Speech Reconstruction from Binary Masked Spectrograms Using Vector Quantized Speaker Models

Michael K. Jensen (s001596)
Søren Skou Nielsen (s001806)

August 7, 2006

Preface

This thesis has been prepared over a six month period at the Section of Intelligent Signal Processing, Department of Informatics and Mathematical Modelling, at The Technical University of Denmark, DTU, in partial fulfillment of the requirements for the degree, Master of Science in Engineering, M.Sc.Eng.

The reader is assumed to posess basic knowledge in the areas of signal processing, geometry, and statistics.

Kgs. Lyngby, August, 2006.

Søren Skou Nielsen and Michael Knöchel Jensen

Acknowledgements

This thesis would never have amounted to what it is, or indeed anything, without the invaluable help and support from the following people.

First and foremost, we thank our project supervisor Lars Kai Hansen, for his ability to inspire and motivate, as well as for helpful explanations and insight into tricky matters.

Andreas Brinch Nielsen, co-supervisor, for being available throughout the span of the project, and for being happy to answer our questions.

Ling Feng, from the Section of Intelligent Signal Processing, for providing assistance related to speech databases.

Anders Petersen, a project room companion and fellow thesis student, for proof reading above and beyond expectations in spite the fact that he was off to start his vacation.

Søren Wulff Kristiansen and Ligia Maria Dias, a good friend and his fiancé, for helping out with graphical issues.

We would like to thank the entire Intelligent Signal Processing Section for providing a pleasant atmosphere.

Most important of all, our families and girlfriends for love and support, as well as for coping without us in the final few days when the loose ends had to come together.

Abstract

Several source separation techniques use binary masking on spectrograms to separate two or more speakers from each other. In this thesis, the possibilities for obtaining the best quality signal, reconstructed from masked spectrograms through vector quantized models of speakers, is investigated. The advantages and disadvantages of such an approach are examined. Additionally, the task of signal reestimation from a spectrogram is investigated using several algorithms.

Vector quantization of speakers can be used to improve on binary masked spectrograms but the approach is not shown to produce high quality speech. It is also concluded that phase information is very important for high quality speech reconstruction, and parameters for optimal phase reestimation are suggested.

Keywords:
signal processing, data clustering, mel filtering, voiced unvoiced detection, k-means, vector quantization, signal estimation, phase reconstruction, spectrogram reconstruction.

Resumé

Flere teknikker til adskillelse af tale fra to eller flere talere anvender binære masker på spektrogrammer. I denne opgave undersøges mulighederne for at opnå den bedste kvalitet for et signal, rekonstrueret fra binært maskerede spektrogrammer ved brug af vektor kvantiseringsmodeller af talere. Fordele og ulemper ved denne fremgangsmåde vil blive undersøgt. Desuden behandles opgaven med reestemering af et signal fra et spektrogram gennem flere algoritmer.

Det vises at vektorkvantisering kan anvendes til at forbedre på maskerede spektrogrammer, men det er ikke lykkedes at få fremgangsmåden til at producere tale i høj kvalitet. Det konkluderes desuden at faseinformationen er meget vigtig i forbindelse med rekonstruktion hvis tale i høj kvalitet er et mål, og parametre for optimal fasegendannelse foreslås.

Nøgelord:
signalbehandling, data gruppering, k-means, mel filtrering, voiced unvoiced genkendelse, vektor kvantisering, signal estimation, fase rekonstruktion, spectrogram rekonstruktion.

Contents

Chapter 1

Introduction to Auditory Signal Analysis

> "One of our most important faculties is our ability to listen to, and follow, one speaker in the presence of others. This is such a common experience that we may take it for granted; we may call it "the cocktail party problem." No machine has been constructed to do just this, to filter out one conversation from a number jumbled together."
>
> - *Colin Cherry, 1957*

Many hearing impaired people, however, seem to have completely lost this ability, making it very difficult for them to have a conversation with another person if multiple speakers are talking at the same time, or a lot of background noise is present.

Seperating speakers from a complex acoustic environment requires auditory scene analysis (ASA) as described by Bregman [1]. Research in this field of signal processing has been extensive in the last 10 years, using two forms of ASA: One form uses the predictability of cues like pitch and spatial location, such as the beam-forming technique, widely used in newer hearing aids, where multiple micropohones make it possible to amplify sound from the direction in which the hearing impaired person is facing. The other form uses stored sound patterns as a supplement.
This leads us to the motivation of our project. Many elderly people only have daily contact with a small handful of different speakers. Suppose we had prior information about a speech signal from a certain speaker. Would it then be easier to extract this particular speaker from a noisy auditory environment?

The hearing aid could be trained to know this handful of speakers by conversations in a noise free environment. Such a technique would not be totally dependent on the location of the speaker, and almost any type of noise would be possible to attenuate if the hearing aid knows exactly what to “listen for”.

There are many ways to analyse a signal for a certain speaker and equally many ways to separate speakers. In this project, we have turned our attention to a fairly recent type of source separation technique that uses spectrograms, a method often used to visually show frequency changes over time, to separate speakers. Sam Roweis [22] described how it is often possible to see a pattern in the spectrogram of two simultaneous speakers and how the two speakers can almost be distinguished by the human eye. Several approaches have been used to extract only the data of a single speaker from the spectrogram of a complex auditory environment, using vector quantization [7] [22] as well as neural networks approaches [27] [11]. None of the methods are perfect, however, and they thus need some form of restoration of the missing data, to be considered useful in a hearing aid application. This restoration of a speech signal from a masked spectrogram is what we will focus on during the length of this report.

Chapter 2

Speech Reconstruction From Binary Masked Spectrograms

We have chosen to model a speaker using vector quantization (VQ), since this seems like an intuitive, fast, and not too complicated way of modelling spectrograms of a speaker. In essence, our approach would be to recognise a speaker, subtract him or her from the mixed spectrogram (using ideal binary masking) and try to repair the masked spectrogram using the prior information about a certain speaker stored in the VQ.
A similar approach has been implemented by Dan Ellis in [7], where he uses k-means vector quantization and mel-filter weighting to separate a speaker of interest from another simultaneous speaker. He does not, however, use the already existing information in the mixed spectrogram, but tries to project the mixed spectrogram onto the trained VQ, and then overwrites the mixed spectrogram data with pure VQ information, which in effect is separation by speech synthesis. We will also use the k-means algorithm, but as opposed to Dan Ellis [7], we will try to maintain as much information as possible and only try to repair the masked parts.

We intend to go into depth on how to reconstruct good quality speech signals from repaired spectrograms, and evaluate several algorithms on the critical task of estimating the lost phase information from the spectrogram. For the vector quantization itself we intend to investigate use of the k-means algorithm for clustering speech data, and different methods for reconstruction. Also we examine if this approach can be improved by extensions such as mel filter bank weighting or splitting spectrograms into portions of voiced, unvoiced, and silence regions.
The results for each experiment will be compared by both a subjective evaluation of sound quality, and a calculation of signal to noise ratio.

2.1 Previous research in this field

There has been a considerable interest in researching areas of ASA since the concept was published by Albert Bregman in 1990 [1]. Much of this work has revolved around segregation of speech. Such systems often consist of a segmentation stage where a signal is decomposed into segments each originating from a different source, and a synthesis where the segmented data is reconstructed into intelligible speech. This is also the case for our approach, only the binary masking has already ocurred.
The following are systems for sound segregation based on computational auditory scene analysis (CASA). These present tools used within the area of our problem, and some even define the foundation for our approach.
Ephraim et.al. (Ephraim et al., 1989) modeled speech using a statistichal approach of Hidden Marchov Modeling (HMM) and noise using an AR model. With this model they inferred approximate clean speech by alternating between Wiener filtering to find the noise and Viterbi decoding in the HMM.In this way they managed to enhance corrupted speech. Several later studies including a large body of work in the subject of speech recognition [ling], [31], [25] incorporate the use of HMM to model speech. HMM could be used in conjunction with VQ to perform a less primitive and more schema-driven reconstruction.
DeLiang Wang and Guy Brown [27] did seperation of speech using a neural networks approach based on oscillatory correlation, which belongs to the category of primitive segregation [1]. Later Guoning Hu and DeLiang Wang [11] did speech segregation using a neural networks approach again, but this time with a schema-driven model based on pitch tracking and amplitude modulation. In both of these studies, speech seperation is done by creating binary masks to remove noisy parts of log spectrograms and so these approaches look like the basis for our project of reconstructing speech spectrograms with missing data.
An approach closely related to our intended approach is used by Dan Ellis in 2004 [7], where a vector quantization and HMM model approach is used for seperating two speakers and for resynthesising the speech of the speaker of interest. Dan Ellis reported that an extreme amount of data would be required in order for the approach to achieve good percieved quality of the reconstructed speech.

The following are systems which focus on spectrogram reconstruction based on computational auditory scene analysis (CASA). These are alternative approaches to solving our problem with different advantages.
Deliang Wang and Soundararajan Srinivasan [25] did spectrogram recon-

struction for corrupted phonemes, by storing phoneme templates and inserting these into corrupted patches of speech using HMM and lexical knowledge for choosing which phoneme to insert. They reported good results for restoration of single corrupted phonemes, and also for reestimating and smoothing corrupted pitch information.

Reyes et. al. 2005 [19] drew inspiration from successful applications in computer vision, and used a layered generative graphical model that describes two components in seperate layers. One layer is for the excitation harmonics, and another is for formants and such resonances. This approach exploits the high correlation between adjacent frames to model successive spectra as transformations of their immediate predecessors. They reported outstanding results in the reconstruction of harmonics as well as formants of voiced speech where border-areas where well defined.

Chapter 3

Motivation and Objectives

What if hearing aids could recognize the voice of a friend or a family member in complex noisy environments, and, based on stored knowledge of that voice, were able to transmit it without any of the noise?

Then hearing aid users would be able to comfortly communicate with their loved ones, and be more at ease in almost any environment. But even state of the art hearing aids are far from performing that task, so much work has to be done. The hearing aid should be able to construct a model of speech for a given speaker, it would have to be able to use this model to seperate the speech of this speaker from that of different speakers or other forms of complex noise, and it would have to synthesise the speech before being able to transmit it to the user.

Methods have already been developed for seperating speech from different speakers present in an environment, using a spectrogram based approach. These methods are designed to produce a binary masked spectrogram, which contain only the spectrogram bins thought to belong to the speaker of interest. Speech directly synthesised from such a spectrogram, however, suffers from poor perceived quality and naturalness.

Thus, the main objective was to:
Investigate means of reconstructing a speech signal from a binary masked spectrogram, with high percieved quality and naturalness as the goal.

3.1 Thesis overview

It became evident in the early stages of our project, that the main objective was best met through two main parts - Signal Estimation, which is used to synthesise a speech signal from a spectrogram, and Vector Quantization which is used to model speech and consequently to reconstruct spectrograms of speech.
To accomodate these two main parts and additionally some general theory and conclusion, this thesis is structured into four parts, where each part requires knowledge from the preceeding parts.

Part I Presents some foundational theory of speech as well as tools for signal analysis used throughout this report. Here, the concept of a spectrogram is explained.

Part II Presents several methods for estimating the speech signal from a spectrogram where knowledge of phase is limited.

Part III Presents a vector quantization approach to reconstructing spectrograms, and also investigates key components of doing vector quantization on speech spectrograms.

Part IV Finally, the thesis work is concluded, and possible expansions of the work are discussed.

3.2 Nomenclature

To ease the understanding of coming theory, variables without an explicit denotation conform to the following nomenclature:

$x(n)$ or x_n	Signal x defined by sample n.
$w(n)$	Window w defined by sample n.
$X_w(kS, \omega)$ or X_k	Fourier transform of the k'th segment of a signal.
$\mathbf{X}$	A spectrogram.
$\mathbf{Y}$	A modified spectrogram.
$y_w(kS, n)$	IFFT of a column in the modified spectrogram.
$\mathbf{x}_k$	k'th segment of a signal, x.
$x_w(kS, n)$	windowed k'th segment of a signal, x.
ω	frequency index of the Fourier transform
S	Segment stepsize.
M	Number of segments in a signal / number of frames in a spectrogram.
N	Number of samples in a signal.
kS	Starting sample of k'th segment.
$\mathbf{F}$	Signal defined by sample n.
L	Window length.
Ω	FFT length.
$\mathbf{S}$	A spectrogram.
$\mathbf{p}_{rand}$	Random phase.
K	Amount of clusters in a codebook.
J_i	Amount of members in the i'th cluster.
f	Frequency.
$\hat{f}$	Mel based frequency.

3.3 Abbreviations

Here, the abbreviations used in this thesis are listed:

VQ Vector Quantization.
ASA Auditory Scene Analysis.
CASA Computational Auditory Scene Analysis.
ERB Equivalent Rectangular Bandwidth.
FFT Fast Fourier Transform.
HMM Hidden Markov Model.
MSE Mean Squared Error.
VUS Voiced / Unvoiced / Silence.
SFM Spectral Flatness Measure.

SNR Signal to Noise Ratio.
MFCC Mel Filter Cepstral Coefficients.
HFCC Human Factor Cepstral Coefficients.
IFFT Inverse Fourier Transform.
STFT Short-Time Fourier Transform.

Part I

Speech Theory and Tools

Chapter 4

Speech and speech signals

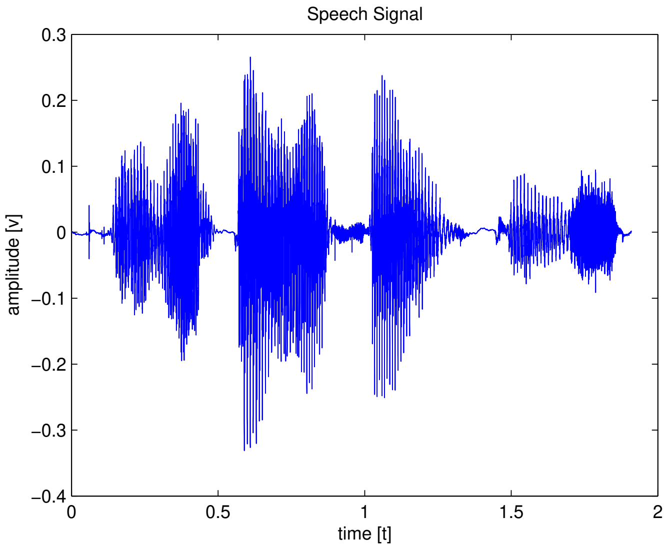


Figure 4.1: A speech signal. The words "Lay red by R5 please" are uttered.

Since speech is the sound source we focus on throughout the project, we will shortly go through the basic theory behind speech and speech signals, and thereby introduce some of the terms used in this thesis.
Speech is produced as air from the lungs is forced through the vocal cords and along the vocal tract.

The vocal tract is the path all the way from the opening of the vocal cords (*latin: glottis*) to the mouth. The vocal tract can be thought of as a filter being able to produce different sounds with resonances. The sound produced depends of the shape of the vocal tract and can be changed by moving the tounge and jaw.

4.1 Voiced and unvoiced sounds

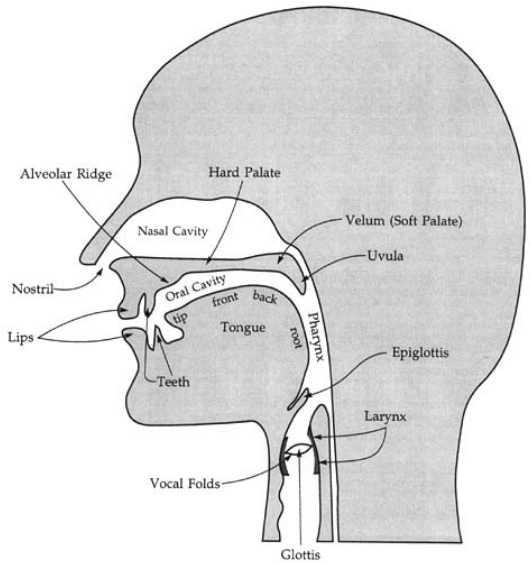


Figure 4.2: Illustration of the human speech apparatus. Figure from [36]

Speech sounds can be broken up into three different classes. Voiced, Unvoiced and Plosive [32].

Voiced sounds occur when the air from the lungs passes the vocal cords (*latin: glottis*) which vibratingly opens and closes (by the vocal folds). This produces the quasi-periodic pulses of air also seen in figure 4.3 (right), the rate of this vibration gives what is known as the pitch of the sound. Examples of voiced sounds are the “A” sound or the “O” sound. If we look at figure 4.3 we see that the pitch period in this example is about 10 ms. The pitch is changed by changing the tension on the vocal cords or the air pressure behind them.

Unvoiced sounds occur when air is forced out of the open vocal cord at high velocities while constricting the vocal tract. An example of this could be when the tounge is pressed against the aveolar ridge (see Figure 4.2), as in an “S” sound, or the teeth against the lower lip, as in an “F” sound. These types of sounds do not show the same length of periodicity as voiced sounds. In stead, they are often similar to noise. They do however still have some short-term correlations due to the vocal tract.

Plosive sounds occur when the vocal tract closes completely, pressure builds up and is then suddenly released. An example of a plosive sound is the first part of the P sound (without the "ee" sound), as in the word "pop".

As we shall see later, these types of sounds distinguish themselves clearly when doing Fourier analysis.

4.2 Stationarity of speech signals

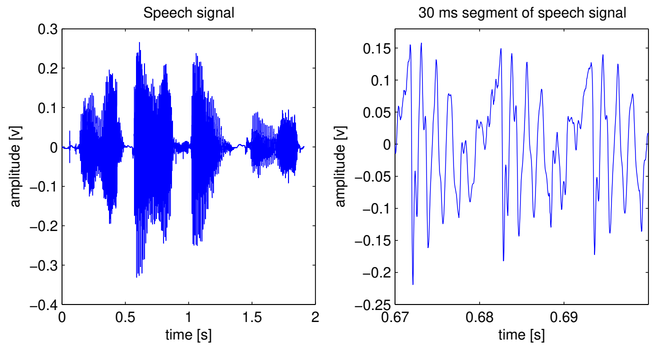


Figure 4.3: Example of the quasi-stationarity of speech signals at small segments of time. To the left is the entire speech signal, which does not look very periodic. To the right is a 30 ms segment of the same speech signal. We clearly see that in this part of the signal, the speech signal looks almost periodic.

Speech signals are non-stationary. This means that the statistical properties of the signal change over time. In contrast, a sinusoid never changes statistical properties but remains the same (periodic). As explained in the previous section, speech is a result of changing the shape of the vocal tract while exciting the vocal chord. Because this movement does not happen at an instant, but in relatively "slow" continuous movements, the speech signal has what is called quasi-stationarity in small segments of time. This means that the signal is "nearly stationary" and only changes very little in a short span of time.
In most litterature [7], [17], a speech signal is assumed quasi-stationary in segments of around 20-40 ms. This is important because many signal processing tools, like the Fouier transform, assumes the signal to be stationary. With nature giving us a 20-40 ms window, we can analyse speech signals almost as we would any other signal, but in small segments at a time. In figure 4.3 we see a close up of a speech segment of 30 ms. It clearly shows the near periodicity of the signal.

Chapter 5

The Short Time Fourier Transform

As explained earlier, if a signal is non-stationary, a normal Fourier transform will not give usable information, since stationarity is assumed when applying a Fourier transform. Some non-stationary signals however, can be assumed stationary over small periods of time, including speech signals, and a Short Time Fourier Transform (STFT) can then be used.

A signal $x(n)$, ex. a speech signal, can be divided into small segments each with a length of L samples called the window length. Due to the nature of the Fast Fourier Transform (FFT) the window length is often chosen as a power of 2, ex. 256 or 512. A segment corresponds to multiplying a square window of length L on a part of the signal. Applying the Fourier tranform on a segment created with a square window causes a ringing effects which severely degrades the resolution of the magnitude spectrum. To avoid ringing effects, a smoothing window function, such as the hamming window function (see figure 5.1), is often used instead of a square window. Figure 5.1 is an illustration of the process of obtaining the windowed segments. The windowed signal in a segment can be described as:

$$x_w(kS, n) = w(kS - n)x(n) \tag{5.1}$$

Where $w(n)$ is the window function that is L points long and non-zero for $0 \leqq n \leqq L - 1$. $k \in [0..M]$ is the segment number, S is the number of samples between each segment, also known as the step-size. kS can be viewed as an indicator of the starting sample of segment k. Equation 5.1 can be thought of as running a signal through a window in steps of S and saving the windowed segments. In other litterature, eq. 5.1 is sometimes written as running a window over the signal, the result is the same. After segmentation,

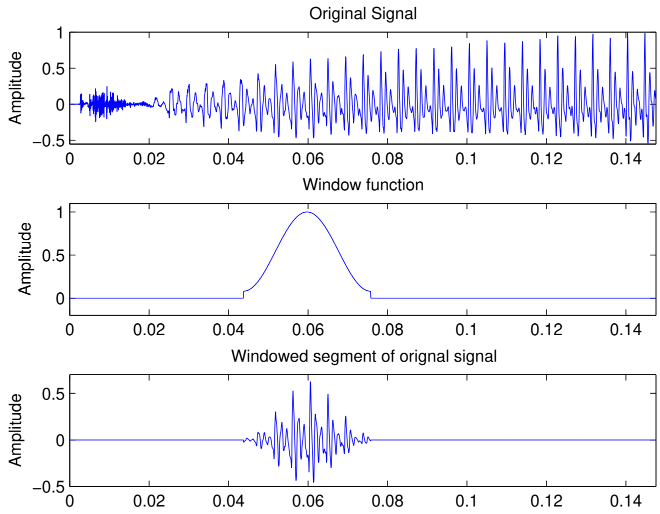


Figure 5.1: Illustration of the window multiplied to the signal to form the segmented part of the signal (frame or segment). The part of the segment that is beyond the boundaries of the window is normally not included in the Fourier transform that is applied after.

the individual segments are Fourier transformed revealing:

$$X_w(kS,\omega) = F_t\left[x_w(kS,n)\right] = \sum_{n=-\infty}^{\infty} x_w(kS,n)e^{-j\omega n} \tag{5.2}$$

Where F_t is the Ω point Fourier transform, and ω is the frequency index in segment k. The segments can now be treated as normal Fourier transforms, but only contain information on a small part of the signal. For speech, this is usually around 30 ms. The STFT can also be done with overlapping regions, which means that S is smaller than the window length L. This way, small parts of the signal are stored in two or more segments, instead of only one. This is often done to increase resolution and will be explained further in later sections.

5.1 The Inverse STFT

The STFT is fully reversible using the inverse STFT to regenerate the signal. The signal is reconstructed by taking the inverse Fourier transform (IFFT) to each of the segments and combining these. The overlapping of the segments and the windowing function that was done before the Fourier transform on the segments, must be taken into account when combining the IFFT of the individual segments. This is done as follows:

To obtain the individual windowed segments, the inverse Fourier transform is applied to each segment.

$$x_w(kS,n) = \frac{1}{2\pi}\sum_{n=-\infty}^{\infty} X(kS,\omega)e^{j\omega n} \quad \text{for all}\, k \in [0,...,M-1] \tag{5.3}$$

To place the segments at the correct location of $x(n)$, the individual segments are delayed to match the place where the segment was taken, and divided by the synthesis window. Remembering equation 5.1 we get:

$$x(n) = \frac{\sum_{k=0}^{M-1} x_w(kS,n)}{\sum_{k=0}^{M-1} w(kS-n)} = \frac{\sum_{k=0}^{M-1} w(kS-n)x(n)}{\sum_{k=0}^{M-1} w(kS-n)} \tag{5.4}$$

using equation (5.4) we can obtain the exact signal from the STFT if it has not been modified. Not all modified STFT's are "valid" STFT's, in that some patterns are not possible signals, especially if the STFT has overlaps.

5.2 The spectrogram

A spectrogram is a way of looking at the frequency content of a 1D signal over a given timelength. If the magnitude of the short time Fourier transformed segments are placed as columns in a matrix, this matrix can be displayed as an image, with a color scale indicating the magnitude at the different frequencies (see Figure 5.2) . The magnitude of the STFT is written as:

$$|X(kS,\omega)| \tag{5.5}$$

Note that some people use the squared magnitude $|X(kS,\omega)|^2$ to describe the spectrogram [37]. In this thesis, the magnitude will be used, unless noted otherwise. The length of the Fourier transform when doing the STFT determines the spectral/frequency resolution (resolution of the y-axis). If the length of the FFT, Ω, is longer than the window length, L, which can be achieved through zero padding, a more smooth spectral resolution can be obtained. The window length L determines the temporal/time resolution

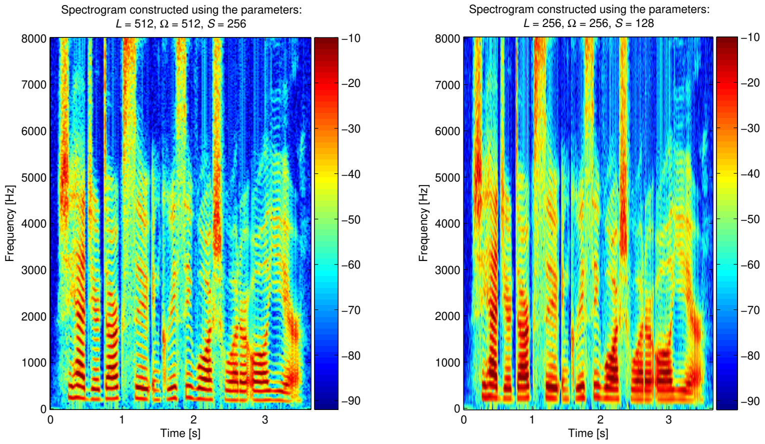


Figure 5.2: Spectrograms made from the same speech signal, using different sets of parameters. The increased resolution as a result of increased FFT, window length and step size can easily be verified. Note that the color scale used to show amplitude is in dB, so what we plot is in fact $20 \cdot \log_{10}(|X(kS, \omega)|)$

(resolution of the x-axis). If the signal is not very long, it can be hard to obtain both a good spectral and temporal resolution. The temporal resolution can be improved however, by using overlapping windows. So instead of splitting the signal in steps of L, it would be possible to use steps of step size, S, so $L - S$ would be the number of overlapping samples between neighbouring segments.

The Fourier transform consists of complex numbers containing a real and an imaginary part which together contain information of the frequency content as well as the phase of the corresponding audio signal. In detail, the magnitude of the real and imaginary parts of the Fourier transform accounts for the frequency content of the signal, while the angle between the real and imaginary parts of the Fourier transform accounts for the phase.
As we have just explained, however, a spectrogram only contains the magnitude of these complex numbers, which means that is holds no phase information. Therefore, when we seek to modify a signal by making changes in the spectrogram, we need to either:

1. Adjust the phase in the original Fourier transforms according to the changes in the corresponding frames of the spectrogram and then recombine it with the spectrogram to form the Fourier transforms needed to make the modified signal. This is difficult and not always possible.

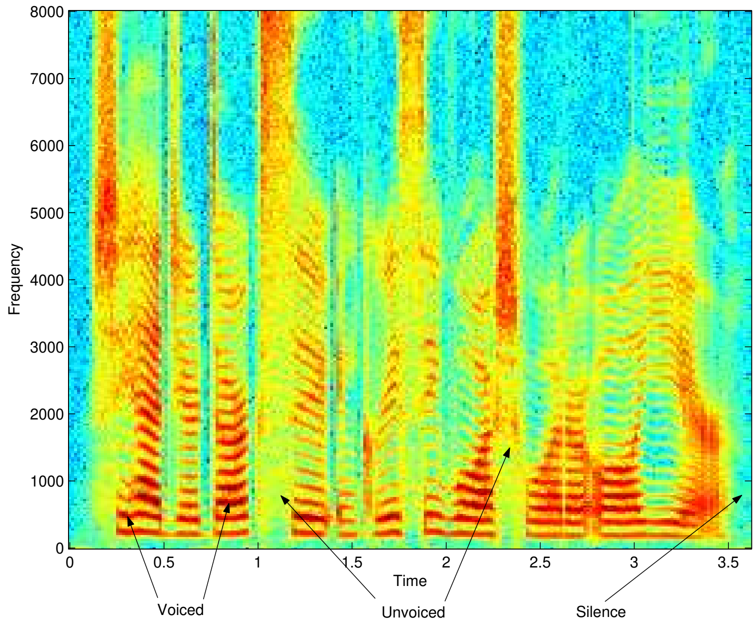


Figure 5.3: “She had your dark suit in greasy washwater all year”

2. Entirely estimate the phase from the frames in the spectrogram.

5.3 Speech in spectrograms

If we look at figure 5.3 we see voiced and unvoiced parts of the speech as discussed earlier. In the voiced parts we clearly see the harmonics as the red lines on top of each other. Looking at the harmonics we can see patterns of concentrations of energy called formants. This is a very distinct characteristic, or spectral cue, of human speech [1]. The unvoiced parts are clearly seen as almost noise, covering a large portion of the spectrum. Periods of silence is shown as blue areas indicating low values. Note that there are periods of silence in speech too, although we do not hear them. Also note that almost all the speech information is contained within 4 kHz, which is why normal landline phones are sampled at 8 kHz.

Chapter 6

Signal and spectrogram evaluation

6.1 Signal-to-noise ratio

It is important to define the signal-to-noise measure that will we be using in the experiments. During the experiments of signal reconstruction from the unmodified spectrogram, it makes sense to use the power signal-to-noise between the original signal x_0 and the reconstructed x_r:

$$SNR_{signal} = 20 \cdot log_{10} \left(\frac{\| x_0 \|_2}{\| x_0 - x_r \|_2} \right) \tag{6.1}$$

Equation 6.1 was also used in [2] to evaluate signal quality when reconstructing the signal from the spectrogram. The SNR_{signal} can be thought of almost as a sample by sample comparison.

If the spectrograms are modified, 6.1 will not be a good measure for the signal-to-noise ration since the original time-signal will sometimes not be available. In those cases, the difference between the spectrograms themselves are used:

$$SNR_{spectrogram} = 20 \cdot log_{10} \frac{\sum_{\omega=0}^{\Omega-1} \sum_{k=0}^{M-1} |X_w(kS, \omega)|^2}{\sum_{\omega=0}^{\Omega-1} \sum_{k=0}^{M-1} \left(|X_w(kS, \omega)| - |R_w(kS, \omega)| \right)^2} \tag{6.2}$$

where $|X_w(kS,\omega)|$ are columns in the original spectrogram and $|R_w(kS,\omega)|$ are columns in the reconstructed spectrogram. Equation 6.1 was also used by [12] to evaluate the quality of the spectrogram reconstruction. [10] uses another error measure, a distance measure, to evaluate the error in "distance" between two spectrograms. This error will also be shown for comparative purposes as the mean square error:

$$MSE = 20 \cdot log_{10} \frac{\sum_{\omega=0}^{\Omega-1} \sum_{k=0}^{M-1} \left(|X_w(kS,\omega)| - |R_w(kS,\omega)|\right)^2}{M \cdot \Omega} \tag{6.3}$$

Where $M \cdot \Omega$ is the total number of bins in the spectrogram, as M is the total number of frames in the spectrogram, and Ω is the Fourier transform length. When comparing the different algorithms for signal reconstruction, the three error functions will be presented. Note that the MSE error may not say more than the $SNR_{spectrogram}$, but can be used to compare directly with [10] that use this measure.

6.2 Peceptual sound quality evaluation (PQE)

Since we have no special algorithm dedicated for the measurement of perceptual sound quality, we introduce a grade system from 1-5 with the definitions as follows:

1. The reconstructed speech is not intelligible and it is even difficult to hear if someone is talking.
2. The reconstructed speech sounds like speech but is not intelligible due to heavy noise.
3. The reconstructed speech is intelligible but is affected by irritating noise or artifacts.
4. The reconstructed speech is intelligible and may contain only weak noise or artifacts. It is still distinguishable from the original.
5. The reconstructed sound is indistinguishable from the original.

Part II

Signal Estimation

Chapter 7

Introduction

In this part we will first describe some of the previous research in Signal estimation, and proceed to set up the theory used in the signal estimation experiments. We will then carefully describe each of those experiments in detail in terms of use of programs, data, and variables. Following this, there is a results section where we will present our results and explain them, and finally we will interpret and discuss our results in the discussion section. A summary of main conclusions from this part of the thesis can be found in section 11.7

7.1 Previous research in signal estimation

As a result of only using the magnitude of the Short Time Fourier Transform (STFT), i.e. the spectrogram, for our VQ approach, the phase information in the synthesised spectrogram is more or less lost. In order to reconstruct the time signal from the spectrogram alone, the phase (or the original signal) must therefore be estimated. Research in this area has been fairly sparse since the first practical algorithm for signal estimation by Griffin & Lim using alternate convex projections between the time-domain and the STFT domain, was introduced in 1983 [10], which however was similar to an earlier algorithm by Fienup'82 [9]. In the last few years, research has picked up due to the growing popularity of using time-frequency masking for source separation, noise removal and time-scaling, and this has resulted in more advanced algorithms such as the probabilistic inference approach by Kannan'04 [12] or the very recent algorithm by Bouvrie'06 [2] which treats the solution to the problem as a solution to a system of non-linear equations.

Although research in entirely new ways to estimate a signal from its STFT magnitude did not receive much attention until recently, a great amount of research and several algorithms have been created over the years to improve on the algorithm by Griffin & Lim or find new ways to reconstruct a changed STFT. These include among others the Syncronized Overlap and Add (SOLA), Roucos '85 [21], Pitch Syncronized Over and add (PSOLA), Valbret '95 [26], Waveform Overlap and Add (WOLA), Verhelst '93 [20]. Allthough it is intuitive to seek an improved version of Griffins & Lims algorithm among these, almost all the algorithms are concerned with ways to improve quality of time-stretched signals. This means copying segments of the STFT to artificially extend the duration of the sound, causing the overlapping segments to start "off phase" and not aligning when simply using the step-size (overlap) of the initial STFT. As the number of overlapping segments and the time between frames in our experiments are the same both before and after the spectrogram modification, very little, if anything, is gained from using these algorithms on our problem of masked spectrogram reconstruction. Ellis'06 [7] suggests a phase vo-coder approach, that uses a combination of techniques from sinusoidal modelling and phase-vocoder algorithms to obtain a better initial estimate to the algorithm by Griffin. The inital estimate, however, was showed to be only a small improvement in SNR after a few iterations of the algorithm by Griffin.

In the following, the background theory for three of these very different approaches ([12], [10] and [2]) will be described in detail. We will start with the algorithm by Griffin & Lim.

Chapter 8

Theory

8.1 Signal estimation using Overlap and Add

The algorithm proposed by Griffin & Lim in [10] is an iterative optimization which tries to minimize the squared distance between the modified spectrogram that we are trying to estimate the phase for, and the resulting spectrogram from a STFT of the inverse STFT (i.e. time signal) with an estimated phase. Since the phase of each frequency index in the STFT causes contructive and destructive interference in the time signal when doing an inverse STFT, a signal with a wrong phase will have a very different spectrogram from the desired. Using the distance measure between these two spectrograms is therefore a good indicator for a good phase estimate.

As explained in chapter 5, the short time Fourier transform (STFT) is defined as:

$$X_w(kS,\omega) = F_t\left[x_w(kS,n)\right] = \sum_{n=-\infty}^{\infty} x_w(kS,n)e^{-j\omega n} \tag{8.1}$$

Where $x_w(kS,n) = w(kS-n)x(n)$. The magnitude of equation 8.1 is $|X_w(kS,\omega)|$, which placed as the columns in a matrix for all $k \in [0..M-1]$, forms the spectrogram. If we define $|Y_w(kS,\omega)|$ as the columns in the modified spectrogram, or magnitude STFT (MSTFT), then the distance measure between the timesignal $x(n)$ and $|Y_w(kS,\omega)|$, can be described as:

$$D\left[x(n), Y_w(kS,\omega)\right] = \sum_{k=-\infty}^{\infty} \frac{1}{2\pi}\int_{\omega=-\pi}^{\pi} |X_w(kS,\omega) - Y_w(kS,\omega)|^2 \tag{8.2}$$

Where $X_w(kS,\omega)$ is the STFT of $x(n)$. Equation 8.2 being a function of $x(n)$ emphasizes that $X_w(kS,\omega)$ is a fully valid STFT, while $|Y_w(kS,\omega)|$ may not be valid, i.e. no real signal may exist that matches the magnitude STFT $|Y_w(kS,\omega)|$. Using Parseval's theorem, eq. 8.2 can be rewritten as:

$$D\left[x(n), Y_w(kS,\omega)\right] = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \left[x_w(kS,n) - y_w(kS,n)\right]^2 \tag{8.3}$$

[10] notes that since 8.3 is in the quadratic form of x(n), the gradient with respect to $x(n)$ can be calculated:

$$0 = \sum_{k=-\infty}^{\infty} 2 \cdot \left[x(n)w(kS-n) - y_w(kS,n)\right] \cdot w(kS-n) \tag{8.4}$$

and solving for $x(n)$ reveals:

$$x(n) = \frac{\sum_{k=-\infty}^{\infty} w(kS-n) y_w(kS,n)}{\sum_{k=-\infty}^{\infty} w^2(kS-n)} \tag{8.5}$$

Equation 8.5 can then be used in an iterative scheme descibed as followed:

An initial guess for the phase is made (random phase or zero phase is often used) and $x(n)$ is derived from Eq. 8.5 after an inverse STFT. The STFT of $x(n)$ is then obtained and the phase of this is now used for the next estimation. The new estimated phase is applied to the modified spectrogram to obtain the estimated STFT:

$$\hat{X}_w^i(kS,\omega) = |Y_w(kS,\omega)| \cdot e^{j\angle X_w^i(kS,\omega)} \tag{8.6}$$

Where i is the iteration number. The inverse STFT is now performed on $\hat{X}_w^i(kS,\omega)$ and the signal is regenerated using eq. 8.5. The process is repeated until the desired number of iterations is reached. Figure 8.1 shows a flowchart of the algorithm. Griffin & Lim [10] notes that they obtained good quality sound on test speech signals for 25-100 iterations. They also note that, although the algorithm is guarrenteed to decrease the squared error between the spectrograms, it is not a guarantee of a global minimum. In other words, local minima are possible and running the algorithm for a very high number of iterations does not guarantee to reveal the original signal.

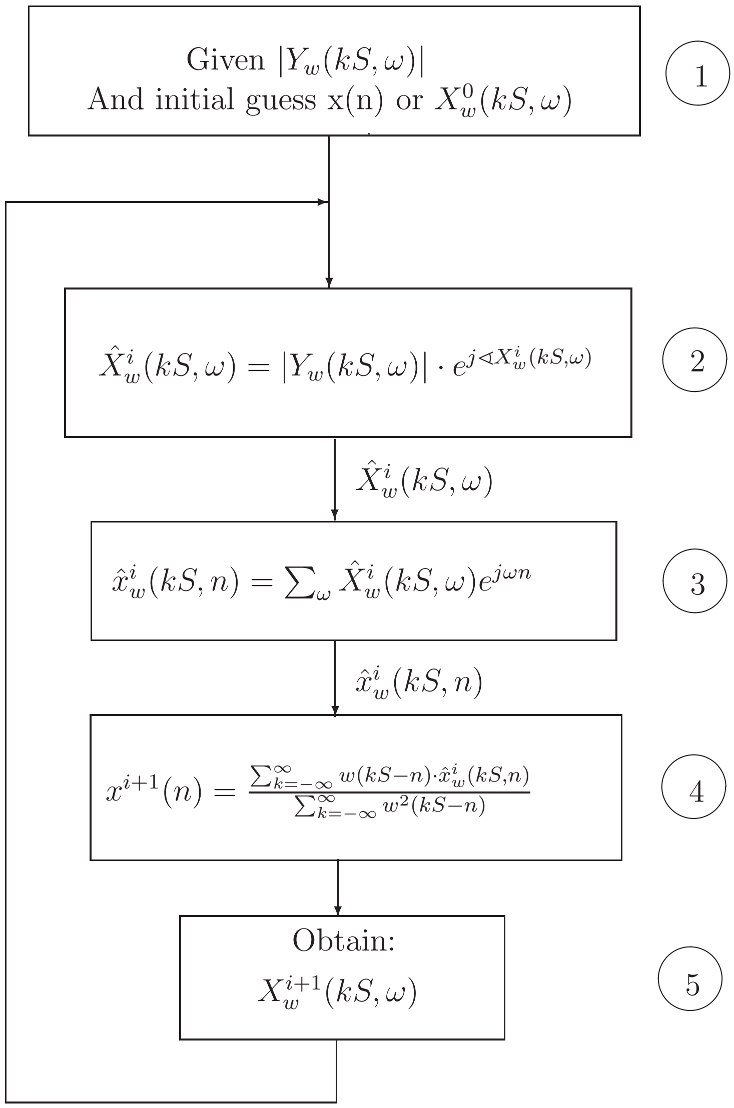


Figure 8.1: Flowchart of the signal estimation from STFT magnitude by Griffin & Lim. Starting from the top at 1. Given a modified spectrogram $|Y(kS,\omega)|$ we make an inital phase guess by constructing a STFT, $X_w^0(kS,\omega)$, with the initial phase, or obtain the STFT from the signal guess $x(n)$. 2. We apply the phase guess to the modified spectrogram, obtaining the STFT $\hat{X}_w^i(kS,\omega)$. 3. We apply the inverse Fourier transform to all the segments of the STFT. 4. We insert the segments into the Overlap and Add equation by Griffin & Lim that ensures a decrease in the distance between $|Y(kS,\omega)|$ and $|\hat{X}_w^i(kS,\omega)|$, obtaining $x^{i+1}(n)$. 5. We obtain the STFT of $x^{i+1}(n)$ and repeat steps 2-5 until the desired number of iterations has been reached.

8.2 The probabilistic model for signal estimation

A different approach to estimating the timesignal from the spectrogram alone, is to use a probabilistic model. [12] uses a conjugated gradient optimizer to maximize the likelihood of an estimated speech signal $x(n)$, given a spectrogram $\mathbf{Y}$. To make the notation easier and more clear, we will use x_n to describe the signal in this section.

$$P(\mathbf{x}, \mathbf{Y}) = P(\mathbf{x})P(\mathbf{Y}|\mathbf{x}) \tag{8.7}$$

Where $P(\mathbf{Y}|\mathbf{x})$ is the likelihood of $\mathbf{Y}$ given the signal $\mathbf{x}$, and $P(\mathbf{x})$ is the prior. For the prior, a uniform distribution can be used, but as suggested by [12] an autoregressive (AR) model can be used to apply prior information about a certain speaker and to try to minimize discontinuities at window boundaries due to phase mismatch between frames. [12] Introduces the prior as:

$$P(x) \propto \prod_{n=1}^{N} \exp\left\{-\frac{1}{2\rho^2}\left(\sum_{r=1}^{R} a_r x_{n-r} - x_n\right)^2\right\} \tag{8.8}$$

where R is the model order, a_r is the AR model coefficients, N the total number of samples and x_n the sample at sample number n. ρ^2 is a constant referring to the variance of the noise in the signal. Since this is a minimization problem, this constant can be omitted. The likelihood can be described as:

$$P(\mathbf{Y}|\mathbf{x}) \propto \prod_{k} \exp\left\{-\frac{1}{2\sigma^2}\||\hat{X}_k| - |Y_k|\|^2\right\} \tag{8.9}$$

where $|\hat{X}_k|$ is the magnitude spectrum of the estimated signal x_n at segment k. and $|Y_k|$ is the magnitude spectrum of the modified spectrogram at frame k. The idea (if using the AR model) is that the described likelihood will favor signals that are close to the modified spectrogram, and the prior will favor solutions that match the AR model.

In order to be able to optimize on the estimated speech signal using the conjugate gradient optimizer, the speech signal x_n must be explicitly described in the model. For the likelihood, using a Fourier transform matrix $\mathbf{F}$ and using the squared magnitude (power) spectrograms instead of plain magnitude (as discussed in section 5.2), we can use the fact that $c \cdot c^* = |c|^2$, where * denotes complex conjugation. This means that the power spectrum can be described as:

$$|\hat{X}_k|^2 = \mathbf{F}\mathbf{x}_k \circ \mathbf{F}^*\mathbf{x}_k \tag{8.10}$$

Where $\circ$ indicates element-wise product, and the vector $\mathbf{x}_k$ is the samples of the k'th segment of the windowed signal. Note that $\mathbf{F}\mathbf{x}_k$ is the same as a simple Fourier transform of $\mathbf{x}_k$. $\mathbf{F}^*$ is however not the same as an inverse Fourier transform but only the conjugated version of the Fourier matrix, the inverse Fourier transform would require a division by the length of the Fourier transform. Using the facts from above, expanding the norm, and taking the logarithm [12] rewrites equation 8.9 to:

$$\log P(\mathbf{x}, \mathbf{Y}) \propto \quad -\frac{1}{2\sigma^2}\sum_k\sum_i\left(\sum_{j=1}^{\Omega}\sum_{j=1}^{\Omega} F_{ij}x_{k(\Omega/2)+j}F^*_{il}x_{k(\Omega/2)+l} - |Y_{ki}|^2\right)^2 \tag{8.11}$$

Where i is the frequency index and j the length index of the Fourier matrix and Ω is the length of the Fourier transform (usually set to the length of the window in the STFT). To apply a gradient optimizer like the one by Carl Rasmussen [30], we need to have the first order derivatives for the samples x_n. The derivative of eq. 8.11 with respect to sample number u is derived from eq. 8.11. Eq. 8.11 was however modified to be able to use window functions different from a simple boxcar window. The window function is noted as w. Please see Appendix A for in detail calculations and optimization considerations for Matlab. The derivates obtained with respect to sample x_u were:

$$\begin{aligned}\frac{\partial - logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto \frac{1}{2\sigma^2}\sum_k\sum_i 2\left(\cdot\sum_{j=1}^{\Omega}\sum_{l=1}^{\Omega} F_{ij}x_{k(\Omega/2)+j}w(j)F^*_{il}x_{k(\Omega/2)+l}w(l) - Y^2_{ki}\right)\\ \cdot\sum_{j'=1}^{\Omega}\sum_{l'=1}^{\Omega} F_{ij'}w(j')\delta_{j'=u-k(\Omega/2)}F^*_{il'}x_{k(\Omega/2)+l'}w(l')\\ + F^*_{il'}w(l')\delta_{l'=u-k(\Omega/2)}F_{ij'}x_{k(\Omega/2)+j'}w(j')\end{aligned} \tag{8.12}$$

A better overview can be obtained by looking at the equation as vectors:

$$\begin{aligned}\frac{\partial logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto -\frac{1}{2\sigma^2}\cdot 2\sum_k\sum_i\left(\mathbf{F}\mathbf{x}_k\mathbf{w}\circ\mathbf{F}^*\mathbf{x}_k\mathbf{w} - |Y_k|^2\right)\\ \cdot\left(\mathbf{F}\mathbf{w}\delta_u\circ\mathbf{F}^*\mathbf{x}_k\mathbf{w} + \mathbf{F}^*\mathbf{w}\delta_u\circ\mathbf{F}\mathbf{x}_k\mathbf{w}\right)\end{aligned} \tag{8.13}$$

where δ_u is a vector with a 1 at the place where the sample u is located in frame k. If sample u is not present in frame k, then δ_u is a vector of pure zeros. $\mathbf{x}_k$ is a vector containing all the samples that corresponds to frame k of the short time Fourier transform.

If we are careful about what k's to sum over, we can optimize the equation a little to reveal:

$$\begin{aligned}\frac{\partial - logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto & \frac{1}{2\sigma^2} \cdot 2 \sum_{k_u} \sum_i \left(\mathbf{F}\mathbf{x}_k\mathbf{w} \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} - |Y_k|^2\right) \\ & \cdot \left(\mathbf{f}_u w_i \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} \ + \mathbf{f}_u^* w_i \circ \mathbf{F}\mathbf{x}_k\mathbf{w}\right)\end{aligned} \tag{8.14}$$

Where $\mathbf{f}_u^* = \mathbf{F}^*\delta_\mathrm{u}$ and $\mathbf{f}_\mathrm{u} = \mathbf{F}\delta_\mathrm{u}$ are vectors consisting of the column that corresponds to the location of sample u (x_u) in $\mathbf{x}_k$. Since the right parathesis in the equation will be zero at frames where x_u doesnt exist, we only sum over k_u which are all the frames where x_u exist.

Further optimization can be done by realising that the term:

$$\mathbf{f}_\mathrm{u} w_i \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} + \mathbf{f}_u^* w_i \circ \mathbf{F}\mathbf{x}_\mathrm{k}\mathbf{w} = 2 \cdot \Re\left\{\mathbf{f}_u w_i \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w}\right\} \tag{8.15}$$

Where $\Re\{\}$ is the real part of the resulting complex vector. This, in turn gives us the equation for the gradient as a function of vectors and matrixes:

$$\frac{\partial logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto -\frac{1}{2\sigma^2} \cdot 2 \sum_{k_u} \sum_i \left(\mathbf{F}\mathbf{s}_k\mathbf{w} \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} - |Y_k|\right)\left(2 \cdot \Re\left\{\mathbf{f}_u w_i \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w}\right\}\right) \tag{8.16}$$

In the same fashion, we can calculate the gradient of the prior with respect to the sample x_u. From Appendix A we get:

$$\frac{\partial - logP(\mathbf{x})}{\partial x_u} \propto \frac{1}{2\rho^2} \sum_n 2 \left(\sum \mathbf{a} \cdot \mathbf{x}_{n-r} - x_n\right) \left(\sum_{r'=1}^{R} a_{r'=t-u} - \delta_{r'=u}\right) \tag{8.17}$$

where $\mathbf{a}$ is a vector containing the AR coefficients of the AR model and $\mathbf{x_{t-r}}$ is a vector with the elements $[x_{t-1}, ..., x_{t-R}]$.

Equation 8.17 and equation 8.16 can be used as gradient evaluation for the conjugated gradient optimizer by [30] which was also used by Kannan [12]. The gradient optimizer uses linesearch and variable step-size which means that the algorithm will have to evaluate the function values and gradients several times before taking the next step towards a minima. For more information about the gradient optimizer we refer to [30].

8.3 Root-finding approach to signal estimation

Bouvrie et al. [2] suggests another approach to estimating the signal from the STFT magnitude alone. The problem is looked upon as a system of nonlinear equations. If we look at the first column in the spectrogram matrix, with frequency spanning the rows, and time spanning columns, the signal segment that we seek to estimate is:

$$\tilde{x}_w(k_0S, n) = w(k_0S - n)x(n) \tag{8.18}$$

If we define $|Y(k_0S, \omega)|$ as the magnitude spectrum column k_0 of the desired modified spectrogram, then we can write the equation:

$$|Y(k_0S, \omega)| = |\sum_{n=0}^{L-1} \tilde{x}_w(k_0S, n)e^{-j\omega n}|, \quad \omega = 0, ..., \Omega - 1 \tag{8.19}$$

Where L is the length of the analysis window, and Ω the length of the Fourier transform. To make the notation easier, we can write 8.19 in a similar form to what was used in the last subsection as:

$$|Y_k| = |\mathbf{F}\tilde{\mathbf{x}}_\mathbf{k}| \tag{8.20}$$

where F is the $\Omega \times L$ Fourier transform matrix and $|Y_k|$ is the magnitude spectrum at column k in the spectrogram matrix. If we move $|Y_k|$ to the other side of the equality sign in eq. 8.20, we get:

$$G(\tilde{x}) \equiv |\mathbf{F}\tilde{\mathbf{x}}_\mathbf{k}| - |Y_k| = 0 \tag{8.21}$$

As we can see in 8.21, finding $\tilde{x}$ is the same as solving the numerical root-finding problem. Since we need L linearly independent equations to solve the L unknowns in $\tilde{\mathbf{x}}$, and the Fourier transform only provides $\Omega_{independant} = \Omega/2$ points (since the Fourier transform is symmetric for real signals), the length of the Fourier transform in the original STFT must be at least $\Omega = 2 * L$. We will then only be using the positive frequency part of the spectrogram for solving the equations, since for any real signal, the spectrogram is symmetric.

Although we now have the number of equations needed for a solution to the system of non-linear equations, there are many possible solutions, including a great deal that does not obtain the original signal. To constrain the algorithm from diverging, two smoothing constraints is suggested by [2]. The first is the fact that using overlapping windows means that part of the signal in segment k, must also be in segment $k+1$. This means that we should only estimate the samples that are not overlapping with the last segment, and keep the overlapping samples fixed. The next constraint is the inital guess of the non-overlapping samples to be estimated. [2] suggests linear extrapolation (prediction) from the last p points of the part of the overlapping samples. As the step size S between segments become larger, a linear model begins to give poor estimates of the remaining part of $\tilde{\mathbf{x}}_{k+1}$. Another possibility is to use zero-order hold, that is, copying the last sample of the overlapping part to the entire h elements of the initial estimate. A good estimate would most likely be to use both methods, predicting the first few values, and holding the last one for the rest of the values. As a last constraint, the signal is assumed possitive to avoid phase sign errors. This means adding a constant factor to the DC elements of the spectrogram, or simply working with a non-negative version of the signal $x(n)$ by: $x(n) = x(n) - min(x(n))$

Since we only solve for a subset of the samples in the segment $\tilde{\mathbf{x}}_k$ we need to solve a non-square system of nonlinear equations, as the number of unknowns are smaller than the number of equations. As described by [2], the problem of solving p nonlinear equations of q unknowns can be solved by treating it as a locally linear squares problem. [2] arrives at the linear minimization problem:

$$\begin{aligned}\mathbf{x}_{k+1} = argmin_{x\in\mathbb{R}}\{&f(\mathbf{x}_k)^T f(\mathbf{x}_k) + 2(\mathbf{x}-\mathbf{x}_k)^T J(\mathbf{x}_k)^T f(\mathbf{x}_k) \\ &+ (\mathbf{x}-\mathbf{x}_k)^T J(\mathbf{x}_k)^T J(\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k)\}\end{aligned} \tag{8.22}$$

Where $J(\mathbf{x}_k)$ is the Jacobian matrix of the samples in segment k, and $f(\mathbf{x}_k)$ can be substituted with equation 8.21. Setting the derivative of this to zero gives us the recursive solution:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (J(\mathbf{x}_k)^T J(\mathbf{x}_k))^{-1} J(\mathbf{x}_k)^T f(\mathbf{x}_k) \tag{8.23}$$

8.23 is also known as the Gauss-Newton solution to the nonlinear least-squares problem. The Jacobian need not be evaluated analytically, but can be estimated numerically [3].

The algorithm can now be explained as follows. The algorithm steps through the STFT magnitude matrix column by column first in the forward direction, then in the backward direction:

1. Set the first initial guess of the overlapping samples for the first segment to $\tilde{\mathbf{x}}_{ol} = \text{rand}(\mathcal{U}\,[0,1])$

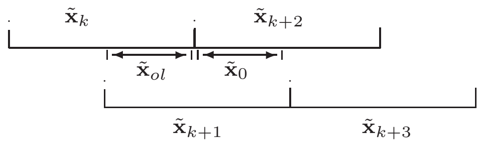


Figure 8.2: The halfsquares all noted by $\tilde{\mathbf{x}}_k$ to $\tilde{\mathbf{x}}_{k+3}$ indicate the windows of a windowed signal with 50% overlap. $\tilde{\mathbf{x}}_{ol}$ is the overlapping points between $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}_{k+1}$. $\tilde{\mathbf{x}}_0$ initial guess for the remaining S points to be estimated. After the points in $\hat{\mathbf{x}}_{est}$ have been estimated from $\tilde{\mathbf{x}}_0$, the algorithm moves on to $\tilde{\mathbf{x}}_{k+2}$ and now uses $\tilde{\mathbf{x}}_{est}$ as the overlapping points $\tilde{\mathbf{x}}_{ol}$ for the next segment and so on.

2. Let $\tilde{\mathbf{x}}_{ol}$ be the $L - S$ overlapping points that overlaps with $\tilde{\mathbf{x}}_{k+1}$ (If at first segment see step 1)

3. Compute the elements of the S long $\tilde{\mathbf{x}}_0$ by extrapolation (Linear or zero-order hold or combination) from the last p points of the overlapping points in $\tilde{\mathbf{x}}_k$

4. Find solution $\hat{\mathbf{x}}_{\mathbf{est}}$ to $|\mathbf{F}\tilde{\mathbf{x}}_{k+1}| - |Y_{k+1}| = 0$ using Gauss-Newton iteration with initial condition $\tilde{\mathbf{x}}_0$. Keep overlapping points fixed so that $\tilde{\mathbf{x}}_{k+1} = \left[\tilde{\mathbf{x}}_{ol}^T\ \hat{\mathbf{x}}_{\mathbf{est}}^{\mathbf{T}}\right]^T$

5. Set $\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_{k+1} - \text{mean}\left[\tilde{\mathbf{x}}_{k+1}\right] + \frac{|Y_{k+1}(0)|}{L}$

6. Repeat step 2-5 for all columns in the STFT magnitude matrix, $|Y_k|, k = 1, ..., M - 1$

7. Repeat step 2-5 going backwards through all the columns $|Y_k|, k = M, ..., 2$ STFT magnitude matrix. Extraplolate in the opposite direction so that $\tilde{\mathbf{x}}_{k-1} = [\hat{\mathbf{x}}_{\mathbf{est}}^{\mathbf{T}}\ \tilde{\mathbf{x}}_{ol}^T]$. Where $\tilde{\mathbf{x}}_{ol}$ now are points in $\tilde{\mathbf{x}}_{k-1}$ that overlap with segment $\tilde{\mathbf{x}}_k$

8. Reconstruct signal by overlap-adding segments $\{\tilde{\mathbf{x}}_k\}_{k=1}^{M}$

Chapter 9

Methods

9.1 Test data

For the signal estimation experiments we used speech data from two databases, the NIST TIMIT corpus and the Interspeech 2006 source separation challenge database [35]. The NIST TIMIT corpus consists of high quality speech sampled at 16 kHz. From TIMIT we mainly used two randomly chosen speakers, a female voice uttering the words “She had your dark suit in greasy washwater all year” (sa1.wav from the fpad0 folder in the dr6 dialect folder), and a male voice uttering the same words (sa1.wav from the mabc0 folder in the dr6 dialect folder). This way we can see if gender has any effect on the reconstruction. To make sure that the main conclusions made from our experiments are not database dependant, we also used a male speaker from the Interspeech 2006 source separation challenge database. This also high quality speech signal is sampled at 25 kHz, and the words “bin blue F two now” are uttered. Before use, all the signals were normalized with zero mean and scaled to unity. For the most of our experiments we will mainly be using an analysis window of 32 ms and an overlap of 16 ms. This is done to capture the harmonic structure of the sound in the spectrogram. Since we need to do vector quantization on the spectrogram itself, we need a good quality spectrogram. The same analysis window length was also used in [7] for the same reasons. We will be evaluating the performance on other analysis window lengths and overlaps to see if this choice is an advantage or a disadvantage.

9.2 Signal estimation from a spectrogram

In the following sections and subsections, we will go through the details of our experimental setups for each of the three signal estimation algorithms. To get a good overview of the performance of the algorithms, all three will be evaluated on the following:

- Reconstruction performance as a function of inital signal estimate. (Random, or zero inital phase)
- Reconstruction performance on unmodified spectrogram. Evaluated on the SNR's as well as visual and auditory percieved appearance
- Reconstruction performance as a function of different overlap- and STFT analysis window length.

Since the setup is a little different for each of the three algorithms, we will go through each of the three performance considerations displayed above for every signal estimation algorithm. In the following sections each of the experiments carried out are described in detail.

9.2.1 Inverse STFT with Zero and Random phase

These experiments will show the effect of simply using the inverse short time Fourier transform (eq. 5.4) to reconstruct signals while using random phase and zero phase for the missing phase information in a spectrogram. They will also help to give an indication of the performance of the three signal estimation algorithms, as we we will examine the SNR's and perceptual sound quality (PQE) in both cases. The experiment will also give an illustration of how important the phase information in the STFT is for intelligibility andfor the quality of the reconstructed sound.

The result will be shown as a comparison of the three signals for original-, zero- and random-phase. To illustrate the changes that occur in the spectrogram when changing the original phase in the STFT, a spectrogram will be presented from the recontructed signal of the TIMIT female speaker.

The experiments that follow will use the same random phase estimate that was used for reconstructions in this experiment (unless noted otherwise) in order to be able to directly compare the signals. Due to the fact that different choices of random phase leads to different SNR values, we will also make a small statistical analysis on the SNR's for the three different signals. The random phase estimate is repeated for 500 runs, and we will then present the mean, minimum and maximum values of the SNR's and show the standard deviation. This way we can show that our random phase choices are not very different from the mean, and show how high SNR's can get by just guessing at a random phase. The experiment is carried out as follows:

Zero Phase

For the zero phase, we will simply do the inverse Fourier transform of the spectrogram itself.

Random Phase

For the random phase, we will create a random phase P_{rand} from uniform white noise and add this to the positive frequency part of the spectrogram by:

$$S(kS, \omega_+) = |S(kS, \omega_+)| \cdot e^{j \sphericalangle \mathbf{P}_{rand}} \tag{9.1}$$

This positive part of the STFT is then flipped around $\omega = 0$ and conjugated. If we had used random phase on the entire positive and negative spectrogram, the resulting STFT would not have been symmetric and the signal would be complex. After applying the new phase to the spectrogram by eq. 9.2.1, the inverse short time Fourier transform (eq 5.4) is used to obtain the reconstructed signal.

9.3 Signal Estimation using Overlap and Add

The algorithm by Griffin & Lim was implemented in MATLAB (*STFTmest.m*), and the script uses the matlab files *spectro.m* and *invspectro2.m* for creating the STFT and the inverse STFT.

9.3.1 Performance as function of initial phase estimate

This experiment will show the effect of using different inital signal estimates for the algorithm by Griffin & Lim. This is done in order to determine whether random or zero phase is the best starting estimate for the reconstruction. We will try to obtain the best possible reconstruction by using the unmodified spectrogram, applying zero or random phase, obtaining the signal and using it as inital estimate for the algorithm. We will be using the TIMIT female and male signals, and the experiment will be carried out as follows:

Zero phase

We simply feed the zero phase signal (spectrogram only) to the Griffin algorithm.

Random phase

The random phase is applied by:

$$S(kS, \omega_+) = |S(kS, \omega_+)| \cdot e^{j \sphericalangle P_{rand}} \tag{9.2}$$

and the spectrogram is mirrored and conjugated to get the right symmetry. The results will be presented as the the number of iterations of the algorithm as a function of the error functions, SNR on the reconstructed signal (SNR_{signal}, SNR on the spectrograms ($SNR_{spectrogram}$), and MSE on the spectrograms. Since the random phase estimate is a uniform random variable, the results will be averaged over 200 runs of different random phases and furthermore the results will be plotted for 500 iterations.

9.3.2 Performance on unmodified spectrogram

We intend to evaluate the performance of the algorithm by comparing the reconstructed spectrogram to the original, and by evaluating the perceptual sound quality of the reconstructed speech signal. Since the spectrograms that need to be reconstructed in the end part of this thesis, are masked spectrograms that have been patched by a clustering algorithm, we want the reconstructed spectrogram to look as close to the modified (or original) as possible. In this experiment we examine the visual and audial perceptual quality of the result of using the algorithm by Griffin & Lim to reconstruct a signal from the spectrogram only.

We will be evaluating on all three test signals. An analysis window of 32 ms with an overlap og 50% will be used. For initial phase we will use the random phase estimate from the zero and random phase experiment in section 9.2.1. The algorithm will be run for three different iteration lengths, 10, 100 and 500. To show the visual changes on the signal and on the spectrogram of the reconstructed signal, we will be evaluating these on the TIMIT female signal. The perceptual sound quality will also be evalutated for all three signals, shown as PQE (see section 6.2).

Artifacts

We will show an example of the algorithm diverging when using a zero phase estimate on a sine wave. This example is discussed in the discussion chapter.

9.3.3 Performance at different overlaps and window lengths

In order to evaluate at which overlap and with which analysis window the algorithm performs the best, we will perform an experiment where we adjust the overlap and window length and run the algorithm for each. We will do these two experiments for the TIMIT female speaker signal, and the Interspeech male speaker signal. The experiments will be carried out in the following way:

Overlap

The TIMIT female signal will be analysed with a 512 sample window (32 ms) and the overlap will be adjusted from 10 to 500 samples for each run with 100 iterations. The result will be plotted as the number of overlapping samples, as a function of the three error functions, $SNR_{spectrogram}$, SNR_{signal} on the reconstructed signal, and MSE on the spectrogram. To evaluate the perceptual sound quality, we will do reconstruction at 25%, 50% and 75% overlap and evaluate on these signals. The algorithm will be started with the random phase estimate from section 9.2.1. The perceptual sound quality experiment will be repeated for the Interspeech male signal using an 800 sample window (32 ms).

Analysis window length

In this experiment we will use the same signal as above. The overlap will be fixed at 50% and the analysis window size will be adjusted from 64 to 1024 samples (4 to 64 ms). The SNR_{signal} and $SNR_{spectrogram}$ will then be shown, and the signals will be perceptually inspected. The same discussion on $SNR_{spectrogram}$ from above applies here.

9.4 Signal estimation using Probabilistic Inference

The signal estimation algorithm by Kannan was implemented in MATLAB. The source code, along with a description of the functions can be found in Appendix B (*kannan.m*). *Kannan.m* calculates the function values and derivatives given a signal estimate. We used the conjugated gradient optimizer by Carl Rasmussen [30] to optimize the initial signal estimate, this can be found in *minimize.m*. Since the derivatives were derived and optimized by hand, we checked our implementation numerically against a small example using Matlab's symbolic calculator. The check confirmed that the derivatives and function values were correctly implemented.

9.4.1 Performance as function of initial phase estimate

This experiment will show the effect of using different initial signal estimates for the algorithm by Kannan et al. [12]. We will obtain the reconstruction by using the unmodified spectrogram, applying zero and random phase, obtain the signal and using it as initial estimate for the algorithm.

The test will be performed on the female and male signal from the TIMIT database, using an analysis window of 512 samples with a 50% overlap. The algorithm will then be run for 200 iterations and the results in form of the $SNR_{spectrogram}$ and MSE will be plotted for a random phase initial estimate and a zero phase initial estimate. The random phase plot will be an average of 200 runs. The perceptual sound quality will be evaluated after 200 iterations on the zero or random phase estimates to see if there is any difference between the two.

9.4.2 Performance on unmodified spectrogram

In this experiment we will evaluate the quality of the reconstruction, both visually by looking at the spectrograms, by perceptually evaluating the sound quality and by the error functions. There are two ways of running the algorithm by [12] i.e with and without a pretrained AR model. We will be evaluataing on all three signals without the AR model. The initial signal estimates will be the random phase signals obtained in the experiment in section 9.2.1. To evaluate the AR model we used a small part of the male TIMIT signal due to the extreme time duration of the algorithm when using the AR model. The experiments were done in the following manner:

Without AR model

Here we will show the quality of the algorithm without using the AR model. The SNR's will be shown for all three test signals at different numbers of iterations (10, 200, and 500) and the perceptual sound quality will be evaluated.

With AR model

We trained a 12th order AR model (as used by [12]) on a different part of the signal than the part that we were recontructing. Since this is a computationally very expensive algorithm (which will be shown later), the AR model test will be performed on a smaller part of the speech file (TIMIT male). We have cut out "Dark Suit" and will use this to evaluate the quality. We will also show the difference on the convergence curves with and without the AR model. Due to the variance in the random phase estimate, the curves will be plotted as an average over 50 runs. The AR model was downweighted by a factor of 10 at the boundaries as described in [12], in order to try to minimize phase discontinuities at the boundaries of the segments in the reconstructed signal. The AR model was trained using the Yule-Walker method on the part of the signal with the utterance "Greasy Washwater". The following AR coefficients were obtained:

Table 9.1: **AR model coefficients**

Coefficient (delay)	(x-1)	(x-2)	(x-3)	(x-4)	(x-5)	(x-6)
Value	-0.9339	0.1154	-0.2391	-0.1415	0.4873	0.09853
Coefficients	(x-7)	(x-8)	(x-9)	(x-10)	(x-11)	(x-12)
Value	0.0247	-0.1617	-0.04109	-0.04929	0.1361	0.03083

The difference in perceptual sound quality will also be commented on.

Artifacts

The probabilistic algorithm proposed by Kannan shows some artifacts in the reconstructed signal. A discontinuity in the signal can be observed at every location of a boundary to a new window in the STFT. An example of this will also be shown.

9.4.3 Performance at different overlaps and window lengths

To evaluate the effect on reconstruction quality as a function of analysis window size and overlap, we will plot the error measures as a function of overlap to show the tendency on the error, as was done with the Griffin & Lim algorithm. In these experiments we will be using the TIMIT female and Interspeech male signal. The two experiments are descibed as follows:

Overlap

The overlap graphs will be shown as an overlap from 10 to 500 samples using a 512 sample window, and 100 iterations for the TIMIT female speaker. The goal is to see a tendency in the SNR's of the reconstruction, and since this algorithm is very computationally expensive, the algorithm was only run up to 100 iterations and the last part of the graphs were only evaluated for 350 400 and 500 sample overlaps.

The sound quality will be evaluated on both the Interspeech male signal and the TIMIT female at 25%, 50% and 75% overlap.

Window length

We will also try out different analysis window length, evaluate the sound quality and comment on the signal-to-noise ratio values. This is done for both the Interspeech male signal and the TIMIT female signal. The window size will be tested using windows from 4 to 128 ms and the overlap will be fixed to 50%.

9.5 Signal Estimation using Non-linear Equations

The signal estimation algorithm by Bouvrie was implemented in Matlab, *bouvrie.m*. The script further uses *funk2.m* and *funk3.m* to calculate the function values of the error function.

9.5.1 Performance as function of initial phase estimate

The algorithm by Bouvrie does not use an initial signal estimate with zero or random phase. Since it is reconstructing the next segment of the STFT based on a previous segment, the first segment requires a guess. Bouvrie [2] uses uniform random noise. This gives rise to artifacts that we will further discussed in the discussion chapter.

9.5.2 Performance on unmodified spectrogram

Since the algorithm by Bouvrie does not estimate the entire signal at each step in the algorithm, it is not possible to create a graph that displays the error function as a function of iterations. Bouvrie's algorithm is a recursive algorithm going from frame to frame in the spectrogram. The iterative part of the algorithm is the estimation of the missing samples in each segment (solving the non-linear equations) which needs to be done in an iterative approach. We used the Gauss-Newton solver *fsolve* in MATLAB to iterate to a solution, the max number of iterations was set to the default value of 100, but the algorithm often found a good solution in very few iterations. We used zero-order hold from the last overlapping samples as a initial guess for the Gauss-Newton solver.

In this experiment we will be evaluating the three test signals and compare the perceptual sound quality. We will further visually inspect the resulting signal and spectrogram of the signal from a spectrogram of the female speaker in the TIMIT database. The algorithm requires the spectrogram to be oversampled x2 to obtain the correct amount of equations to the number unknown samples. We did this by doing a L $\times$ 2 Fourier transform of the speech signal, where L is the length of the analysis window. In this case, we used 512 sample analysis window with a 50% overlap. We also forced positivity in the signal, by adding the largest negative value in the signal to the entire signal. The algorithm by Bouvrie is only described for boxcar (square) windows. We intend to show what effects this has and discuss it later in the discussion section.

In order to be able to compare the reconstruction of the spectrograms with the reconstruction done by the two other algorithms, the $SNR_{spectrogram}$ will be based on the difference between the spectrogram of the original signal with a 32 ms analysis window with 50% overlap (the Fourier transform length equal to the window length), and a spectrogram of the reconstructed signal with the same parameter values.

9.5.3 Performance at different overlaps and window lengths

The paper by Bouvrie et al. [2] had a single graph showing the SNR_{signal} as a function of step-size. Since this is a very time consuming graph to do for larger window sizes (Bouvrie uses a window length of only 100 samples in the paper) we refer to [2] for an overlap graph which shows the same tendencies as out experiments for the Griffin and Kannan algorithm. Since our window length is very different, however, we will evaluate the sound quality at different overlaps.

Overlap

In this experiment we show the SNR's and evaluate the perceptual sound quality at 25%, 50% and 75% overlap. We evaluated on all three signals with the window length constant at 512 samples (32 ms) for the TIMIT speech signals, and at 800 samples (32 ms) for the Interspeech signal.

Window length

The effects of using different analysis window length in the STFT will also be shown for the algorithm. The window sizes will again be varied from 4 to 64 ms and the perceptual quality and signal-to-noise ratios will be evaluated. The experiment will be performed on the TIMIT female and Interspeech male signal. The overlap will be kept constant at 50%.

9.6 Run times of the different algorithms

To evaluate the computational requirements of the different algorithms we will show the time it takes to run the three algorithms on different lengths of signals. We will furthermore also evaluate the run time for different lengths of overlap (25%, 50% and 75%).

The result will of course be dependent on the amount of processing power on the computer the algorithm is run on, and on implementation style and programming language. For our implementation we used MATLAB and made sure to try to implement the code as optimally as possible, using fast matlab commands and matrix evaluations instead of for-loops and other structures not suited for matlab. These results may not show the most optimal timing of the algorithms (which would require real time processing) but will be a rough guide as to the amount of calculations needed in each algorithm.

The machine used was an AMD64 3500+ 2.2 GHz, with 1024 MB of RAM, running Windows XP SP 2. All time-graphs will be shown in seconds as a function of the signal length in samples.

9.7 Algorithm Comparison

As a final comparison, we will compare the results from all three methods side by side.

The aim is to obtain the best possible perceptual quality from all three algorithms in this reconstruction. We will use the 500 iteration reconstructions from the algorithms by Griffin & Lim and Kannan et. al which will also include a 12th order AR model trained on a different signal with the same speaker. We will compare the signals and spectrograms from the TIMIT female speech signal, and all three algorithms will be evaluted visually by inspecting the signals and spectrograms, and graded using our perceptual sound quality evalutation scale (PQE). The results will be discussed and concluded on in the discussion chapter.

Chapter 10

Results

10.1 Signal estimation from a spectrogram

In the following we show the results of the experiments described in the methods chapter. We will decribe in detail what we see in every result, and discuss the results and our oppinions to what may cause these results in the discussion chapter that follows.

10.1.1 Zero and Random Phase

As we see in figure 10.1, the signals have been greatly distorted by using a phase that is not the same as the original. Perceptually, it is very difficult to say which is better, random phase may sound more human-like but is more distorted than the zero phase signal.

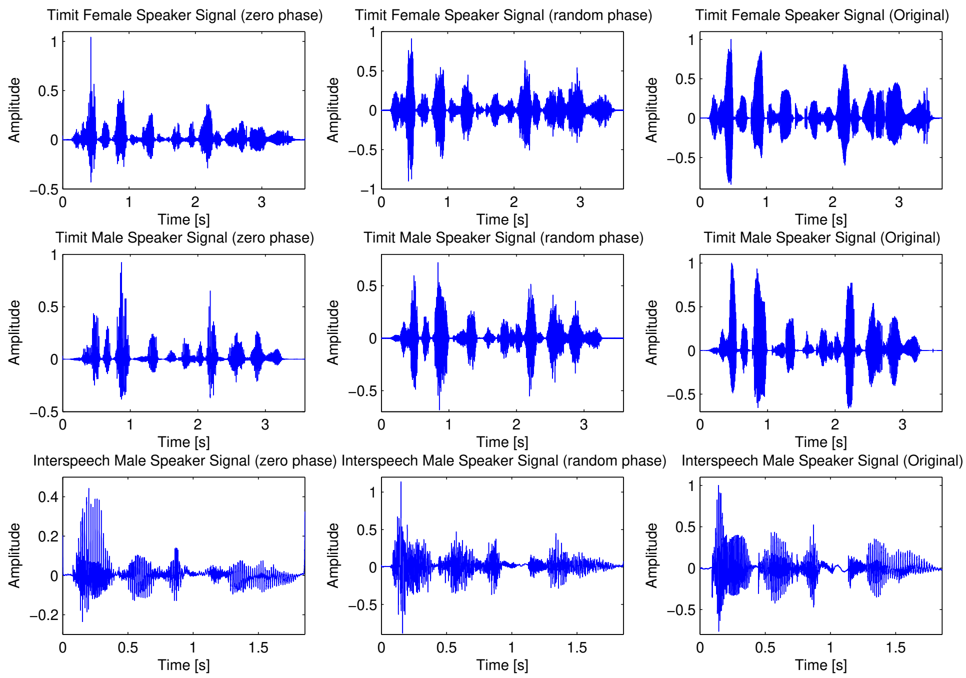


Figure 10.1: This figure shows the effect of doing an inverse short time Fourier transform on the STFT after replacing the original phase with zero- (left column) and random (middle column) phase. Both estimates seem to have some similarity with the original signal. For the zero phase estimate, the signal seems to have almost captured the positive envelope of the original signal, while the variance is wrong. It almost looks as if looking at the absolute values of the signal. Looking at the random phase estimate, we see that this time, the variance of the reconstructed signal is similar to the original. It does however not caputer the finer details in the original signal.

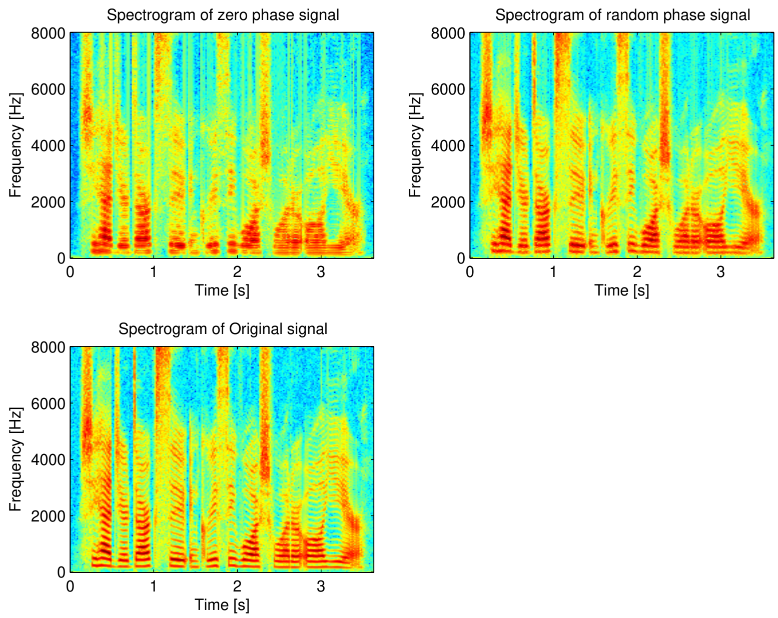


Figure 10.2: This figure shows the spectrograms of the TIMIT female signal shown in figure 10.1 (top). We clearly see differences between the spectrograms with the wrong phase (top right and top left) and the original spectrogram (bottom left). The spectrogram with zero phase seems very smeared. The harmonics, that appear clear on the original spectrogram, can however still be spotted. The random phase spectrogram is also smeared, but the resolution seems a little higher and the spectrogram captures some of the finer details in the original spectrogram.

Table 10.1: **Timit Female** - Random and Zero phase signals, window length 512, 50% overlap

Phase	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Random	-5.14 dB	17.33 dB	37.16 dB	3
Zero	-0.88 dB	4.13 dB	50.36 dB	3

Table 10.2: **Timit Male** - Random and Zero phase signals, window length 512, 50% overlap

Phase	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Random	-4.77 dB	16.63 dB	33.03 dB	3
Zero	-1.68 dB	4.55 dB	45.11 dB	3

Table 10.3: **Interspeech Male** - Random and Zero phase signals, window length 800, 50% overlap

Phase	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Random	-4.61 dB	17.38 dB	34.45 dB	3
Zero	-1.75 dB	4.45 dB	47.38 dB	3

Table 10.1, 10.2 and 10.3 show the signal to noise ratios of the different signals with random and zero phase reconstructions. In all cases, zero phase obtained the highest SNR_{signal}, while the random phase obtained the highest $SNR_{spectrogram}$. PQE was 3 in all cases. We were able to understand what was said, but sometimes with difficulty due to noise and artifacts.

Table 10.4: **Timit Female** - Random phase statistics, window length 512, 50% overlap, 500 runs

	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$
Mean	-5.09	16.94	37.54
Max	-3.49	18.65	39.57
Min	-6.55	14.92	35.84
Std.	0.56	0.61	0.61

Table 10.5: **Timit Male** - Random phase statistics, window length 512, 50% overlap, 500 runs

	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$
Mean	-5.07	17.07	32.59
Max	-3.26	18.67	34.75
Min	-6.46	14.91	30.99
Std.	0.53	0.60	0.60

Table 10.6: **Interspeech Male** - Random phase statistics, window length 800, 50% overlap, 500 runs

	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$
Mean	-5.09	17.92	33.91
Max	-2.20	20.18	36.71
Min	-7.26	15.12	31.66
Std.	0.78	0.91	0.91

Tabel 10.4, 10.5 and 10.6 show the statistics of choosing different random phases. All signals had a standard deviation of less than 1 dB, and the high and low values seem to be around 2 dB on each side of the mean for the $SNR_{spectrogram}$ and 2-3 dB for the SNR_{signal}.

10.2 Signal Estimation Griffin

This section presents the results for the experiments using the algorithm by Griffin & Lim [10] to reconstruct the speech signal from a spectrogram. The experiments are described in detail in the methods section.

10.2.1 Performance as function of inital phase estimate

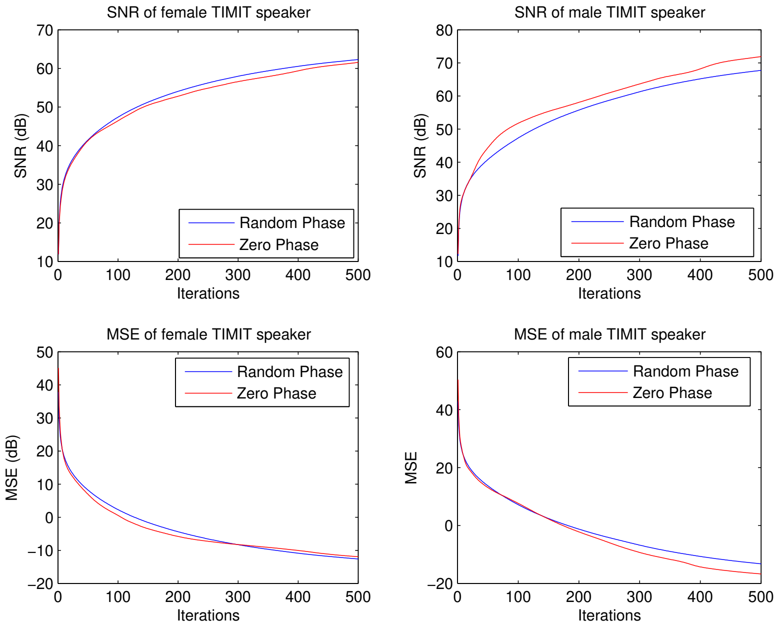


Figure 10.3: This figure shows the error functions as a function of iterations of the algorithm by Griffin & Lim, and as a function of initial signal estimates with random or zero phase. The result is different for the two different signals (left and right). If we look at the SNR on the spectrograms, we see that for the female speech signal (top left), the curves for the zero and random initial phase estimates are very close. They stay close to each other all the way to 500 iterations, with the random phase estimate on top, and the difference between to two curves are 0.7 dB. For the male signal (top right) on the other hand, the initial estimates are close in the begining but the zero phase estimate gets higher values after a few iterations. The two curves then run almost parallel and ends up at 500 iterations with a difference of 4.18 dB, this time with the zero phase estimate giving the best SNR. If we look at the MSE function for the two signals, we see the same tendency, but the crossing of the curves, happens at different iterations. Another thing to note is that at the first 10 iterations the zero phase estimate always gives lower SNR than the random phase estimate (this can be difficult to se on the figures). The reconstructed sound was indistinguishable between the two phase estimates after 100 iterations.

10.2.2 Performance on Unmodified Spectrogram

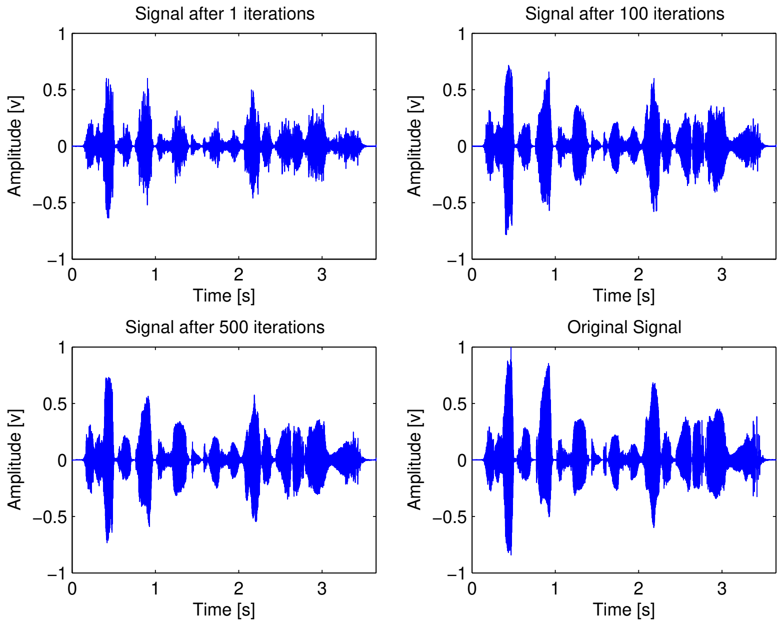


Figure 10.4: This figure shows the reconstructed signals after 10, 100 and 500 iterations of the signal estimation algorithm by Griffin and Lim together with the original signal. We see that all reconstructed signals, even for 10 iterations, has clear similarities to the original signal. An improvement on the reconstructed signal can be seen from 10 iteration to 100 iterations. From 100 to 500 iterations, improvement can still be spotted but is a little difficult to see.

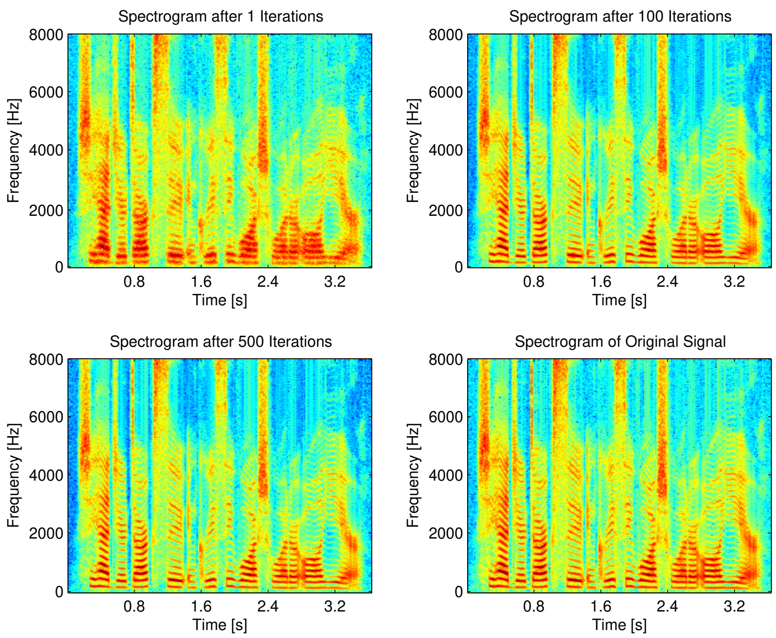


Figure 10.5: This figure shows the corresponding spectrograms to the signals in figure 10.4. We see that the spectrogram of the reconstructed signal comes closer to the original as more iterations are performed. The improvement on the spectrogram as the number of iterations increases is clear. The difference from 10 to 100 iterations is large, compared to the difference between 100 and 500 iterations. We also see that after 500 iterations the spectrogram is close to the original spectrogram, but not an exact match.

Table 10.7: **Timit Female Speaker** - Quality of different number of iterations (512, 50% overlap, random phase)

Iterations	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
10	-6.06 dB	29.27 dB	-2.49 dB	3
100	-6.49 dB	46.42 dB	-19.64 dB	4
200	-6.48 dB	53.57 dB	-26.79	4
500	-6.41 dB	65.87 dB	-39.09 dB	4

Table 10.8: **Timit Male Speaker** - Quality of different number of iterations (512, 50% overlap, random phase)

Iterations	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
10	-5.12 dB	30.69 dB	-8.0 dB	3
100	-5.01 dB	43.94 dB	-25.34 dB	4
200	-4.65 dB	50.38 dB	-31 dB	4
500	-4.56 dB	59.11 dB	-37.8 dB	4

Table 10.9: **Interspeech Male Speaker** - Quality of different number of iterations (800 50% overlap, random phase)

Iterations	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
10	-5.84 dB	31.70 dB	-6.94 dB	3
100	-6.30 dB	59.23 dB	-20.19 dB	4
200	-6.47 dB	61.65 dB	-26.63 dB	4
500	-6.53 dB	64.44 dB	-35.36 dB	4

Tables 10.7, 10.8 and 10.9 shows the signal to noise ratios of the different signals after a different number of iterations. The SNR_{signal} drops from 10 to 100 iterations in Table 10.7 but then rises a little after more iterations. In Tabel 10.8 the SNR_{signal} generally rises over all iterations, and in Tabel 10.9 the SNR_{signal} generally drops over all iterations. For all the signals, the $SNR_{spectrogram}$ kept rising for each iteration and the $MSE_{spectrogram}$ dropped. The sound quality was poor for all signals after 10 iterations. It did improve very slightly from 100 to 500 iterations, but all signals contained artifacts that made it clear that this was not the original signal.

10.2.3 Performance at different overlaps and window lengths

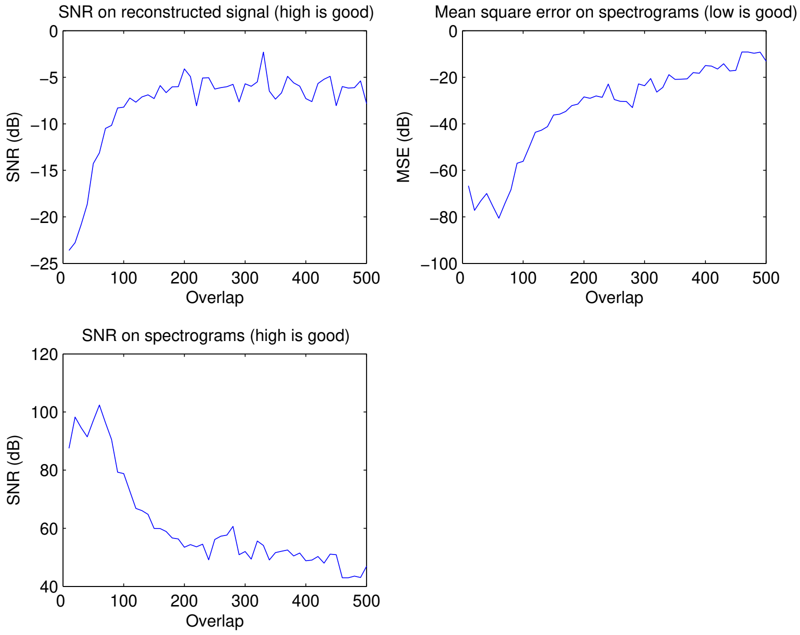


Figure 10.6: Error as function of overlap using 100 iterations, (analysis window size of 512) on male TIMIT speaker. We see that for small overlaps, less than 100 samples (20%), the SNR on the reconstructed signal (top left) is very low. The SNR is fairly constant (very small climb) as we go past the 200 sample overlap (40%). If we look at the SNR on the spectrograms we see high values for small overlaps, and the SNR in this case drops as the number of overlapping samples is increased. If we look at the MSE error, we see the same tendency, a low error at small overlaps, which increases as the overlap increases.

Table 10.10: **Timit Female Speaker** - Varying the overlap (200 iterations, window length = 512)

Overlap (samples)	128	256	384
Overlap (%)	25	50	75
SNR_{signal}	-6.83 dB	-6.48 dB	-6.69 dB
$SNR_{spectrogram}$	61.4 dB	53.57 dB	46.05 dB
PQE	3	4	4

Table 10.11: **Interspeech Male Speaker** - Varying the overlap (200 iterations, window length = 800)

Overlap (samples)	200	400	600
Overlap (%)	25	50	75
SNR_{signal}	-6.11 dB	-6.47 dB	-5.01 dB
$SNR_{spectrogram}$	76.40 dB	59.23 dB	50.42 dB
PQE	3	4	4

Tables 10.10 and 10.11 shows the signal-to-noise ratios and the evaluation of perceptual sound quality at different overlaps. In Table 10.10 . Going below 50% overlap decreased the sound quality. At a 128 sample overlap the sound was rated a 3. At 50% overlap and above, the sound quality only changed slightly, at an overlap of 384 samples (75%), the sound was still rated a 4. The same applied to the Interspeech male signal for the perceptual quality. The $SNR_{spectrogram}$ followed the tendency shown in Figure 10.6 for both signals. The SNR_{signal} was also fairly constant in both cases.

Table 10.12: **Timit Female Speaker** - Varying the analysis window size (200 iterations, 50% overlap)

$WindowSize(samples)$	64	128	256	512	1024
$WindowSize(ms)$	4	8	16	32	64
$SNR_{signal}(dB)$	-6.06	-5.82	-5.40	-6.48	-6.27
$SNR_{spectrogram}$ (dB)	49.23	53.37	46.78	53.57	50.12
PQE	3	3	3-4	4	4

Table 10.13: **Interspeech Male Speaker** - Varying the analysis window size (200 iterations, 50% overlap)

$WindowSize(samples)$	100	200	400	800	1600
$WindowSize(ms)$	4	8	16	32	64
$SNR_{signal}(dB)$	-4.46	-5.65	-5.24	-4.20	-6.73
$SNR_{spectrogram}$ (dB)	40	47.6	51,29	55.7	59.49
PQE	3	3	3	4	3-4

If we look at Tables 10.12 and 10.13 we see that the sound quality changes as the analysis window is changed from a 4 ms to a 64 ms window. The perceptual quality was best at a 32 ms window in both the TIMIT female speaker signal and the Interspeech male speaker signal. Looking at the SNR_{signal} we see that they do not seem to follow a special pattern. The $SNR_{spectrogram}$ rises in the Interspeech male signal, but show no general improvement in the TIMIT female signal. The signals would sound very distorted at the small window lengths, with irritating noise, while being fairly pleasant at 512 and 1024 sample windows. A window length of 1024 did however contain very small artifacts.

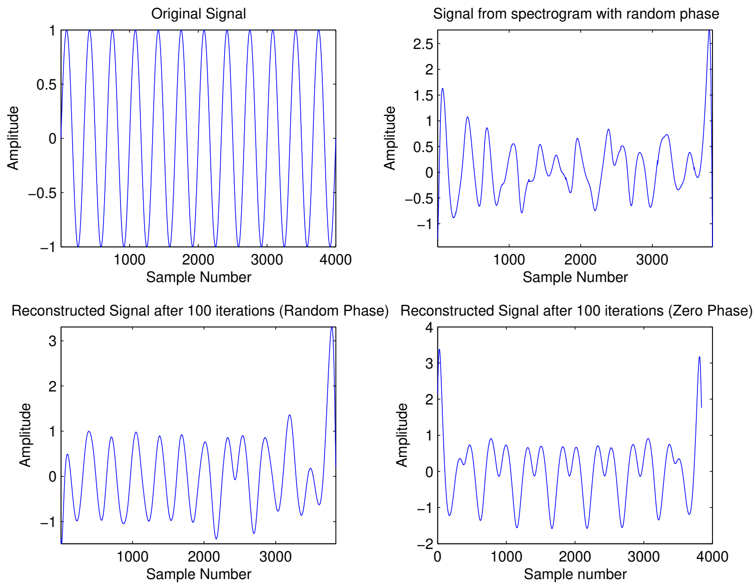


Figure 10.7: Result of doing signal reconstruction using the algorithm by Griffin & Lim. This figure displays the difference of the different initial phase guesses when working on a sine wave.

On Figure 10.2.3 we see a sine wave with a random phase applied to its STFT. The random phase estimate seem to have gotten closer to the original signal after 100 iterations. Starting from zero phase, however, the signal did not improve beyond 60 iterations, but maintained the same estimate (bottom right) after 100 iterations, where the random phase estimate continued to improve.

10.3 Signal Estimation using Probabilistic Inference

In this section we will present the results of our tests using the algorithm by Kannan et al.[12]. The details of these experiments can be found in the methods chapter.

10.3.1 Performance as function of inital phase

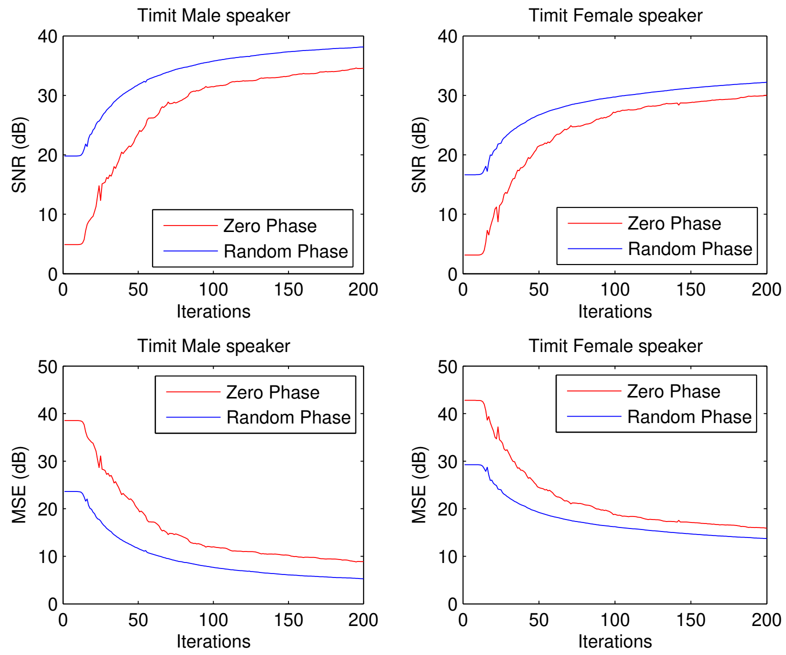


Figure 10.8: Signal reconstruction using probabilistic algorithm by Kannan at different initial phase guesses, without using a pretrained AR model. Random phase displayed is averaged over 200 runs. In this figure we see the $SNR_{spectrogram}$ and $MSE_{spectrogram}$ of the reconstructed spectrograms. The curves are not entirely smooth because the errors were evaluated at each function evaluation, and not at the current minimum of the algorithm. This was done to maintain a feeling of the number of iterations (in this case function evaluations) needed. If we look at the $SNR_{spectrogram}$ on the male speech sample, we see that the random phase initial estimate starts at a higher value than the zero phase estimate. Although the zero phase estimate seems to climb faster in the beginning, the gap is still 4.5 dB after 200 iterations. The same tendency can be seen on the female signal as well. Here the gap ends up being 2.3 dB. Looking at the MSE curves, the tendency is also the same, the zero phase estimate never comes really close to the random phase estimate. The sound quality was also different for the two initial signal estimates, with random phase being better than the zero phase.

10.3.2 Performance on unmodified spectrogram

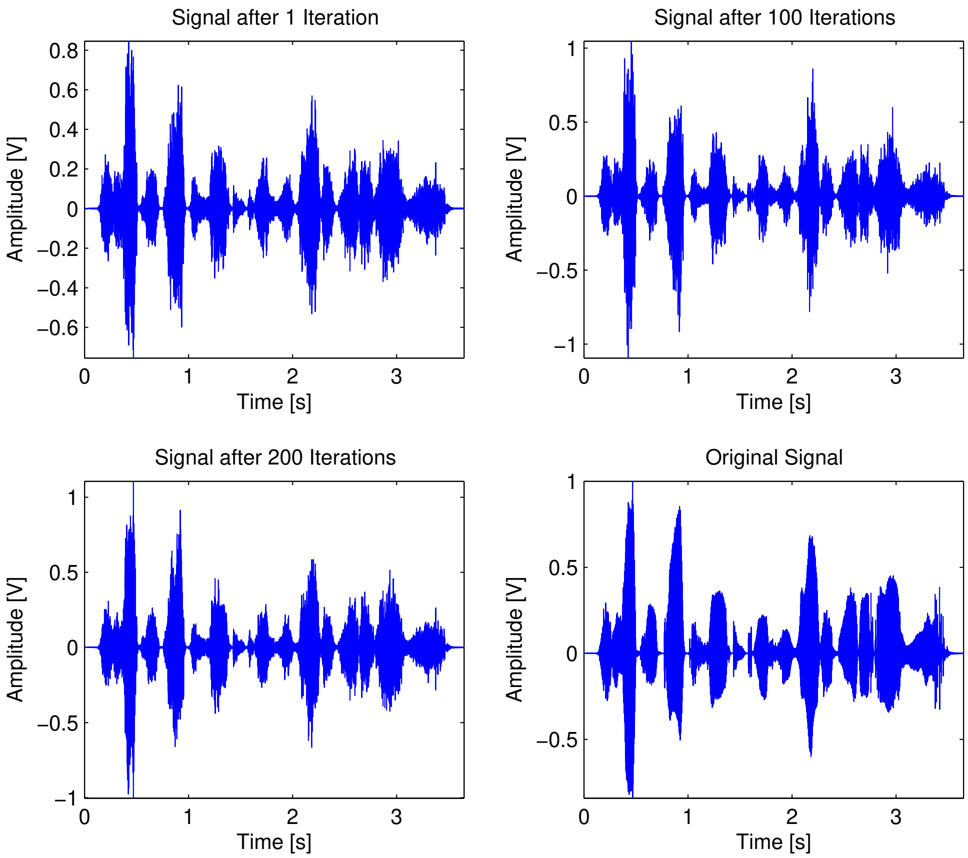


Figure 10.9: Reconstructed signals for different number of iterations. As we can see, it looks as if there are small improvement from 10 iterations to 100, and from 100 to 200. After 200 iterations the signal still has spikes that is also present on the 10 iteration signal.

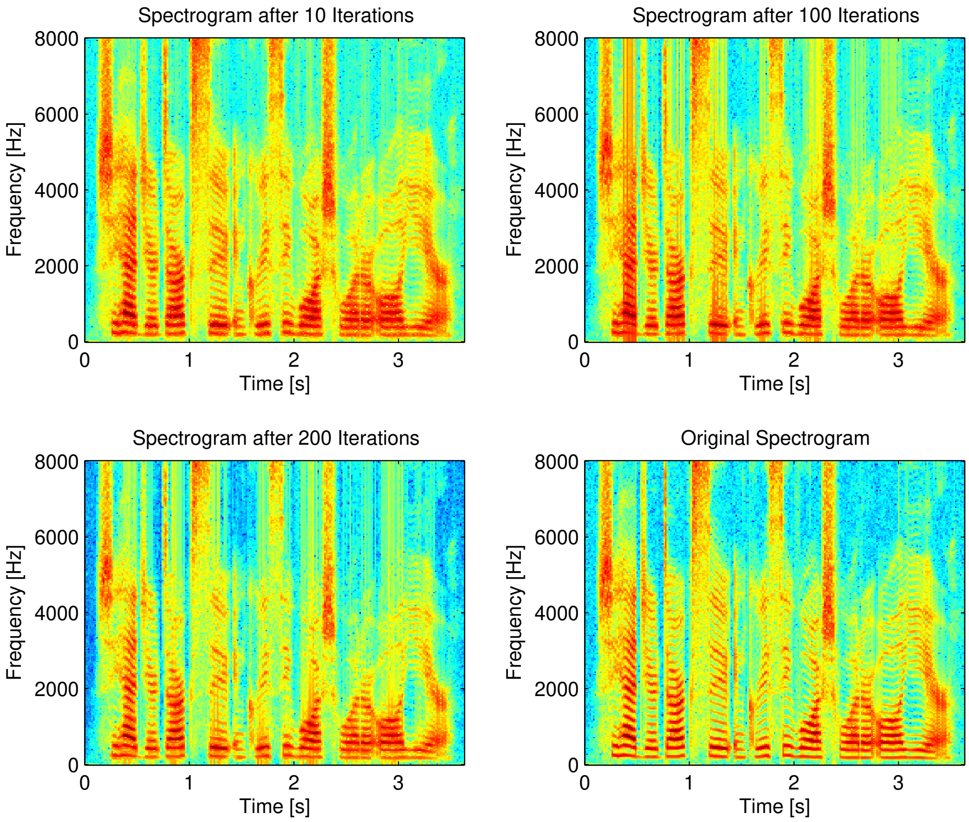


Figure 10.10: Reconstructed spectrograms for different number of iterations. After 10 iterations, the spectrogram looks coarse and smeared. We clearly see an improvement from 10 to 100 and from 100 to 200 iterations, but we also see what appears to be artifacts as the number of iterations increase. We see dark or bright lines through voiced parts of the spectrogram after 100 iterations. These lines are also present after 200 iterations, but not quite as clear.

Table 10.14: **Timit Female Speaker** - Quality of different number of iterations (512 window, 50% overlap)

Iterations	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
10	-5.15 dB	17.33 dB	37.6 dB	3
100	-6.71 dB	26.34 dB	28.15 dB	3
200	-6.51 dB	30.03 dB	24.45 dB	3
500	-6.46 dB	32.81 dB	21.68 dB	3

Table 10.15: **Timit Male Speaker** - Quality of different number of iterations (512 window, 50% overlap)

Iterations	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
10	-4.77 dB	16.63 dB	33.03 dB	3
100	-5.16 dB	27.49 dB	22.17 dB	3
200	-4.8 dB	30.34 dB	19.32 dB	3
500	-4.78 dB	34.45 dB	15.21 dB	3

Table 10.16: **Interspeech Male Speaker** - Quality of different number og iterations (800 window, 50% overlap)

Iterations	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
10	-5.08 dB	16.94 dB	34.91 dB	3
100	-7.24 dB	30.83 dB	21.02 dB	3
200	-7.53 dB	33.61 dB	18.24 dB	3
500	-7.44 dB	36.58 dB	15.21 dB	3

Table 10.14, 10.15 and 10.16, show the evaluations after different number of iterations. For the SNR_{signal} we see no special tendency. The TIMIT female signal and the Interspeech male signal seems to have dropped a little in SNR_{signal}, while the TIMIT male signal stays about the same. In all three signal reconstructions, the $SNR_{spectrogram}$ improved over all iterations. The $MSE_{spectrogram}$ also improved, by dropping at every iteration. The perceptual sound quality was not very good, with artifacts plaguing the signals. There was also only a small improvement in sound quality from 100 iterations to 500. This showed to be true for all of the three signals.

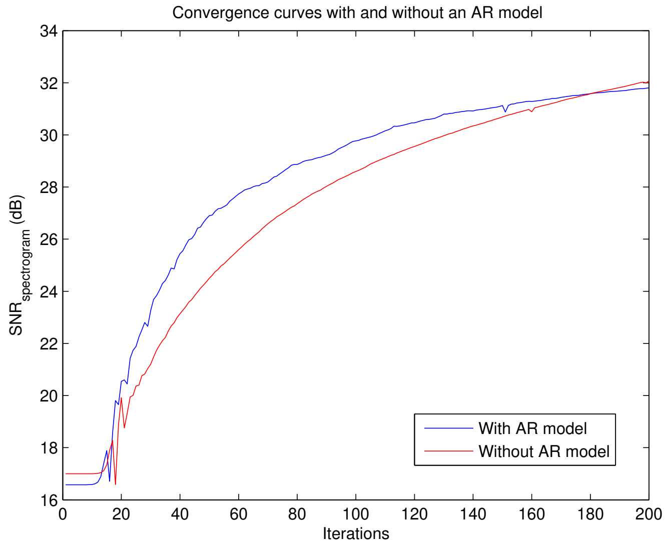


Figure 10.11: Convergence curves with and without the AR model. We see that running the algorithm with the AR model does make a difference in the convergence curves. The curve with the AR model rises faster than the curve without the AR model. This means that the error between the original and the reconstructed spectrogram is less if 200 iterations are used. Note however, that the AR model curve seems to flatten more after 200 iterations, than the curve without the AR model. The reconstruction was run at random phase, and the curves shown were averaged over 25 runs.

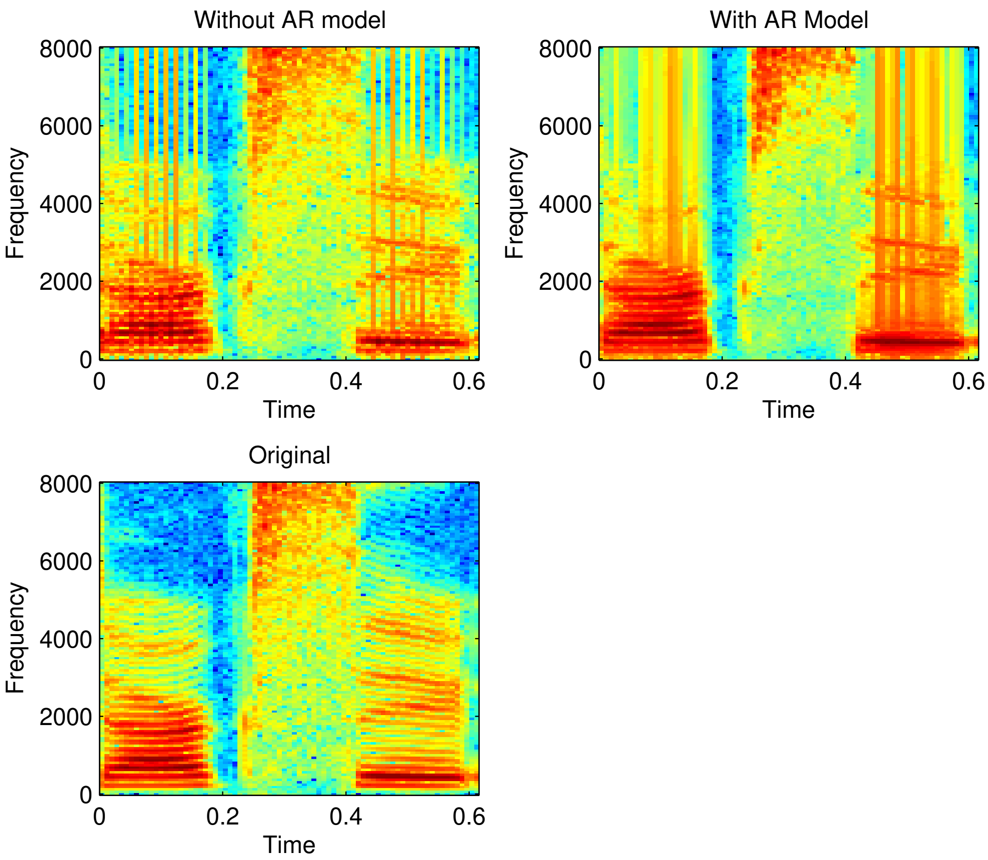


Figure 10.12: Reconstructed spectrograms with and without the AR model. This figure shows the spectrogram after 200 iterations with and without the AR model on the small TIMIT male signal clip with the utterance "Dark Suit" . As we can see, without the AR model, the spectrogram is more coarse than with the AR model. The artifacts also noted in 10.10 are clearly visible in this case. The AR model spectrogram looks more smeared, but seems to have made the harmonics more visible.

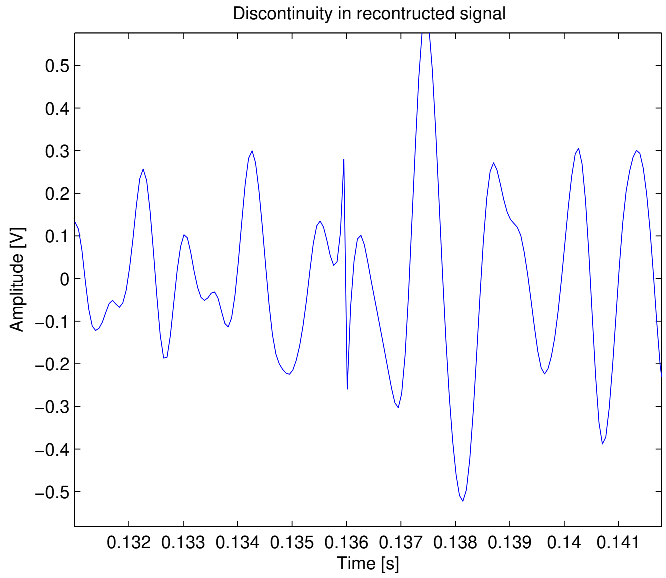


Figure 10.13: This figure shows a zoom-in of the discontinuity in the reconstructed signal. We see a clear discintinuity in the signal on this figure at around 0.136 seconds. The discontinuity appears at the location of every beginning of a new window in the STFT throughout the signal and vary in size. The discontinuity is clearly audible when playing the reconstructed signal. Note that it is not just two points out of place, there are 2-4 points on each side of the two peaks, that seem to go towards the peaks.

10.3.3 Performance at different overlaps and window lengths

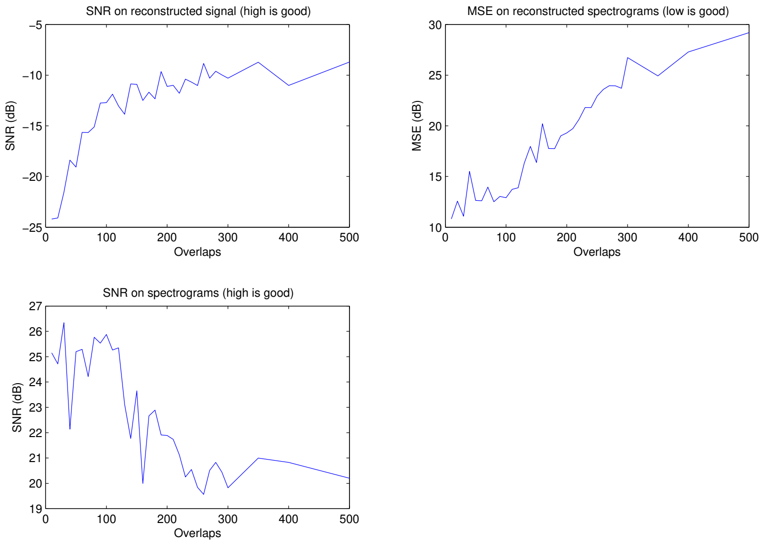


Figure 10.14: This figure shows the error functions as a function of overlapping samples. We see the same tendency as for the same graph for the Griffin algorithm. The curves are more coarse due to the fact that we only used 100 iterations to show the tendency. The SNR_{signal} rises with increasing number of overlapping samples, and the $SNR_{spectrogram}$ drops, while the MSE rises. The perceptual sound quality was also following the same tendency as in the Griffin graph. Low SNR_{signal} at low overlaps gave very poor quality sound, while the quality improved with higher overlaps.

Table 10.17: **Timit Female Speaker** - Varying the overlap (200 iterations, 512 sample window

Overlap (samples)	128	256	384
Overlap (%)	25	50	75
SNR_{signal}	-7.20	-6.51	-5.75
$SNR_{spectrogram}$	41.88	30.03	33.92
PQE	3	3	3-4

Table 10.18: **Interspeech Male Speaker** - Varying the overlap (200 iterations, 800 sample window

Overlap (samples)	200	400	600
Overlap (%)	25	50	75
SNR_{signal}	-5.23	-7.53	-7.89
$SNR_{spectrogram}$	41.88	33.61	38.88
PQE	3	3	3-4

Table 10.17 and 10.18 shows the effects of different lengths of overlap. The signals still had artifacts with a 75% overlap, but seemed to have improved slightly in quality. The SNR_{signal} seems to rise from the 25% to the 75% overlap on the TIMIT female signal, while dropping on the Interspeech male signal. The $SNR_{spectrogrm}$ does not seem to follow a special pattern and notice that it is not possible to evaluate the sound quality from these to SNR measures.

Table 10.19: **Timit Female Speaker** - Varying the analysis window size (200 iterations, 50% overlap)

$WindowSize(samples)$	64	128	256	512	1024
$WindowSize(ms)$	4	8	16	32	64
$SNR_{signal}(dB)$	-5.83	-5.75	-7	-6.51	-6.6
$SNR_{spectrogram}$ (dB)	34,97	38.62	30.3	30.03	29.7
PQE	3	3	3	3	3

Table 10.20: **Interspeech Male Speaker** - Varying the analysis window size (200 iterations, 50% overlap)

$WindowSize(samples)$	100	200	400	800	1600
$WindowSize(ms)$	4	8	16	32	64
$SNR_{signal}(dB)$	-5.93	-5.87	-3.95	-7.53	-7.57
$SNR_{spectrogram}$ (dB)	22.41	30.37	33.74	33.61	33.02
PQE	3	3	3	3	3

Table 10.19 and 10.20 shows the performance as we change the analysis window length. It is difficult to see a pattern in the signal to noise ratios. The perceptual sound quality, although all experiments were rated a 3 due the noise in the reconstructed signals, did improve from 64 to 512 samples, and worsened at 1024 samples. The tendency in quality is thereby similar to the one in Table 10.12 for the Griffin algorithm. Note that it is not obvious from the SNR's that the 32 ms window gave the best quality reconstruction. The quality of the reconstruction had the same tendency for both signals.

10.4 Signal estimation using Non-linear Equations

In this section we will present the results of our tests using the algorithm by Bouvrie et al.[2]. The details of these experiments can be found in the methods chapter.

10.4.1 Performance on unmodified spectrogram

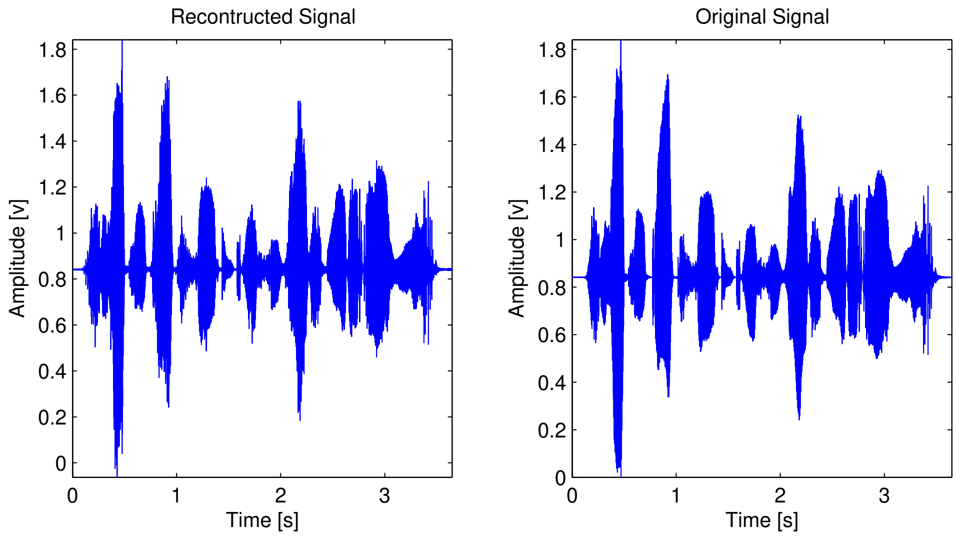


Figure 10.15: Reconstructed signals (Timit Female) after using [2]. As we can see the signals looks very similar. The reconstructed signal though seem to have a few extra small spikes compared to the original signal.

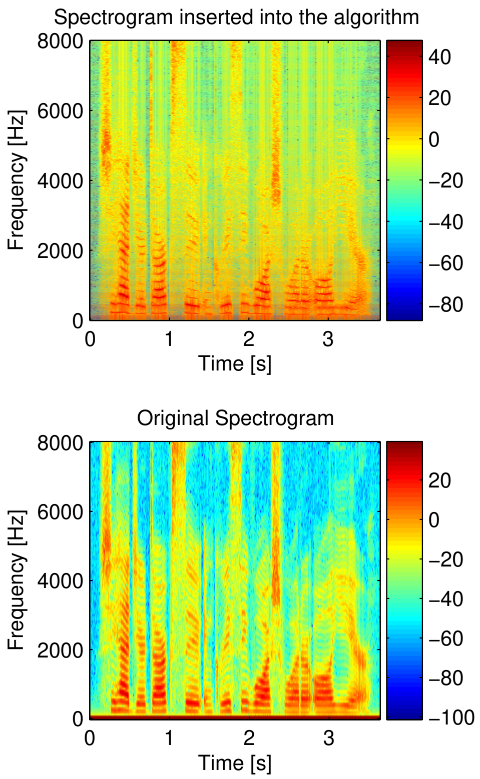


Figure 10.16: Reconstructed Spectrograms (TIMIT Female) after using the algorithm by Bouvrie et al. [2]. In the first spectrogram (top left), we see the the spectrogram being fed to the algorithm by Bouvrie. As we can see, the spectrogram seems more dark and smeared due to the fact that it was created with a square analysis window, and a Fourier transform length which was double the length of the analysis window. If we analyse the reconstructed signal by obtaining the spectrogram and compare it to the spectrogram of the original signal, we get the two spectrograms (top right and bottom left). As we can see, these two spectrograms look very similar.

Table 10.21: **Speakers and Errors.** Timit analysis window length 512, Interspeech window length 800, 50% overlap.

	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Timit Female	16.28 dB	35.19 dB	27.32 dB	3-4
Timit Male	21.56 dB	42.60 dB	15.12 dB	3-4
Interspeech Male	20.53 dB	38.32	21.51 dB	3-4

Looking at table 10.21 we see the evaluation results for the three different signals. We see very high values for the SNR_{signal} and good values for the $SNR_{spectrogram}$. The perceptual quality is good, but the signals contain artifacts. The first part of the reconstructed signal contains noise, while the last part is almost indistinguishable from the original signal. We will go into depth on this in the discussion section. The noise is present in all the three signals, but most noticable with the TIMIT female signal. Due to the noise, the signals all recieve a 3-4 PQE grade.

10.4.2 Performance at different overlaps and window lengths

Table 10.22: **Timit Female Speaker** - Varying the overlap (512 sample window)

Overlap (samples)	128	256	384
Overlap (%)	25	50	75
SNR_{signal}	3.89	16.28	50.03
$SNR_{spectrogram}$	34.67	35.19	60.59
PQE	3	3-4	4-5

Table 10.23: **Timit Male Speaker** - Varying the overlap (800 sample window)

Overlap (samples)	128	256	384
Overlap (%)	25	50	75
SNR_{signal}	6.98	21.56	156.43
$SNR_{spectrogram}$	39.30	42.60	156.91
PQE	3	3-4	5

Table 10.24: **Interspeech Male Speaker** - Varying the overlap (800 sample window)

Overlap (samples)	200	400	600
Overlap (%)	25	50	75
SNR_{signal}	6.64	20.53	121.02
$SNR_{spectrogram}$	30.05	38.32	125.08
PQE	3	3-4	5

Table 10.22, 10.23 and 10.24 show the effects of using different overlap on all three signals. All the signals followed the same tendency in both the SNR's and the PQE's. At 25% overlap, the algorithm performed poorly, the reconstructed speech was contaminated with heavy artifacts and noise. With a 50% overlap it improved, but still had noise in parts of the reconstructed signal. With a 75% overlap, perfect reconstruction was achived on both the

TIMIT male speaker, and the interspeech signal. The TIMIT female signal was also very close to the original, with only very slight noise.

Table 10.25: **Timit Female Speaker** - Varying the analysis window size (200 iterations, 50% overlap)

Window Size (samples)	64	128	256	512	1024
Window Size (ms)	4	8	16	32	64
$SNR_{signal}(dB)$	23.67 dB	14.5 dB	17.5 dB	16.28 dB	3.91 dB
$SNR_{spectrogram}$	35.64 dB	37.38 dB	41.68dB	35.19 dB	31.23 dB
PQE	3-4	3-4	3-4	3-4	3

Table 10.26: **Interspeech Male Speaker** - Varying the analysis window size (200 iterations, 50% overlap)

Window Size (samples)	100	200	400	800	1600
Window Size (ms)	4	8	16	32	64
$SNR_{signal}(dB)$	32.73 dB	21.35 dB	43.37 dB	20.53 dB	3.91 dB
$SNR_{spectrogram}$	44.60 dB	32.22 dB	62.57 dB	38.32 dB	31.23 dB
PQE	4	4	4-5	3-4	3

Table 10.25 and 10.26 show the effects of using different analysis window lengths, while maintaining the 50% overlap. All of the reconstructed signals for different window lengths contained noise in the beginning of the reconstructed signal, and the noise seems to get worse with larger window lengths. The noise apeared a bit worse on the female signal, and we seem to get better reconstruction on the Interspeech signal. The SNR_{signal} seem to go up and down, but became very low at the 64 ms window which also lead to the poorest quality sound. The best quality sound was at 16 ms for the Interspeech male speaker. Here the SNR_{signal} and $SNR_{spectrogram}$ are both high compared to the results for the other window lengths.

10.5 Run times of the different algorithms

These are the results of the analysis of complexity of the three different signal estimation algorithms. The signal length experiments were carried out on the TIMIT female speaker signal at different lengths, using a 512 window with a 50% overlap.

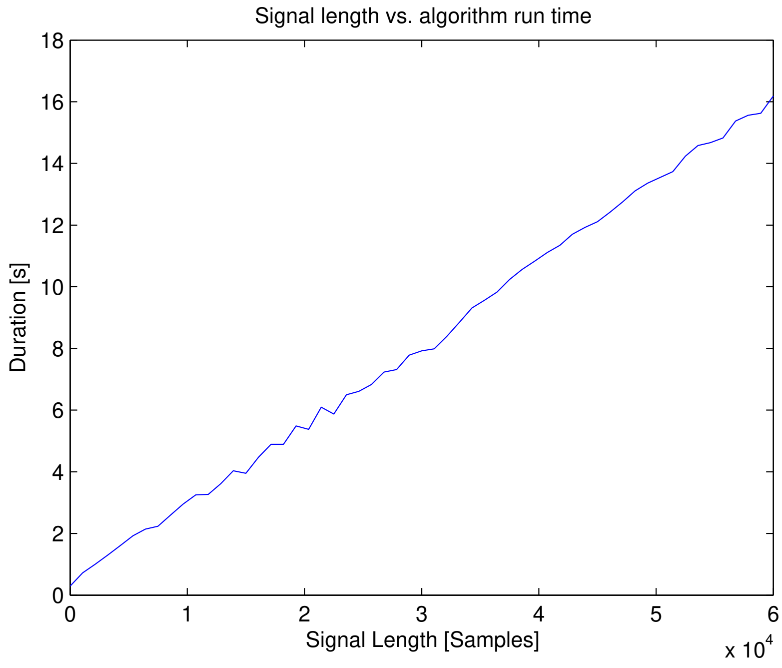


Figure 10.17: This figure shows duration of the algorithm by Griffin & Lim as a function of the number of samples (100 iterations). As we can see, the time duration of the algorithm is fairly linear. The curve stretches from 0.69 seconds for 1000 samples, to 16.2 seconds for 60.000 samples

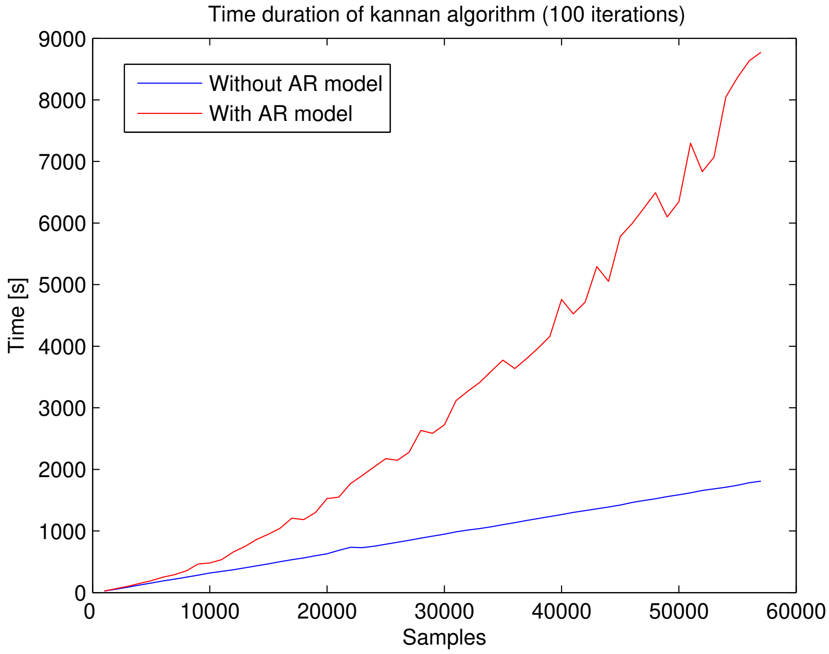


Figure 10.18: This figure shows duration of the algorithm by Kannan as a function of the number of samples using 100 iterations. As we can see, the time duration of the algorithm is fairly linear without the AR model. The curve stretches from 321 seconds for 1000 samples, to 2000 seconds for 60.000 samples. With the AR model the run time increases dramatically, ending at around 9000 seconds for 60.000 samples.

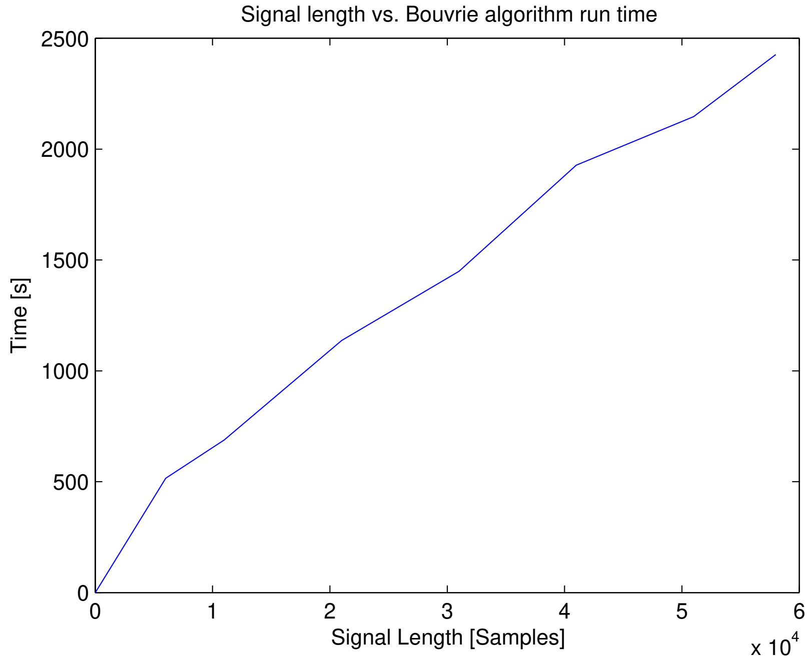


Figure 10.19: This figure shows duration of the algorithm by Bouvrie as a function of the number of samples. As we can see the run time is also fairly linear for this algorithm. The algorithm has a run-time of around 600 seconds for a 10.000 sample signal and stretches all the way to around 2500 seconds for 60.000 samples.

Table 10.27: **Time Comparison** Different overlaps (200 iterations, 512 window)

Overlap (samples)	128	256	384
Overlap (%)	25	50	75
Griffin et al. (time)	21.7 sec	31.8 sec	62.5 sec
Kannan et al. (time)	3140 sec	3920 sec	19200 sec
Bouvrie et al. (time)	9116 sec	2659 sec	550 sec

If we look at Table 10.27 we see the run-time of the algorithms at different overlaps. 200 iterations was used for the Griffin and Kannan algorithm. As we can see, both the Griffin and Kannan algorithm increases a great deal in run time at larger overlaps, while the Bouvrie algorithm run time is dramatically decreased.

10.6 Comparison of signal estimation algorithms

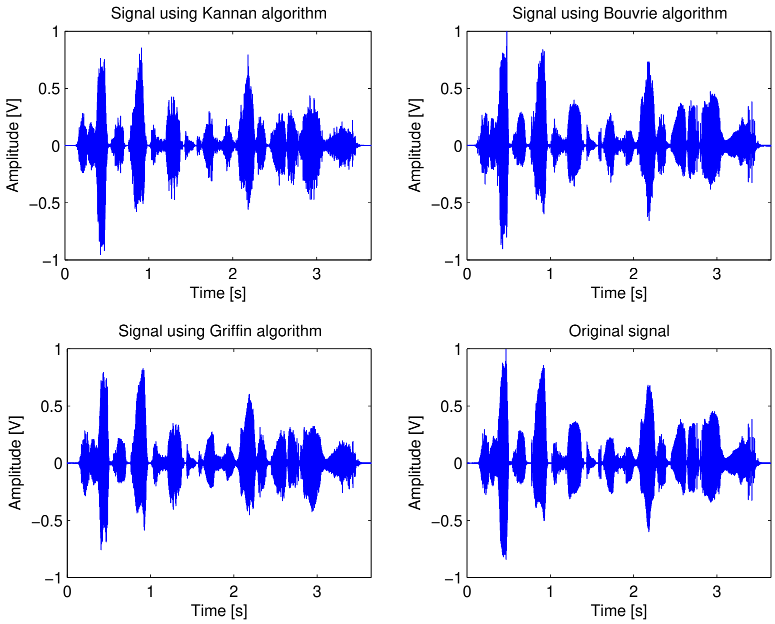


Figure 10.20: Reconstructed signals for each of the three algorithms using 500 iterations on the TIMIT female speaker signal (512 window with 50% overlap). We see that the Griffin algorithm and the Bouvrie algorithm gets closest to the original signal, with Bouvrie being very close to the original. The algorithm by Kannan resembles the original signal, but has certain spikes that are not present on the original signal. These spikes are also to some extent present on the result obtained with the algorithm by Griffin & Lim, but not to the same extent

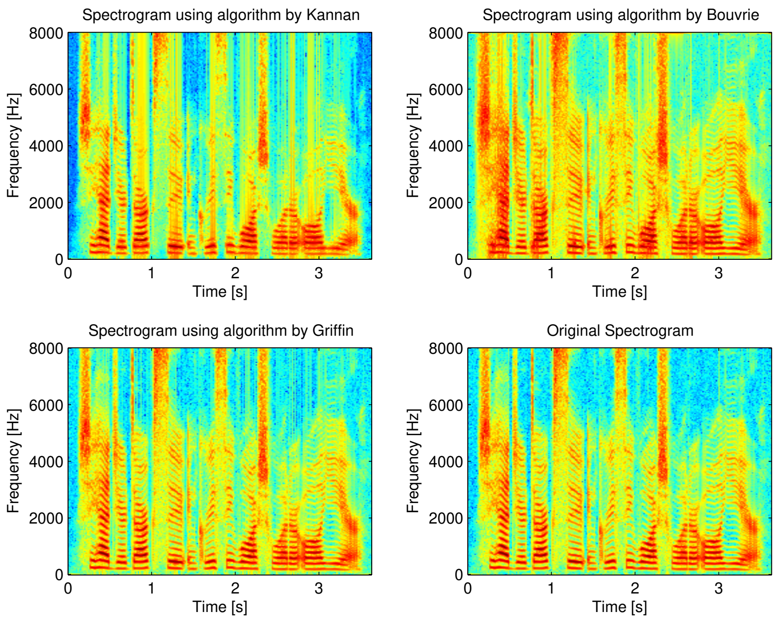


Figure 10.21: Reconstructed spectrograms for each of the three algorithms using 500 iterations on the TIMIT female speaker signal (512 window with 50% overlap. We see that the best match to the original spectrogram is obtained by the Griffin (bottom left) and Bouvrie algorithm (top right). The Bouvrie algorithm, however, seem to have some difference from the original at the voiced parts at about 0.25 and 0.9 seconds and at 2.1 seconds. These difference can be heard as noise in the reconstructed signal. The spectrogram (top left) by the Kannan algorithm does not seem to capture all of the finer details in the original spectrogram and appears less sharp at the voiced parts.

Table 10.28: **Timit Female Speaker** - Reconstruction Comparison (512 window, 50% overlap)

Algorithm	*Iterations*	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Random Guess	-	-5.14 dB	17.33 dB	37.16 dB	3
Zero Guess	-	-0.88 dB	4.13 dB	50.36 dB	3
Griffin et. al.	500	-6.41 dB	65.87 dB	-39.09 dB	4
Kannan et. al	500	-6.38 dB	33.78 dB	20.7 dB	3
Bouvrie et al.	-	16.28 dB	35.19 dB	27.32 dB	3-4

Table 10.29: **Timit Male Speaker** - Reconstruction Comparison (512 window, 50% overlap)

Algorithm	*Iterations*	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Random Guess	-	-4.77 dB	16.63 dB	33.03 dB	3
Zero Guess	-	-1.68 dB	4.55 dB	45.11 dB	3
Griffin et. al.	500	-4.56 dB	59.11 dB	-37.8 dB	4
Kannan et. al	500	-4.58 dB	35.65 dB	16.41 dB	3
Bouvrie et al.	-	21.56 dB	42.60 dB	15.12 dB	3-4

Table 10.30: **Interspeech Male Speaker** - Reconstruction Comparison (512 window, 50% overlap)

Algorithm	*Iterations*	SNR_{signal}	$SNR_{spectrogram}$	$MSE_{spectrogram}$	PQE
Random Guess	-	-4.61 dB	17.38 dB	34.45 dB	3
Zero Guess	-	-1.75 dB	4.45 dB	47.38 dB	3
Griffin et. al.	500	-6.53 dB	64.44 dB	-35.36 dB	4
Kannan et. al	500	-7.41 dB	33.58 dB	18.3 dB	3
Bouvrie et al.	-	20.53 dB	38.32	21.51 dB	3-4

If we look at Tables 10.28, 10.29 and 10.30 we see that the Griffin algorithm obtained the best $SNR_{spectrogram}$ while the Bouvrie algorithm obtained the best SNR_{signal}. We also see that all algorithms have improved much over both the zero phase guess and the random phase guess when observing the $SNR_{spectrogram}$. The zero phase guess showed higher SNR_{signal} than both the Griffin algorithm and the Kannan algorithm, but had much worse quality sound that any of the reconstruction algorithms. The best perceptual sound quality, when using the 32 ms window with a 50% overlap was found using the Griffin algorithm, since both the Bouvrie and Kannan algorithm had noise that was unpleasant.

Table 10.31: **Timit Female Speaker** - Comparison (512, 75% overlap)

Algorithm	*Iterations*	SNR_{signal}	$SNR_{spectrogram}$	PQE
Griffin et. al.	200	-6.69 dB	46.05 dB	4
Kannan et. al	200	-5.75 dB	33.92 dB	3
Bouvrie et al.	-	50.03 dB	60.59 dB	4-5

Table 10.32: **Interspeech Male Speaker** - Comparison (800, 75% overlap)

Algorithm	*Iterations*	SNR_{signal}	$SNR_{spectrogram}$	PQE
Griffin et. al.	200	-5.01 dB	50.42 dB	4
Kannan et. al	200	-7.89 dB	38.88 dB	3
Bouvrie et al.	-	121.02 dB	125.08 dB	5

Tables 10.31 and 10.32 shows the side-by-side comparison of the three algorithms with a 32 ms window and a 75% overlap. In both signals, the Bouvrie gets both the highest $SNR_{spectrogram}$ and SNR_{signal}. While the Griffin and Kannan algorithm only improved very little by the extra overlap, the Bouvrie algorithm now produced almost perfect reconstruction. Note that these results are for 200 iterations.

Chapter 11

Discussion

In the following we shall discuss and interpret the results of each of the signal estimation experiments. In order to easily refer to the three different algorithms, we will use the terms Griffin algorithm, Kannan algorithm and Bouvrie algorithm to denote the algorithms by Griffin & Lim, Kannan et al. and Bouvrie et al. respectively.

11.1 Zero and Random phase

Applying a different phase than the orignial to a spectrogram and then reconstructing the original signal with an inverse short time Fourier transform showed to have great concequences on the sound quality of the signal. In some research fields, like automatic speech recognition [8], phase is often not considered to be important. As our perceptual sound quality evaluation grade shows, all reconstructed speech was indeed intelligable. It was, however, not very pleasant to listen to. Phase information is obviously very important for the sound quality of the reconstructed signals. The human auditory system iseasily irritated or stressed by even a slight change in the original phase. On the other hand, a computer trying to recognise speech may be almost unaffected by a wrong phase. This only goes to show that the phase information is valued differently in different research areas.

We noted in Table 10.1, 10.2 and 10.3, that the two phase guesses yielded different values on both SNR_{signal} and $SNR_{spectrogram}$. A random phase signal seems to achieve a spectrogram that is closer to the original than a zero phase signal, and a zero phase signal estimate achieved a signal which was closer to the original signal in terms of SNR_{signal} than a random phase estimate. Since the Griffin and Kannan algorithm optimize on the spectrogram, this might already be an indication that a random phase estimate is better than a zero phase estimate for these two algorithms.

Table 10.4, 10.5 and 10.6 showed that taking a random guess at the phase, could lead to a difference between maxium and minimum $SNR_{spectrogram}$ of around 4-5 dB, with the same applying to the SNR_{signal}. The standard deviation of 0.91 however showed that these random guesses was often close to the mean. The mean values also show that our stored random phase signals which were used in most experiments (Table 10.1, 10.2 and 10.3) are close to what is normally obtained with a random phase guess. This ensured that our experiments were carried out with the "typical" random guess for the phase, and that most guesses lie in this area.

11.2 Signal estimation using Overlap and Add

11.2.1 Performance as a function of initial phase

Section 10.2.1 showed the performance of the Griffin algorithm using zero and random phase signals as initial estimates for the algorithm. Figure 10.3 showed that using random- or zero phase gave different results for different signals. The female signal had a higher SNR on the reconstructed signal, when using the random phase initial estimate, while the male speaker had a higher SNR on the zero phase signal. This clearly shows that the choice of zero or random phase seems dependant on the signal. The perceptual sound quality was however almost indistinguishable between zero and random phase on both signals after 100 iterations. The conclusion of this experiment is therefore that a random phase estimate is just as good as a zero phase estimate as long as the number of iterations are over 100. We did, on the other hand, see an example of the algorithm ending in a local minimum in Figure 10.2.3 when starting with a zero phase estimate. This result was however obtained on a perfect sinusoidal signal, which may be more prone to local minima. To be safe, it seems resonable to use random phase as initial phase for all signals.

11.2.2 Performance on unmodified spectrogram

The algorithm by Griffin & Lim generally obtained good quality recontruction. We saw that the reconstructed signals looked close to the original in Figure 10.4, and that signal did improve after an increasing number of iterations. If we look at Table 10.7, 10.8 and 10.9 we see that the algorithm achieved good quality sound after 100 and 200 iterations. We used 200 iterations for most of our experiments because this number of iterations ensured good quality reconstruction on all the signals we tested.

The SNR_{signal} did not change much over the 500 iterations on the TIMIT female speaker. It fluctuated a little up and down before landing at -6.41 dB. On the TIMIT male speaker, the SNR_{signal} steadily increased at the displayed iterations. The Interspeech male speaker on the other hand steadily decreased at the displayed iterations. This rise and fall in SNR can be explained by an overall phase difference between the recontructed and the original signal. A 90 degree phase shift on the entire signal would cause a decrease in the SNR_{signal} at every iteration. This also shows that SNR_{signal} mat not be a very good indicator of the quality of the reconstruction, since all three signals had the same perceptual quality.

Looking at the spectrograms in Figure 10.5, we saw visual improvements for increasing number of iterations. The reconstructed spectrograms looked very close to the original, and also caputures some of the finer details of the original spectrogram. The close match of the reconstructed spectrograms to the original spectrogram was also noted in Table 10.7, 10.8 and 10.9. The $SNR_{spectrogram}$ increased at each iteration, and the MSE kept decreasing. This was all a good indication that the algorithm was working properly and converged to a solution where the difference between the original and the estimated spectrogram got closer to each other. It was also shown that 10 iterations with the Griffin algorithm does not generally lead to a good reconstruction, but suffers from more severe artifacts which were beginning to be unpleasant.

Even though we could recieve very high $SNR_{spectrogram}$ on all the test signals, the reconstructed speech, even after 500 iterations, still contained small artifacts which made it clear that this was not exactly the same as the original speech. The artifact sounded like a very fine reverb effect (like talking inside a box), but was not unpleasant. Since the sound quality of the reconstruction did not improve significantly from 200 to 500 iterations, even though the $SNR_{spectrogram}$ increased to higher values, we concluded that 200 iterations was enough to obtain close to, if not the best reconstruction, the Griffin algorithm could provide.

11.2.3 Performance at different overlaps and window lengths

In Figure 10.6 we saw the changes in the error functions SNR_{signal}, $SNR_{spectroram}$ and MSE as we changed the number of overlapping samples. The SNR_{signal} was low up until a overlap of 40%, and the sound quality seemed to follow this curve well. The quality was very low up to 40%, and kept improving with a larger overlap. We felt, however, that the difference between 50% and 90% overlap was not very significant. We also saw high $SNR_{spectrogram}$ and low MSE at the overlaps from 1 to 150 samples. This does not indicate good quality sound, but is rather an image of how the algoithm can get a good match to a very low quality spectrogram. We believe that a low quality spectrogram makes it "easier" to find a signal to match the spectrogram, but as we saw in the SNR_{signal} graph, the reconstructed signal was very different from the original signal. This is also an example of $SNR_{spectrogram}$ not always being a good measure of the general quality of the reconstructed signal. $SNR_{spectrogram}$ together with SNR_{signal} may say more about the signal than either alone. A very high SNR_{signal} and a very high $SNR_{spectrogram}$ would probably indicate a good reconstruction, while a very low SNR_{signal} and high $SNR_{spectrogram}$ would indicate a poor reconstruction.

We displayed the results of testing the quality at 25%, 50% and 75% overlap in Tables 10.10 and 10.11, and there were no difference between the Interspeech male and TIMIT female speech signal. Both obtained a poor result at 25% overlap. At 50% overlap the quality was very close to that with 75% overlap. This means that the extra calculations needed to do a 75% overlap in stead of 50% is maybe not justified when considering the sound quality.

The length of the analysis window proved to be important for the quality of the reconstruction. Tables 10.12 and 10.13 showed that at the 50% overlap, an analysis window of 32 ms gave the best quality signal. The result was clear for both the signals from the two different sound databases, the TIMIT with the female speaker, sampled at 16 kHz, and the male speaker from the Interspeech database sampled at 25 kHz. It seems that the best reconstruction is obtained with the spectrogram that has the best resolution in both the frequency and time axis. 512 samples with a 50% overlap yields a good resolution on both frequency and time axis with the TIMIT database. Since we need a high resolution spectrogram for the vector quantization, it is fortunate that the Griffin algorithm also performs best on high spectrogram resolution. Generally we did not notice any difference on reconstruction of the three different speech signals, which implies that neither gender or sampling frequency affects the reconstruction.

11.3 Signal estimation using Probabilistic Inference

11.3.1 Performance as a function of initial phase

Looking at Figure 10.8 the algorithm by Kannan gave a similar result for both the male and female signal. The experiment suggests that a random phase signal is the best initial signal estimate to start the algorithm with. The perceptual sound quality between using zero and random phase was also different, with random phase giving better quality. The test was also run on a signal with a male speaker from the interspeech database, with the same result. Compared to the algorithm by Griffin & Lim we clearly see from Figure 10.8 that this algorithm converges much slower.

11.3.2 Performance on unmodified spectrogram

In Figure 10.9 we saw that the signal improved as the number of iterations went from 10 to 200. In Table 10.14, 10.15 and 10.16 the SNR_{signal} is displayed for different number of iterations. It is difficult to see any pattern in the SNR_{signal}'s . The perceptual quality did improve over an increasing number of iterations, but still carried artifacts which caused the lower grade in PQE. The improvement however small after 100 iterations, however, and this is another example of the SNR_{signal} not being a solid indicator for the perceptual quality. The signal improves, but the SNR_{signal} does not. Again, an overall phase error could contribute to the low SNR_{signal}. In any case, this shows that this algorithm, as with the Griffin algorithm, obtains a reconstruction that does not have the exact same starting phase as the original.

The spectrograms in figure 10.10 clearly showed improvements with increasing number of iterations. Table 10.14, 10.15 and 10.16 also showed that the $SNR_{spectrogram}$ kept rising for increasing number of iterations. This is a good indication that the algorithm is functioning correctly. The sound quality also improved with increasing number of iterations, but it did also show artifacts. Those artifacts are especially clear in the top right spectrogram (after 100 iterations) at the voiced areas in figure 10.10. They can be seen as lines spanning the entire frequency band which suggests a discontinuity in the signal. Figure 10.13 shows the discontinuity that occurs at the boundaries of the segments in the STFT. The discontinuities are also very noticable in the reconstructed signal.

The AR model was tested on a small part of the male speaker signal from the TIMIT database (due to the high run time of the algorithm with an AR model, also shown in Figure 10.18). The reconstruction with the AR model made the algorithm converge faster, as we saw in figure 10.11. The artifacts, however, are still there, and are also clearly visible in the spectrograms on Figure 10.12. We tried to do multiple tests on different signals. Sometimes the AR model would help a little, and other times it would not do much. A reason why the AR model may not give good results could be that the AR model did not capture a tendency in the signal on which it was trained, that could be used to improve the signal being reconstructed.

The artifact that plagued all of the reconstructions, shown in figure 10.13 is difficult to explain. It seems to have something to do with how the algorithm updates the individual samples, since the discontinuities are still present, and sometimes even worse, after many iterations. We tried applying a lowpass filter to the estimated signal in order to remove the discontinuities, but the cut-off frequency needed to make a significant difference, was so low that the quality of the speech signal was affected to a degree where the unfiltered signal was better. We also tried running the algorithm with a randomly scaled version of the original signal to see if the problem occured if there were no phase mismatch. The reconstructed signal had no discontinuities.

Kannan et al. [12] did note the problem with the discontinuities indirectly, and suggested the AR model to compensate by smoothing the signal at the boundaries of each segment. Their SNR result show that the AR model improved the reconstruction by 0.27 dB, but did not remove the discontinuities entirely. This is evident, as [12] also note that they tried using spline smoothers along the boundaries to further improve the sound quality of the reconstruction. Using a spline smoother does not necessarily improve the signal on every boundary unless different spline smoothers are used at every boundary. We refrained from using a spline smoother, because we did not think that this could be done automatically and at the same time ensure improved sound on all speech or sound reconstructions. We also noted that [12] contains the same artifact lines in their spectrograms as we saw in ours.

The general sound quality of the algorithm was not what we had expected, and our conclusions are different from the ones shown in [12]. Kannan et al. [12] reaches the conclusion that the probabilistic model has far superior sound quality to the algorithm by Griffin [10]. Disregarding the artifact described above, we found it possible to obtain a fair reconstruction when using many iterations, but not better than the Griffin algorithm using the same number of iterations. [12] show time signals and the spectrograms of reconstructed signals from both the suggested algorithm and the one by Griffin &

Lim. Neither the reconstructed signal nor the spectrogram obtained from the Griffin algorithm, match the quality that we obtained in our experiments. It is difficult to say how these results in [12] were produced, since there is no mention of the number of iterations used or how many is needed to obtain good quality reconstruction with the suggested algorithm. But even after 100 iterations (as suggested by [10]), the quality of the spectrogram produced by the Griffin algorithm is much higher than the one portrayed in [12]. We further noted that [12] used utterances not only from the Wall Street Journal corpus, but also from the TIMIT database which we used. They, however, used a window length of 256 samples (16 ms) with a 50% overlap. As we showed in Table 10.12, a 16 ms window gives a poorer sound quality compared to a 32 ms window. This could maybe, to some extent, explain some of the poor results they obtained for the Griffin algorithm.

11.3.3 Performance at different overlaps and window lengths

Both the overlap experiment (Figure 10.14) and the window length experiment (Table 10.19) showed the same tendency as for the Griffin algorithm. Although the PQE grade was 3 for all window lengths, we found that the best sound quality was at a window length of 32 ms (512 samples) for a 50% overlap. For the overlaps shown in Table 10.17 and 10.18 we also saw the same tendency as with the Griffin algorithm. A 75% overlap gave a very slight increase in sound quality. This proved to us that only very little is gained by using a 75% overlap with these two algorithms. It was not obvious from the $SNR_{spectrogram}$ and SNR_{signal} that a 32 ms window would give the best result. This again shows that the SNR's are not good indicators of the quality of the reconstructed sound in this algorithm either.

11.4 Signal estimation using Non-linear Equations

11.4.1 Performance on unmodified spectrogram

Figure 10.15 showed the reconstructed signal compared to the original. The reconstructed signal is almost an exact match to the original. Looking at the spectrograms of the reconstructed and the original signal we also saw a very good approximation to the original spectrogram. In Table 10.21 the SNR's for all three test signals were shown. We saw an impressive SNR_{signal} on all signals. This indicates that the reconstructed signal is indeed very close

to the original, and that this algorithm achieves a good sample-by-sample match to the original signal in the reconstruction. The $SNR_{spectrogram}$ was not very high compared to the values of the Griffin algorithm, but the reconstructed spectrogram still looked close to the original. This shows that the algorithm was functioning correctly.
The sound quality was also very high, except for the fact that there were noise in the beginning of the reconstructed signal. This noise can be explained by the random guess needed for the first S samples of the first segment (see section 8.3). The noise was most noticeable on the female speaker signal, and could cause some stress to the listener. The reason why the noise sounded less severe with the Interspeech signal, could be that the noise was lower in frequency, and this male speaker had a very "dark" voice. It was difficult to grade the signals that had noise in the beginning and were otherwise almost perfect. In these cases, the grade was set to 3-4 due to the noise.

Looking at the spectrogram in Figure 10.16 (top left) we see the effects of using an FFT length of two times the window length, and the effect of using a boxcar (square) analysis window. The zero padded extra length in the Fourier transform may artificially increase the resolution on the frequency axis, but the use of a boxcar window causes the spectrogram to be fuzzy and obtain a "ringing effect" which appears when a Fourier transform is applied to a square window. The use of a boxcar window is not an option in our case, since we need a high quality spectrogram with good resolution forn our vector quantization algorithms. The algorithm was unfortunately only described for boxcar windows, and due to time constraints, only a few failed attempts were made to extend the algorithm for use with other window types as well.

11.4.2 Performance at different overlaps and window lengths

Table 10.25 and 10.26 showed the effects of using different analysis window lengths on the TIMIT female speaker signal and the Interspeech male signal. What captured our eyes at first, was the fact that the perceptual sound quality did not change significantly at any window lengths from 64 to 512 samples. The SNR_{signal} showed that the reconstructed signal gets closer to the original signal even if small window lengths are used. The perceptual quality was fairly constant over the different window lengths, but with different noise in the first part of the signal, due to the mentioned random signal guess. Again the Interspeech signal had a more pleasant sound, due to the low frequency voice of the speaker and the low frequency noise. The best quality signal was obtained by the Interspeech speaker at a 16 ms window. We also saw that both the SNR_{signal} and $SNR_{spectrogram}$ was high in this case. This could suggest that these two error measures may provide a good hint as to the quality of the reconstructed speech by this algorithm.

Table 10.22, 10.23 and 10.24 showed the effect of changing the overlap size. The algorithm performed much better with a larger overlap. At 75% overlap the result was as good as perfect with only very slight noise in the TIMIT female signal. With the TIMIT male and Interspeech male signals, the reconstructed signal was not distinguishable from the orignal.

11.5 Run times of the different algorithms

In the results section 10.5 we showed the run time of the different algorithms using different signal lengths. It was very clear, that using algorithms which rely on the 1. order derivatives of the problem, increases the amount of calculations needed, and thereby the run time of the algorithm. 100 iterations with the Griffin algorithm only took 16 seconds, while the same number of function evaluations lasted about 2000 seconds (33 mins) with the Kannan algorithm. Both algorithms were linear with regard to the length of the signal. The time consumption of using the AR model in the Kannan algorithm was also very high, which was why we chose to do our experiment on a smaller part of the signal when a very large number of reconstructions was needed (a 50 run mean for instance). Notice that these times are for only 100 iterations. The Bouvrie algorithm took nearly 2500 seconds to reconstruct a 60.000 sample signal using a 512 sample window with a 50% overlap and the curve was almost linear for smaller signals.

Table 10.27 showed some very interesting results. Both the Griffin and Kannan algorithm run time increased as the overlap was made larger. This is due to the fact that the spectrogram becomes larger as a result of the increase in overlap, and more calculations are therefore needed. For the Kannan algorithm this also means longer calculations for the derivatives, since the samples are included in more frames of the spectrogram. This is why we see the dramatic increase in run time for the Kannan algorithm. The Bouvrie algorithm shows quite the opposit. Using a 75% overlap decreased the running time to 550 seconds. This is due to the fact, that as more segments overlap, less iterations are needed to get a good solution to the equations (using the Gauss-Newton method) in the Bouvrie algorithm.
Bouvrie et al. [2] describe the algorithm as being comparable in run time

to the Griffin algorithm for small window sizes. This is true if the goal is to obtain the same amount of error in the spectrograms for both algorithms, but since our results showed that the perceptual quality of the algorithm by Griffin, does not improve much above 200 iterations we do not agree entirely. The Griffin algorithm can give a fairly good result in a fraction of the time needed by the other two algorithms.

11.6 Comparison of the signal estimation algorithms

The results section 10.6 showed a side by side comparison of the three algorithms. Figure 10.20 showed the visual comparison of the reconstructed signals. It is clear that the Bouvrie algorithm obtains the best sample by sample match to the original spectrogram. The algorithm by Griffin also captures the details in the signal well, but with noticable differences. The signal from the Kannan algorithm also resemble the original signal, but with more noticable differences, especially the small peaks that are not present on the original signal. This figure tells us, that the best sample by sample match to the orignal signal is obtained with the Bourvrie algorithm.

The spectrograms in Figure 10.21 showed, that the Griffin and Bouvrie algorithms both gave very close matches to the original spectrograms, with the fine details in the orignal spectrogram being captured very well. We also see the noise in the Bouvrie algorithm spectrogram which was explained earlier. The Kannan algorithm produced a spectrogram that did not capture all of the finer details in the original spectrogram. Looking at table 10.28 we see that at a 512 window with a 50% overlap, the Griffin algorithm gets a higher $SNR_{spectrogram}$ than any of the two other algorithms. The Bouvrie algorithm, however, obtains the highest SNR_{signal}. Due to the noise in the reconstructed signal using the Bouvrie algorithm, we felt that the reconstruction by the Griffin algorithm gave a more pleasant sound, although it did not sound as “correct” as the Bouvrie signal. Tables 10.31 and 10.32 told a different story.

At 75% overlap the Bouvrie algorithm was the best by a great margin, creating perfect reconstructions of the original signals. The results were consistent for both test signals. The random and zero guess shown in Tables 10.28, 10.29 and 10.30 are the simple inverse STFT on the signal with a different phase. As we can see, all of the algorithms yielded a much better result on the

$SNR_{spectrogram}$ and also achieved much better sound quality. This was also for the Kannan algorithm, even though the same grade of 3 is given.

The different algorithms have their advantages and disadvantages. The Griffin algorithm is very fast and obtains fairly good quality estimates in a very short time. It is however clear when comparing the original sound to the reconstructed, that the Griffin algorithm does not obtain the original signal. It can be discussed if this is that important in speech, but listening to music may be an entirely different matter.

The Kannan algorithm proved to have very slow convergence and the recontruction we obtained was plagued by artifacts. The algorithm does however have a very interesting aspect; the prior. The prior can be changed to not only contain an AR model, but other information about the speaker being reconstructed. This is an interesting point that could be beneficial when working on a problem such as ours i.e. reconstructing speech from a known speaker. Unfortunately, we did not get the sound quality we had hoped for, and the results leave much to be desired. We still believe, however, that this algorithm holds great potential.
The Bouvrie algorithm is very effective in getting reconstructions that are very close to the original, sample by sample. The results were impressive, but the algorithm is very slow compared to the Griffin algorithm. Even at 75% overlap, we get a run time which is almost ten times as long as with Griffin. The algorithm is new (will be published at Interspeech 2006 nov.14.) and seems to hold a great deal of potential. The noise issues may be improved by post processing and the ideas for the Bouvrie algorithm may help to improve other algortihms, as for instance the Kannan algorithm, by using overlap information in the same way. The algorithm also needs to be modified to be able to use other windows than a square (boxcar) window for it to be usable in a vector quantization scheme such as ours.

11.7 Summary of Main Conclusions

The main conclusions of part II (Signal Estimation) were:

- Phase information proved to be very important for good perceptual sound quality.
- Random phase is the best inital phase estimate for both the algorithm by Griffin et al. and the algorithm by Kannan et al.
- A window length of 32 ms leads to the best reconstruction when using the Griffin or Kannan algorithm.
- At a 50% overlap with a 32 ms window, the Griffin algorithm achieves the most pleasant reconstructed sound.
- At a 75% overlap with a 32 ms window, the Bouvrie algorithm achieves almost perfect reconstruction.
- The Bouvrie algorithm is not affected by changes in window length and takes less time to run at larger overlaps
- The Griffin algorithm is much faster than the two algorithms that use the 1.st derivative
- The Kannan algorithm proved to be the slowest of the three algorithms when using the AR model, and had the poorest reconstruction
- The signal-to-noise measurements used were not sufficient to describe the sound quality of the reconstructed signals.

Part III

Vector Quantization

Chapter 12

Introduction

Using binary masking to remove unwanted noise or a speaker from a spectrogram, results in a spectrogram with large holes as seen in the data on Figure 14.1, which has been masked with simulated masks.
Vector quantization (VQ) is a mapping of training vectors from a vector space to a finite number of areas within that space. These areas are called clusters and are represented by their center vectors or centroids. A collection of centroids, which represents the whole vector space, is called a codebook. VQ can be a useful tool for resynthesizing the missing data and one of the most popular VQ algorithms is k-means clustering [6].

In this part we will first describe some of the previous research in vector quantization, and proceed to set up the theory used in the data clustering and spectrogram reconstruction experiments. We will then carefully describe each of those experiments in detail in terms of use of programs, data, and variables. Following this, there is a results section where we will present our results and explain them, and finally we will interpret and discuss our results in the discussion section. A summary of main conclusions from this part of the thesis can be found in section 16.3.

12.1 Previous research in vector quantization of muti-dimensional data

Many different variations of the k-means algorithm have been proposed. In fact ever since the mid 1960's an overwhelming body of work has been the subject of studies, either strengthening ease of use, precision, or speed. They all spring from the same principal steps usually attributed to a paper written by Lloyd in 1964 (published in 1982) [13]. These are summarised in section 13.1 and a graphical example can be seen in Figure 13.1. They can usually be divided into two major categories. Exact k-means algorithms and approximate k-means algorithms.

The exact k-means algorithms apply geometric reasoning to reduce the number of necessary distance calculations in the clustering process. Many papers in this category have been published on speeding up the algorithm using inequalities that are specific for Euclidian distance [28], [15], [18], but another well explored approach is to use triangle inequalities to eliminate unnecessary distance calculations [16], [6].

Approximate k-means algorithms, as the title would suggest, uses approximations to reduce the number of calculations and thereby speed up the process. Algorithms in this category will not be treated in this thesis, since clustering quality is of importance in our application, and since the exact algorithm proposed by Elkan promises to be very fast [6].

Chapter 13

Theory

13.1 Simple *k*-means algorithm

The principal steps of the k-means algorithm are summarised in the following, and a graphical example can be seen in Figure 13.1. For the purpose of visualisation of the algorithm, we will go through the theory as it would be applied to data in the form of points in a two dimensional space, however it is trivial to expand the approach into a form with the amount of dimensions of the spectrogram training frames.

Step 1 Begin with a decision on the value of K = number of clusters

Step 2 Select any initial partition that classifies the M_{train} training points into K clusters. The training points can be assigned randomly, or systematically using the "furthest first" method [4] in the following manner:

1 Assign the first centroid to be the mean of the dataset, $\mathbf{X}$.

2 Assign the rest of the centroids by continually choosing the training vector farthest from any of the already chosen centroids. Or, considering $\mathbf{C}$ to be the set of already chosen centroids, max $\{\min\{d(\mathbf{x},\mathbf{c}) : \mathbf{c} \in \mathbf{C}\} : \mathbf{x} \in \mathbf{X}\}$.

3 Assign each of the M_{train} training points to the cluster with the nearest cluster centroid. After each assignment, recompute the centroid of the gaining cluster.

Step 3 Take each point in sequence and compute its distance from the centroid of each of the clusters. If a point is not currently in the cluster with the closest centroid, switch this point to that cluster and update the centroids of the cluster gaining the new point and the cluster losing the point.

Step 4 Repeat step 3 until convergence is achieved, that is until a pass through the training points causes no new assignments.

Furthermore, for the specific case where the number of training points is less than the number of clusters, we assign each training point as a cluster centroid. Each centroid will have a cluster number.

MATLAB has a generic function, *kmeans*, for performing data clustering, which is easy to use, and by doing more than one reinitialisation step similar to Step 2 it is quite effective at avoiding poor cluster quality. Simpler, faster algorithms without this feature will some times end up in poor local minima. However, it takes a long time to run the MATLAB generic function with large numbers of K and M_{train} mainly due to the amount of distance calculations. Therefore we will use an accelerated k-means algorithm proposed by Elkan since it is effective for datasets with up to 1000 dimensions and high numbers of K [6]. And for $K \geq 20$ it is many times faster than other known accelerated K-means methods [6].

13.2 Accelerated *k*-means algorithm

The central operation in the k-means algorithm is to find the nearest centroid for each training vector. [6]. One approach for accelerating this operation is to use triangle inequalities to eliminate unnecessary distance calculations [6]. There has been several earlier attempts at this but here we will focus on the promising version proposed by Elkan in 2003[6].
Again, like in the previous section, for the purpose of visualisation of the algorithm, we will go through the theory as it would be applied to data in the form of points in a two dimensional space.

The triangle inequality illustrated on Figure 13.2 states that for any three points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$:

$$d(\mathbf{x}, \mathbf{z}) \leqq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \tag{13.1}$$

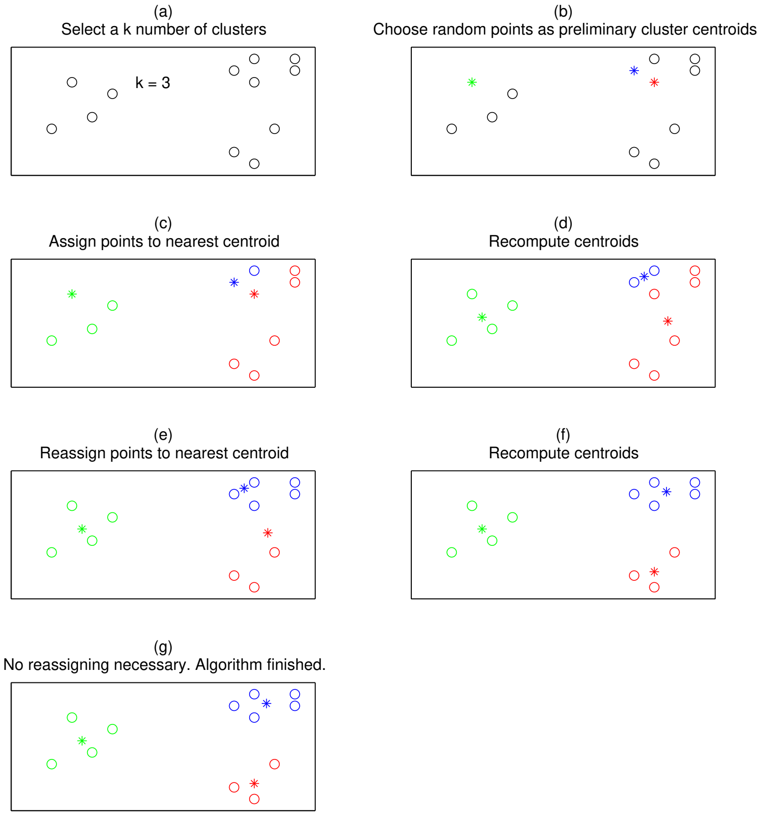


Figure 13.1: A two dimensional example using the k-means algorithm to divide a 12 point data set into 3 clusters. The events are ordered alphabetically from (a) to (g). o is used to mark a training point, while a * marks a centroid.

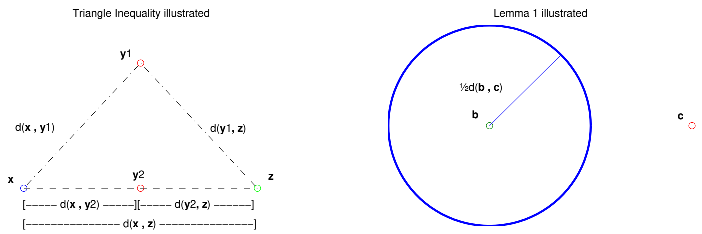


Figure 13.2: Triangle inequality (right) using two examples of a possible point $\mathbf{y}$ and an illustration for Lemma 1 (left)

Equation 13.1 is used for proving the following:

Lemma 1: Let $\mathbf{x}$ be a point and let $\mathbf{b}$ and $\mathbf{c}$ be centroids. If $d(\mathbf{b},\mathbf{c}) \geq 2d(\mathbf{x},\mathbf{b})$ then $d(\mathbf{x},\mathbf{c}) \geq d(\mathbf{x},\mathbf{b})$. This means, that on Figure 13.2 we know a point located within the blue circle is closer to $\mathbf{b}$ than to $\mathbf{c}$

Proof: We know that $d(\mathbf{b},\mathbf{c}) \leq d(\mathbf{b},\mathbf{x})+d(\mathbf{x},\mathbf{c})$. So $d(\mathbf{b},\mathbf{c})-d(\mathbf{x},\mathbf{b}) \leq d(\mathbf{x},\mathbf{c})$. Consider the left-hand side: $d(\mathbf{b},\mathbf{c}) - d(\mathbf{x},\mathbf{b}) \geq 2d(\mathbf{x},\mathbf{b}) - d(\mathbf{x},\mathbf{b}) = d(\mathbf{x},\mathbf{b})$. So $d(\mathbf{x},\mathbf{b}) \leq d(\mathbf{x},\mathbf{c})$.

Lemma 2: Let $\mathbf{x}$ be a point and let $\mathbf{b}$ and $\mathbf{c}$ be centroids. Then $d(\mathbf{x},\mathbf{c}) \geq \max\{0, d(\mathbf{x},\mathbf{b})-d(\mathbf{b},\mathbf{c})\}$.

Proof: We know that $d(\mathbf{x},\mathbf{b}) \leq d(\mathbf{x},\mathbf{c})+d(\mathbf{b},\mathbf{c})$. So $d(\mathbf{x},\mathbf{c}) \geq d(\mathbf{x},\mathbf{b})-d(\mathbf{b},\mathbf{c})$. Also, $d(\mathbf{x},\mathbf{c}) \geq 0$.

It follows from Lemma 1, that if the distance between any data point $\mathbf{x}$ and its designated centroid $\mathbf{c}$ is less than or equal to half the distance from the centroid $\mathbf{c}$ to a new centroid $\mathbf{c}'$, then the data point will be closer to, or equally close to $\mathbf{c}$ as $\mathbf{c}'$. That is: If $\frac{1}{2}d(\mathbf{c},\mathbf{c}') \geq d(\mathbf{x},\mathbf{c})$ then $d(\mathbf{x},\mathbf{c}') \geq d(\mathbf{x},\mathbf{c})$. In this case it is not necessary to calculate the distance from $\mathbf{x}$ to $\mathbf{c}'$.

Suppose that we do not know $d(\mathbf{x},\mathbf{c})$ excactly, but we do know an upper bound u such that $u \geq d(\mathbf{x},\mathbf{c})$. Then we need to compute $d(\mathbf{x},\mathbf{c}')$ and $d(\mathbf{x},\mathbf{c})$ only if $u > \frac{1}{2}d(\mathbf{c},\mathbf{c}')$.

If $u \leq \frac{1}{2}\min(d(\mathbf{c},\mathbf{c}'))$, for all $\mathbf{c}' \neq \mathbf{c}$, then $\mathbf{x}$ must remain assigned to $\mathbf{c}$, and all distance calculations for $\mathbf{x}$ can be avoided.

Lemma 2 is applied as follows. Let $\mathbf{x}$ be any data point, let $\mathbf{b}$ be any centroid, and let $\mathbf{b}'$ be the same centroid as it was in the previous iteration. Suppose that we in the previous iteration knew a lower bound l' so that $d(\mathbf{x}, \mathbf{b}') \leq l'$. Then we can infer a lower bound l for the current iteration:

$$d(\mathbf{x}, \mathbf{b}) \geq \max\{0, d(\mathbf{x}, \mathbf{b}') - d(\mathbf{b}, \mathbf{b}')\} \geq \max\{0, l' - d(\mathbf{b}, \mathbf{b}')\} = l$$

Suppose $u(\mathbf{x}) \geq d(\mathbf{x}, \mathbf{c})$ is an upper bound on the distance between $\mathbf{x}$ and the centroid $\mathbf{c}$ to which $\mathbf{x}$ is currently assigned, and suppose $l(\mathbf{x}, \mathbf{c}') \leq d(\mathbf{x}, \mathbf{c}')$ is a lower bound on the distance between $\mathbf{x}$ and some other centroids $\mathbf{c}'$. If $u(\mathbf{x}) \leq l(\mathbf{x}, \mathbf{c}')$ it is not necessary to calculate $d(\mathbf{x}, \mathbf{c})$ or $d(\mathbf{x}, \mathbf{c}')$ since $d(\mathbf{x}, \mathbf{c}) \leq u(\mathbf{x}) \leq l(\mathbf{x}, \mathbf{c}') \leq d(\mathbf{x}, \mathbf{c}')$.

The accelerated k-means algorithm is outlined as follows:

1. For all centers $\mathbf{c}$ and $\mathbf{c}'$, compute $d(\mathbf{c}, \mathbf{c}')$. For all centroids $\mathbf{c}$, compute $s(\mathbf{c}) = \frac{1}{2}\min_{\mathbf{c}' \neq \mathbf{c}} d(\mathbf{c}, \mathbf{c}')$.

2. Identify all points $\mathbf{x}$ such that $u(\mathbf{x}) \leq s(c(\mathbf{x}))$.

3. For all remaining points $\mathbf{x}$ and centroids $\mathbf{c}$ such that

   i $\mathbf{c} \neq \mathbf{c}(\mathbf{x})$ and

   ii $u(\mathbf{x}) > l(\mathbf{x}, \mathbf{c})$ and

   iii $u(\mathbf{x}) > \frac{1}{2}d(\mathbf{c}(\mathbf{x}), \mathbf{c})$:

   (a) If $r(\mathbf{x})$ then compute $d(\mathbf{x}, \mathbf{c}(\mathbf{x}))$ and assign $r(\mathbf{x}) =$ false. Otherwise, $d(\mathbf{x}, \mathbf{c}(\mathbf{x})) = u(\mathbf{x})$.

   (b) If $d(\mathbf{x}, \mathbf{c}(\mathbf{x})) > l(\mathbf{x}, \mathbf{c})$ or $d(\mathbf{x}, c(\mathbf{x})) > \frac{1}{2}d(\mathbf{c}(\mathbf{x}), \mathbf{c})$ then compute $d(\mathbf{x}, \mathbf{c})$. If $d(\mathbf{x}, \mathbf{c}) < d(\mathbf{x}, \mathbf{c}(\mathbf{x}))$ then assign $\mathbf{c}(\mathbf{x}) = \mathbf{c}$.

4. For each centroid $\mathbf{c}$, let $m(\mathbf{c})$ be the mean of the points assigned to $\mathbf{c}$.

5. For each point $\mathbf{x}$ and centroid $\mathbf{c}$, assign

   - $l(\mathbf{x}, \mathbf{c}) = \max\{l(\mathbf{x}, \mathbf{c}) - d(\mathbf{c}, m(\mathbf{c})), 0\}$.

6. For each point $\mathbf{x}$ and centroid $\mathbf{c}$, assign

   - $u(\mathbf{x}) + d(m(\mathbf{c}(\mathbf{x})), \mathbf{c}(\mathbf{x}))$.
   - $r(\mathbf{x}) =$ true.

7. Replace each centroid $\mathbf{c}$ by $m(\mathbf{c})$.

13.3 Pre-emphasis

Pre- and de- emphasis are necessary with regard to data clustering because in the spectrum of a human speech signal, the energy in the signal decreases as the frequency increases as shown on Figure 5.3 [14]. Pre-emphasis increases the energy in parts of the signal by an amount inversely proportional to its frequency. Thus, as the frequency increases, pre-emphasis raises the energy of the speech signal by an increasing amount. This process serves to flatten the signal spectrum, and the flatter spectrum allows the data analysis to model the speech segment more accurately. Without pre-emphasis, any comparisons in k-means would incorrectly focus on the lower frequency components of speech, losing important information about certain sounds.

The pre-emphasis filter where x is the original speech signal, y is the filtered speech signal and n denotes the sample number:

$$y(n) = x(n) - \alpha \cdot x(n-1) \tag{13.2}$$

The inverse filter for de-emphasis is then

$$x(n) = y(n) + \alpha \cdot x(n-1) \tag{13.3}$$

The pre-emphasis factor α is computed as

$$\alpha = e^{-2Ft} \tag{13.4}$$

Where F is the frequency above which the spectral slope will increase by 6 dB per octave and where t is the sampling period of the sound [14].

13.4 Voiced - unvoiced - silence detection

To reduce the size of the k-means codebook after training, we wish to automatically remove the silence periods from the training set, and distinguish between unvoiced and voiced training vectors. A simple method for determining voice-unvoiced-silence regions (VUS) with good performance on clean speech is the spectral flatness measure, SFM [24]. SFM is defined as [24],

$$\mathrm{SFM}(k) = \frac{\left(\prod_{\omega=1}^{\Omega} X_k(\omega)\right)^{\frac{1}{\Omega}}}{\frac{1}{\Omega}\sum_{\omega=1}^{\Omega} X_k(\omega)} \tag{13.5}$$

The SFM is the ratio between the geometric mean and the arithmetic mean of windowed speech. The value is bound between zero and one, where unity SFM means the spectrum is flat, while an SFM close to zero means it is a peaky spectrum [24]. To improve the robustness of this VUS detection method a two level Schmidt trigger is used to decide the state based on the previous state. Quite simply, if the SFM value is in the uncertain range between the trigger levels, the state is set equal to the previous state.

13.5 Mel filterbank

A popular speech feature extractor used in ASR applications is the mel frequency cepstral coefficient algorithm (MFCC). Basically it is a filterbank approach first presented by Davis and Mermelstein [5]. They proposed a filterbank comprised of triangular filters of equal height with 10 linearly spaced centers below 1 KHz and 10 log-spaced filters above 1 KHz. They determine the base width of each filter by using the center frequency of the previous filter, as shown in Figure 13.3. Since then, improved variations of this algorithm has been proposed with different or variable number of filters [29], [31], [23], with equal area filters [29], variable frequency range [31], [24] and filter bandwidth related to filter center frequency [23].

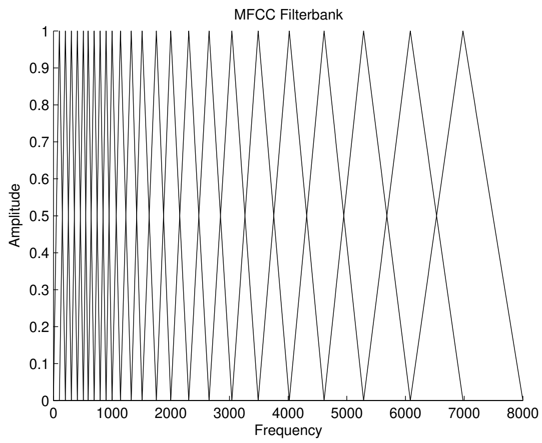


Figure 13.3: The MFCC filterbank as proposed by Davis and Mermelstein [5], but with 24 filters to accomodate a 16 kHz sampling rate. Center frequencies of filters 1-10 linearly spaced between 100 and 1000 Hz, and filters 11 - 24 are log-spaced at 5 filters per octave.

In this report we will focus on two filterbank approaches. The MFCC filterbank used in HTK [31], and the one used by Skowronski [23] which he terms HFCC or Human Factor Cepstral Coefficients.

13.5.1 HTK MFCC

The HTK MFCC is based on the following formula for relating linear frequency, f, to mel frequency, $\hat{f}$:

$$\hat{f} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{13.6}$$

The filter centers are then equally spaced between $\hat{f}_{\min}$ and $\hat{f}_{\max}$ (typically assumed to be 0 and the Nyquist rate respectively):

$$\begin{aligned} \Delta\hat{f} &= \frac{\hat{f}_{\min} - \hat{f}_{\max}}{I-1} \\ \hat{f}_{c_i} &= \hat{f}_{\min} + i\Delta\hat{f} \qquad i = 1, ..., I \end{aligned} \tag{13.7}$$

where I is the number of filters. Since our sample data has a sampling rate of 16 kHz, we construct our filterbank with $I = 32$ to retain $\Delta\hat{f} \approx 85.84$ Hz, which was used by Young [31] and Skowronski [23]. Filter bandwidth is determined in the same fashion as for Davis and Mermelstein [5], and all filters are triangular with unity heigth. The MFCC filterbank used in this report can be seen in Figure 13.4.

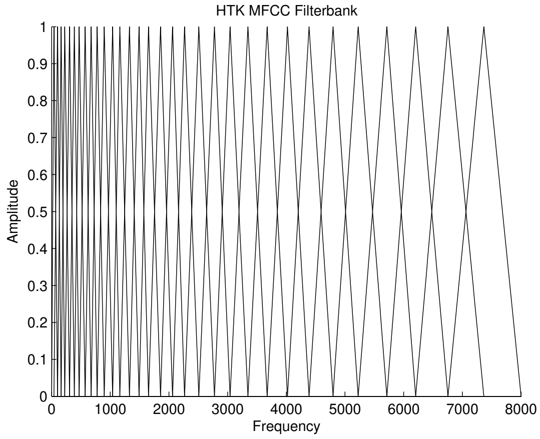


Figure 13.4: The MFCC filterbank used in this report. Similar to the one used by Young in his HTK [31], but with 32 filters to accomodate a 16 kHz sampling rate without changing the original bandwidth or spacing between filters.

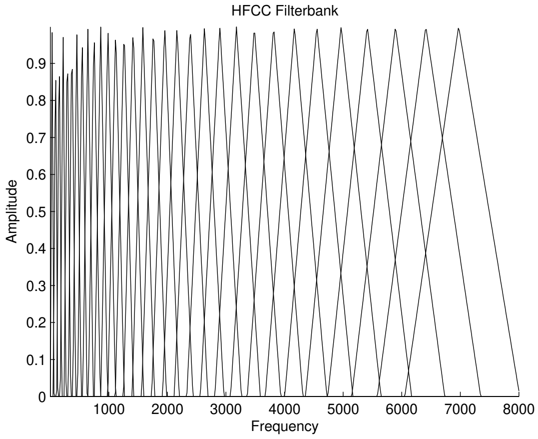


Figure 13.5: The HFCC filterbank with 32 filters. Notice the increased overlap between adjacent filters compared to those of the MFCC filters in Figure 13.3. The height variations are due to round-offs adjusting to the integer indexing of the x-axis.

13.5.2 HFCC

For his HFCC algorithm Skowronski determines filter bandwidth through equivalent rectangular bandwidth (ERB), which is defined by the following formula attributed to a paper by Moore and Glasberg in 1983, but described by Skowronski [23] since we were unable to retrieve the original paper:

$$ERB = \frac{\int |H(f)|^2 \mathrm{d}f}{|H(f)|^2} \tag{13.8}$$

where $|H(f)|$ is the amplitude of the filter transfer function.

Also we know [23]:

$$ERB = af_c^2 + bf_c + c \tag{13.9}$$

where center frequency f_c is in Hz with the curve fit between 100 and 6500 Hz, and:

$$\begin{aligned} a &= 6.23 \cdot 10^{-6} \\ b &= 93.39 \cdot 10^{-3} \\ c &= 28.5 \end{aligned} \tag{13.10}$$

With these formulas in place we proceed to construct the filterbank as follows:

1. In the implementation of the HFCC algortihm, Skowronski does not simply place the first and last filter center frequencies at $\hat{f}_{\min}$ and $\hat{f}_{\max}$ since there is no benefit from an overlap outside the frequency range.
   The calculation of $\hat{f}_{c_1}$ and $\hat{f}_{c_N}$, the first and last filter center frequencies, is placed in Appendix B, section B.1.

2. Additional filter centers are then equally spaced in mel frequency, so using $\hat{f}_{c_1}$ and $\hat{f}_{c_N}$ in Equation 13.7 it yields:

$$\begin{aligned} \Delta\hat{f} &= \frac{\hat{f}_{c_N} - \hat{f}_{c_1}}{N+1} \\ \hat{f}_{c_i} &= \hat{f}_{c_1} + (i-1)\Delta\hat{f} \qquad i = 2, ..., N-1 \end{aligned} \tag{13.11}$$

3. The actual center frequencies f_{c_i} are then calculated by the inverse of Equation 13.6:

$$f_{c_i} = 700 \cdot (10^{(\frac{\hat{f}_{c_i}}{2595})} - 1) \tag{13.12}$$

4. ERB_i is then calculated for f_{c_i} from Equation 13.9.

5. f_{l_i} and f_{h_i}, the low and high frequencies of the i'th filter is then calculated.
   The calculation is placed in Appendix B, section B.2.

6. Construct the filters in the frequency domain by connecting straight lines between f_{l_i} and f_{c_i}, and between f_{c_i} and f_{h_i}. The triangle has zero height at each end, and unity height at f_{c_i}.

The resulting HFCC filter bank is shown in Figure 13.5.

Chapter 14

Methods

In this section we will go through our experiments and describe in detail the setup for each experiment, and why the experiment was performed. For all experiments, we used Matlab Version 7.0.1.24704 (R14) Service Pack 1, run on Microsoft Windows XP Professional SP2. The machine used in this part was a AMD64 3700+ 2.2 GHz, with 1024 MB of RAM.
Due to the complexity of windows based systems, we do not have complete access to distributing system resources, and for instance algorithm run time experiments will be subject to fluctuations in allocated CPU and memory for any given task. We try to reduce this effect by increasing the priority of such tasks, which should be enough to show the run time tendencies.

14.1 Test data

All the test spectrograms for reconstruction are made with FFT-length = 512, Window length = 512, overlap = 256 unless it is stated otherwise.

The TIMIT database [33] was initially selected for use, but along the way it became clear we would need more speech data from each individual person to be able to do some of the tests properly, and so we started using the elsdsr database [34] in these cases. The Timit data, however, is a better quality than the elsdsr data, which is why we retaing using Timit for any experiment that does not require as large amounts of data from each person.
We mainly use speech from fpad0 from the dr6 folder in Timit, and the FMEV and MKBP from elsdsr. Others may be used for experiments needing data from multiple people.

The speech spectrograms for reconstruction using VQ will be submitted to the following set of masks to remove data.

1. Ideal binary mask: This is constructed as described in section 14.1.1, to approximate the effect of speech segregation by binary masking.

2. Random mask: This is constructed with the MATLAB *rand* function to randomly remove close to 70% of the pixels in a spectrogram, which we use to supplement the ideal binary mask, since this one will remove some of the high energy pixels.

3. Box mask: This simply removes a rectangular patch of data in a spectrogram, which we also use to supplement the ideal binary mask, since this one will remove some of the high energy pixels, as well as surrounding pixels in a larger area than for random mask. This mask also captures the effect of removed noise which was completely disrupting an entire frequency band.

For reproduceability, the "random mask" presented here has been produced and stored beforehand, and so will be the same one used each time. On Figure 14.1 the test data spectrograms with applied masking are shown.

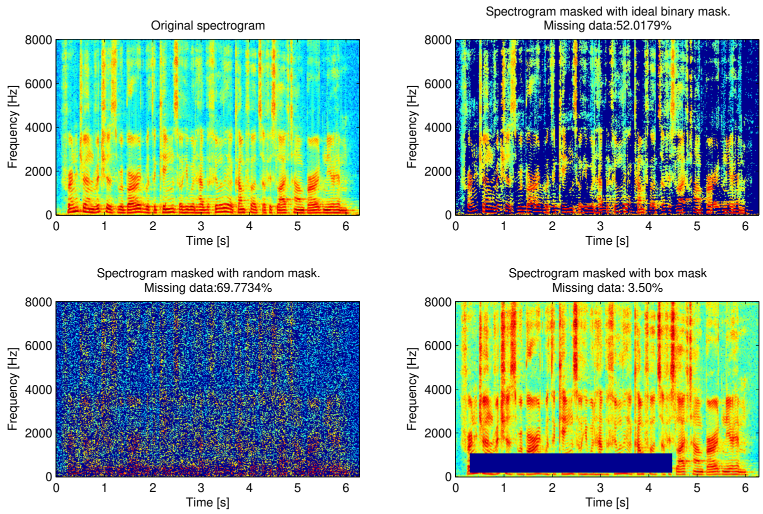


Figure 14.1: Types of missing data applied in reconstruction experiments. Note, that the random mask has been saved, and the exact same one will be used every time. The speech signal is the MKBP_Sr22.wav which will be used in several experiments.

Almost all reconstructions will be performed on the masked signals shown in Figure 14.1, except the female codebook reconstruction which will be done on the masked signals on Figure 14.2.

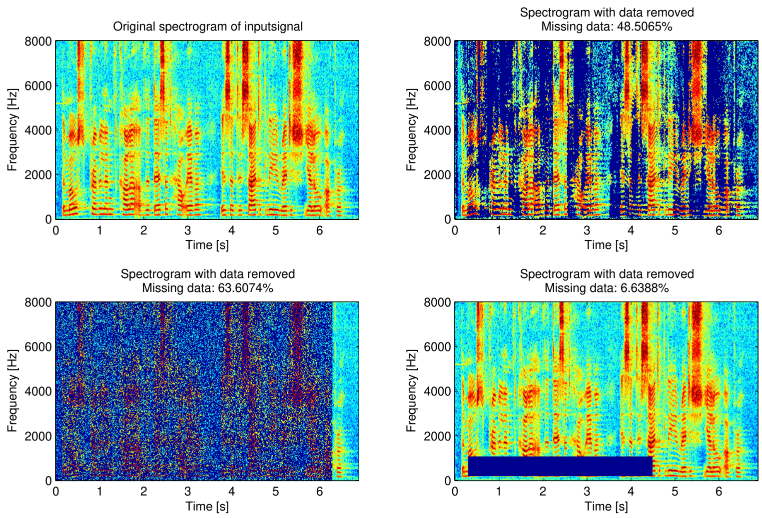


Figure 14.2: Types of missing data applied in female reconstruction experiments. Note, that the random mask and box masks are the exact same ones as used in the data on Figure 14.1. The speech signal is the FMEV_Sr9.wav which will be used in female reconstruction experiments.

The fact that we use clean speech data recorded in near noise-free environments, makes it easier for our algorithms to work on the relevant data without introducing artifacts in the speech signals based on noise, and furthermore, it allows us to better be able to notice noise originating from our reconstruction algorithms since these will not be drowned out by noise in the original signals.

14.1.1 Ideal binary masking

To simulate speech seperation by binary masking and to produce spectrograms of seperated speech data in a simple fashion, "ideal binary masking" can be used [27]. It is based on starting out with two seperate speech signals and calculating their respective log spectrograms $\mathbf{X}_A$ and $\mathbf{X}_B$, which are then added together. The resulting log spectrogram $\mathbf{X}_{Mix}$ corresponds to one found, had the two signals been recorded together [27].
If we consider the speaker from the log spectrogram $\mathbf{X}_A$ to be the speaker

of interest, the seperated log spectrogram is made from $\mathbf{X}_{Mix}$ by inserting zeroes at each bin where $\mathbf{X}_A$ contributed less energy than $\mathbf{X}_B$. That is:

- $\mathbf{X}_A + \mathbf{X}_B = \mathbf{X}_{Mix}$ and
- If $|\mathbf{X}_A(k,\omega)| < |\mathbf{X}_B(k,\omega)|$, then assign $\mathbf{X}_{Mix}(k,\omega) = 0$.
  $k \in [1,..,M-1]$, $\omega \in [1,..,\Omega-1]$

The reason that this is a good simulated segregation of speech signals, is that the log spectrogram of the mixture is almost exactly the maximum of the individual log spectrograms, with the maximum operating over small time-frequency regions. This is verified visually in Figure 14.3.

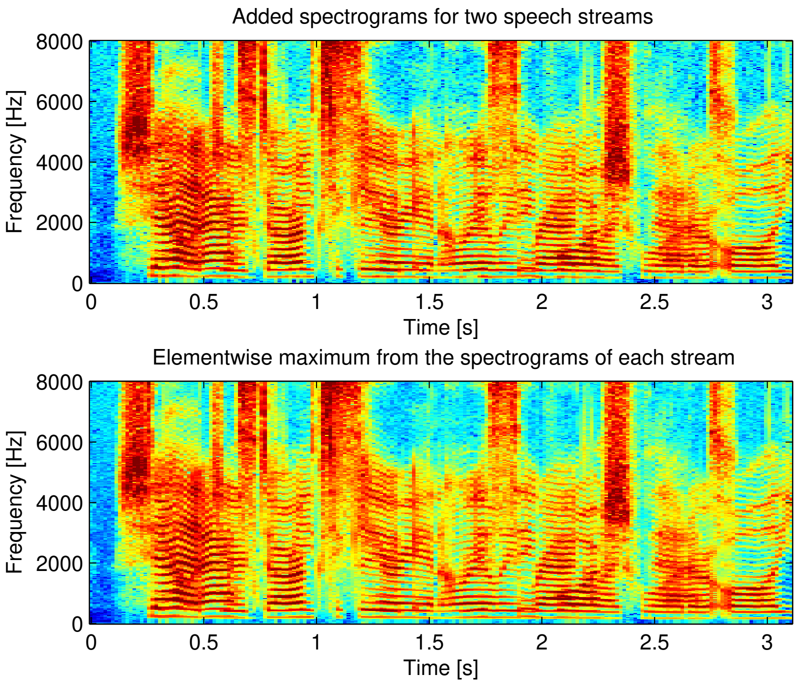


Figure 14.3: A spectrogram made by elementwise addition of two spectrograms of seperate speech signals (top). And a spectrogram made from the elementwise maximum values of two seperate speech signals. As we see, the two spectrograms are very similar, even in this extreme case where both signals are actually made by the same speaker. The speech signals used are from the test signals 'sa2.wav' and 'sa1.wav' from fpad0 in the dr6 test data folder of the TIMIT database.

The Ideal binary mask shown in Figure 14.1 is made from the speech signals MKBP_Sr22.wav, corresponding to $\mathbf{X}_A$), and MKBP_Sr21.wav, corre-

sponding to $\mathbf{X}_B$.
The Ideal binary mask shown on Figure 14.2 is made from the speech signals FMEV_Sr9.wav, corresponding to $\mathbf{X}_A$, and FMEV_Sr10.wav, corresponding to $\mathbf{X}_B$.

14.2 Data clustering

14.2.1 Pre-emphasis

Since for a human speech signal, the energy in the signal decreases as the frequency increases [14], any comparisons in k-means would incorrectly focus on the lower frequency components of speech, losing important information about certain sounds. So to allow the data analysis to more accurately model the speech segment we:

1. Flatten the spectrum through pre-emphasis filtering before doing the analysis
2. And recalculate the actual spectrum through de-emphasis, after doing the analysis

We do the pre-emphasis filtering directly on the original speech signal, before constructing the spectrogram, with the command:

$$filter([1, -15/16], 1, si);$$

where si is the signal. To revert back using de-emphasis, we filter the resulting reconstructed speech signal with the command:

$$filter(1, [1 - 15/16], si_r);$$

where si_r is the reconstructed signal.
To better appreciate the filtering effect, the spectrogram and a spectrum are plotted from a speech signal before and after pre-emphasis. These plots can be seen in section 15.1.1.

14.2.2 Generic Matlab *kmeans* function versus accelerated *k*-means

In this section we wish to show the advantages and disadvantages of using the accelerated k-means algorithm compared to using the generic MATLAB

kmeans function.
We will compare the algorithms with regard to both run-time and quality, while changing the parameters for amount of training data points, M_{train}, codebook size, K, and initialization. For this purpose we need a large trainingset from a single person, so we choose to use a person from the elsdsrs database. The MKBP set was chosen randomly.

Clustering run-time

We compare the run-time by plotting it as functions of K and M_{train} respectively, for both algorithms. Since we expect the mel filtered version (see section 14.2.3) to have an impact on run time, due to both reduced dimensionality for distance calculations, and additional multiplications while applying the filter, we choose to plot this as well.

The run-time is captured by using the *tic* and *toc* commands just before and after the respective k-means function is run. For comparison purposes both the functions are initialized to a fixed clustering using the furthest first method [4] as described in Section 13.1.
However, the initialisation is a part of the function for accelerated k-means and is therefore included in its run time, whereas it is calculated beforehand for the generic MATLAB *kmeans* function. Rather than changing the original code, though, or risking additional time penalties added due to internal matlab issues when running several functions within the *tic*, *toc* commands, we simply disregard the initialization time for the generic function. We expect that this will present no problems given that the generic function is likely to have the highest run times regardless.

Run-time is data-dependant since different data sets, and the sizes of the data set will play a role in both the amount of distance calculations in each iteration, as well as the amount of iterations performed before convergence is achieved. This means, that it is not possible to compare the exact run times of clusterings using different data sets, and to some extent that is what we are doing when we perform experiments with an increasing amount of training frames. However, we should be able to show a general trend for increasing M_{train}.
The nature of complex computer systems makes it difficult to have a constant amount of processing power, and so the run-time for either algorithm is not the same for each run. Not even with the exact same initialisations. This is expected to have little effect on the comparison between the different algorithms, though, so to reduce the already considerable overall run

time for the plots, we decide not to do multiple runs with each parameter set.

The MATLAB code for plotting run time as functions of K and M_{train} is *ktime.m*. The functions for doing accelerated k-means made by Elkan [6] are downloadable from *http://www-cse.ucsd.edu/ elkan/fastkmeans.html* and have only been slightly modified to allow for random initialisation.

Clustering quality

In this section we wish to show how close to optimal solutions the two algorithms offer, and how dependant this is to initialisation and to the number of clusters, K. The quality of a solution is based on the mean summed cluster distances, MSCD, which is calculated as follows.

If K is the number of clusters, J_i is the number of members in the i'th cluster, and $d(\mathbf{x}_i(j), \mathbf{c}_i)$ is the distance between $\mathbf{c}_i$, the centroid of the i'th cluster, and $\mathbf{x}_i(j)$, the j'th member of the i'th cluster. Then MSCD is defined as:

$$\mathrm{MSCD} = \frac{\sum_{i=1}^{K} \sum_{j=1}^{J_i} d(\mathbf{x}_i(j), \mathbf{c}_i)}{K} \tag{14.1}$$

We use this particular measure to asess quality, mainly due to practical reasons, since it is one of the outputs of the generic MATLAB *kmeans* function. The measure, however is highly dependant on the average amount of members per cluster, we will not be able to compare clusterings using different amounts of training vectors, M_{train}. We do not think this is a big issue however, since we want to avoid any comparison of clustering performance using different data sets, because it would also be very dependant on the homogeneity of the respective data.

Graphs showing MSCD with respect to the number of clusters for clustering the full set of training vectors provided by MKBP from the elsdsr database are plotted. This is to gain insight into what settings may provide good data clustering.
Also we compare the clustering results visually by plotting histograms showing the distribution of members for each cluster. After viewing some of these histograms, we realise that there usually is a dominant cluster with a lot of members, which leads us to plot a few codebooks along with the corresponding histograms to analyse the members of this particular cluster.

The MATLAB code for calculating and plotting quality as functions of K is *plotqual2.m* and *calcQual.m*. Generating histograms is trivial in MATLAB using the *hist* function.

14.2.3 Mel filtering used in clustering

As stated by Dan Ellis [7] the full spectra of speech may not be suited very well for efficient data clustering or for good reconstruction of missing data in speech. And in any case, half of the FFT length is a very high dimensionality to perform clustering on. Dan Ellis [7] suggests the use of the mel frequency based feature vectors. Two different feature extractors have been used in our tests, the MFCC filterbank described in section 13.5.1 and the HFCC filterbank described in section 13.5.2. The reason we have both filterbanks implemented is also to compare the older more used MFCC to the newer HFCC.

The MATLAB code for calculating and plotting the MFCC filterbank is *MFCCFilterMatrix.m*. The original function for creating the HFCC filterbank can be downloaded from http://www.cnel.ufl.edu/∼markskow/software/HFCC.zip, the file used in our case is fbInit.m

It would be nice to be able to just calculate the mel filtered spectrogram and then do clustering on the mel feature vectors and in turn use these to reconstruct a flawed mel spectrogram. Unfortunately it is not possible to recalculate the speech signal from a mel spectrogram, without paying a heavy toll on the speech quality. So instead of using nothing but mel filtered data, it has been used for mapping the clustering process.
We could have done this by modifying the distance calculation from standard squared euclidian distance between vectors, to squared euclidian distance between mel filtered vectors. This, however, would mean that a multiplication between the filter and the vector(s) would have to take place each time a distance is calculated, which we would like to avoid. So, in stead, we perform the clustering by running k-means on mel filtered data, and then afterwards we calculate the full cluster centroids from the map of cluster members in each cluster.

An example of clustering performed with mel filtering is plotted and shown in section 15.1.3, as well as comparisons of run time in section 15.1.2 and clustering quality in section 15.1.2. Note, that clustering quality is still calculated for the full cluster centroids in the same manner described in section 14.2.2.

The MATLAB code for applying a mel filterbank to the data, clustering it, and calculating the full cluster centroids, can be seen in *kmain.m*.

14.2.4 Voiced / unvoiced detection

In this section we exploit the fact that voiced and unvoiced sounds have distinctly different spectra as explained in section 5.3. This should allow for dividing the clustering codebook into two seperate codebooks. One for voiced and one for unvoiced speech. Such a division should help speed up the k-means algorithm, even though it would be run in two seperate instances, since it allows for less training vectors, M_{train}, and smaller amounts of clusters, K.
Also, we would expect the quality of the clustering to improve since we can be sure we do not have clusters sharing voiced as well as unvoiced members. An example of such a cluster might be low energy voiced frames grouped together with silent clusters.
Finally, the ideal binary mask will in some places leave the harmonics in a voiced part of the spectrogram, while removing the lower value bins in between. Then, when we calculate the euclidian distance between the remaining (high value) bins and possible templates, we may find that the match best to a remplate that has a flat spectrum of high values like some unvoiced spectra. But with seperate voiced / unvoiced codebooks, this should not be an issue.

We perform the voiced / unvoiced decision based on the SFM measure as described in section 13.4. One thing we have to do, however, when using spectral flatness to distinguish between voiced and unvoiced, is to make sure we do not filter the data with pre emphasis before calculating the SFM, since this will lead to flatter spectra and poor results.

The Schmidt trigger levels used are 0.36 and 0.47 respectively, which was found to work well for Skowronski [24], and seem to perform quite well on our data.

The method has been tested on a few files, and a resulting SFM and voiced / unvoiced decisions have been plotted. Also, the method has been attempted on a pre-emphasis filtered spectrogram as well as an ideally masked spectrogram.

The MATLAB code for calculating SFM and for making the voiced / unvoiced decision is *sfm.m*.

14.2.5 Speaker dependancy

In this section we wish to show the data clustering performance when we use speaker independant codebooks. Here we will present clustering quality graphs for gender-specific codebooks as well as a non-gender-specific codebook. This will give us a hint as to whether gender-specific codebooks are better than non-gender-specific codebooks. The graphs will be made according to the calculation of quality introduced in section 14.2.2.

We use as much speech as possible from up to 6 speakers (it has prooved impossible to go beyond this number due to the memory limitations of our available computer systems). For the quality measure to be comparable to each other we need to have an equal amount of training vectors, so each of the training sets are cropped to the size of the smallest one.

Due to the nature of the quality measure, we cannot directly compare the clustering quality graphs made here, to the graphs made for single speakers, without reducing the multiple speaker training sets to the same size as for the single speaker. Rather than doing that however, we will later investigate the performance during reconstructions in section C.4 which will be more readily comparable.

14.3 Spectrogram reconstruction using clustered data

In this section we basically want to try and reconstruct the data shown on Figure 14.1, with different settings, to determine the potential of the vector quantization approach. We mainly use a method for reconstruction, where only the remaining bins in a frame is used to select a template from. But, alternatively, we suggest and perform a few reconstructions using two other approaches. Each of these three approaches are explained in detail in sections 14.3.1, 14.3.2, and 14.3.3. After the description of the reconstruction methods, the different approaches regarding the handling of voiced / unvoiced codebooks in reconstruction are explained in section 14.3.4.

In common for each of the reconstruction methods are:

- That for ideal binary masked spectrograms, complete vectors of data can have been removed, and too few values for the comparison can easily result in very bad template decisions. To deal with this, we exploit the fact that an entirely removed vector is likely to have had low energy for all frequencies (ex. silence), and therefore we use the codebook centroid with the least energy as a template if there are below 10 values for comparison.

- That after each spectrogram reconstruction is complete, the phase is reconstructed using Griffin & Lim as described in section 9.3 with 200 iterations. The purpose of this is that we will then be able to perform a subjective evaluation of percieved speech quality using the grade system defined in section 6.2.

The MATLAB code for handling the vector quantization and calling the correct reconstruction function is *kmain.m*.

14.3.1 Remaining Bins Spectrogram Reconstruction

In this section we describe our main reconstruction method of using only the bins that were not affected by the masking process. This method assures us that we select a template based on knowledge we know to be true, which is an asset. However, it has the drawback of not being very reliable in cases where most of the bins in a frame have been removed in the masking process.

We perform this type of reconstruction one frame at a time as follows:

- The squared euclidian distance between the frame and each centroid in the codebook is calculated, using only the bins that were not set to zero in the masking process.
- The centroid with the shortest distance is chosen to be the template.
- The missing bins in the frame being reconstructed are filled out so they match the corresponding bins in the template.

For each of the three masked spectrograms shown on Figure 14.1 the following tests have been made using the remaining bins method, and the resulting $SNR_{spectrogram}$ as well as the PQE of the reconstructed speech signals have been noted in Appendix C.

- Overfitted reconstruction experiments where the number of clusters, K, is equal to the number of training vectors, M_{train}. Also for the case where the test signal itself is included in the training set. This will show the performance of the reconstruction method in itself, when the codebooks contain either perfect data, or at least data that has not been averaged over several frames respectively.

- Reconstruction experiments using a codebook with 500 clusters. The clustering process as well as the reconstruction will independantly be employed using HFCC-, MFCC- and a non-filtered approach to test all combinations of filter settings. This will show a more realistic performance of the reconstruction method than the overfitted experiments and should be able to hint at a good setting for mel filters.

- Reconstruction experiments using training sets for more than one speaker. For these cases, all training will be carried out using MFCC filtering, and the reconstruction will be done without any mel filter. We perform experiments with 3 types of codebooks. One for female speakers, one for male speakers, and one for a mix of the two.
  For varying complexity in the codebooks, we construct a set of them for 2, 4, and 6 people, remembering of course to balance the amount of male and female speakers in the mixed experiments. Each codebook is constructed with 500 clusters for the purpose of comparison with the single speaker experiments. Female codebook reconstruction, of course, should not be reconstructed using the three masked spectrograms shown on Figure 14.1, and so they will be done on the masked signals on Figure 14.2 in stead. These experiments will hint at how speaker dependant the vector quantization approach is.

When we use mel filtering in the reconstruction, we simply choose what template to use based on the squared euclidian distance between mel filtered frames and centroids, in stead of the regular full frequency frames. This will reduce the dimensionality and may be better at selecting templates that "sound" correctly.

Of these mentioned reconstruction experiments, the results, when using 500 clusters and MFCC filtering during training on each of the masks shown on Figure 14.1, are also to be found in section 15.2 for comparison with the other two reconstruction methods.

The MATLAB code for doing remaining bins reconstruction is *recon_rembin.m*.

14.3.2 Linear Interpolation Spectrogram Reconstruction

In this section we reconstruct the data shown on Figure 14.1 using linear interpolation to initially guess at the missing bins before we choose which codebook centroid to use for a template.

We use this method mainly as an attempt of working around the problems of using the remaining bins method on frames where few bins survived the masking process.
We perform this type of reconstruction one frame at a time as follows:

- A dummy vector is made by filling out missing bins through linear interpolation between remaining bins in the frame which is to be reconstructed.
- The squared euclidian distance between the dummy vector and each centroid in the codebook is calculated, using all the bins in the entire frequency range.
- The centroid with the shortest distance is chosen to be the template.
- The missing bins in the frame being reconstructed are filled out so they match the corresponding bins in the template.

Reconstructions with this method were performed for each of the masks shown on Figure 14.1. For these experiments we cluster the data to 500 clusters. We use MFCC filter during clustering but the reconstruction in itself is performed using only full length vectors and centroids.

The MATLAB code for doing remaining bins reconstruction is *recon_interp.m*.

14.3.3 Weighted Predecessor Spectrogram Reconstruction

In this section we basically want to try and reconstruct the data shown on Figure 14.1 in a less primitive way.

This reconstruction method seperates itself from the one used in section 14.3.1 by selecting a centroid from the codebook as a template using not only the remaining bins in the frame being reconstructed, but also data from its immediate predecessor. This approach, though grossly simplified, draws inspiration from the deformable spectrograms approach [19], where the high correlation between adjacent frames in a speech spectrogram is also exploited.

We perform this type of reconstruction one frame at a time as follows:

- A dummy vector is made from a weighted sum of the frame under reconstruction and its predecessor. This process is described by Equation 14.2.

- The squared euclidian distance between the dummy vector and each centroid in the codebook is calculated, using all the bins in entire frequency range.
- The centroid with the shortest distance is chosen to be the template.
- The missing bins in the frame being reconstructed are filled out so they match the corresponding bins in the template.

The first frame is reconstructed in the same way as described in section 14.3.1 since it has no predecessor.

If $|X_k|$ is the k'th frame in a spectrogram, $|X_{k-1}|$ its predecessor while w_1 and w_2 are their respective weights, then the dummy vector, $\mathbf{d}$ is calculated in the following way:

$$\mathbf{d} = w_1 \cdot |X_k| + w_2 \cdot |X_{k-1}| \tag{14.2}$$

where

$$w_1 + w_2 = 1$$

Reconstructions with this method were performed for each of the masks shown on Figure 14.1, with $w_1 = w_2 = 0.5$. For these experiments we cluster the data to 500 clusters. We use MFCC filter during clustering but the reconstruction in itself is performed using only full length vectors and centroids.

The MATLAB code for doing remaining bins reconstruction is *recon_weight.m*.

14.3.4 Voiced / unvoiced reconstruction

Reconstruction experiments using voiced / unvoiced codebooks are performed in three ways since we do not have a reliable method for making voiced / unvoiced decisions on the masked spectrograms:

Merged One way is simply the best match centroid from either codebook, in much the same way as had there only been one complete codebook. This will still have the advantage that the voiced and unvoiced parts were not mixed in the clustering.

Ideal Another way, is to do a voiced / unvoiced decision based on the original test signal before masking. This of course is impossible in a real scenario where we would not have the original signal, but it goes to show what benefits might come from divided codebooks, if a reliable method for deciding between voiced and unvoiced for a masked spectrogram can be made.

Linear interp. The last way, is to do a voiced / unvoiced decision based on the masked spectrogram after the bins which did not survive the masking process have been filled out using linear interpolation in each frame.

Reconstructions with each of these three methods were performed for each of the masks shown on Figure 14.1 using the remaining bins method described in section 14.3.1. For these experiments we cluster the data to 500 clusters, 300 voiced and 200 unvoiced. We use MFCC filter during clustering but the reconstruction in itself is performed using only full length vectors and centroids.

The MATLAB code for voiced / unvoiced reconstruction is *kmain.m* with *sfm.m* and *recon_rembin.m*.

Chapter 15

Results

In this chapter we show the results of the experiments described in Chapter 14. We will decribe in detail what we see in every result, and discuss these results and our opinions to what may cause them in the discussion chapter that follows.

15.1 Data clustering

15.1.1 Pre-emphasis

In this section we simply wish to visually verify that a speech spectrogram becomes more "flat" through pre emphasis filtering. For this purpose, a spectrogram as well as a spectrum from before and after pre emphasis have been plotted in Figure 15.1.

15.1.2 Generic matlab *kmeans* function versus accelerated *k*-means

In this section we wish to show the advantages and disadvantages of using the accelerated k-means algorithm compared to using the generic MATLAB *kmeans* function.
We will compare the algorithms with regard to both run-time and quality, while changing the parameters for the amount of training data frames, M_{train}, codebook size, K, and initialisation.

Clustering run-time

In this section we wish to compare the run-time by plotting it as functions of K and M_{train} respectively. This is done for both algorithms with and without mel filtering. Figure 15.2 and Figure 15.3 show this.

Clustering quality

In this section we wish to show how close to optimal solutions the two algorithms offer, and how this depends on initialisation and number of clusters, K. The quality of a solution is based on the mean summed cluster distances (MSCD), calculated as described in section 14.2.2, and will also be visualised with histograms showing cluster sizes in terms of amount of members.

For a general idea on the difference between the performance of the algorithms, as well as the effect of initialisation differences, cluster histograms, and the corresponding values of MSCD are shown on Figure 15.4. Note that high quality is indicated by a low value of MSCD.

On Figure 15.5 the mean summed cluster distances of results from the accelerated k-means algorithm are plotted with respect to different amounts of clusters for two different initialisation methods. Hereby we wish to gain insight into which initialisation to choose, as well as investigate the behaviour of random initialisation.

On Figure 15.6 a few of the resulting codebooks are plotted to explain the behaviour of the dominant cluster noticed on Figure 15.4.

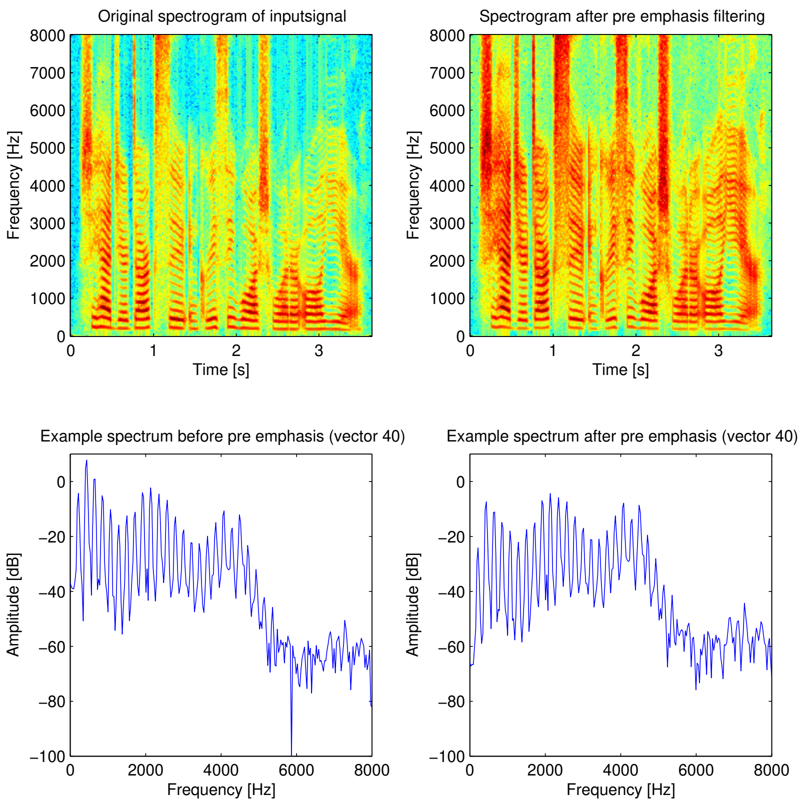
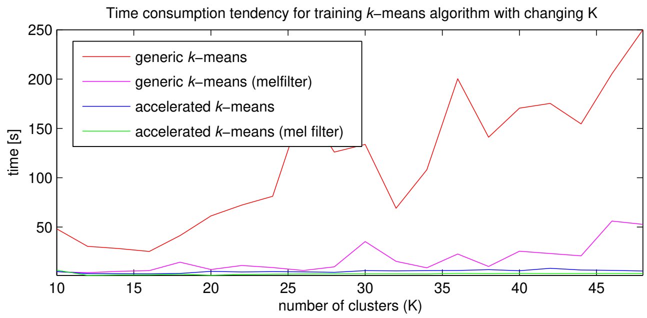


Figure 15.1: Spectrogram of TIMIT database speech signal from a female speaker before and after pre emphasis. File used is 'sa2.wav' from fpad0 in the dr6 test data folder.

On Figure 15.1 we see that the pre emphasis has negated the spectral slope and thereby made the spectra more “flat”, which is what we expected it to do. The selected spectrum is from a voiced speech section, and we see that after pre emphasis the peaks (harmonics) are more level with each other.



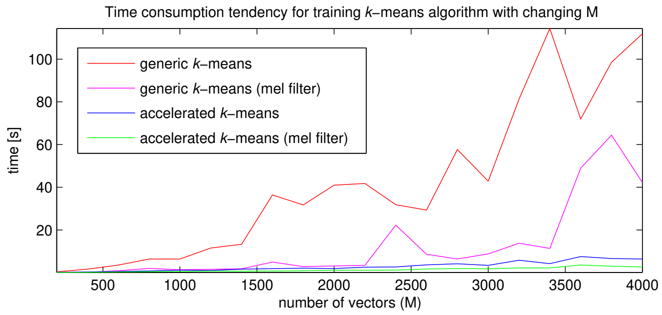


Figure 15.2: Time consumption with increasing amount of cluster centers K, and training data points M_{train}. The data used are the FMEV recordings from the elsdsr database ($M_{train} = 6178$). The generic function has been run with the furthest first initialisation input, in addition to the required inputs (training data and desired amount of clusters). The run times for the algorithms are shown with mel filtering toggled on and off, since this is expected to affect run-time.

On Figure 15.2 it can be seen that the accelerated k-means algorithm has a significantly lower run-time than that of the generic MATLAB function. It can also be seen that in most cases it is faster to perform clustering on the mel filtered data. Finally it can be seen that especially the run-time for the generic MATLAB function does not increase in a monotonous fashion. Due to the amount of time it takes to run the generic algorithm with high numbers of K, we have limited this graph to $K = 48$ and plotted extended graphs of the fast k-means run times on Figure 15.3.

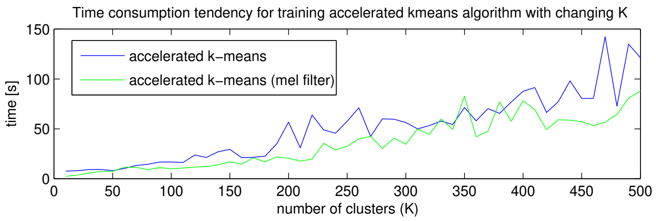


Figure 15.3: Time consumption with increasing amount of cluster centers K, for accelerated k-means. This is an extension of the graph on Figure 15.2, without the time consuming MATLAB generic function. The data used are the FMEV recordings from the elsdsr database ($M_{train} = 6178$)

On Figure 15.3 it can be seen that using mel filtering for training the codebook with accelerated k-means generally decreases run time compared to the unfiltered version.

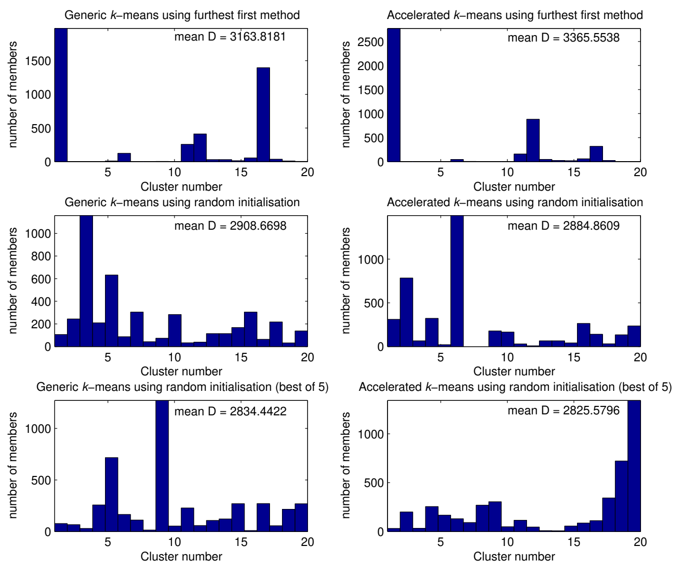


Figure 15.4: k-means clustering quality, defined as the mean of the sum of distances from each data point in a cluster to its cluster center (mean D on the figure). Since the random based initialisation will tend to have varying end results, it has been run 5 times with both the accelerated algorithm and the MATLAB generic function, and the best results have been selected for the bottom right hand and left hand examples respectively. The histograms allow us to view the amount of members in each cluster. The data used are the FMEV recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 6178$)

On Figure 15.4 an example of clustering for different initialisations of the generic Matlab *kmeans* function, as well as a comparison example of a clustering performed by the accelerated k-means algorithm, is shown . The histograms show the amount of members of each cluster, and the calculated MSCD. From the histograms we see that the resulting clustering depends on the initialisation, and that there are several local minima for the algorithms to end up in. Each of these local minima yields a different MSCD.

The smallest MSCD is achieved by running k-means with several random initialisations, and it is seen that the largest MSCD stems from the accelerated algorithm. It can also be seen that the resulting clusterings done by the generic function and the accelerated algorithm while using the same furthest first initialisation are not identical as expected.

Another thing noted from Figure 15.4 is that every clustering has a dominant cluster in which a large part of the training frames are members. On Figure 15.6 this is investigated further.

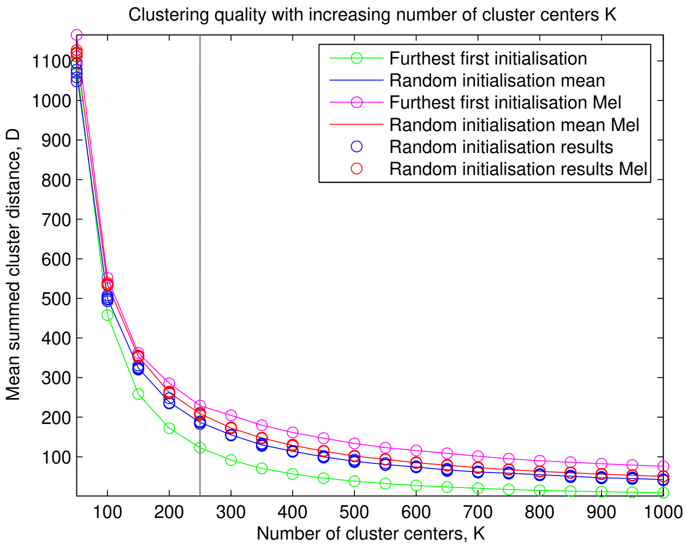


Figure 15.5: k-means clustering quality, defined as the mean of the sum of distances from each data point in a cluster to its cluster center (D on the figure). The data used are the MKBP recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 4367$).

The dependance of the MSCD, on number of clusters, K, is plotted on Figure 15.5. It can be verified that increasing the number of clusters decreases the MSCD, which is to be expected since the mean amount of frames per cluster will have decreased. It is seen that random initialisation yields a lower MSCD than furthest first initialisation with fewer than 100 clusters, while the opposite is the case for more than 100 clusters, and that the random initialisation does end up in different local minima, which are however very close, with a maximum deviation from the mean of 6.78% and an average deviation of 1.56%. For the Mel filtered version, the maximum deviation from mean is 1.74%, and the average deviation is 0.42%.
It is also seen that the graphs all decrease significantly until K reaches a certain point, and then stabilise with the addition of more clusters. The vertical line marks that K.

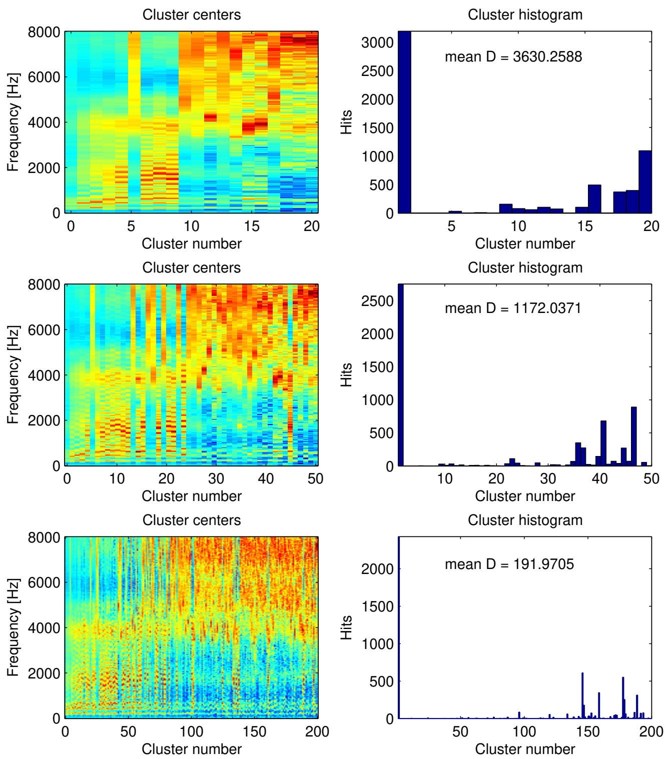


Figure 15.6: Codebooks sorted by frame energy and corresponding histograms for furthest first initialised training with 20, 50 and 200 clusters, K. The data used are the FMEV recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 6178$)

It can be seen on Figure 15.6 that the centroid of the cluster with the most members in each training instance is low in energy across the entire frequency spectrum, as the case is with silence. This is what we expected, since the silent parts of any of our speech signals constitute a significant portion and the fact that they are all low across the entire frequency spectrum make them very much alike.

15.1.3 Mel filtering

In this section we wish to show the basic visualisation of what mel filtering actually does to a spectrogram.
Figure 15.7 shows a spectrogram together with a MFCC and a HFCC filtered version, and Figure 15.8 shows an example of a mel filtered codebook.

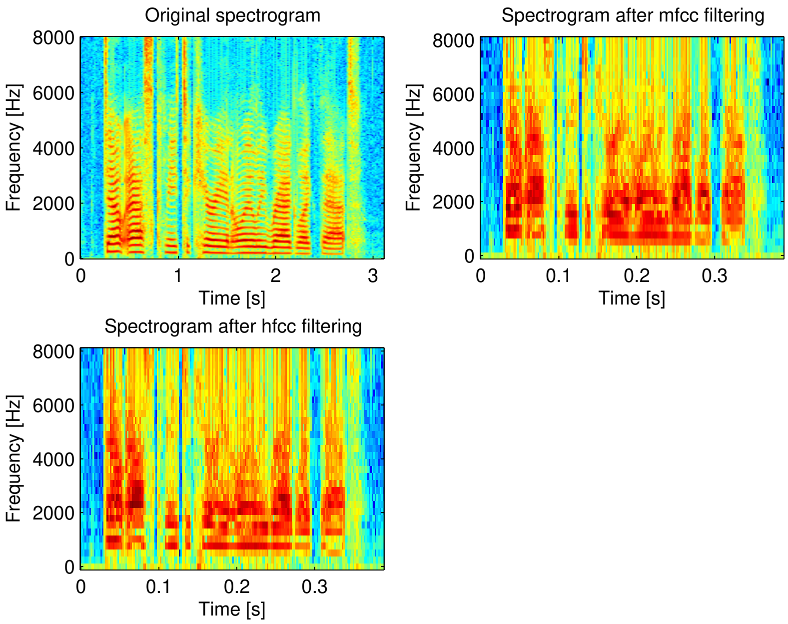


Figure 15.7: Spectrogram of a TIMIT database speech signal from female speaker before and after mel filtering. File used is 'sa2.wav' from fpad0 in the dr6 train data folder. HFCC and MFCC filters can be seen in section 13.5

It can be seen on Figure 15.7 how the spectrogram is affected by mel filtering. The filtering reduces the vector length, and the lower frequency parts of the spectra become more dominant in that they take up a larger percentage of space.

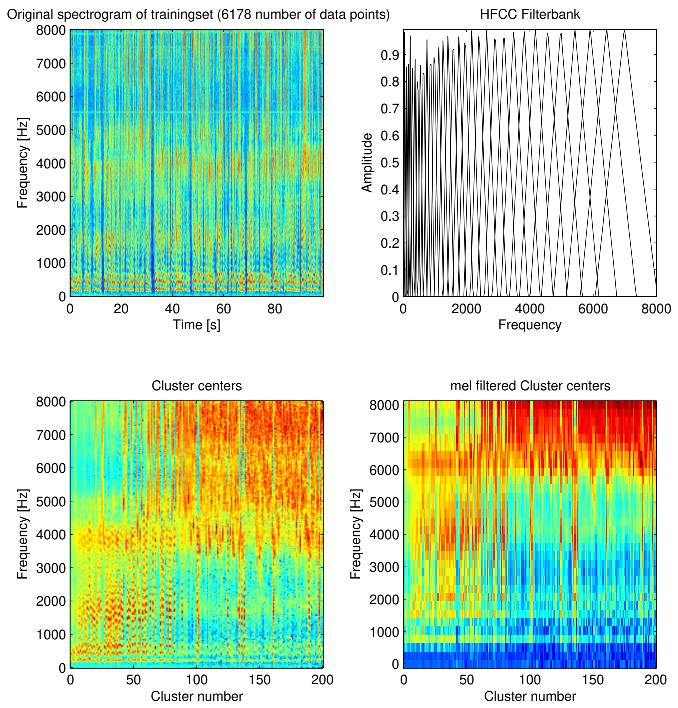


Figure 15.8: Example of training performed with HFCC filtering and $K = 200$. The data used are the FMEV recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 6178$).

An example of training with HFCC filtering can be seen on Figure 15.8. We can visually verify the reduced dimensionality of each frame in the codebook, and to some extent we can visually verify that the low frequency content takes up a larger percentage of bins in the mel filtered codebook.

15.1.4 Voiced / unvoiced detection

In this section we wish to show the performance, of the SFM measure in determining voiced and unvoiced parts of a speech signal.
Figure 15.9 shows an example of a divided voiced / unvoiced codebook.
Figure 15.10 shows the calculated SFM for a speech signal together with the corresponding voiced / unvoiced decision and the spectrogram. Similar plots are shown on Figure 15.11 and Figure 15.12 for a pre-emphasis filtered spectrogram and an ideal binary masked spectrogram respectively. Also on

Figure 15.13 the calculated SFM for the ideal binary masked spectrogram, which has been linearly interpolated is compared to the SFM calculated for the original spectrogram.
Figure 15.14 and 15.15 shows run time and quality respectively as a comparison of using one overall codebook or a divided voiced / unvoiced codebook.

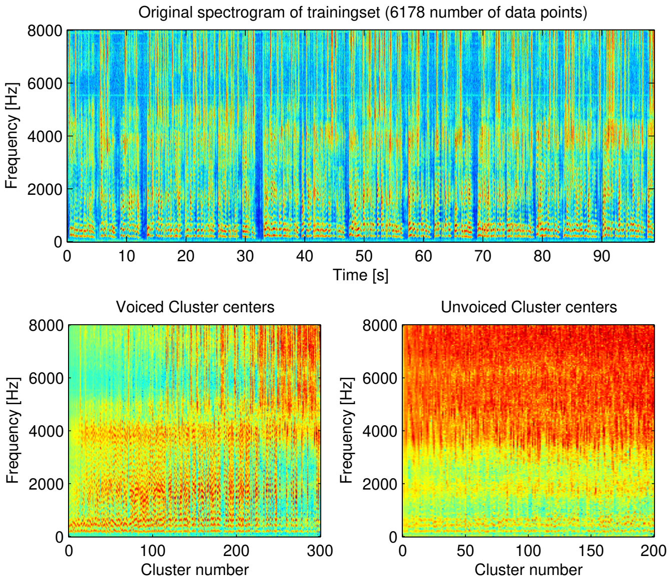


Figure 15.9: Example of training performed with $K_{voiced} = 100$ and $K_{unvoiced} = 50$. The data used are the FMEV recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 6178$)

An example of a training divided into voiced and unvoiced codebooks can be seen on Figure 15.9. It is seen that some unvoiced speech is represented in the voiced codebook, and some voiced speech is likewise represented in the unvoiced codebook.

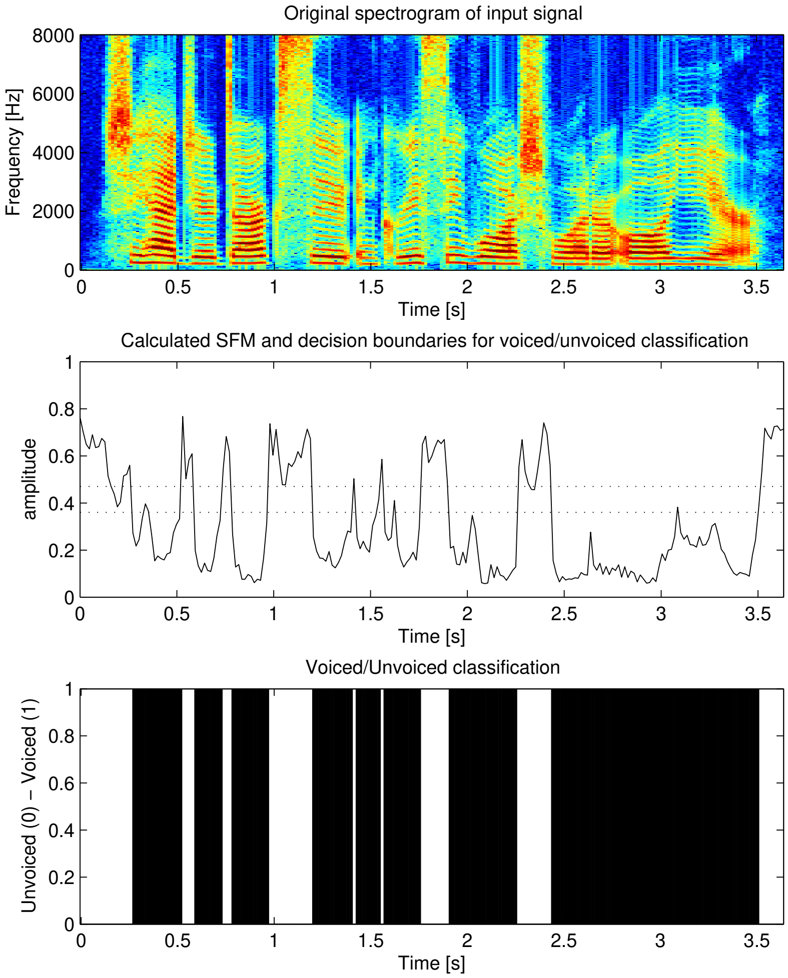


Figure 15.10: Spectrogram of TIMIT database speech signal from female speaker with calculated SFM and voiced/unvoiced decisions. File used is 'sa1.wav' from fpad0 in the dr6 train data folder. In the voiced unvoiced classification plot, the dark areas correspond to voiced frames in the spectrogram.

It can be seen on Figure 15.10 that the schmidt trigger on SFM performs the voiced / unvoiced decision reasonably well, however there are misclassifications such as the prolonged voiced decision in the end of the signal.

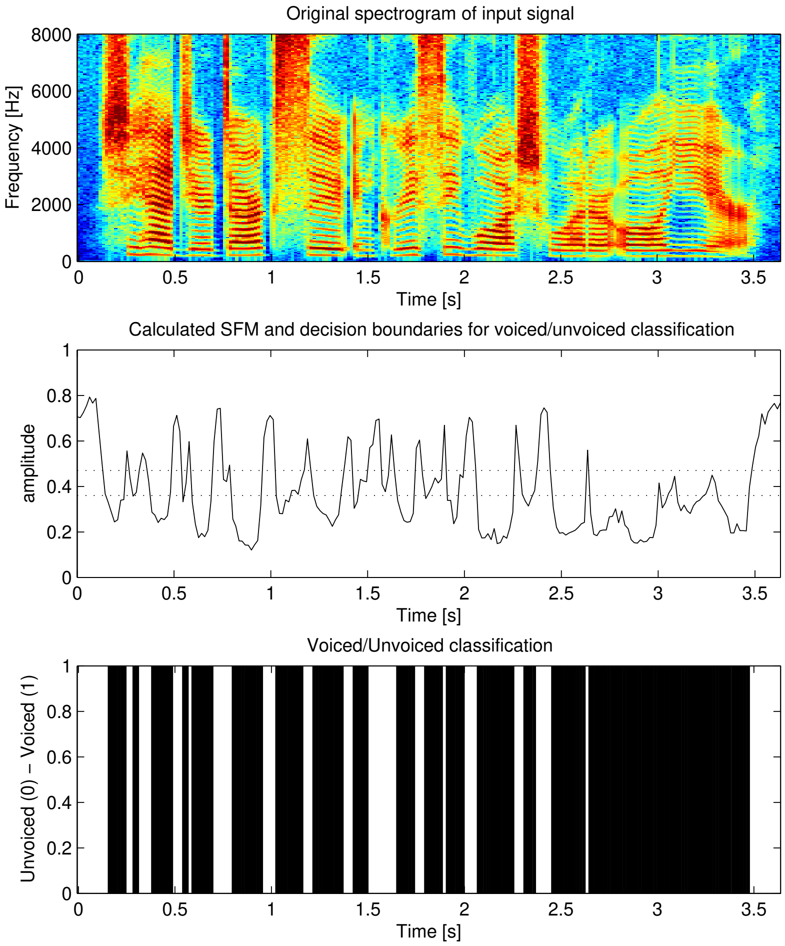


Figure 15.11: Spectrogram of pre emphasis filtered TIMIT database speech signal from female speaker with calculated SFM and voiced/unvoiced decisions. File used is 'sa1.wav' from fpad0 in the dr6 train data folder. This is basically the same as Figure 15.10, but with pre emphasis filtering. In the voiced unvoiced classification plot, the dark areas correspond to voiced frames in the spectrogram.

It can be seen on Figure 15.11 that the schmidt trigger on SFM performs the voiced / unvoiced decision quite poorly. The calculated SFM has become less robust and is not suited to the trigger levels.

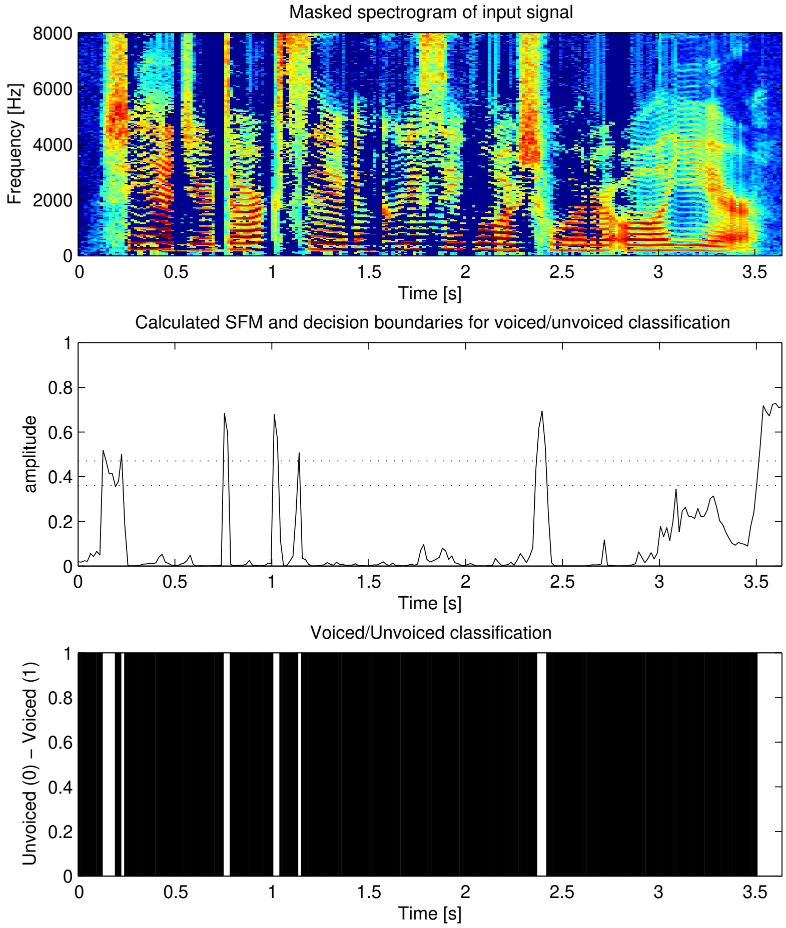


Figure 15.12: Spectrogram of ideal binary masked TIMIT database speech signal from female speaker with calculated SFM and voiced/unvoiced decisions. File used is 'sa1.wav' from fpad0 in the dr6 train data folder, which has been masked from a mixture with 'sa2.wav' from the same folder. In the voiced unvoiced classification plot, the dark areas correspond to voiced frames in the spectrogram.

It can be seen on Figure 15.12 that the schmidt trigger on SFM performs the voiced / unvoiced decision very poorly. The calculated SFM is zero most of the time, and low in the entire span of masked data.

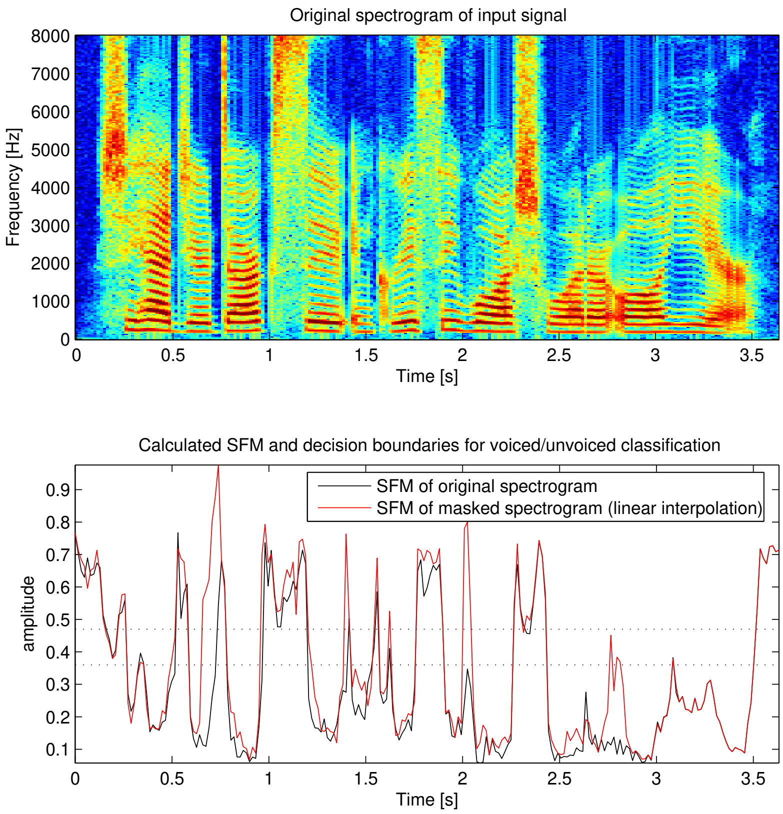


Figure 15.13: Spectrogram of TIMIT database speech signal from female speaker with calculated SFM of the original spectrogram as well as of the ideal binary masked version which has had the missing bins filled out using linear interpolation. File used is 'sa1.wav' from fpad0 in the dr6 train data folder, which has been masked from a mixture with 'sa2.wav' from the same folder.

It can be seen on Figure 15.13 that the calculated SFM of a masked and linearly interpolated spectrogram resembles that of the original spectrogram, especially compared to the SFM shown on Figure 15.12. However there are intervals where they each fall on different sides of the Schmidt trigger levels.

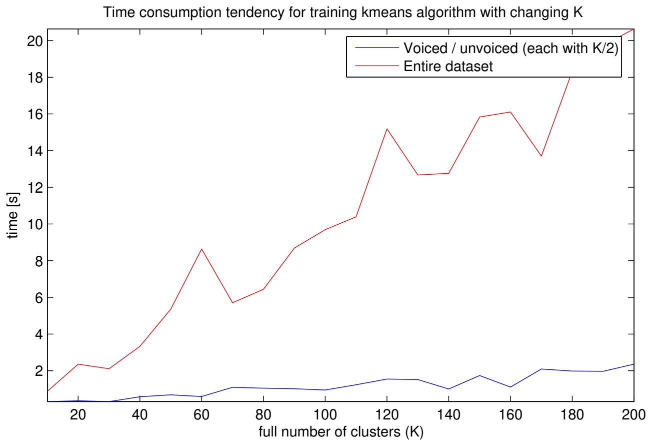


Figure 15.14: Graphs showing run time for the accelerated k-means algorithm for either a single codebook, or the divided voiced / unvoiced codebooks. For any given K on the x-axis, the voiced and unvoiced codebooks are run with $\frac{K}{2}$ for a better comparison between the graphs. The data used are the MKBP recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 4367$, divided as $M_{voiced} = 3260$, and $M_{unvoiced} = 1107$.)

On Figure 15.14 is is seen, that dividing the data clustering into two seperate runs decreases the overall run time significantly for the same complete amount of codebook clusters and that the higher the amount of clusters get, the more time is saved by using seperate voiced / unvoiced codebooks.

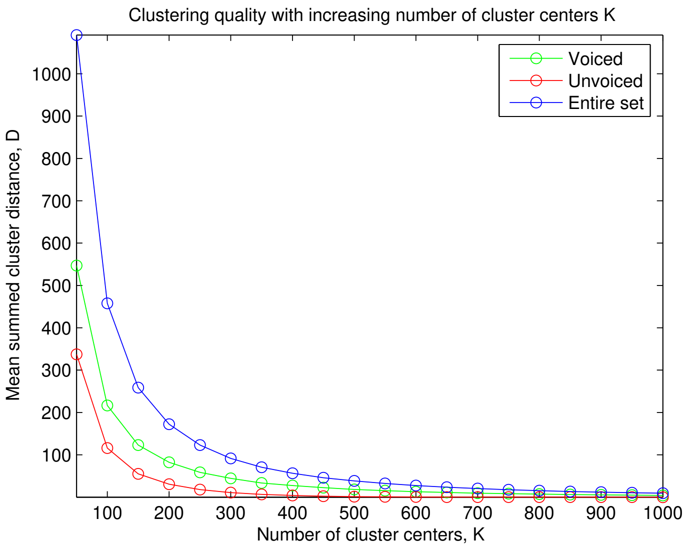


Figure 15.15: Graphs showing resulting clustering quality for the accelerated k-means algorithm for either a single codebook, or the divided voiced / unvoiced codebooks. The data used are the MKBP recordings from the elsdsr database. Training was performed on the full set ($M_{train} = 4367$, divided as $M_{voiced} = 3260$, and $M_{unvoiced} = 1107$). Note that a low value for D (MSCD), means a high clustering quality.

On Figure 15.15 is is seen that dividing the data clustering into two seperate runs does not reduce the quality of the clustering. In fact for fewer than 500 clusters it can be verified that seperating into voiced / unvoiced codebooks with 250 clusters in each would yield higher quality.

15.1.5 Speaker dependancy

In this section we wish to show the data clustering performance when using speaker independant codebooks. Here we will present clustering quality graphs for gender-specific codebooks as well as a non-gender-specific codebook.
Due to the nature of the quality measure, we cannot directly compare the clustering quality graphs made here to the graphs made for single speakers, without reducing the multiple speaker training sets to the same size as for the single speaker. Rather than doing that however, we will later investigate the performance during reconstructions in section 15.2 and C.4.

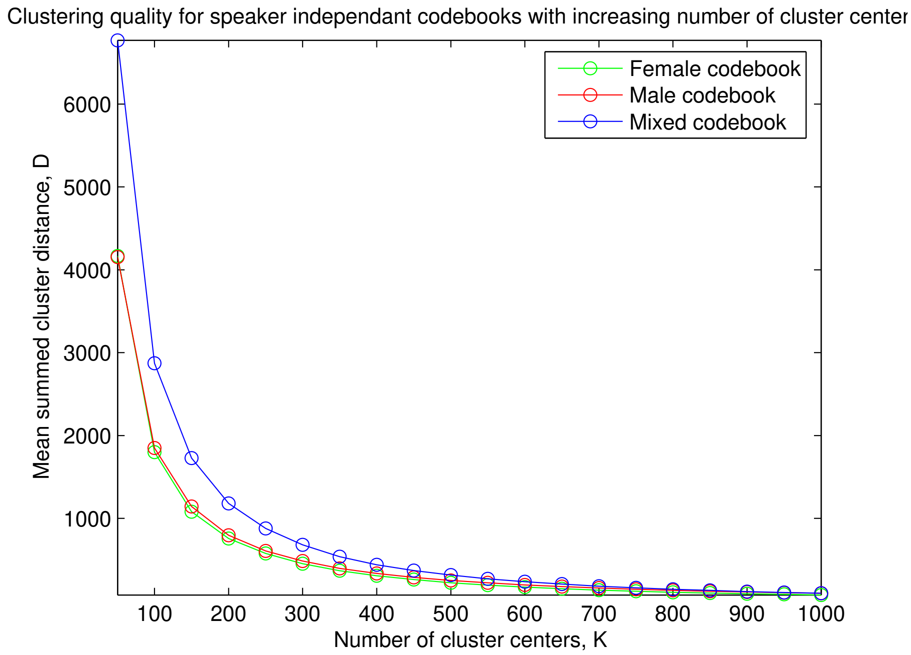


Figure 15.16: Graphs showing resulting clustering quality for speaker independant codebooks. The data used are the male and the female recordings from the elsdsr database. Training was performed on the alphabetically first 6 speakers cropped to the same length. ($M_{train} = 24317$). Note that a low value for D (MSCD), means a high clustering quality.

On Figure 15.15 is is seen that the mixed codebook has a poorer quality for any given K, however with high amounts of clusters, the difference between the graphs is very small. The female and male codebooks are very close to being of the same quality.

15.2 Spectrogram reconstruction using clustered data

In this section we show results from trying to reconstruct the speech shown in the top left of Figure 14.1 from the different masked versions. The purpose of this is to be able to evaluate the performance of the algorithms used, and what effect the different settings for filtering, and codebook type have. The amount of data generated by doing reconstructions with all the different settings, however, is substantial, so for the pupose of organising the results in a comprehensible way, the bulk of the reconstruction results have been placed in Appendix C as stated in section 14.3.1, while a set of results with the most successful filter settings is presented here as well.
The reconstruction results are evaluated through $SNR_{spectrogram}$ as well as PQE as described in Chapter 6, and these values are noted in tables in Appendix C for the following experiments performed on each of the three masks:

1. Overfitted reconstruction experiments where the number of clusters, K, is equal to the number of training vectors, M_{train}. Also for the case where the test signal itself is included in the training set.

2. Reconstruction experiments using a codebook with 500 clusters. The clustering process as well as the reconstruction will independantly be employed using HFCC-, MFCC- and a non-filtered approach to test all combinations of filter settings.

3. Reconstruction experiments using training sets for more than one speaker. We perform experiments with 3 types of codebooks. One for female speakers, one for male speakers and one for a mix of the two.
   The test data for male and mixed codebook reconstructions are the ones shown on Figure 14.1, while the test data for female codebook reconstructions are the ones shown on Figure 14.2.

Except for the speaker dependancy reconstruction experiments, training has been carried out with the full set of training data from MKBP recordings from the elsdsr database.
After each spectrogram reconstruction, the phase was estimated using Griffin & Lim with 200 iterations. This is also the case for the masked spectrograms themselves.

The results in this section are organized as follows:

- First in the following subsection, for the purpose of comparison, there are the results of using the masked spectrograms themselves to make the speech signal from, so that we can better analyse the reconstruction results.

- In the same subsection, we will present the performance of the different types of reconstructions. We use the remaining bins, linearly interpolated frames, and weighted predecessor methods, for a fixed filter and codebook setting.

- In the end of same subsection, we will present the performance of the different ways of determining VUS in the masked spectrogram. Reconstruction is done, using remaining bins method for a fixed filter and codebook setting.

- Then, in subsection 15.2.2 we will show a selection of reconstructed spectrograms along with their respective original and masked spectrograms.

- Finally, in subsection 15.2.3, there will be a summary of the reconstruction results, including the ones drawn from the results in Appendix C

15.2.1 Reconstruction tables

The tendencies noticed in the tables of this section are summed up in subsection 15.2.3.

Table 15.1: **Masked spectrograms** - Masked spectrograms for comparison purposes

Type of mask	*Ideal binary mask*	*Random mask*	*Box mask*
$SNR_{spectrogram}$	18.65 dB	2.98 dB	15.71 dB
PQE	3	3	3

Table 15.2: **Spectrogram reconstruction methods** - Reconstructions, where $K = 500$, MFCC filtering was used in training and the methods described in sections 14.3.1, 14.3.2, and 14.3.3 respectively, were used to do reconstruction.

Type of mask	*Ideal binary mask*	*Random mask*	*Box mask*
Remaining bins			
$SNR_{spectrogram}$	11.85 dB	8.61 dB	20.07 dB
PQE	3	3	3
Linear interpolation			
$SNR_{spectrogram}$	16.62 dB	8.94 dB	19.13 dB
PQE	3	3	3
Weighted predecessor			
$SNR_{spectrogram}$	19.24 dB	6.33 dB	19.38 dB
PQE	3	3	3

Table 15.3: **Reconstruction VUS methods** - Reconstructions, where $K_{voiced} = 300$, $K_{unvoiced} = 200$, MFCC filtering was used in training and VUS was done in the different ways described in section 14.3.4.

Type of mask	*Ideal binary mask*	*Random mask*	*Box mask*
Ideal VUS			
$SNR_{spectrogram}$	12.85 dB	8.80 dB	19.59 dB
PQE	3	3	3
Merged codebook			
$SNR_{spectrogram}$	10.80 dB	8.89 dB	19.55 dB
PQE	3	3	3
Linear Interp. VUS			
$SNR_{spectrogram}$	7.28 dB	7.10 dB	17.34 dB
PQE	3	3	3

15.2.2 Reconstruction examples

In this section we show selected examples of reconstructions, to support visual understanding of the reconstruction process. The examples have been chosen to cover mainly experiments with rather good performance, since these provide the best insight. No mel filtering was used in the reconstruction part for any of the examples in this section, and unless otherwise stated, 500 clusters have been used, and training was performed using MFCC filtering.

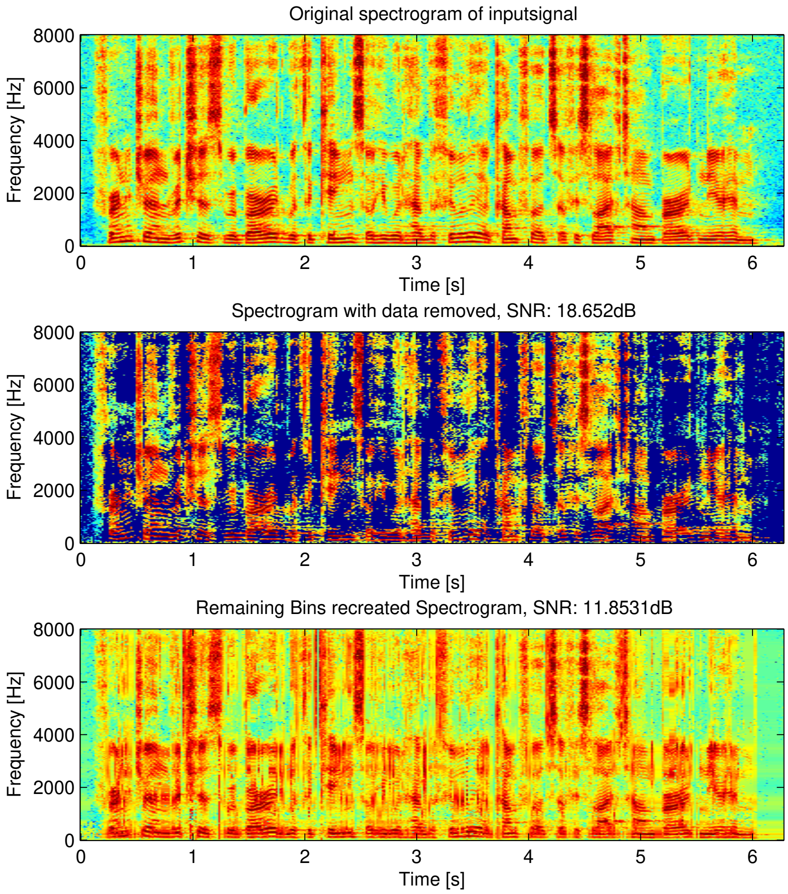


Figure 15.17: Reconstruction of the test signal with ideal binary masking using the remaining bins methods. Training was performed on the full set of MKBP training recordings from the elsdsr database.

On Figure 15.17 an example of a reconstruction from a signal that has been masked with the ideal binary mask is seen. After the reconstruction the spectrogram has regained some resemblance to the original spectrogram, but a lot of the details have been lost, and much of the inserted data seems to contain a higher energy level than that of the original spectrogram.
On Figure 15.18 examples are shown of reconstructions of the masked spectrogram on Figure 15.17, using alternative methods. We see that for each of the reconstructions the spectrogram has regained alot in resemblance to the original spectrogram shown on Figure 15.17.
With the VUS reconstruction, however, we see that a few of the former low energy areas have been reconstructed with high energy content, and though it has better $SNR_{spectrogram}$ it resembles the remaining bins reconstruction shown on Figure 15.17.

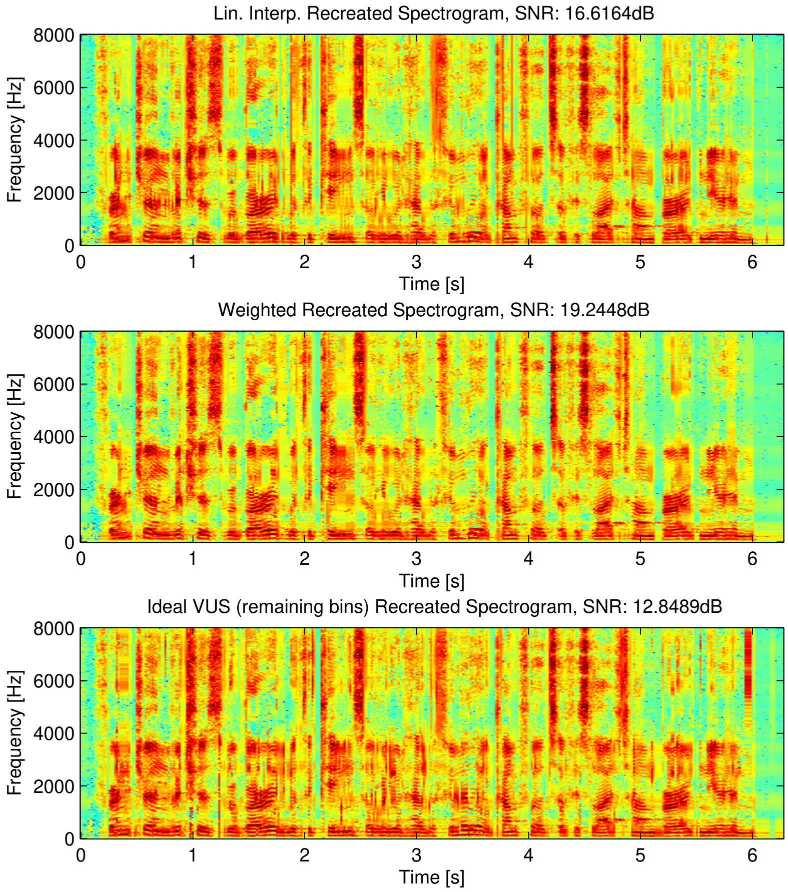


Figure 15.18: Reconstruction of the test signal with ideal binary masking using the alternative reconstruction methods including ideal VUS reconstruction. Training was performed on the full set of MKBP training recordings from the elsdsr database. This original and masked spectrogram for these reconstructions can be seen on Figure 15.17.

On Figure 15.18 with the Linear interpolated reconstruction we see that it has a tendency in some places to "draw vertical lines" in the spectrogram, meaning that in some instances of voiced speech, where all the high frequency bins have been removed by the mask, the linear interpolation method may choose an unvoiced template with high energy content for most of the high frequency bins. Other than that, the reconstruction using the linear interpolation method looks more like the original than the remaining bins reconstruction shown on Figure 15.17 does. It also has a higher $SNR_{spectrogram}$.
With the weighted predecessor method we plainly see a closer resemblance to the original spectrogram shown on Figure 15.17 than for any of the other reconstructions on Figure 15.18 as well as for the one on Figure 15.17.

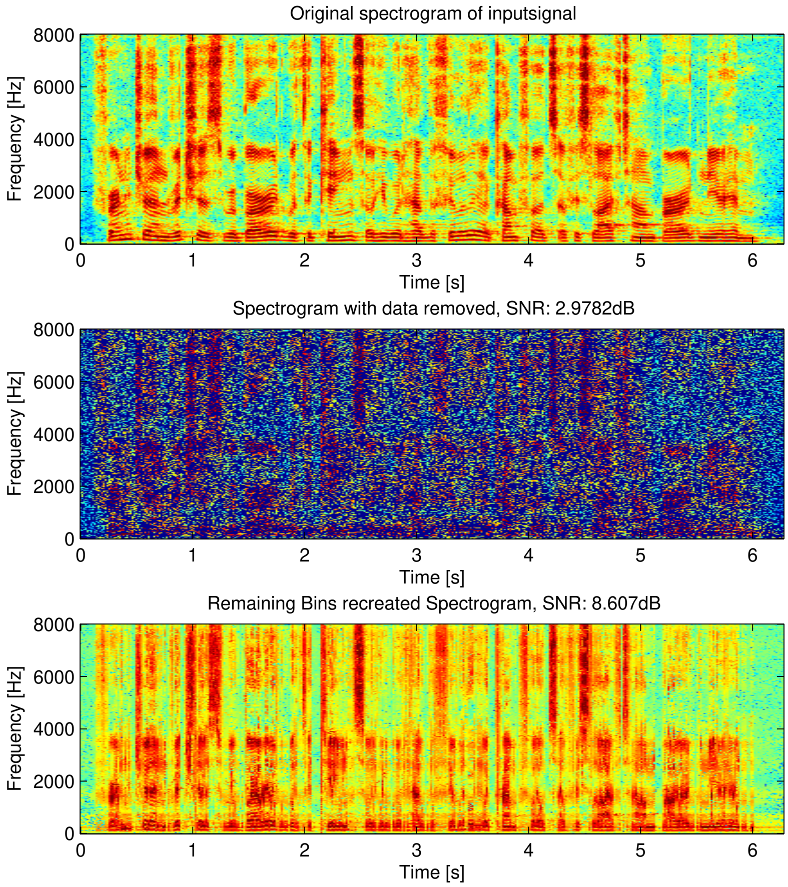


Figure 15.19: Reconstruction of the test signal with random masking using the remaining bins methods. Training was performed on the full set of MKBP training recordings from the elsdsr database.

On Figure 15.19 an example of a reconstruction from a signal that has been masked with the random mask is seen. After the reconstruction the spectrogram has regained some resemblance to the original spectrogram, but a lot of the details and structure have been lost. The harmonics for instance seem not to have survived the masking and reconstruction process. Generally the same tendencies are seen in the related reconstructions using alternative methods in Figure 15.20. None of the methods have been able to perform a good reconstruction of the harmonics, allthough they have certainly improved on the masked spectrogram.

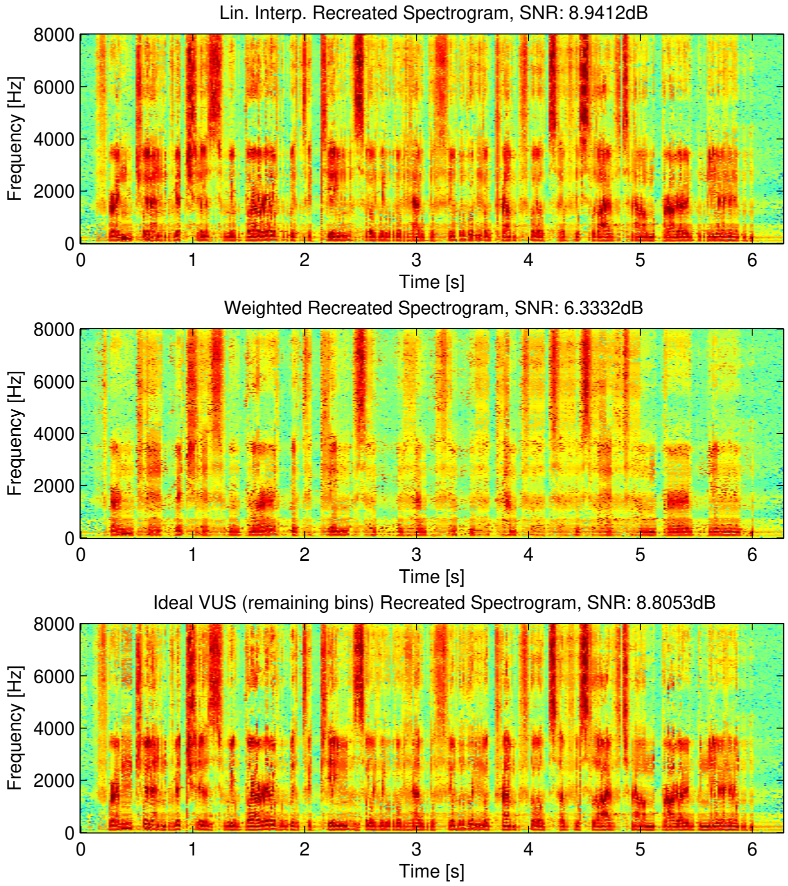


Figure 15.20: Reconstruction of the test signal with random masking using the alternative reconstruction methods including ideal VUS reconstruction. Training was performed on the full set of MKBP training recordings from the elsdsr database. This original and masked spectrogram for these reconstructions can be seen on Figure 15.19.

On Figure 15.20 we notice that the only method which visually stands out with reconstruction of the random masked spectrogram is the weighted predecessor method. It has an even poorer performance than the rest of the methods, and we see that almost no detail has been achieved.

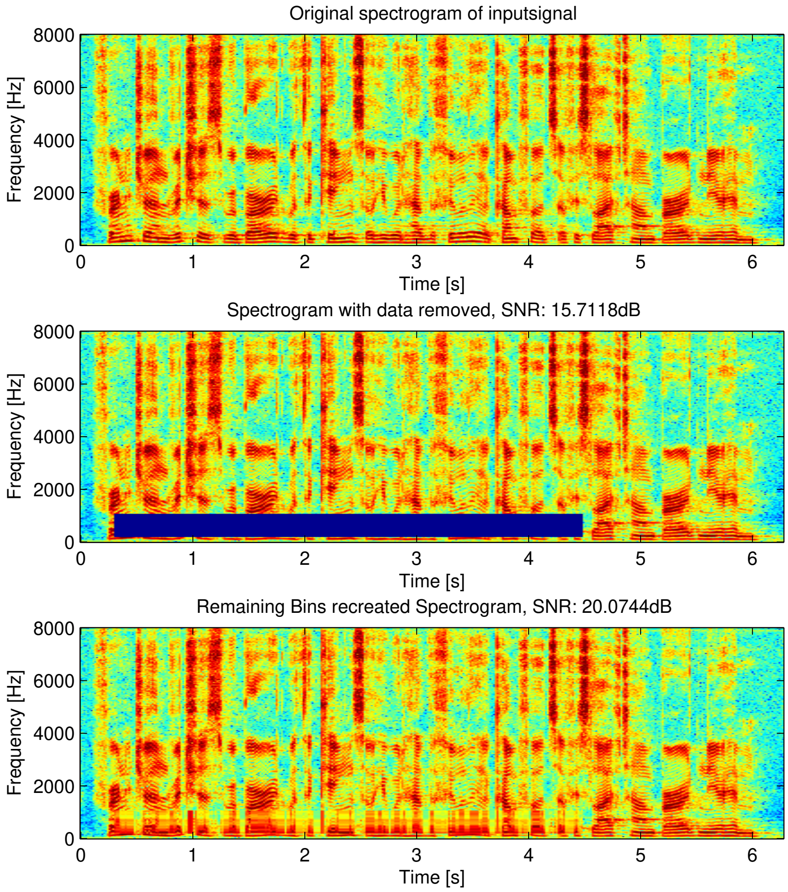


Figure 15.21: Reconstruction of the test signal with box masking using the remaining bins methods. Training was performed on the full set of MKBP training recordings from the elsdsr database.

On Figure 15.21 an example of a reconstruction from a signal that has been masked with the box mask is seen. After the reconstruction the spectrogram has regained some resemblance to the original spectrogram, but a lot of the details and structure have been lost. The harmonics for instance seem not to have survived the masking and reconstruction process. Generally the same tendencies are seen in the related reconstructions using alternative methods in Figure 15.22. None of the methods have been able to perform a good reconstruction of the harmonics, allthough they have certainly improved on the masked spectrogram.

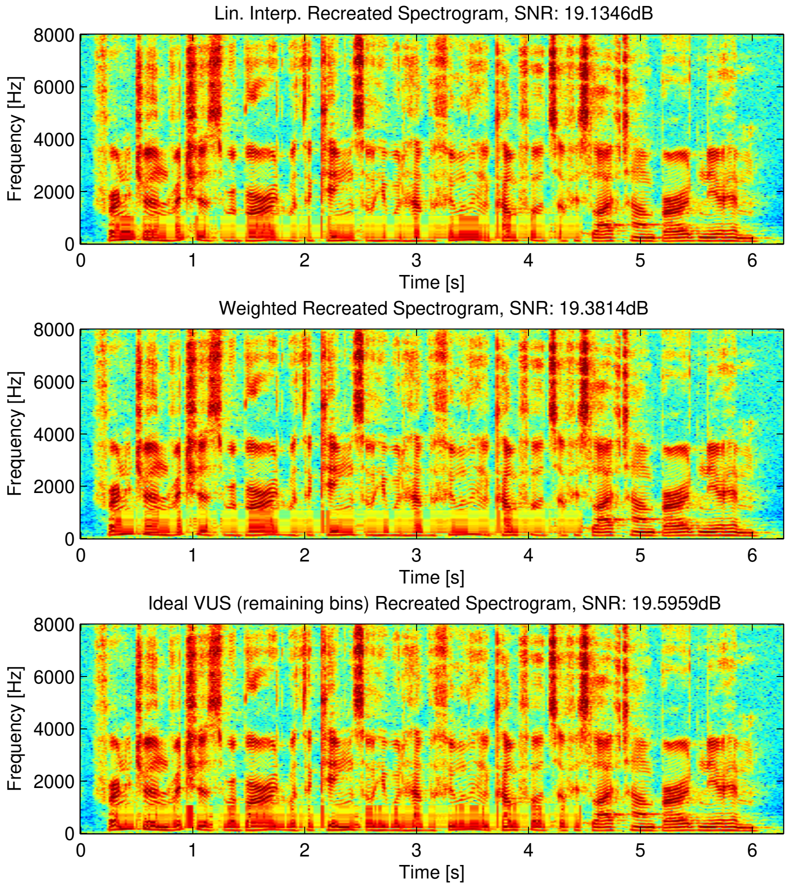


Figure 15.22: Reconstruction of the test signal with box masking using the alternative reconstruction methods including ideal VUS reconstruction. Training was performed on the full set of MKBP training recordings from the elsdsr database. This original and masked spectrogram for these reconstructions can be seen on Figure 15.21.

15.2.3 Summary of reconstruction results

The following tendencies are showed by Tables C.1 through C.9 in Appendix C:

- Spectrogram reconstructing without any mel filtering clearly achieves the best $SNR_{spectrogram}$. This is seen in every experiment conducted.
- Employing mel filtering for doing data clustering generally achieves the best $SNR_{spectrogram}$. This was the case for all the experiments, except for the box masked reconstructions using no filter.
- The ideal binary masked spectrogram was generally made poorer by the reconstruction. This was the case for all experiments except for the ideal case of overfitting with testdata included in training, and the weighted experiment in Table 15.2.
- Both the random masked signal and the box masked signal were generally improved upon by the filterless reconstruction. This was the case for all experiments.
- In more cases than not the MFCC achieves better $SNR_{spectrogram}$ than HFCC for reconstruction purposes. For training purposes they each achieve the better $SNR_{spectrogram}$ an equal amount of times.
- Speaker independant codebooks, have allowed for much larger training sets, but achieve poor $SNR_{spectrogram}$ in our ideal binary masked signal experiments, and in general for the male codebooks. For the female random and box masked signal experiments, the $SNR_{spectrogram}$ seems to increase with the amount of speakers which relates to the size of the training set.
- In general, the mixed gender codebooks achieve a higher $SNR_{spectrogram}$, the more speakers are added, but for the ideal masked signal experiments as well as the random masked signal experiments, even a large 6 speaker training set achieves poorer $SNR_{spectrogram}$ than the single speaker reconstructions. The box masked signals, however, get excellent $SNR_{spectrogram}$, but casual listening reveals that the speaker voice has been warped.

The following tendencies are shown by the tables in subsection 15.2.1:

- The remaining bins reconstruction improved on the masked spectrogram $SNR_{spectrogram}$ in the case of random and box masks, whereas it actually reduced the $SNR_{spectrogram}$ of the ideal binary masked spectrogram.

- The interpolated reconstruction did well in the ideal binary masked spectrogram example, as well as in the random masked spectrogram while it did rather poorly for the box masked spectrogram in comparison with the remaining bins method.

- The weighted reconstruction also did well in the ideal binary masked spectrogram example, while it did rather poorly for the box masked and random masked spectrograms in comparison with the remaining bins method.

- The voiced / unvoiced divided codebooks have poor $SNR_{spectrogram}$ with no knowledge of which codebook to use, however, with reliable knowledge of which codebook to use, it achieves very close to the same $SNR_{spectrogram}$ as the regular codebook does. A little bit better in the ideal binary mask and random masked cases, even.
  We also see that the VUS decision based on SFM calculated on interpolated frames in no way performs as well as the ideal case, and not even as well as the merged codebook approach.

Chapter 16

Discussion

16.1 Data clustering

We see on Figure 15.5 that for the maximum amount of training vectors with the MKBP training data (4367), a reasonable amount of clusters to use is $K = 250$, but to be on the safe side, we have chosen to include even more of the cluster quality improvement, which is why we mainly use $K = 500$ throughout the reconstruction experiments. The amount of clusters is data dependant, and we cannot say anything about the optimal amount of clusters for any other dataset, especially if they consist of more or less training vectors.
We see by the histograms on Figure 15.4, that we get some clusters with extremely few members, even for low numbers of K. This is not optimal, but while we use a relatively large amount of clusters to seperate the training vectors, it is bound to happen. Rather than reducing the amount of clusters, which would lead to poor ability for the resulting codebook to capture the details of a speech spectrogram, we would have liked to be able to increase the amount of training vectors, M_{train}. Unfortunately we do not have that large sets of training data for single speakers, and we have been unable to find any with good quality speech. So we will make do with the elsdsr database.
We notice that a single cluster is always very large compared to the others. This cluster always contains the low energy silence or near-silence frames, since these are all so similar.

16.1.1 Generic matlab *kmeans* function versus accelerated *k*-means

As can be seen on Figure 15.2 the accelerated k-means algorithm is, as expected, very fast compared to the generic MATLAB function. In fact, even for small amounts of clusters the generic takes many times the run-time of the accelerated algorithm. The more clusters you divide into, and the more training vectors you cluster, the more time is gained from using the accelerated algorithm.
With respect to quality we see on 15.4 that the generic k-means is better than the accelerated version for both random initialisation as well as when using furthest first initialisation, however multiple random initialisations work well to fix this issue. Of course, with random initialisations there is no guarantee you at any given time will end in a good local minima, but this is the case for both of the algorithms. We notice on Figure 15.4 that the two algorithms end up in different local minima even though they are initialised the same, which is not what we expected. Elkan has stated that the end result as well as any iteration step would be the same for the accelerated algorithm as for a standard k-means algorithm [6]. The reason they do not get the same end result is that the generic function does some further processing when it reaches a local minima.
Since the accelerated k-means algorithm is so much faster than the generic function, that it can conceivably be run with a large number of random initialisations and still reduce run-time, we feel that it is better suited for our project.

16.1.2 Mel filtering used in clustering

On Figure 15.2 and Figure 15.3 we see that mel filtering actually reduces the run time of the accelerated k-means algorithm even though there are additional calculations of multiplying the filter on the data, and computing the full filter cluster centroids as described in section 14.2.3. This is because of the reduced dimensionality of the distance calculations in the algorithms which is especially reducing the run-time of the generic function since this one does many more of those calculations.
On Figure 15.5 we see that mel filtering generally reduces the quality (higher MSCD), but we expected that to be the case. We implemented the mel filter in order to group vectors that sound alike in stead of just vectors that have the shortest distance to each other, and it is therefore natural that the quality measure, which is based on the distance between full vectors, shows a poor

result. The spectrogram reconstruction part of this report will help clarify if using mel filtering for data clustering in speech is a good idea despite the increased MSCD.

16.1.3 Voiced / unvoiced

On Figure 15.10 we see that the schmidt trigger on SFM performs voiced / unvoiced reasonably well in the shown example. For our use a flawless classification is not necessary, since we only seek to roughly divide the data to show if improvements can be gained in this fashion. We use the same voiced unvoiced decision for most of our VUS reconstruction experiments, and therefore the misclassified data used in the codebooks will likely end up being used for reconstructing misclassified data in the reconstruction part.
On Figure 15.14 we see that the overall run-time decreases significantly for the same complete amount of codebook clusters using the same training set. In fact, the more clusters wanted in the codebooks, the more time is saved by using divided data. The reduction in run-time does not only come from a reduction in number of clusters but also from the reduction in training vectors for each clustering. That this would yield a reduction in run-time can be verified on Figure 15.2.
On Figure 15.15 we see that the for a low amount of clusters a seperation of voiced and unvoiced data into two different codebooks would decrease the MSCD and therefore increase clustering quality. This is important since it shows us that even though there are misclassifications in the VUS decision, it does not have a significant negative effect on the clustering.

On Figure 15.11 we see that when a speech signal has been pre emphasis filtered, we do not get as good voiced / unvoiced classifications. One could argue that a new set of schmidt triggers would be able to account for the generally flatter spectra, but we also see that the calculated sfm has become less robust in that it spikes up and down a lot for both states, and so better for us to simply limit the voiced / unvoiced classification to signals that have not yet been pre emphasis filtered.

On Figure 15.12 we see that when a speech signal has been masked, we do not get useful voiced / unvoiced classifications. This is because of the sharp transitions between the remaining data and the data that has been removed by the mask. It might help to insert random white noise of a energy level suited for silent areas, though this would be a problem when high energy content has been removed. Also, it might help to do a linear interpolation between the remaining data points, though this might make the spectrum flatter when for instance the areas between harmonics have been removed. On Figure 15.13, however, we see that when a speech signal has been masked, and then linearly interpolated we get a much more useful voiced / unvoiced classification than for SFM on the masked spectrogram in itself. Based on this, we have tried to do VUS reconstructions based on an SFM calculated in this manner.

16.1.4 Speaker dependancy

On Figure 15.15 is is seen that the mixed codebook has a poor quality for any given K, however with high amounts of clusters the difference between the graphs is very small. The female and male codebooks are very close to being of the same quality. Due to the dependancy between the data set and the MSCD measure, it is not certain that the graphs would show the same tendencies for all data sets. However, the tendencies we do see in this example correspond well with what we would expect from the differences of the voices represented in the training data sets; that the more different the voices represented are, the larger the MSCD value for any given number of clusters would be, and the more clusters would be needed in order to obtain the same quality as for data sets of only speech from a single speaker.

16.2 Spectrogram reconstruction

16.2.1 Mask type issues

In the result tables in Section 15.2.1 and C.1 we see that the ideal binary masked signal was generally made poor by the reconstruction. This was the case for all experiments except for the weighted predecessor method and for the ideal case of overfitting with testdata included in training. We expected this to be the case, since in ideal binary masking we almost only remove data with low energy, and as such the masked spectrogram is actually a close approximation. Therefore any data inserted with high energy is likely to decrease the $SNR_{spectrogram}$ and the percieved quality of the speech.
We knew beforehand that it would be hard improving on a signal masked with this type of mask, since it is an ideal case, where much of the energy content of the original speech was untouched by the mask. We expect to get a better picture of the performance of the reconstruction when important speech data has been removed, as the case is with random and box masks, which are used to remove data regardless of energy level.

Both the random masked signal and the box masked signal were generally improved upon SNR-wise by the filterless reconstruction. This was the case for all experiments in Section 15.2.1 and Section C.2 and Section C.3, which goes to show that the vector quantization approach can restore some of the data removed from a spectrogram. However, the reconstruction failed to produce any speech signals with a PQE of more than 3, for non-overfitted experiments, and therefore failed to produce speech signals without artifacts percieved as annoying.

16.2.2 Mel filtering

In the result tables in Section C.1 and in Section C.2 we see that doing data clustering using either mel filter results in better values of $SNR_{spectrogram}$. This was not the case in the tables in Section C.3 where a box mask was used to remove much of the information of the harmonics in the speech. Also we note that for all the experiments in those 3 sections, mel filtering used in reconstruction had very poor results compared to reconstruction without mel filtering.

The idea behind mel filtering was basically that the mel feature vectors represent the important features in each vector, and therefore we would like to cluster vectors that have similar mel features. This should ensure that the members in a cluster all sound alike, which can account for the good results we have gotten from performing mel filtered clustering. But it seems that due to the reduced dimensionality of the mel filtered vectors we may simply end up blurring the harmonics when the centroid of each cluster is calculated. This would account for poor performance in Section C.3 since it is primarily the harmonics being reconstructed there, and is supported by the smeared look of the reconstructed parts in the figures of section 15.2.1.
Also, the reduced dimensionality might acount for the problems in filtered reconstruction, since the excact position of details like harmonics may be smeared.

In the results in Sections C.1, C.2, and C.3, in most cases, the MFCC achieves better SNR than HFCC for reconstruction purposes. For training purposes they each achieve the better SNR an equal amount of times. Based on the performance of each of the filters we cannot say that either one is better than the other, but rather that their performances are data dependant.
The idea that filter bandwidth should be based on knowledge of human perception, as is the case with HFCC [24], is appealing, but the practical implementation of the filter in this case is problematic due to the numeric

nature of MATLAB as seen in Figure 13.5 where the filters should have been of equal height.

16.2.3 Voiced / unvoiced

In Table 15.3 we see that voiced / unvoiced divided codebooks have poor $SNR_{spectrogram}$ with no knowledge of which codebook to use, however, with reliable knowledge of which codebook to use, it achieves very close to the same $SNR_{spectrogram}$ as the non voiced / unvoiced approach does. A little bit better in the ideal binary mask and random masked cases, even.

We also see that the VUS decision based on SFM calculated on interpolated frames is in no way as good as the ideal case, and not even as good as the merged codebook approach.

This tells us, that for the division of the codebook to work it is important to use a robust means of doing a VUS decision on masked data, and that if such a means could be found, it would indeed be beneficial to use voiced and unvoiced codebooks.

From the reconstruction examples in section 15.2.2, particularly on Figure 15.18 we see poor reconstruction of certain frames for the voiced / unvoiced codebook. In the original spectrogram, the frames were silence, but only low frequency bins survived the masking process. Because of this, unvoiced frames with high energy at the high frequency bins, were chosen as a templates. This is in fact a problem generated by the remaining bins method, and therefore it could be alleviated by using a different reconstruction method than "remaining bins", since the other methods are more robust against this type of error, or it could be alleviated by even stricter rules for the selected templates for frames with few surviving bins.

16.2.4 Speaker dependancy

In Section C.4 we see that speaker independant codebooks, have allowed for much larger training sets, but perform poorly in our ideal binary masked signal experiments, and in general for the male codebooks.

In the female random and box masked signal experiments, despite the added complexity of the codebook with more speakers, the performance increases as the amount of training data grows.

This goes to show that the more data you have, the more likely it is that you will find a good match for the vector you are trying to reconstruct. Of course, though, since we are using a constant number of cluster centroids, we need the data to have some common characteristics, so that we do not end up having to group vectors with vastly different features. This seem to be the case for the female voices used, but we would expect this to be very dependant on data set.

In general the mixed gender codebooks perform better, the more speakers are added, but for the ideal masked signal experiments as well as the random masked signal experiments, even a large 6 speaker training set performs poorer than the single speaker reconstructions. This goes to show, that when the data set is not homogenous we end up grouping vectors with vastly different features. This could be fixed by increasing the amount of clusters however, keeping in mind, though, that this will increase run-time as seen on Figure 15.2 and Figure 15.3.
The box masked signals for mixed gender codebooks, however, get excellent SNR, again showing that the more data you have, the more likely it is that you will find a good match for the vector you are trying to reconstruct. But casual listening tests reveals that the speaker voice has been warped.

16.2.5 Linear Interpolation reconstruction

In Table 15.2 we see that the Linear Interpolation method did well in the ideal binary masked spectrogram example, as well as for the random mask, while it did rather poorly for the box masked spectrogram.
The fact that it did poorly for the box mask is to be expected since the straight line initial guess made by linear interpolation across the height of the box is a poor estimate of the harmonics which were removed.
The performance gained in the ideal binary masked experiment compared to the remaining bins method is likely due to the fact that frames with few surviving bins from the masking in most cases will only be filled with low energy bins due to the linear interpolation across the span of bins. And in that way, we avoid the problems with unvoiced high energy frames selected to template what in fact was supposed to be silence.
This method performs well with the random masked experiments which is likely because there are rarely a large number of consecutive bins which did not survive the masking process, and therefore the linear interpolation in many cases come close to the original enough to be an asset in the template selection.

16.2.6 Weighted reconstruction

In Table 15.2 we see that the weighted reconstruction did well in the ideal binary masked spectrogram example, while it did rather poorly for the box masked and random masked spectrograms.

We would expect the method to do well in reconstruction of voiced data since the harmonics are often similar in neighbouring spectra, and also we would expect that it reduces the tendency of inserting unvoiced data, with high energy in for high frequencies, in a silent area. It seems to do the latter, but the former seems not to be the case; for instance when looking at Figure 15.22.

It is evident that the approach performs poorly when the data from the same frequencies are missing from a number of consecutive spectra in the spectrogram where changes occur since any errors are carried over in the weighted contribution from previous spectra. this explains why the box masked experiment did poorly, and to a degree also why the random masked experiment did poorly, since this particular mask has had a great deal of data removed. A crude solution might be to simply only carry over weighted contributions for original data and not data that was reconstructed in the previous spectrum.

The reason for the improved performance of reconstructing the ideal binary masked spectrogram is very likely just caused by the fact that the silent areas no longer get higher energy unvoiced data inserted.

One of the reasons that this method does not do well in reconstructing the voiced harmonics is probably found in the making of the codebook through clustering. In section 16.2.2 we noted that the reduced dimensionality of mel feature vector based clustering had a smearing effect on the codebook data, an effect that does not seem to be entirely avoided in the regular full vector based clusterings as seen in the examples of section 15.2.2. To reduce this problem we need to:

- Either, increase the number of clusters in the data clustering, and consequently also the amount of training data vectors, to get clusters with similar harmonics only.

- Or in stead of using the cluster centroid, which in essence is a mean of the members in an entire cluster, one could usie the actual unaltered cluster member closest to the cluster mean, as a representative of the cluster. Unfortunately this has not been attempted due to the time constraints of this project.

- Or, as in [19] not try to get seperate codebook states for every possible spectral configuration through data clustering, but only a limited set of "sharp" states that can still cover the full spectral variety of a source via the contribution of the previous frame.

16.3 Summary of Main Conclusions

The main conclusions of part III (Vector Quantization) were:

- The clustering process using spectrogram frames as training data vectors, will usually end up in local minima, but this can be alleviated by performing the clustering process multiple times, each time with a different random initialisation.

- The accelerated k-means algorithm as used in our experiments has no protection against ending in local minima, and is therefore more affected by the above, than the generic MATLAB *kmeans* function, which does have some protection.

- The accelerated k-means algorithm has a significantly lower run time than the generic MATLAB *kmeans* function. This has more significance, the larger the values of K and M_{train} used.

- The silent frames in any training set are so alike that a clustering will group these together in a single large cluster

- Mel filtering in the clustering process reduces run time, and in most cases it also improves the resulting $SNR_{spectrogram}$ of the reconstructions. While used in the reconstruction itself, mel filtering significantly reduces the $SNR_{spectrogram}$.

- VUS divided codebooks reduces the run time significantly, but unless a robust means of performing VUS decisions on masked spectrograms, is developed, this approach does not perform well during reconstruction.

- For the investigated sizes of K, the codebooks constructed are quite gender specific, but evidence suggests they can be speaker independant for voices with similar characteristics.

- The remaining bins reconstruction method can be used to improve on masked spectrograms, but does not always select good templates for frames that have few remaining bins after the masking process.

- The linear interpolation reconstruction method can also be used to improve on masked spectrograms, but does not always select good templates for frames that have had many consecutive bins of a frame removed in the masking process.

- The weighted predecessor reconstruction method can also be used to improve on masked spectrograms, but performs poorly when the masking process has removed the same bin in several consecutive frames.

- Our clustering approach of using cluster centroids in the codebook produces “smeared” harmonics (along with other details), and therefore it is speculated that larger amounts of clusters along with larger training sets would be beneficial to our approach, and it is furthermore speculated that a different cluster representative approach might be also be beneficial.

Part IV

Conclusion and Future Work

Chapter 17

Summary of Main Contributions

The main objective was to:
Investigate means of reconstructing a speech signal from binary masked spectrogram, with high percieved quality and naturalness as the goal.

In this thesis, several components for reconstructing a speech signal has been investigated, that comply to the objective. The components belong to the two main parts of the objective - Signal estimation used to synthesise speech signals from spectrograms and vector quantization used to reconstruct missing data in spectrograms:

17.1 Signal Estimation

This part of the objective was realised through an in depth analysis of existing algorithms for signal estimation and the parameters they depend on. The algorithms are those proposed by Griffin & Lim [10], Kannan et al. [12], and Bouvrie et al. [2].
For each of the algorithms, the following investigations were conducted:

- Performance as function of initial phase estimate.
- Performance on unmodified spectrograms.
- Performance as function of different spectrogram parameters.

17.2 Vector Quantization

For a vector quantization approach to reconstructing a binary masked spectrogram, the k-means algorithm has been implemented, key components in data clustering have been implemented and investigated, as well as different methods for reconstruction and voiced / unvoiced handling.

Data clustering

- An accelerated k-means algorithm was used to speed up the clustering process.
- Pre emphasis filtering was employed to balance the energy dependant focus of the data analysis.
- A well proven MFCC mel filterbank as well as a rather new HFCC filterbank were implemented and used.
- A simple and fast method for voiced / unvoiced detection was employed for codebook size reduction ultimately in order to reduce run time.

Spectrogram reconstruction

- Three different types of simulated binary masks were used to consider the lack of data that may occur in the process of bnary masking.
- Three different methods for template selection during reconstruction were considered and implemented.
- The already mentioned mel filterbanks were used for reconstruction purposes and optimal filter combinations were investigated.
- The voiced / unvoiced detection method was used in conjunction with linear interpolation to provide a possible means of determining voiced and unvoiced areas in a binary masked spectrogram.

Chapter 18

Suggestions for Future Work

- At the end of our project there were still possibilities for improvement that we unfortunately did not have time to implement and test. The most obvious for the signal estimation part was the ability to use STFT analysis windows other than boxcar windows in the Bouvrie algorithm. This is needed to reconstruct sound from a good quality spectrogram for the vector quantization.

- Work still need to be done on the Kannan algorithm to identify what causes the phase discontinuities at the borders of the overlapping segments. The Kannan algorithm could potentially be very powerful if better reconstruction was achived since the prior could maybe be used indirectly to fix some of the erros that may occur in the regeneration of the spectrogram using the vector quantization.

- A representative member approach to constructing the clustering codebooks would be easy to implement, and might offer a way to reduce the “smeared” effect of the data inserted during reconstruction.

- For the vector quantization, we belive that a Hidden Markov Model may help to improve the reconstruction of the spectrogram. A statistical approach such as this, would help to avoid placing obviously wrong frames at the wrong place, ex. placing an s sound or silence in the middle of a voiced frame, even though the distance in the k-means codebook was the shortest.

- A less simplified version of the deformable spectrograms approach [19] could be used in conjunction with vector quantization. We feel it is likely such an approach could:

1. Combine the performance in reconstruction of voiced speech from the deformable spectrograms approach with the overall recognition of frames using a VQ template approach.
2. Work around the drawback of poor coherence between neighbouring frames using VQ, as well as the drawback of not being able to use deformable spectrograms on frames bordering between voiced and unvoiced speech.

Chapter 19

Conclusion

The reconstructions we obtained using our vector quantization approach leaves much to be desired. Even though we showed that it was possible to improve on the masked spectrograms through vector quantization, we did not manage to produce speech which was pleasant to listen to. Due to the timeframe of this project, we have only been able to investigate some of the possible extensions for this approach, and it is therefore likely that even better reconstructions can be obtained in future versions.

The quality of the reconstructed signal after spectrogram regeneration and the following signal estimation, is a result of the performance of these two independent steps in the speech signal regeneration process. Since both the methods used had sources of error, the reconstructed signal never became better than the weakest link in the chain. In our case, it was especially difficult to obtain a good spectrogram reconstruction using simple k-means data clustering.

In our examination of the algorithms for signal reconstruction, we showed that the algorithm by Griffin & Lim [10] generally achieves good results in a very short time. The algorithm by Bouvrie et al. [2] was however able to achieve perfect reconstruction for a larger overlap, but was also much slower. Certainly one should also take into account the reconstruction of the spectrogram in the choice of the signal estimation algorithm. Perfect signal estimation is no good, if the spectrogram is poorly reconstructed. During the project, the processing power of our PC's were time and again pushed to the limit. This is a good indication of the computational demands of these advanced algorithms, and it shows that there is still a long way to go before a practical solution for hearing aid applications exists.

A big problem in the area of speech quality evaluation is the lack of a robust objective quality measure. The signal to noise ratio measures used in this report are often used in papers to compare the performance of different algorithms. Throughout our report, and especially for the signal estimation part, we note that these measures can often be misleading in terms of the perceptual sound quality. The problem is well known, and our results confirm the need for a objective perceptual quality measure.

The entire project was an immense learning experience for us in the field of speech separation and reconstruction, and getting to know a few pieces of the puzzle only left us craving for more. The idea, of making computers able to completely seperate the speech of a single speaker from background noise or competing speech, is now over 50 years old. The use of modern computers as well as the extensive research in the past years have pushed the limits what can be achieved, and each year we come closer to the goal.

The benefits of this research area may even go beyond help for hearing impaired people. As research progresses, it may one day not be a question of trying to match the capabilities of the brain, but to improve and extend the senses we are born with.

Appendix A

The Probabilistic Signal Estimation Model In Detail

A.1 The Model

In the following we will explain in detail the calculations of the algorithm suggested by [12] for signal estimation from the STFT magnitude. We will also discuss the optimizations needed to use this very computational intensive algorithm in Matlab (see subsection A.2).

The idea behind the algorithm is to maximize the joint probability:

$$P(\mathbf{x}, \mathbf{Y}) = P(\mathbf{Y}|\mathbf{x})P(\mathbf{x}) \tag{A-1}$$

where $\mathbf{x}$ is a signal, and $\mathbf{Y}$ is a spectrogram. $P(\mathbf{x}, \mathbf{Y})$ is the joint probability of the signal $\mathbf{x}$ and the spectrogram $\mathbf{Y}$. $P(\mathbf{x})$ is known as the prior, and functions as a weight that changes according to some prior known information about the signal. If no prior information about the signal is known, then all signals er equally probable, and the prior can be set to 1. $P(\mathbf{Y}|\mathbf{x})$ is known as the likelihood and describes the probability of observing the spectrogram $\mathbf{Y}$ given the signal $\mathbf{x}$. The likelihood function and the prior are given by [12] as:

$$P(\mathbf{x}) \propto \prod_{n=1}^{N} \exp\left\{-\frac{1}{2\rho^2}\left(\sum_{r=1}^{R} a_r x_{n-r} - x_n\right)^2\right\} \tag{A-2}$$

Where R is the model order, a_r are the AR model coefficients, N the total number of samples and x_n the sample at sample number n. ρ^2 is the noise in the prior.

$$P(\mathbf{Y}|\mathbf{x}) \propto \prod_k \exp\left\{-\frac{1}{2\sigma^2}\||\hat{Y}_k| - |Y_k|\|^2\right\} \tag{A-3}$$

Where $|\hat{Y}_k|$ is the magnitude spectrum of the estimated signal $\mathbf{x}$ at frame k. and $|Y_k|$ is the magnitude spectrum of the modified spectrogram at segment k. δ^2 is the noise in the observed spectra. Note that the noise variances need not be known in order to optimize on the joint probability since they are only constants being multiplied on the liklihood and prior, and can therefore be omitted in an optimization algorithm. The joint probability can now be written:

$$P(\mathbf{x}, \mathbf{Y}) = \prod_k \exp\left\{-\frac{1}{2\sigma^2}\||\hat{Y}_k| - |Y_k|\|^2\right\} \prod_{n=1}^{N} \exp\left\{-\frac{1}{2\rho^2}\left(\sum_{r=1}^{R} a_r x_{n-r} - x_n\right)^2\right\} \tag{A-4}$$

Since we need $|\hat{Y}_k|$ to be expressed as a function of the individual samples in frame k, x_k, [12] descibes $|\hat{Y}_k|$ as:

$$|\hat{Y}_k| = \mathbf{F}\mathbf{x}_k \circ \mathbf{F}^*\mathbf{x}_k \tag{A-5}$$

$\mathbf{F}$ is the Fourier matrix which is simply the matrix version of a Fourier transform. $\mathbf{F} \cdot \mathbf{x}$ is equal to $X(k) = \sum_n^{N-1} x(n)e^{-i2\pi kn/N}$. $*$ denotes complex conjugation and $\circ$ denotes element wise product. Since the product of a complex number c and its conjugate c^* gives $|c|^2$ what we get is, the squared magnitude of the Fourier transform. Kannan et al. apparently use the squared magnitude for their definition of the spectrogram, where we only use the magnitude. This means that if we want the same result, we should change our $|\hat{Y}_k|$ with our definition to $|\hat{Y}_k|^2$. Equation (A-4) can then be rewritten as:

$$P(\mathbf{x}, \mathbf{Y}) = \prod_k \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{F}\mathbf{x}_k \circ \mathbf{F}^*\mathbf{x}_k - |Y_k|^2\|^2\right\} \prod_{n=1}^{N} \exp\left\{-\frac{1}{2\rho^2}\left(\sum_{r=1}^{R} a_r x_{n-r} - x_n\right)^2\right\} \tag{A-6}$$

To use an optimization algorithm to maximize P($\mathbf{x}$,$\mathbf{Y}$) we need to have the likelihood and prior as aditive functions instead of products. This is obtained by taking the logarithm, which gives us:

$$\begin{aligned}\log P(\mathbf{x},\mathbf{Y}) = &-\sum_k \left\{ \frac{1}{2\sigma^2} \|\mathbf{F}\mathbf{s}_k \circ \mathbf{F}^* x_k - |Y_k|^2\|^2 \right\} \\ &-\sum_{n=1}^{N} \left\{ \frac{1}{2\rho^2} \left(\sum_{r=1}^{R} a_r x_{n-r} - x_n \right)^2 \right\}\end{aligned} \tag{A-7}$$

Since most optimization algorithms use minimization and not maximization, the negative logarithm is often used:

$$\begin{aligned}-\log P(\mathbf{x},\mathbf{Y}) = &\frac{1}{2\sigma^2} \sum_k \left\{ \|\mathbf{F}\mathbf{x}_k \circ \mathbf{F}^* \mathbf{x}_k - |Y_k|^2\|^2 \right\} \\ &+\frac{1}{2\rho^2} \sum_{n=1}^{N} \left\{ \left(\sum_{r=1}^{R} a_r x_{n-r} - x_n \right)^2 \right\}\end{aligned} \tag{A-8}$$

Expanding the norm of (A-8) we get

$$\begin{aligned}-\log P(\mathbf{x},\mathbf{Y}) \propto &\frac{1}{2\sigma^2} \sum_k \sum_i \left(\sum_{j=1}^{\Omega} \sum_{j=1}^{\Omega} F_{ij} x_{k(\Omega/2)+j} F^*_{il} x_{k(\Omega/2)+l} - |Y_{ki}|^2 \right)^2 \\ &+\frac{1}{2\rho^2} \sum_{n=1}^{N} \left\{ \left(\sum_{r=1}^{R} a_r x_{n-r} - x_n \right)^2 \right\}\end{aligned} \tag{A-9}$$

Where j and i are the indexes of the Fourier matrix, and Ω then length of the Fourier transform. Note that the indexes on x are different from kannan's notation since their index does not start at x_0 but on $x_{\Omega/2}$ or $x_{-\Omega/2}$ depending on their starting value of k. We also need to expand equation (A-9) to include a window function w that we used to make $|Y_k|^2$ (see section 5). This changes equation (A-9) to:

$$\begin{aligned}-\log P(\mathbf{x},\mathbf{Y}) \propto &\frac{1}{2\sigma^2} \sum_k \sum_i \left(\sum_{j=1}^{\Omega} \sum_{j=1}^{\Omega} F_{ij} x_{k(\Omega/2)+j} w(j) F^*_{il} x_{k(\Omega/2)+l} w(l) - |Y_{ki}|^2 \right)^2 \\ &+\frac{1}{2\rho^2} \sum_{n=1}^{N} \left\{ \left(\sum_{r=1}^{R} a_r x_{n-r} - x_n \right)^2 \right\}\end{aligned} \tag{A-10}$$

In order to use (A-10) in a conjugate gradient optimizer, we need the derivatives with respect to each sample x_n. Since the likelihood and prior are now additive, we can take the derivative of each of them by themselves and add them in the end. Starting with the likelihood, we find the derivatives with respect to sample x_u:

$$\begin{aligned}\frac{\partial - logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto \frac{1}{2\sigma^2}\sum_k\sum_i 2&\left(\cdot\sum_{j=1}^{\Omega}\sum_{l=1}^{\Omega} F_{ij}x_{k(\Omega/2)+j}w(j)F^*_{il}x_{k(\Omega/2)+l}w(l) - |Y_{ki}|^2\right)\\ &\cdot\sum_{j'=1}^{\Omega}\sum_{l'=1}^{\Omega} F_{ij'}w(j')\frac{\partial x_{k(\Omega/2)+j'}}{\partial x_u}F^*_{il'}x_{k(\Omega/2)+l'}w(l')\\ &+F^*_{il'}w(l')\frac{\partial x_{k(\Omega/2)+l'}}{\partial s_u}F_{ij'}x_{k(\Omega/2)+j'}w(j')\end{aligned} \tag{A-11}$$

$\frac{\partial x_{k(\Omega/2)+j'}}{\partial x_u}$ Corresponds to a delta function at index $u = k(\Omega/2) + j' \Rightarrow j' = u - k(\Omega/2)$. We then rewrite eq.(A-11) to:

$$\begin{aligned}\frac{\partial - logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto \frac{1}{2\sigma^2}\sum_k\sum_i 2&\left(\cdot\sum_{j=1}^{\Omega}\sum_{l=1}^{\Omega} F_{ij}x_{k(\Omega/2)+j}w(j)F^*_{il}x_{k(\Omega/2)+l}w(l) - |Y_{ki}|^2\right)\\ &\cdot\sum_{j'=1}^{\Omega}\sum_{l'=1}^{\Omega} F_{ij'}w(j')\delta_{j'=u-k(\Omega/2)}F^*_{il'}x_{k(\Omega/2)+l'}w(l')\\ &+F^*_{il'}w(l')\delta_{l'=u-k(\Omega/2)}F_{ij'}x_{k(\Omega/2)+j'}w(j')\end{aligned} \tag{A-12}$$

In the same way, we continue to obtain the partial derivatives with respect to sample x_u of the prior:

$$\frac{\partial - logP(\mathbf{x})}{\partial x_u} \propto \frac{1}{2\rho^2}\sum_n 2\left(\sum_{r=1}^{R} a_r x_{n-r} - x_n\right)\left(\sum_{r'=1}^{R} a_{r'}\frac{\partial x_{n-r}}{\partial x_u} - \frac{\partial x_n}{\partial x_u}\right) \tag{A-13}$$

As before, $\frac{\partial x_{n-r}}{\partial x_u}$ can be described as a kronecker delta function. We insert these delta functions to reveal:

$$\frac{\partial - logP(\mathbf{x})}{\partial x_u} \propto \frac{1}{2\rho^2}\sum_n 2\left(\sum_{r=1}^{R} a_r x_{n-r} - x_n\right)\left(\sum_{r'=1}^{R} a_{r'}\delta_{r'=n-u} - \delta_{r'=u}\right) \tag{A-14}$$

We are then finally able to write the partial derivatives for P($\mathbf{x}$,$\mathbf{Y}$) as:

$$\begin{aligned}\frac{\partial - logP(\mathbf{x},\mathbf{Y})}{\partial x_u} \propto \frac{1}{2\sigma^2}\sum_k\sum_i 2 &\left(\cdot\sum_{j=1}^{\Omega}\sum_{l=1}^{\Omega} F_{ij}x_{k(\Omega/2)+j}w(j)F^*_{il}x_{k(\Omega/2)+l}|Y_{ki}|^2\right)\\ &\cdot\sum_{j'=1}^{\Omega}\sum_{l'=1}^{\Omega} F_{ij'}w(j')\delta_{j'=u-k(\Omega/2)}F^*_{il'}x_{k(\Omega/2)+l'}w(l')\\ &+F^*_{il'}w(l')\delta_{l'=u-k(\Omega/2)}F_{ij'}x_{k(\Omega/2)+j'}w(j')\\ +\frac{1}{2\rho^2}\sum_n 2&\left(\sum_{r=1}^{R}a_r x_{n-r}-x_n\right)\left(\sum_{r'=1}^{R}a_{r'}\delta_{r'=n-u}-\delta_{r'=u}\right)\end{aligned} \tag{A-15}$$

Theses partial derivatives must be calculated for each iteration of a optimization algorithm like the one we used by Carl Rasmussen [30]. As we can see from equation (A-15), this is a very computationally expensive algorithm. Signals of a few seconds can easily mass up a total number of samples of 40000 or more. Which means that 40000 partial derivative must be done at each iteration. We will continue in the next section to describe how we can be most effecient when doing some of the calulations.

A.2 Optimization tips for Matlab

In order to use this computationally expensive algorithm in Matlab, we need to optimize it as much as possible using matrix and vector evaluations and matlab functions. Just entering the sums in equation (A-15) as for-loops could easily make the program run until the end of the next millinium for even a fairly small signal. First we need to translate equation (A-15) into using vectors and matrixes instead of sums. The likelihood can be written as:

$$\begin{aligned}\frac{\partial logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto -\frac{1}{2\sigma^2}\cdot 2\sum_k\sum_i &\left(\mathbf{F}\mathbf{x}_k\mathbf{w}\circ F^*\mathbf{x}_k w - |Y_k|^2\right)\\ &\cdot\left(\mathbf{F}\mathbf{w}\delta_u\circ\mathbf{F}^*\mathbf{x}_k\mathbf{w}+\mathbf{F}^*\mathbf{w}\delta_u\circ\mathbf{F}\mathbf{x}_k\mathbf{w}\right)\end{aligned} \tag{A-16}$$

here $\mathbf{x}_k$ is a vector containing all samples that are included in frame/segment k of the STFT. To ensure we had the correct samples at any amount of overlap, we created a matrix that contained the index of the used samples in each frame when we did the STFT. The delta function is now a vector with the same length as the length of the fourier transform (or fourier matrix). The vector is pure zeros except for a 1 at the position where sample x_u is located in frame k. Equation (A-16) can however be optimized if we only sum over those frames k_u that contains the sample x_u. Further we can also remove the calculations that result in a zero due to the delta functions. Since the delta function multiplied with the window w corresponds to the value of the window at w_i and the delta function multiplied on the Fourier matrix gives a sigle column of the matrix, we can write:

$$\begin{aligned}\frac{\partial - logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto & \frac{1}{2\sigma^2} \cdot 2 \sum_{k_u} \sum_{i} \left(\mathbf{F}\mathbf{x}_k\mathbf{w} \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} - |Y_k|^2 \right) \\ & \cdot \left(\mathbf{f}_u w_i \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} + \mathbf{f}_u^* w_i \circ \mathbf{F}\mathbf{x}_k\mathbf{w} \right)\end{aligned} \tag{A-17}$$

Equation (A-17) can be reduced a little since the last part of the equation can be reduced to:

$$\begin{aligned}\frac{\partial - logP(\mathbf{Y}|\mathbf{x})}{\partial x_u} \propto & \frac{1}{2\sigma^2} \cdot 2 \sum_{k_u} \sum_{i} \left(\mathbf{F}\mathbf{x}_k\mathbf{w} \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} - |Y_k|^2 \right) \\ & \left(2 \cdot \Re \left\{ \mathbf{f}_u w_i \circ \mathbf{F}^*\mathbf{x}_k\mathbf{w} \right\} \right)\end{aligned} \tag{A-18}$$

Where $\Re$ is the real part of the result in the parenthesis. Instead of using the Fourier matrix we can use the matlab function fft() to do our fourier transforms. Simply replace $\mathbf{F}\mathbf{x}_k$ with fft($\mathbf{x}_k$). ifft() can be used instead of $\mathbf{F}^*$, so $\mathbf{F}^*\mathbf{x}_k\mathbf{w}$ is equal to ifft$(\mathbf{x}_k\mathbf{w}) \cdot \Omega$. Notice that we need to multiply with the length of the Fourier transform, Ω, in order to use ifft(), since ifft() is defined as $\frac{1}{\Omega}\sum x(n) \cdot e^{i2\pi nk/\Omega}$. $\mathbf{f}_u$ can be calculated simply by using fft() on the delta function vector described earlier.

If we take a look at the prior and write it as:

$$\frac{\partial - logP(\mathbf{x})}{\partial x_u} \propto \frac{1}{2\rho^2} \sum_{n} 2 \left(\sum \mathbf{a} \cdot \mathbf{x_{n-r}} - \mathrm{x_n} \right) \left(\sum_{r'=1}^{R} a_{r'=n-u} - \delta_{r'=u} \right) \tag{A-19}$$

where $\mathbf{a}$ is a vector containing the AR coefficients of the AR model and $\mathbf{x_{t-r}}$ is a vector with the elements $[x_{n-1}, ..., x_{n-R}]$. We can now finally write the joint probability as:

$$
\begin{aligned}
\frac{\partial - logP(\mathbf{x}, \mathbf{Y})}{\partial x_u} \propto & \frac{1}{2\sigma^2} \cdot 2 \sum_{k_u} \sum_{i} \left(\mathbf{F} \mathbf{x}_k \mathbf{w} \ \circ \mathbf{F}^* x_k \mathbf{w} - |Y_k|^2 \right) \\
& \cdot \left(2 \cdot \Re \left\{ \mathbf{f}_u w_i \circ \mathbf{F}^* \mathbf{x}_k \mathbf{w} \right\} \right) \\
& + \frac{1}{2\rho^2} \sum_{n} 2 \left(\sum \mathbf{a} \cdot x_{n-r} - x_n \right) \left(\sum_{r'=1}^{R} a_{r'=n-u} - \delta_{r'=u} \right)
\end{aligned}
\tag{A-20}
$$

For-loops can not be avoided all togehter, mostly due to extreme memory requirements if we were to do 3D matrix calculations on large signals, but by being careful on what to calculate (not doing calculations that will end up being zero and using the same for-loop to do different calculations that are independent of each other) we were able to reduce the number of calculations significantly. The major time consumer is the calculation of the gradients at each and every iteration of the optimization algorithm, which can take a long time even when handling signals of a few seconds. Please look in APPENDIX B, kanrow.m for our implementation of the calculations of the likelihood, prior and gradients.

Appendix B

HFCC Filter Parameters

B.1 HFCC filter centers

For a frequency range between $f_{\min}$ and $f_{\max}$, the center frequencies of the first and last filters are determined as follows.
Define f_{l_i}, f_{c_i} and f_{h_i}, as the low, center and high frequencies for the ith filter in linear frequency. The triangular filters are equilateral in mel frequency. That is,

$$\hat{f}_{c_i} = \frac{1}{2}(\hat{f}_{h_i} + \hat{f}_{l_i}) \tag{B-1}$$

Unwarping to linear frequency using Equation 13.6 yields

$$\begin{aligned}
\log[1 + \frac{f_{c_i}}{700}] &= \frac{1}{2}(\log[1 + \frac{f_{h_i}}{700}] + \log[1 + \frac{f_{l_i}}{700}]) \\
\log[1 + \frac{f_{c_i}}{700}] &= \frac{1}{2}\log[(1 + \frac{f_{h_i}}{700})(1 + \frac{f_{l_i}}{700})] \\
(700 + f_{c_i})^2 &= (700 + f_{h_i})(700 + f_{l_i}) \qquad \text{(B-2)}
\end{aligned}$$

which, solved for f_{h_1} and f_{l_N}, in terms of f_{c_1} and f_{c_N}, given $f_{l_1} = f_{\min}$ and $f_{h_N} = f_{\max}$, yields

$$f_{h_1} = \frac{700 + f_{c_1}^2}{700 + f_{\min}} - 700 \tag{B-3}$$

$$f_{l_N} = \frac{700 + f_{c_N}^2}{700 + f_{\max}} - 700 \tag{B-4}$$

Furthermore from Equation 13.8 for a triangular filter with $|H(f_c)| = 1$,

$$
\begin{aligned}
ERB &= \int_{f_l}^{f_c} \frac{f - f_l}{f_c - f_l} \mathrm{d}f + \int_{f_c}^{f_h} \frac{f - f_h}{f_c - f_h} \mathrm{d}f \\
ERB &= \frac{1}{2}(f_h - f_l) \qquad \text{(B-5)}
\end{aligned}
$$

so from Equation B-5 and Equation 13.9,

$$
af_{c_i}^2 + bf_{c_i} + c = \frac{1}{2}(f_{h_i} - f_{l_i}) \qquad \text{(B-6)}
$$

for f_{c_i} in Hz, given the values in Equation 13.10.

For $f_{\min}$ or $f_{\max}$ consider values $\hat{a}$, $\hat{b}$ and $\hat{c}$, so that

$$
af_{c_i}^2 + bf_{c_i} + c = \hat{a}f_{c_i}^2 + \hat{b}f_{c_i} + \hat{c} \qquad \text{(B-7)}
$$

then

$$
f_{c_i}^2 + \bar{b}f_{c_i} + \bar{c} = 0 \qquad \text{(B-8)}
$$

where

$$
\begin{aligned}
\bar{b} &= \frac{b - \hat{b}}{a - \hat{a}} \\
\bar{c} &= \frac{c - \hat{c}}{a - \hat{a}}
\end{aligned}
$$

so f_{c_1} and f_{c_N} can be found by solving

$$
f_{c_i} = \frac{1}{2}(-\bar{b} \pm \sqrt{\bar{b}^2 - 4\bar{c}}) \qquad \text{(B-9)}
$$

where only a + sign leads to meaningful center frequencies.

The values for $\hat{a}$, $\hat{b}$ and $\hat{c}$ are calculated for the first and last filter respectively, by inserting each of Equations B-3 and B-4 into B-6

For the first filter, where $f_{l_1} = f_{\min}$,

$$
\begin{aligned}
\hat{a} &= \frac{1}{2}\frac{1}{700 + f_{\min}} \\
\hat{b} &= \frac{700}{700 + f_{\min}} \\
\hat{c} &= -\frac{f_{\min}}{2}(1 + \frac{700}{700 + f_{\min}}) \qquad \text{(B-10)}
\end{aligned}
$$

For the last filter, where $f_{h_N} = f_{\max}$,

$$
\begin{aligned}
\hat{a} &= -\frac{1}{2}\frac{1}{700 + f_{\max}} \\
\hat{b} &= -\frac{700}{700 + f_{\max}} \\
\hat{c} &= \frac{f_{\max}}{2}(1 + \frac{700}{700 + f_{\max}})
\end{aligned}
\tag{B-11}
$$

B.2 HFCC bandwidth

When the ERB_i and the center frequencies f_{c_i} have been determined, the upper and lower filter frequencies are determined as follows. From Equation B-5 the upper frequencies are isolated,

$$f_{h_i} = f_{l_i} + 2\mathrm{ERB}_i \tag{B-12}$$

which can be put into Equation B-2 to yield

$$(700 + f_{c_i})^2 = (700 + f_{l_i} + 2\mathrm{ERB}_i)(700 + f_{l_i}) \tag{B-13}$$

then, since we have already determined ERB_i and f_{c_i}, we can solve the quadratic function with respect to the lower frequencies

$$f_{l_i} = -(700 + \mathrm{ERB}_i) \pm \sqrt{(700 + \mathrm{ERB}_i)^2 + f_{c_i}(f_{c_i} + 1400)} \tag{B-14}$$

where only a + sign leads to meaningful results. Equation B-12 is used to find f_{h_i}

Appendix C

Reconstruction Results

C.1 Ideal binary masked signal reconstruction

Table C.1: **Reconstruction 1** - Overfitted reconstruction, where $K = M$.

reconstruction filter	*mfcc*	*hfcc*	*none*
Testdata included in training			
$SNR_{spectrogram}$	6.00 dB	5.38 dB	23.44 dB
PQE	2	2	4
Testdata excluded from training			
$SNR_{spectrogram}$	5.27 dB	5.42 dB	12.70 dB
PQE	2	2	3
Voiced / unvoiced			
$SNR_{spectrogram}$	6.42 dB	6.24 dB	15.12 dB
PQE	2	2	3

Table C.2: **Reconstruction 2** - Reconstruction, where $K = 500$.

reconstruction filter	*mfcc*	*hfcc*	*none*
No mel filter used in training			
$SNR_{spectrogram}$	-3.87 dB	-3.91 dB	9.57 dB
PQE	1	1	3
HFCC filter used in training			
$SNR_{spectrogram}$	3.21 dB	2.23 dB	11.92 dB
PQE	1	1	3
MFCC filter used in training			
$SNR_{spectrogram}$	3.55 dB	3.52 dB	11.85 dB
PQE	1	1	3

C.2 Random masked signal reconstruction

Table C.3: **Reconstruction 5** - Overfitted reconstruction, where $K = M$.

reconstruction filter	*mfcc*	*hfcc*	*none*
Testdata included in training			
$SNR_{spectrogram}$	1.19 dB	0.46 dB	26.83 dB
PQE	1	1	4
Testdata excluded from training			
$SNR_{spectrogram}$	0.87 dB	0.28 dB	8.45 dB
PQE	1	1	3
Voiced / unvoiced			
$SNR_{spectrogram}$	1.16 dB	0.89 dB	8.38 dB
PQE	1	1	3

Table C.4: **Reconstruction 6** - Reconstruction, where $K = 500$.

reconstruction filter	*mfcc*	*hfcc*	*none*
No mel filter used in training			
$SNR_{spectrogram}$	-6.86 dB	-7.68 dB	8.38 dB
PQE	1	1	3
HFCC filter used in training			
$SNR_{spectrogram}$	0.61 dB	-0.61 dB	8.67 dB
PQE	1	1	3
MFCC filter used in training			
$SNR_{spectrogram}$	-0.07 dB	-0.59 dB	8.61 dB
PQE	1	1	3

C.3 Box masked signal reconstruction

Table C.5: **Reconstruction 8** - Overfitted reconstruction, where $K = M$.

reconstruction filter	*mfcc*	*hfcc*	*none*
Testdata included in training			
$SNR_{spectrogram}$	13.98 dB	14.21 dB	47.06 dB
PQE	3	3	5
Testdata excluded from training			
$SNR_{spectrogram}$	13.97 dB	14.03 dB	19.40 dB
PQE	3	3	3
Voiced / unvoiced			
$SNR_{spectrogram}$	14.88 dB	14.89 dB	19.46 dB
PQE	3	3	4

Table C.6: **Reconstruction 9** - Reconstruction, where $K = 500$.

reconstruction filter	*mfcc*	*hfcc*	*none*
No mel filter used in training			
$SNR_{spectrogram}$	12.22 dB	13.88 dB	20.22 dB
PQE	3	3	3
HFCC filter used in training			
$SNR_{spectrogram}$	14.81 dB	14.34 dB	19.83 dB
PQE	3	3	3
MFCC filter used in training			
$SNR_{spectrogram}$	14.11 dB	14.37 dB	20.07 dB
PQE	3	3	3

C.4 Speaker independant reconstruction

Table C.7: **Female reconstruction** - Here we use female codebooks, $K = 500$.

Number of speakers	*1*	*2*	*4*	*6*	*Masked Spectrogram*
$M =$	*6178*	*12372*	*23060*	*33520*	
Ideal binary mask					
$SNR_{spectrogram}$	21.47 dB	19.91 dB	17.14 dB	15.51 dB	30.29 dB
PQE	3	3	3	3	4
Random mask					
$SNR_{spectrogram}$	8.98 dB	9.26 dB	9.31 dB	9.27 dB	3.42 dB
PQE	3	3	3	3	3
Box mask					
$SNR_{spectrogram}$	21.36 dB	21.11 dB	21.95 dB	23.05 dB	19.41 dB
PQE	3	3	3	3	3

Table C.8: **Male reconstruction** - Here we use male codebooks, $K = 500$.

Number of speakers	*1*	*2*	*4*	*6*	*Masked Spectrogram*
$M =$	*4367*	*9477*	*19502*	*29404*	
Ideal binary mask					
$SNR_{spectrogram}$	11.85 dB	10.84 dB	10.13 dB	10.25 dB	18.65 dB
PQE	3	3	3	3	3
Random mask					
$SNR_{spectrogram}$	8.61 dB	8.74 dB	8.49 dB	9.17 dB	2.98 dB
PQE	3	3	2	2	3
Box mask					
$SNR_{spectrogram}$	20.07 dB	20.23 dB	19.73 dB	19.82 dB	15.71 dB
PQE	3	3	3	3	3

Table C.9: **Mixed reconstruction** - Here we use mixed codebooks containing both genders, $K = 500$.

Number of speakers	*2*	*4*	*6*
$M =$	*10547*	*21851*	*31039*
Ideal binary mask			
$SNR_{spectrogram}$	9.04 dB	11.21 dB	11.66 dB
PQE	3	3	3
Random mask			
$SNR_{spectrogram}$	7.88 dB	8.29 dB	8.23 dB
PQE	2	2	2
Box mask			
$SNR_{spectrogram}$	19.94 dB	30.15* dB	31.24* dB
PQE	3	3	3

* *It is noted through casual listening tests, that even though the SNR is high for these cases, the speaker voice seems warped.*

Bibliography

[1] Albert S. Bregman, *Auditory Scene analysis* The MIT Press, Massachussetts Institute og Technology, Cambridge, Massachussetts, 1990.

[2] J. Bouvrie, T. Ezzat, *An Incremental Algorithm for Signal Reconstruction from Short-Time Fourier Transform Magnitude*, Center for Biological and Computational Learning Department of Brain and Cognitive Sciences, MIT., To appear at Interspeech, 2006.

[3] C. G. Broyden, *A Class of Methods for Solving Nonlinear Simultanious Equations*, Mathematics of Computation, vol. 19, pp. 577-593, 1965.

[4] Sanjoy Dasgupta, *Performance guarantees for hierarchical clustering*, Journal of Computer and System Sciences, vol. 70(4), pp. 555-569, 2005. http://www.phillong.info/publications/hier.pdf

[5] S.B. Davis & P. Mermelstein, *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*, IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 28, pp. 357-366, 1980.

[6] Charles Elkan, *Using the Triangle Inequality to Accelerate k-Means*, In Proceedings of the Twentieth International Conference on Machine Learning (ICML'03), pp. 147-153, 2003.
http://www.cse.ucsd.edu/users/elkan/kmeansicml03.pdf

[7] Daniel P. W. Ellis., Ron J. Weiss, *Model-based Monaural Source Seperation Using a Vector-Quantized Phase-Vocoder Representation*, Acoustics, Speech and Signal Processing (ICASSP'06), pp. V-957-960, 2006.

[8] N. Evans, J. Mason, W. Liu, B. Fauve, *An Assessment on the Fundamental Limitations of Spectral Subtraction*, Acoustics, Speech and Signal Processing, (ICASSP'06), vol 1, pp. 145-148, 2006.

[9] J.R. Fienup, *Phase retrieval algorithms: a comparison*, Appl. Opt., vol. 21(15), pp. 2758-2769, 1982.

[10] Daniel W. Griffin, Jae S. Lim, *Signal Estimation from Modified Short-Time Fourier Transform*, IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 32, pp. 236-242, 1984.

[11] Guoning Hu, DeLiang Wang, *Monaural Speech Segregation Based on Pitch Tracking and Amplitude Modulation*, IEEE Transactions on Neural Networks, vol. 15, pp. 1135-1150, 2004.
http://www.cse.ohio-state.edu/~dwang/papers/Hu-Wang.tnn04.pdf

[12] Kannan Achan, Sam T. Roweis, Brendan J. Frey, *Probabilistic Inference of Speech Signals from Phaseless Spectrograms*, Neural Information Processing Systems 16 (NIPS'03), pp. 1393-1400, 2003.
http://www.cs.toronto.edu/~roweis/papers/cgsp_nips2003.pdf

[13] Stuart P. Lloyd, *Least squares quantization in PCM*, IEEE Trans. on Information Theory, vol 28, 129-137, 1982.

[14] J. Markel, and A. Gray, *Linear Prediction of Speech*, Springer-Verlag, Berlin, 1976.

[15] Jarno Mielikainen, *A novel full-search vector quantization algorithm based on the law of cosines*, IEEE signal processing letters, vol. 9(6), pp. 175-176, 2002.

[16] Michael T. Orchard, *A fast nearest-neighbor search algorithm*, Acoustics, Speech, and Signal Processing, (ICASSP-91), vol 4, pp. 2297-2300, 1991.

[17] K.K. Paliwal, L.D. Alsteris, *Usefulness of Phase in Speech Processing*, Speech Communication, vol. 45, pp. 153-170, 2005.

[18] Z. Pan, K. Kotani, T. Ohmi, *An Improved Full-Search-Equivalent Vector Quantization Method Using the Law of Cosines*, IEEE signal processing letters, vol. 11, pp. 247-250, 2004.

[19] Manuel Reyes-Gomez, Nebojsa Jojic, and Daniel P.W. Ellis, *Deformable spectrograms*, AI & Statistics 2005, pp. 285-292. 2005.
http://www.ee.columbia.edu/~dpwe/pubs/aistats05-defspec.pdf

[20] M. Roelands, W. Verhelst, *Waveform Similarity Based Overlap-Add (WSOLA) for Time-Scale Modification of Speech: Structures and Evaluation*, European Conference on Speech Communication and Technology, pp. 337-340, 1993.

[21] Roucos, S & A. M. Wilgus, *High quality time-scale modification for speech*, Acoustics, Speech and Signal Processing (ICASSP'85), pp. 493-496, 1985.

[22] Sam T. Roweis, *One Microphone Source Seperation*, Neural Information Processing Systems 13 (NIPS'00), pp. 793-799, 2000. http://www.cs.toronto.edu/∼roweis/papers/onemic.pdf

[23] Mark D. Skowronski and John G. Harris, *Exploiting independent filter bandwidth of human factor cepstral coefficients in automatic speech recognition*, J. Acoustical Society of America, vol. 116, pp. 1774-1780, 2004.

[24] Mark D. Skowronski, *Biologically Inspired Noise-Robust Speech Recognition For Both Man And Machine*, Ph.D. dissertation, Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, 2004. http://www.cnel.ufl.edu/∼markskow/papers/mdsThesisMain.pdf

[25] Soundararajan Srinivasan, DeLiang Wang, *A schema-based model for phonemic restoration*, Speech Communication, vol. 45, pp. 63-87, 2005. http://www.cse.ohio-state.edu/∼dwang/papers/Sriniv-Wang.spcomm05.pdf

[26] H. Valbret, E. Moulines, J. P. Tubach, *Voice transformation using PSOLA technique*, Speech Communication, vol. 11, pp. 175-187, 1992.

[27] DeLiang Wang, Guy J. Brown, *Seperation of Speech from Interfering Sounds Base on Oscillatory Correlation*, IEEE Transactions on Neural Networks, vol. 10, pp. 684-697, 1999. http://www.cse.ohio-state.edu/∼dwang/papers/Wang-Brown.tnn99.pdf

[28] Kuang-Shyr Wu, Ja-Chen Li, *Fast VQ encoding by an Efficient Kick-out Condition*, IEEE trans. on circuits and systems for video technology, vol. 10(1), pp. 59-62, 2000.

[29] Malcolm Slaney, *Auditory Toolbox version 2*, Interval Research Corporation, Technical Report, 1998. http://cobweb.ecn.purdue.edu/∼malcolm/interval/1998-010/AuditoryToolboxTechReport.pdf

[30] Carl Edward Rasmussen, *Conjugated Gradient Optimizer for Matlab (minimize.m).* http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/

[31] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, P. Woodland, *The HTK Book (for HTK Version 3.0)*, Cambridge University Engineering Department, 2000.

[32] Web Page *The Basic Properties of Speech*, University of Southampton, Communications Research Group
http://www-mobile.ecs.soton.ac.uk/speech_codecs/speech_properties.html

[33] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, *DARPA TIMIT Accoustic Phonetic Continous Speech Corpus CDROM* NIST, 1993.

[34] Ling Feng, *English Language Speech Database for Speaker Recognition*, Department of Informatics and Mathematical Modelling, Technical University of Denmark http://www2.imm.dtu.dk/~lf/ELSDSR.htm

[35] Interspeech 2006, Source Separation Challenge Database.
http://www.dcs.shef.ac.uk/ martin/SpeechSeparationChallenge.htm

[36] Figure of the vocal tract.
http://www.jcarreras.homestead.com/files/PhoneticsVocalTract.jpg

[37] Spectrogram description at Wikipedia.
http://en.wikipedia.org/wiki/Spectrogram