight improvement for the system with the 2048-mixture UBM, compared to when only the speech model was adapted. However, the adaption of the system did not show as big improvements as expected. For the system with the 1-mixture UBM, it was not possible to find an adaption matrix as the mean values of the noise model were too small, resulting in a non-invertible adaption matrix. 102 adaption samples of speech were used to obtain the adaption matrix W and the log likelihood values showed a clear maximum which resulted in an optimal W matrix. For the adaption of the noise model on the other hand, only 27 adaption samples of noise were available and the log likelihood values were alternating. It was therefore likely that the adaption matrix W was not the most optimal. More adaption data would be necessary for finding absolute optimal adaption matrices.

The difference in performance between the two noise models was not statistically significant enough to choose one over the other. Overall, obtaining more data, especially keyboard noise, for training, adaption and testing is likely to improve the system performances. The system with 1-mixture UBM that was adapted for the speech model reached 5.6% EER at 500ms inputs, indicating that this system was working well for offline use. There are however clearly room for improvement if the systems are to be used in real-time, as the EER is currently too high for providing good accuracy and thereby a good user experience.

Chapter 8

Future Work

One of the most important approaches for improving the system is to obtain more noise and speech data for training, testing and adaptation. That way, stronger models can be trained, and the performance can be evaluated more thoroughly. Collecting noise from keyboard clicking for training the noise model is likely to improve the overall system performance. Working further on developing an fully adapted system for the system with the noise model with one mixture is likely to provide good results. Other methods can be tried out, as there are several other approaches for sound classification that can potentially provide better results.

The final system should be one where source localization and classification are connected. Further work should therefore be to connect source localization and classification to build a connected system that actively searches for the speaker in a room, based on inputs from the microphone array. The connected system should be computationally fast in order to provide the best user experience, if used in real-time.

To built further on the system, a potential addition would be to attenuate the noise in order to improve the system performance. That way, the system can not only find and switch between speakers, but also lower the noise that is found in the room. The final system should be optimized to provide a good user experience.

Bibliography

[1] Cowling, M. (2004). *Non-Speech Environmental Sound Classification System for Autonomous Surveillance*. Griffith University, Gold Coast Campus, pp.9-109

[2] Van Trees, H. (2002). *Detection, estimation, and modulation theory*. New York: Wiley-Interscience, pp.17-89.

[3] McCowan, Iain. (2001) *Robust Speech Recognition using Microphone Arrays*, Queensland University of Technology

[4] Krol, D., Swiecinski, R. and Lorenc, A. (2015). *Detecting Laterality and Nasality in Speech with the Use of a Multi-Channel recorder.*

[5] Dey, N. and Ashour, A. (2017). *Direction of arrival estimation and localization of multi-speech sources*. Springer, pp.5-22.

[6] Se.mathworks.com. *Grating Lobes*. [online] Available at: `https://se.mathworks.com/help/antenna/ug/grating-lobes.html` [Accessed 10 Apr. 2018].

[7] Antenna-theory.com. *Grating Lobes*. [online] Available at: `http://www.antenna-theory.com/arrays/weights/gratinglobes.php` [Accessed 11 Apr. 2018].

[8] Se.mathworks.com. (2010). *Developing an Isolated Word Recognition System in MATLAB*. [online] Available at: `https://se.mathworks.com/company/newsletters/articles/developing-an-isolated-word-recognition-system-in-matlab.html` [Accessed 12 Apr. 2018].

[9] Lutter, M. (2014). *Mel-Frequency Cepstral Coefficients*. [online] Recognize-speech.com. Available at: `http://recognize-speech.com/feature-extraction/mfcc` [Accessed 29 Mar. 2018].

[10] Manolakis, D. and Ingle, V. (2011). *Applied Digital Signal Processing.* Cambridge: Cambridge University Press.

[11] Ahmed, N., Natarajan, T. and Rao, K. (1974). *Discrete Cosine Transform.* IEEE Transactions on Computers, C-23(1), pp.90-93.

[12] Strand, O. and Egeberg, A. (2004). *Cepstral mean and variance normalization in the model domain.* [online] Available at: `https://pdfs.semanticscholar.org/0de2/7e275803a000babcfa5c06c0683ee1df76e0.pdf` [Accessed 22 May 2018].

[13] Huang, X. (2001). *Spoken language processing.* Upper Saddle River, NJ: Prentice Hall PTR.

[14] Brilliant.org. *Gaussian Mixture Model.* [online] Available at: `https://brilliant.org/wiki/gaussian-mixture-model` [Accessed 12 Apr. 2018].

[15] VanderPlas, J. (2016). *Python data science handbook.* New York: O'Reilly Media.

[16] Akaike, H. (1974). *A new look at the statistical model identification.* IEEE Transactions on Automatic Control, 19(6), pp.716-723.

[17] Hu, S. (2007). *Akaike Information Criterion.* Center for Research in Scientific Computation, North Carolina State University

[18] Reynolds, D. A. (2008). *Universal Background Models.* Encyclopedia of Biometric Recognition, MIT Lincoln Laboratory

[19] Mahier, J., Pasquet, M., Rosenberger, C. and Cuozzo, F. (2009). *Biometric Authentication.* Encyclopedia of Information Science and Technology, Second Edition, pp.346-354.

[20] Dominique Fohr, Odile Mella, Irina Illina. (2017). *New Paradigm in Speech Recognition: Deep Neural Networks.* IEEE International Conference on Information Systems and Economic Intelligence, Marrakech, Morocco.

[21] Myrvoll, T. A. (2002). *Adaption of Hidden Markov Models using Maximum a Posteriori Linear Regression with Hierarchical Priors.* Norwegian University of Science and Technology, pp. 49-55.

[22] Michelsanti, D. and Tan, Z-H. (2017). *Conditional Generative Adversarial Networks for Speech Enhancement and Noise-Robust Speaker Verification,* INTERSPEECH, Stockholm, Sweden.

[23] nb.no. (2016). *Språkbanken: NB Tale - en grunnleggende akustisk-fonetisk tale-database for norsk.* [online] Available at: `https://www.nb.no/sprakbanken/show?serial=oai\%3Anb.no\%3Asbr-31\&lang=nb` [Accessed 27 Mar. 2018].

[24] *RWCP Sound Scene Database in Real Acoustical Environments,* Copyright (c) 1998-2001 Mitsubishi Research Institute, Inc. [Accessed 4 May 2018]

[25] se.mathworks.com. *Phased Array System Toolbox* [online] Available at: `https://se.mathworks.com/help/phased/index.html` [Accessed 27 May 2018]

[26] se.mathworks.com. *Statistics and Machine Learning Toolbox* [online] Available at: `https://se.mathworks.com/help/stats/index.html` [Accessed 28 May 2018]

[27] Falkena, W. and Mo, X. (2012). *xml2struct* [online] Available at: `https://se.mathworks.com/matlabcentral/fileexchange/28518-xml2struct?focused=5233246&tab=function` [Accessed 27 May 2018]

[28] se.mathworks.com. *resample* [online] Available at: `https://se.mathworks.com/help/signal/ref/resample.html` [Accessed 27 May 2018]

[29] Ellis D. (2012). *PLP and RASTA (and MFCC, and inversion) in Matlab using melfcc.m and invmelfcc.m* [online] Available at: `https://labrosa.ee.columbia.edu/matlab/rastamat/` [Accessed 28 May 2018]