University of Erlangen-Nuremberg

Multimedia Communications and Signal Processing

Bachelor Thesis

Extension of the REMOS concept for Noise-Robust Frequency Filtering Features

Marek Malewicz

April 2014

Supervisors:

Prof. Dr.-Ing. Walter Kellermann

Prof. Dr.-Ing. Elmar Nöth

M.Sc. Roland Maas

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

_____ _____

Ort, Datum Unterschrift

Contents

Chapter 1

Introduction

Robust speech recognition in a distant-talking environment is highly desirable these days. Due to these distance properties of a speaking environment, the automatic speech recognition (ASR) system receives a signal that is a combination of the speaker voice and the reverberation. This effect degrades the recognition rate drastically. Reverberation has a dispersive effect on the features used in ASR-algorithms. With increased correlation, the conditional independency assumption of the HMMs is violated. Hence, the recognition error increases by a high margin. REMOS (Reverberation Modeling for Speech Recognition) is a framework combining Hidden Markov Models (HMM) with a reverberation model. While the HMMs are used to model the clean speech, REMOS handles the reverberated part in the feature domain. The speech recognition is done with the extended Viterbi-Algorithm. Each step of the algorithm, containing an inner optimization problem subject to a nonlinear constraint, estimates the highest contribution of each HMM to the current set of observation parameters. In this thesis, we will mainly focus on extending the REMOS concept for Frequency Filtering Features, the explanation and implementation of a specific quadratic-programming-based solver that can handle the optimization problem with better efficiency when compared to a straight forward, brute force approach. Both will then be evaluated in terms of computational time. We will take a look at the average performance, as well as, some particular cases with a specific parameterization of the objective function.

Chapter 2

Review of Automatic Speech Recognition

2.1 Feature Extraction

This section is based upon [Sehr09]. Obtaining the speech signal information relevant for our ASR is a vital part of the recognition process. Due to the so called 'curse of dimensionality' it is advised to keep the amount of features as low as possible. Only the parameters which are helpful in making a distinction between non-identical phonemes and words. Otherwise the data will become sparse and the classification more difficult to conduct.

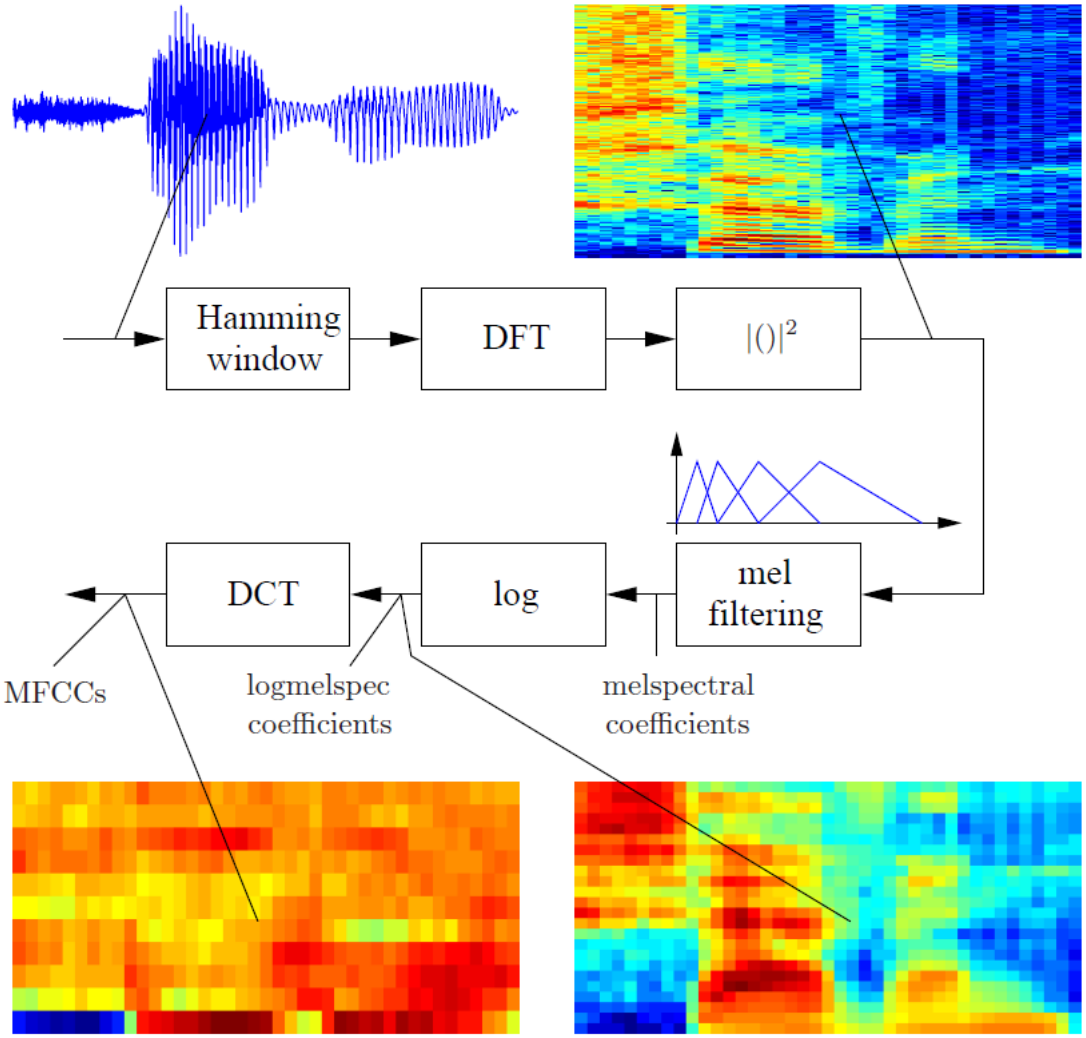

Figure 2.1: Speech recognition system [Sehr09]

After several steps of preprocessing the actual feature extraction takes place. A form of short time analysis is applied to the spectral representation by using overlapping frames of the speech signal in combination with a Hamming window followed by the Short-time Fourier transform (STFT), as can be seen in Fi. 2.1. The exemplary frame length and shift would lie between 20ms-40ms and 10ms-25ms respectively.

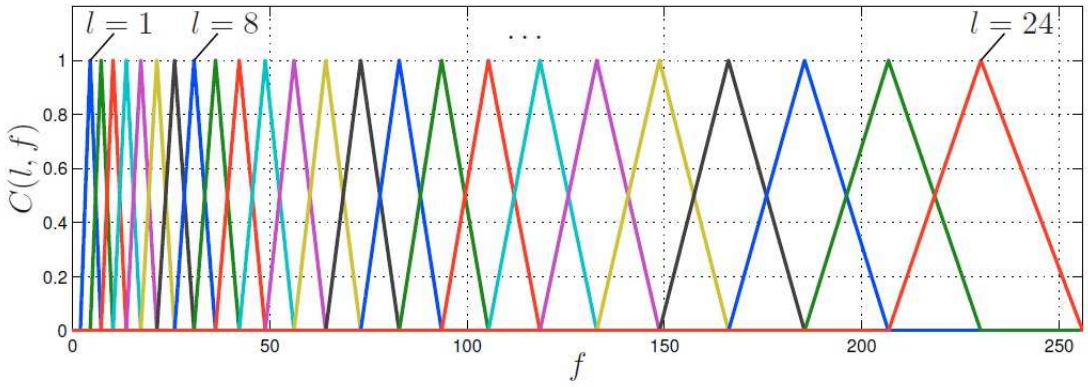

Figure 2.2: Triangular weighting function [Sehr09].

The human sense of hearing has a better resolution for low frequencies. This can be emulated by using the mel filter bank, as shown in Fig. 2.2. The melspec coefficients are further improved by applying the logarithm function. The resulting values are described as the logmelspec coefficients. Now the logmelspec features can be used for Gaussian Mixture Models (GMM). Choosing a special type of GMMs with diagonal covariance matrices reduces the number of parameters needed for the training of the acoustic model. The Decorrelation is done by applying the Discrete Fourier Transform. This is how the MFCCs are created. The 0-th cepstrum coefficient represents the logarithmic energy of the frame, whereas the rest is showing the grade of diversity in respect to that energy over other mel channels. The objective of feature extraction is to obtain the envelope of the time-frequency representation. It is therefore sufficient to consider only the 13 first MFCCs.

The short time spectral changes have a significant influence on the distinction between models [Sehr09]. The first/second order derivatives are also very useful for recognition purposes. The derivatives are then said to be the dynamic features and the MFCCs are referred to as the static features. Yet, as it has proven to be highly challenging to clearly evaluate the MFFC domain, logmelspec features will be used when explaining signal feature characteristics. Because MFCCs and logmelspec coefficients correlate in a manner that one can be transformed into the other by using a linear matrix

transformation.

2.2 Viterbi Decoding

The recognition algorithm uses HMM-based networks containing the information about grammar and pronunciation, as depicted in Fig. 2.3.

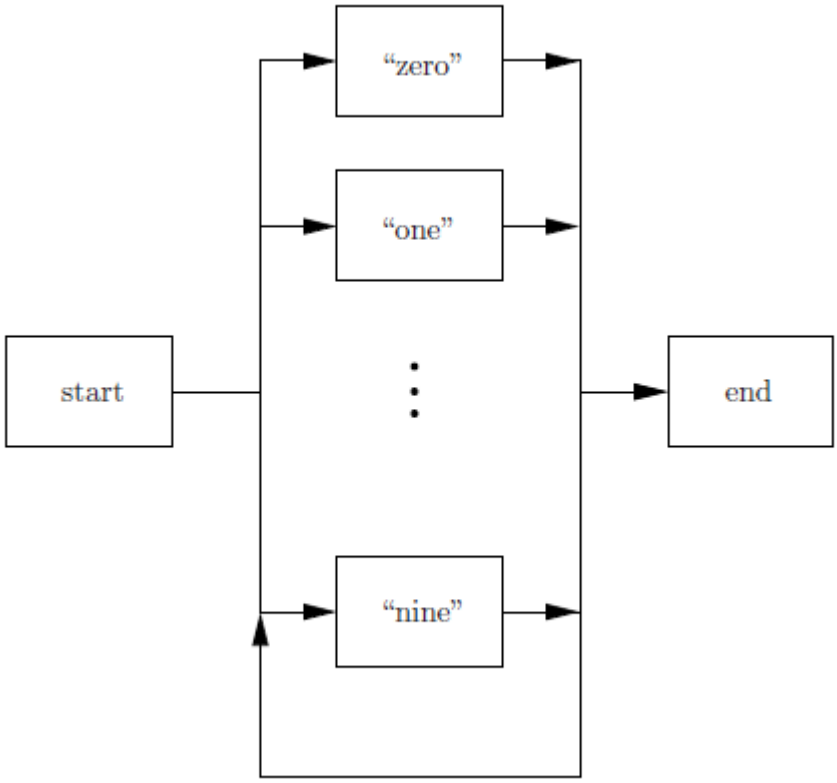

Figure 2.3: Word level grammar [Sehr09]

When an utterance is given, the ASR attempts to find the optimal state sequence of an HMM for a given observation. The score can be computed using the following equation:

$$p(y_{1:n}) = \sum_{q_{1:K}} p(y_{1:n}, q_{1:n}) \tag{2.1}$$

. Due to the nature of the score computation function, which has a high cost in terms

of performance, the Viterbi algorithm is used instead.

$$\sum p(y_{1:n}i, q_{1:n}) = \max_{q_{1:N-1}} p(y_{1:n}, q_{1:n}) \tag{2.2}$$

The acoustic score function calculates the probability of the path most likely characterizing the original observation. In the particular case of word level HMMs, this would yield a group of consecutive words, e.g. 'one', 'two','ten'. For better comprehension, a trellis diagram is used. It shows a state transition possibility. Note that allowed transitions are only to a state with the current index k+1 or looping back to the current state. This way the HMM can 'stretch' itself and adapt to a possible temporal extension of a phoneme.

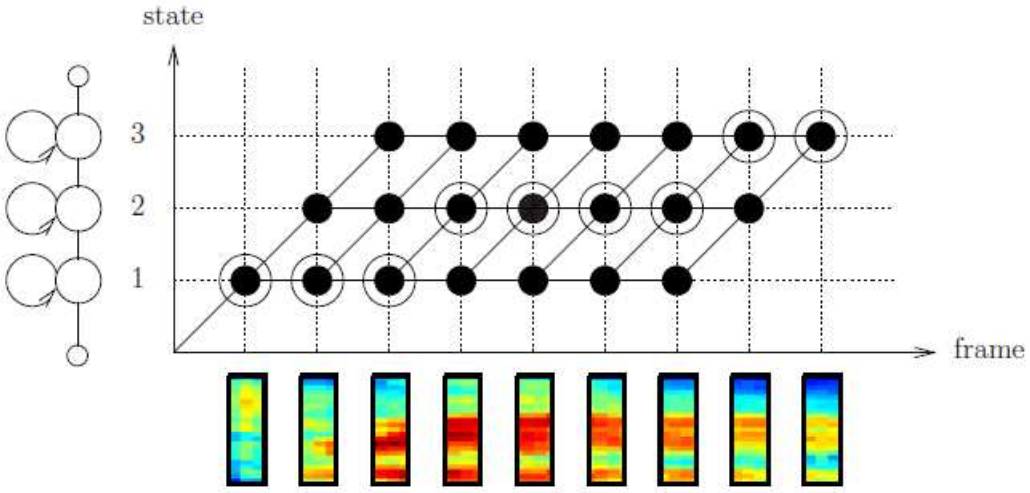

Figure 2.4: Viterbi Decoder [Maas14]

The recognized HMM would have a direct mapping of its states to each observation frame. Starting with the first state that corresponds to the beginning of the observation, the final frame ought to be assigned to the last HMM state. The Viterbi score represents the likelihood of a observation frame 'j' at a given point in time considering the partial HMM sequence being currently in the state 'k'. The implementation has the form of a

recursive function:

$$\tilde{\gamma}_j(k) = p(y_n|q_n) \max_i \{p(q_n \| q_n n - 1 \gamma_i(k-1)\alpha_{ij}\} \tag{2.3}$$

Iterating over all time frames(1:N) yields the score estimation for the whole HMM. The best partial path is contributed via the backtracking function (forward/backward probability).

2.3 Effect of Reverberation

Reverberation is one of the main reasons for ASR underperformance. In general, deploying a recognition system in a distant talking environment will need it to cope with other speakers, background noise as well as the already mentioned reverberant component. The recording unit has to deal with additional speaker signal that has been reflected off walls and other surfaces within the room. These characteristics are described by the room impulse response (RIR).

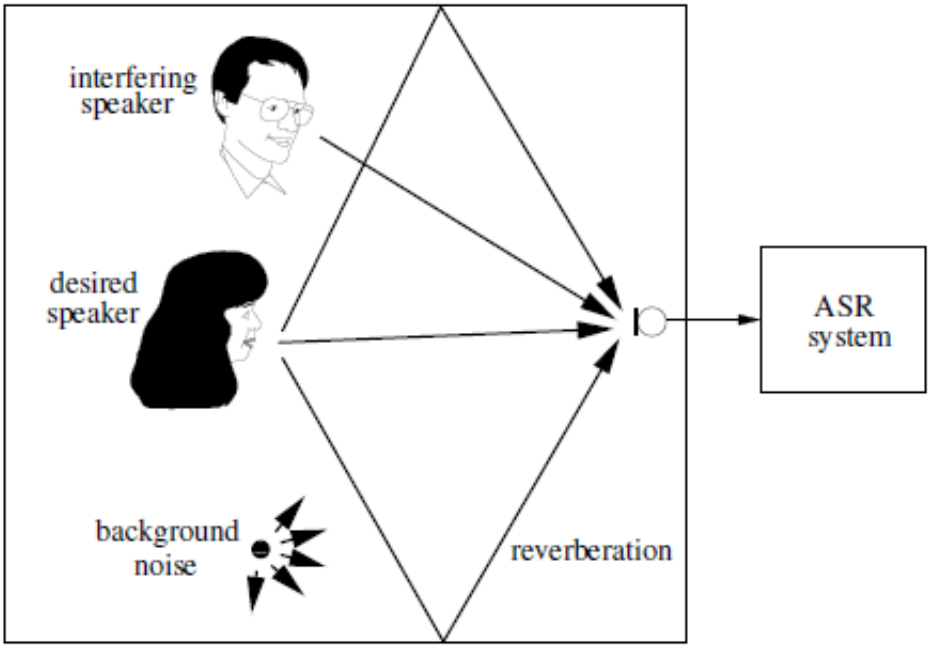

Figure 2.5: Reverberation [Sehr09]

In the time domain, there is a convolutional dependency between the clean speech signal x(t), the RIR h(t) and the reverberated signal y(t) being received by the recording system. The length of the room impulse response is T_{RIR}.

$$y_(t) = \sum_{\tau=0}^{T_{RIR}} h(\tau)q(t-\tau) \tag{2.4}$$

The RIR can be examined under the aspect of its early reflection and late reverberation parts. During the first 50ms the RIR is influenced by the distance relation between the speaker and microphone as well as the AD/DA conversion within the microphone itself. In the second phase more overlapping takes place as the magnitude decays in an exponential manner. By the time the energy value a reaches 60dB decrease compared to the direct signal, the sound energy has decayed to such a low value that it does not need to be considered anymore. The time interval between the beginning of the signal and the point where the magnitude is small enough, is called reverberation time

T_{60}. This is sufficient in most experimental and practical environments, even if the theoretical length of a RIR reaches infinity.

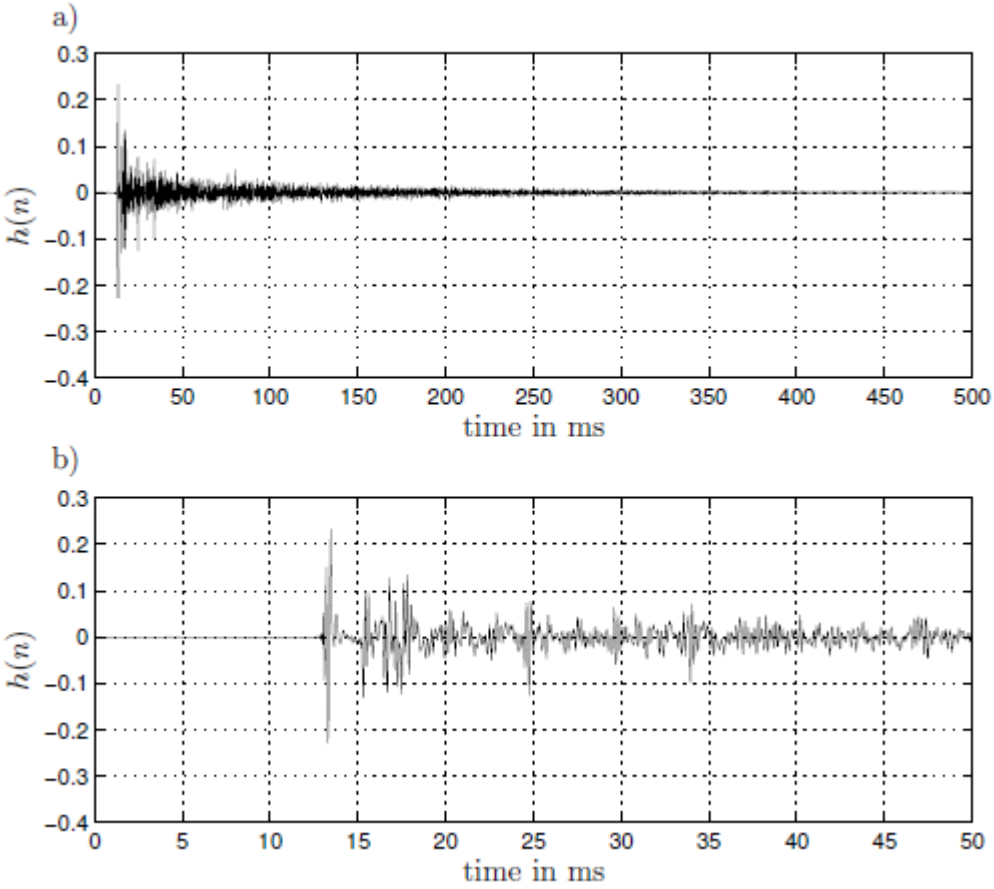

Figure 2.6: Room impulse response (RIR)[Sehr09].

An alternative way of describing the reverberant signal is the summation of early and late reverberant components. The human hearing cannot fully distinguish the early reflections and therefore is classifies the signal before the first 50ms as direct. After that point in time it becomes increasingly aware of the echoic part [Sehr09]. As distinguishing between the two terms of echo and reverberation is a non trivial matter to some, it would be perhaps appropriate to point out the differences. The difference between reverberation and acoustic echoes lies within the fact that a reverberant signal is a set of copies of the original signal. The echoic part consists of multiple copies of

interfering loudspeaker signals. ASR systems are only aware of the reverberant signal. They are working without any information on the positioning or the characteristics of the reverberant environment. The original signal is being sought via blind approach. The nature of solving a dereverberation task is a substantial challenge when compared to echo cancellation. Late reflection signals can be interpreted as additive noise. The is because they are expected to be uncorrelated in reference to each other due to the observation that autocorrelation coefficients for for time frames greater 50ms tend to remain small and therefore unimportant [YSD12]. The autocorrelation coefficients of he reverberant signal y(t), on the other hand, are high enough that they allow to estimate the the late reverberation based on pre-acquired partial knowledge from the past. The only issue with this statement is that the HMMs are assumed to be conditionally independent. A substantial problem is also the fact that the early reflections have the form of a non-stationary signal.

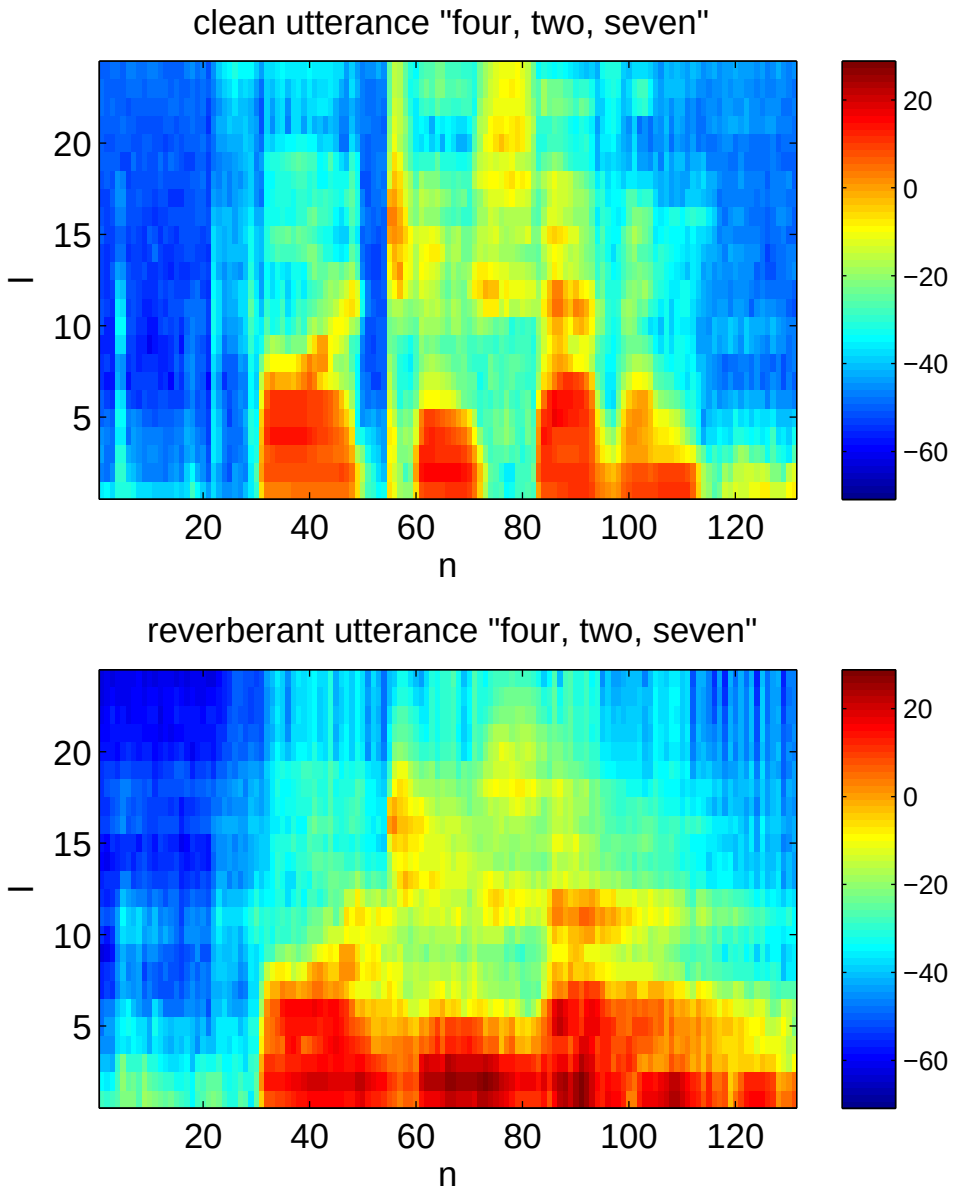

Figure 2.7: Logmelspec representation of an clean and reverberant utterance [Maas14].

The figure 2.7 shows the correlation of the reverberant signal in comparison to the clean speech signal. This is caused by the nature of short-time Fourier Transform [YSD12] employed in feature extraction and brings us to the point where it is not possible to describe the reverberant signal as a multiplication in the feature domain.

The autocovariance coefficients of reverberant signals decay slower than their counter-

parts of the direct clean signals. ASR frameworks are told to be less capable of fulfilling their task in cases where the value of the reverberation time T_{60} is increased [Sehr09]. While it is said that late reverberation may cause problems for the ASR, the early reflection can be considered helpful [SHMK10]. HMM based dereverberation approaches are difficult to incorporate due to the requirement of conditional independence neighboring feature vectors. One way to tackle this problem would be to increase the rank order of HMMs. Therefore, at least in terms of current technological standings, incorporating higher order HMM solutions is out of the question [YSD12].

Chapter 3

REMOS optimization problem

This Chapter is based on [Maas14]. The Reverberation Modeling for Speech Recognition (REMOS) is an uncertainty decoding framework that exploits the properties of reverberation and allows to modify the HMMs, so that can overcome their disadvantages regarding the conditional dependencies and perform under reverberant conditions. By approximating the late reverberation segments, the altered Viterbi algorithm achieves relaxed conditional independence.

3.1 Extended Viterbi Decoding

REMOS is using a melspectral domain based observation model [Sehr09] [RNS06]. The reverberation model is defined as a discrete melspec convolution.

$$y_n^{mel} = \sum_{i=1}^{L} \alpha_l^{mel} * x_{n-l}^{mel} \tag{3.1}$$

The variables x, y describe clean and reverberant speech, respectively and the RIR characteristics are given by α, as depicted in Fig. 3.1 and 3.2. However, details like frame overlap, phase contribution and cross terms are not being taken into account as the melspec convolution is only an estimation of the features and not the actual representation. While it is not able to precisely depict the reverberant frame cycle, the

envelope is modeled quite accurate [Sehr09].

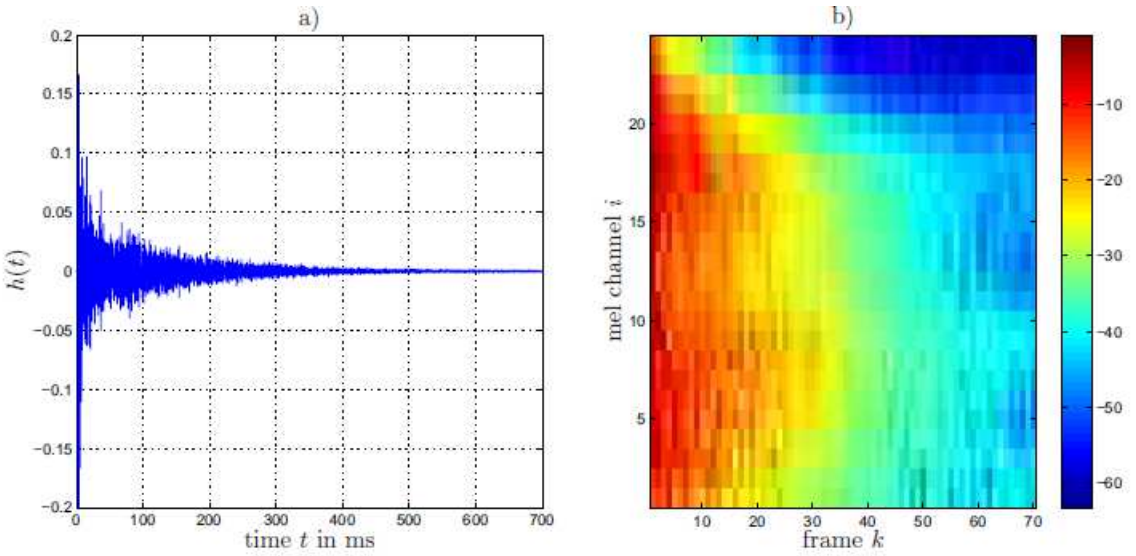

Figure 3.1: a)Room impulse response (RIR) b) Reverberation model in the melspec domain [Sehr09]

The melspec convolution coefficients are, just as it is the case with the MFCCs, static features. Nevertheless, the RIR is not time invariant so that the need for more flexibility arises. In this case, the early and late part of the room impulse response can be described as a normal distribution and a Gaussian, respectively. Early and late components of the RIR are presumed to be independent.

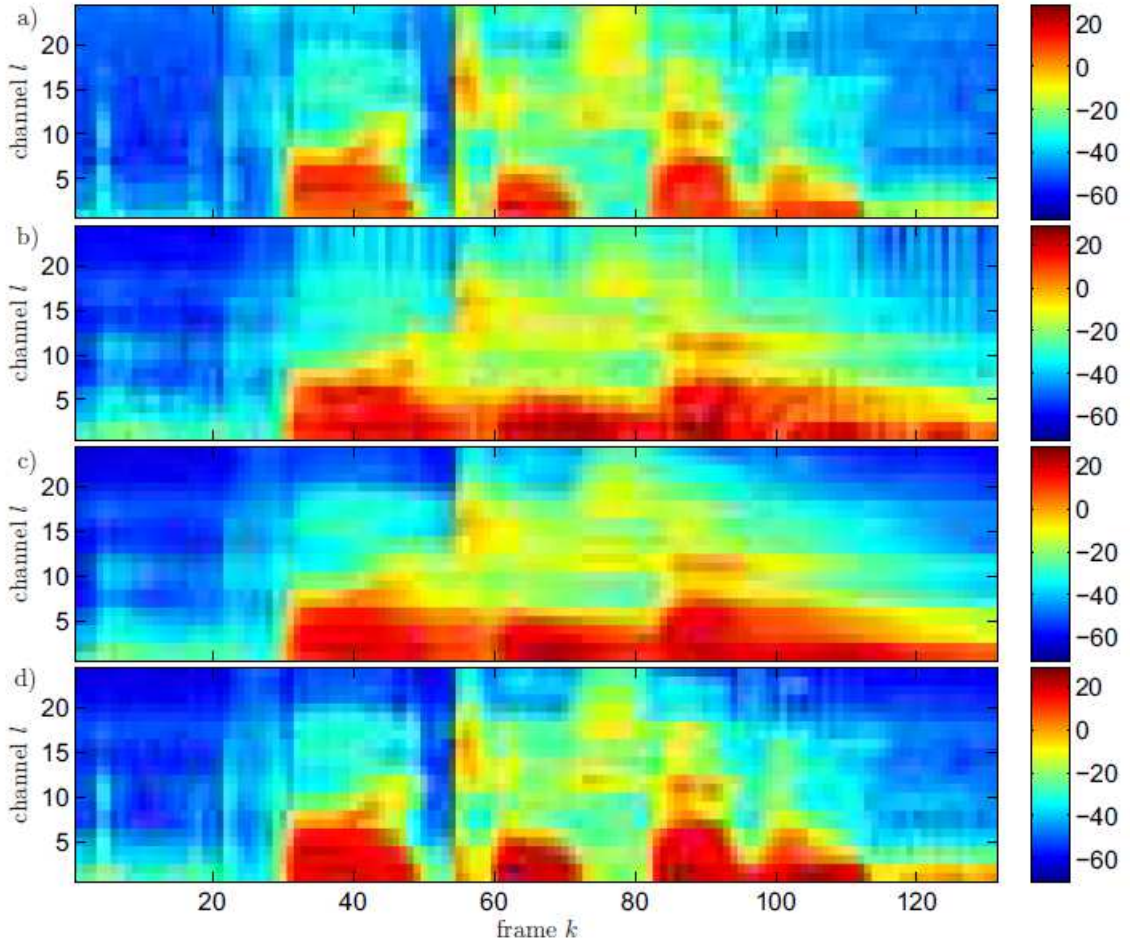

Figure 3.2: Melspec feature vector sequence using a dB color scale: a) clean speech, b) reverberant speech and c) according to (3.1) [Sehr09].

Another crucial task that needs to be tackled is dealing with background noise. In regard to additive signal distortions, a logarithmic model can be used. Analogous to the melspec convolution, the logmelspec is also to be considered an approximation [HRR13] , yet again using normal distributions to model additive noise.

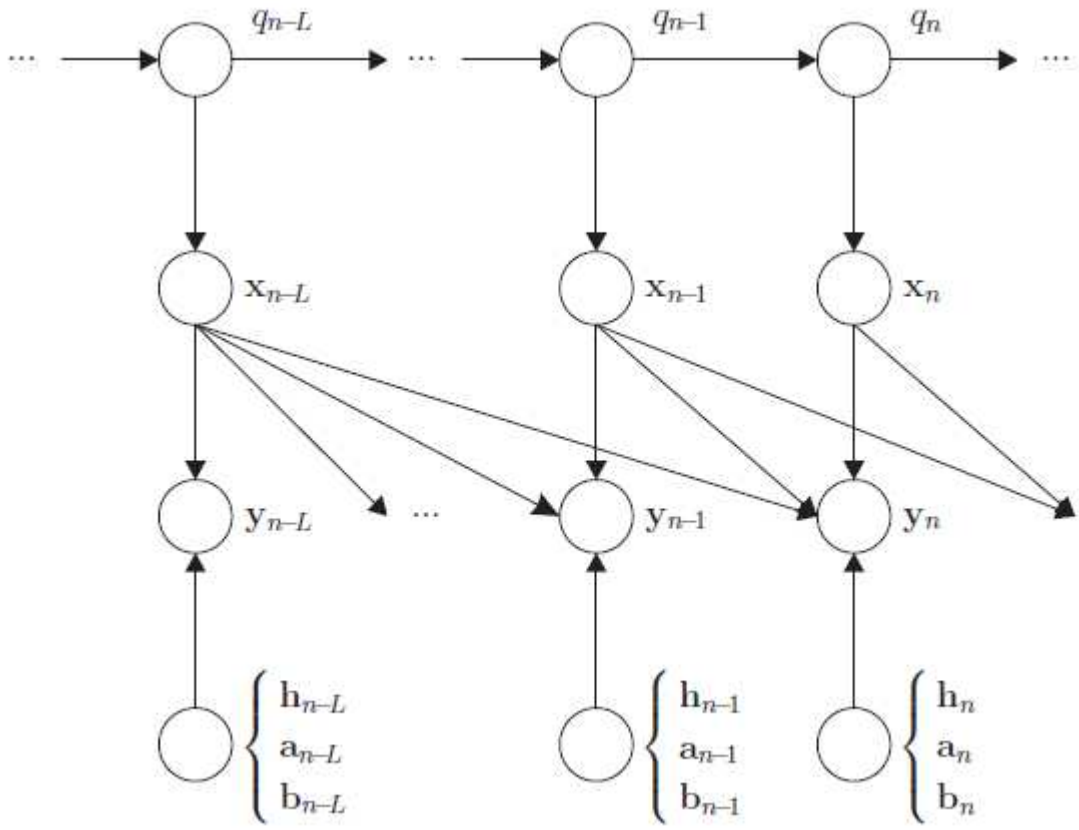

Figure 3.3: REMOS concept as a Bayesian network [Maas14].

Fig 3.3 shows a Bayesian network representing the probabilistic relationships between the random variables $\mathbf{a_n}, \mathbf{b_n}, \mathbf{h_n}$ and their conditional dependencies. The variables $\mathbf{a_n}, \mathbf{b_n}, \mathbf{h_n}$ describe the uncertainty of the late reverberation part, additive noise component and the RIR , respectively.

In conclusion, the logmelspec observation model would look as follows,

$$y_n = \log(\exp(h_n + x_n) + \exp(a_n + r_n(x_n - L_{n-1})) + \exp(b_n)) \tag{3.2}$$

with $\mathbf{r_n}$ being the reverberation tail.

The new observation model leads to an updated formula for the Viterbi decoder cal-

culation. Thus, we obtain the acoustic score:

$$\sum_{q_1:N} p(y_{1:N}, q_{1:N}) = \max_{q_1:N-1} p(y_{1:N}, q_{1:N}) \tag{3.3}$$

. These newly introduced variables in Fig. 3.3 are not directly observed. It is rather the case that they are being inferred from and aim to explain the already observed data.

$$\max_{q_1:N-1} p(y_{1:N}, q_{1:N}) = \max_{q_1:N-1} \{ \int p(y_N \| x_{N-L:N}) p(x_N \| q_N) p(y_{1:N-1}, q_{1:N-1}, x_{1:N}) dx_{1:N} \} \tag{3.4}$$

where y_N, x_N and q_N all share a mutual conditional independency. The formula can be modified, such that it can be integrated into the original Viterbi decoder. The extended Viterbi score of the REMOS decoder can be written as

$$\left\{ \begin{array}{c} \ddot{\gamma}(y_{1:n}, q_n) := \max_{q_1:n-1} \{ \max_{d_n, a_n, b_n} \\ \delta(y_n - F(d_n, x_{n-\hat{L}:n-1}, a_n, b_n)) p(a_n) p(b_n) p(d_n \| q_n) p(q_n \| q_{n-1}) \ddot{\gamma}(y_{1:n}, q_{n-1}) \} \end{array} \right\} \tag{3.5}$$

with $F(d_n, x_{n-\hat{L}:n-1}, a_n, b_n$ representing the overall observation model in the logmelspec domain. In essence, the extended Viterbi score $\ddot{\gamma}(\mathbf{y_{1:n}}, \mathbf{q_n})$ grants the optimal state sequence $\hat{\mathbf{q}}$, just as it was in the case of the fundamental Viterbi decoder. In addition, it provides the feature vectors $\hat{\mathbf{d}}_{\mathbf{q_n}}, \hat{\mathbf{a}}_{\mathbf{q_n}}, \hat{\mathbf{b}}_{\mathbf{q_q}}$ which allow the extraction of the optimal clean speech estimate $\hat{\mathbf{x_n}}(\mathbf{q_n})$ from the direct path $\mathbf{d_n}(\hat{\mathbf{q}}_\mathbf{n})$ with $\mathbf{q_n}$ representing the current state and $\mathbf{y_{1:n}}$ being the observed feature vector sequence.

$$\hat{x}_{qn} := argmax_{x_n} p(\hat{d}_n(q_n) - x_n) p(x_n \| q_n) \tag{3.6}$$

Finally, the late reverberation part can be computed with

$$r_n(x_{n-\hat{L}:n-1}) = \log(\sum_{i=1}^{L} \exp(\alpha_i) + \hat{x}_{n-i}) \tag{3.7}$$

with the recursion step taking the partial estimation of the previous state q_{n-1}.

In conclusion to this section, what we need to solve, is a constrained optimization

problem with the objective function

$$\max d_n, a_n, b_n p(d_n \| q_n) p(a_n) p(b_n) subject to y_n = F(d_n, \hat{x}_{n-L:n-1}, a_n, b_n) \tag{3.8}$$

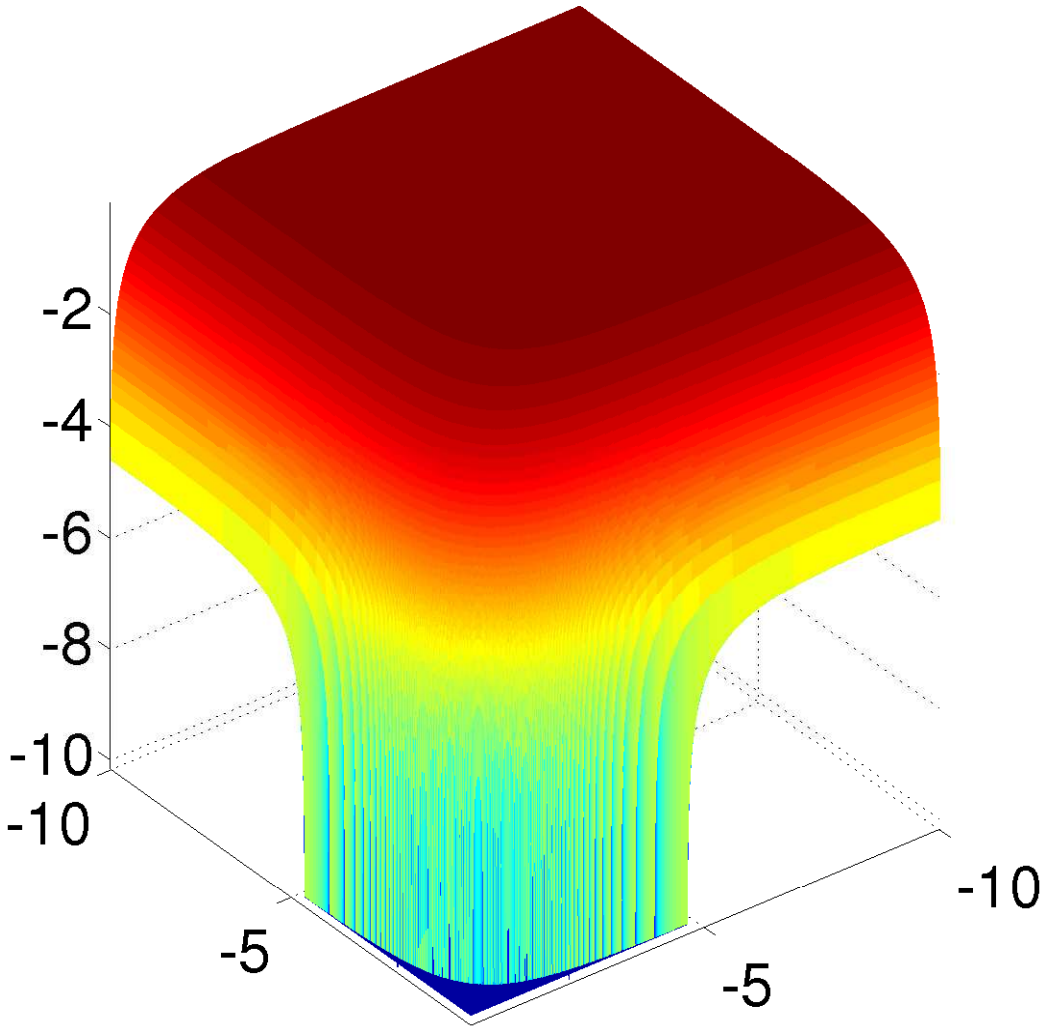

Figure 3.4: Illustration of the optimization problem [Maas14] (3.8)

3.2 Frequency-Filtering domain with a noise model

According to (3.8), a non-linear optimization problem needs to be solved during each step of the Viterbi algorithm . The covariance matrices of the emission pdfs are as-

sumed to be diagonal. Although the logmelspec features have proven to have a higher channel correlation than their MFCC counterparts, what is the reason for them being incorporated in ASR systems nowadays. Our problem here is that diagonal covariance matrices in the MFCC domain yield full covariance matrices in the logmelspec domain. In result, the original optimization problem 3.1 can be formulated as:

$$\max_{u,v,w} f(u,v,w) := \tfrac{1}{2}(u-\mu_u)^T\mathbf{C}_u(u-\mu_u) + \tfrac{1}{2}(v-\mu_v)^T\mathbf{C}_v(v-\mu_v) + \tfrac{1}{2}(w-\mu_w)^T\mathbf{C}_w(w-\mu_w)$$
$$\text{subject to:}(u^{(i)},v^{(i)},w^i) \in \text{ G with i: 1...L} \tag{3.9}$$

where the piecewise planar estimation 3.5 having K plane segments $\mathbf{G_{1:K}}$ will be applied to each feature (1...L) separately.

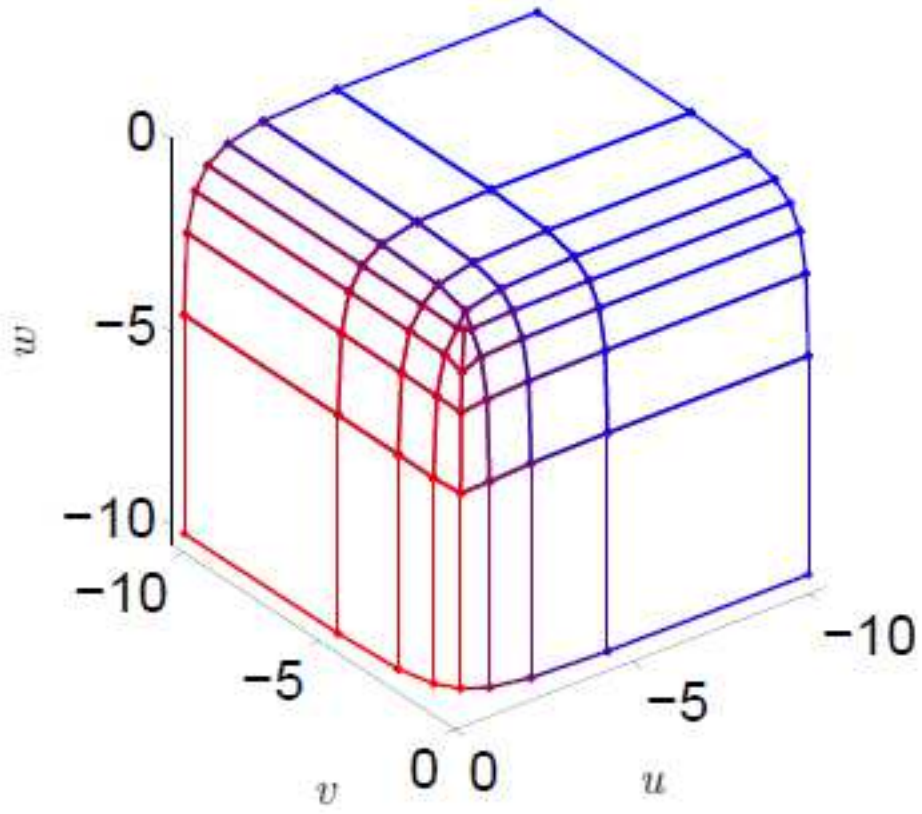

Figure 3.5: Piecewise planar approximation of the nonlinear constraint [Maas14].

We could solve this problem with a straight forward approach by systematically enumerating all possible candidates and picking up the best solution. However, due to the high computation time the brute force solution does not seem appropriate in this case. Hence, a more simple solution is needed. The main point is to obtain a set of parameters that are able to decorrelate the logmelspec channels and split the problem into smaller ones, which then are easier and faster to compute.

3.2.1 Adaptation

The feature channels are assumed to be decorrelated by using a transform matrix $\mathbf{S}$.

$$\begin{aligned} \mathbf{C}_{u.dec} &= \mathbf{S}\mathbf{C}_u^T \\ \mathbf{C}_{v.dec} &= \mathbf{S}\mathbf{C}_v^T \\ \mathbf{C}_{w.dec} &= \mathbf{S}\mathbf{C}_w^T \end{aligned} \tag{3.10}$$

, where $\mathbf{C_u}, \mathbf{C_v}, \mathbf{C_w}$ are the covariance matrices and $\mathbf{C_{u,dec}}, \mathbf{C_{v,dec}}, \mathbf{C_{w,dec}}$ describe diagonal covariance matrices related to the logmelspec domain. The vectors $\mathbf{u_{dec}}, \mathbf{v_{dec}}, \mathbf{w_{dec}}$ contain the MFCC features. In general, the transform matrix is block diagonal, filled with smaller DCT matrices. Therefore, the same applies to the structural behavior of the covariance matrices which also have a block diagonal representation. By choosing this form we can replace the optimization problem with a set of subproblems having a lower dimension.

$$\mathbf{S} = \begin{pmatrix} A_{1,1} & A_{1,2} & 0 & 0 & 0 & 0 \\ A_{2,1} & A_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & B_{1,1} & B_{1,2} & 0 & 0 \\ 0 & 0 & B_{2,1} & B_{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{1,1} & C_{1,2} \\ 0 & 0 & 0 & 0 & C_{2,1} & C_{2,2} \end{pmatrix} \tag{3.11}$$

This lead to a decomposition into 3 subproblems $\in R^2 \times R^2$. There are some prerequisites in terms of characteristics that have to be met by the transformation matrix. It has to be regular and have the rank L. Would it not be the case then the matrices would be singular with an infinite amount of global optima. Hence, the vectors $\vec{u}, \vec{v}$ and $\vec{w}$ could not be estimated and in consequence prevent the Viterbi decoder from working properly.

The newly derived features are also partially decorrelated and yield by using a regular transformation matrix of a block diagonal structure. Aside of that, an optimized numerical solver will be introduced. It is specialized in coping with the big optimization

problem after it has been broken down into smaller subproblems. As shown in the end, it has a lower computational load compared to the brute force approach by a sizable margin.

The adaptation process expects the features to have such characteristics that they can be distinguished from one another. Analyzing and organizing data in high dimensional spaces is difficult. With increasing dimensionality the volume of the classification space grows rapidly, often exponentially, causing the data to become sparse. Consequently, it becomes troublesome, if not impossible, to obtain a viable classification. Also, the classifier itself is expected to characterize the data components accordingly. From us it would mean having a robust representation of the pdf models. In addition, using the REMOS framework requires a certain degree of trade off in relation to the first two prerequisites. Data modeling efficiency suffers in terms of recognition accuracy rate when the logmelspec features are represented by Gaussian Mixture Models (GMMs)[NHG95]. They need to be decorrelated. In other words, the cepstral variance is to be made constant[JRW87]. The logmelspec domain has the characteristics of a stationary process with a cepstral variance decaying inversely proportional to the squared quefrency bin.

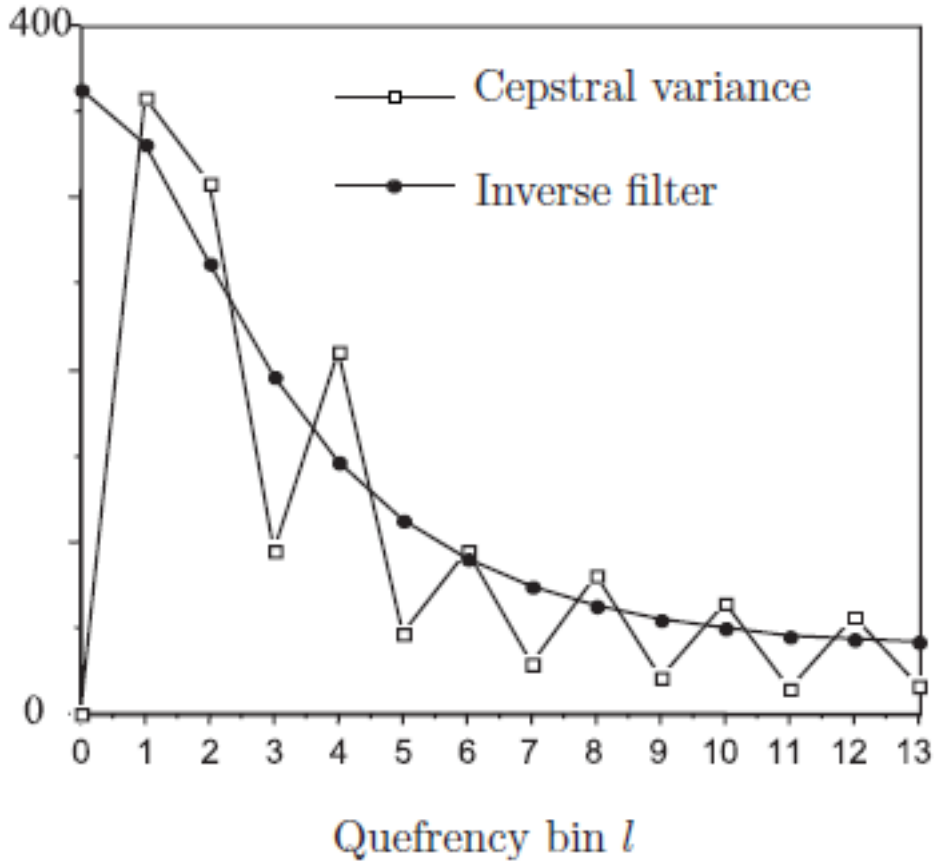

Figure 3.6: Cepstral variance compared to $T(z) = 1 - \frac{1}{2}y^{-1}$ [NHG95]

This can be done by using a high pass filter, for instance one like $T(z) = 1 - \frac{1}{2}z^{-1}$. It is stated, that holding the cepstral variance at a approximately constant level, is more a necessary, and not a sufficient requirement. Nevertheless, according to several theories [NHG95] [NMH01], it leads to a certain degree of decorrelation that we find acceptable and are able to work with. It starts being more problematic when we realize that each cepstral coefficient comes with a distinct degree in how divergent its values can get. The higher the quefrency, the lesser the recognition reliability [JRW87] [NMH01]. As a satisfactory result for both decorrelation of the features and phoneme discrimination cannot be attained, we try to find the middle ground by taking the approach of high and low quefrency deemphasis with $T(z) = 1 - z^{-2}$. If we apply that thought for

Frequency Filtering (FF) scheme and pack the following FF transformation matrix:

$$\mathbf{S} = \begin{pmatrix} 0 & -1 & & & \\ 1 & 0 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & -1 \\ & & & 1 & 0 \end{pmatrix}$$

and modify it so that it complies to the requirement of having a block diagonal form [MWST11], we get

$$\mathbf{S}_{Diag} = \begin{pmatrix} 1 & -1 & & & & \\ 1 & 1 & & & & \\ & & 1 & -1 & & \\ & & 1 & 1 & & \\ & & & & 1 & -1 \\ & & & & 1 & 1 \end{pmatrix}$$

The averaging filter $T(z) = 1 + z^{-2}$ massively contributes to the final form of the block diagonal matrix S_{Diag}. It helps to include the original information about the spectral shape of the logmelspec coefficients. At the current point, the structural expectation of REMOS is met and we can proceed with optimization problem.

3.2.2 Solver

Going back to solving the the subproblems means that we have to deal with a sequence of quadratic programs.

$$\min_{u,v,w \in R^2} f(u,v,w) := \tfrac{1}{2}(u-\mu_u)^T C_u (u-\mu_u) + \tfrac{1}{2}(v-\mu_v)^T C_v (v-\mu_v) + \tfrac{1}{2}(w-\mu_w)^T C_w (w-\mu_w)$$
$$\text{subject to } exp(u) + exp(v) + exp(w) = 1 \tag{3.12}$$

Because u,v and w each denote a 2D vector, the constraint condition ought to be seen as a combination of two equations.

$$exp(u_1) + exp(v_1) + exp(w_1) = 1 \tag{3.13}$$

$$exp(u_2) + exp(v_2) + exp(w_2) = 1 \tag{3.14}$$

As the equations are mutually independant, we can estimate the two constraints separately. Hence, the piecewise plane approximation is applied 3.5.

The planes For every possible combination of plane pairs we estimate the subproblem equals solving the optimization problem We estimate the over“ After reformulating the subproblem we now have to run a nested iteration in order to solve the optimization for each pair of plane segments $(P_{1:K,1:K}), G_K^2 = G_K x G_K$ where K stands for the number of planes.

$$\begin{aligned} \min_{u,v,w \in R^2} f(u,v,w) := & \tfrac{1}{2}(u-\mu_u)^T \mathbf{C}_u (u-\mu_u) + \tfrac{1}{2}(v-\mu_v)^T \mathbf{C}_v (v-\mu_v) \\ & + \tfrac{1}{2}(w-\mu_w)^T \mathbf{C}_w (w-\mu_w) \\ & \text{subject to } (u_1, u_2, v_1, v_2, w_1, w_2) \in G^2 \end{aligned} \tag{3.15}$$

During each step, we solve the plane pair $(P_{i,j})$ with

$$\begin{aligned} & \min_{u,v,w \in R^2} f(u,v,w) \\ & \text{subject to } (u_1, v_1, w_1) \in G_i (u_2, v_2, w_2) \in G_j \end{aligned} \tag{3.16}$$

where G_i, G_j are the planes being currently processed. We end up with K^2 distinct iteration steps. As we try to split the set of optimization problems into subsets of an even more unique nature, we come across the characteristic of feasibility for a convex hull space. Four new cases emerge. The first one considers the convex hull scenario for

both of the constraint equations $(P_{\triangle,\triangle})$

$$\begin{aligned} &\min_{u,v,w \in R^2} f(u,v,w) \\ &\text{subject to } (u_1,v_1,w_1) \in conv(G)(u_2,v_2,w_2) \in conv(G) \end{aligned} \tag{3.17}$$

A further method considers a mixture of the convex hull feasibility for one of the constraints and iterating over the range j=1...K for the other constraint, what leaves it unchanged compared to the problem it originated from. With $(P_{\triangle,1:\bar{K}})$

$$\begin{aligned} &\min_{u,v,w \in R^2} f(u,v,w) \\ &\text{subject to } (u_1,v_1,w_1) \in conv(G)(u_2,v_2,w_2) \in \bigcup j = 1^K aff(G_j) \end{aligned} \tag{3.18}$$

By swapping the vector components we can easily derive $(P_{\overline{1:K},\triangle})$. The only difference is that the vector parameters have their constraint association shifted towards the 'other' equation. When computing the quadratic program, we need to be careful and remember one thing. The convex hull approach $\mathbf{conv}(\mathbf{G})$ contributes to the inequality constraints, whereas the affine restriction on $\bigcup_{j=1}^{K} aff(G_j)$, i.e. iterating over the plane set 1...K, is to be considered as part of an equality constraint equation.
In the last variation we apply the affine restriction for both vector components and get $(P_{\overline{1:K},\overline{1:K}})$

$$\begin{aligned} &\min_{u,v,w \in R^2} f(u,v,w) \\ &\text{subject to } (u_1,v_1,w_1) \in \bigcup_{i=1}^{K} aff(G_i)(u_2,v_2,w_2) \in \bigcup_{j=1}^{K} aff(G_j) \end{aligned} \tag{3.19}$$

Any globals solution of $(P_{1:K,1:K})$ can be found within one of its subproblems.

3.3 Implementing the optimization

In last section, we have found a decomposition to $(P_{1:K,1:K})$ in form of a set of methods to compute a specific quadratic program, a quadri-variate quadratic function $f : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$. The numerical solver $(P_{1:K,1:K})$ can be written as shown in Fig. 3.7:

Algorithm 2.12 (Solver for $(P_{1:K,1:K})$).

if $\boldsymbol{\mu} \notin \mathrm{conv}(\mathbb{G})^2$ ***then***
  solve $(P_{\triangle,\triangle})$.
  if *global solution of* $(P_{\triangle,\triangle})$ *is infeasible* ***then***
    solve $(P_{\triangle,\overline{1:K}})$.
    if *global solution of* $(P_{\triangle,\overline{1:K}})$ *is infeasible* ***then***
      solve $(P_{\overline{1:K},\triangle})$.
      if *global solution of* $(P_{\triangle,\overline{1:K}})$ *is infeasible* ***then***
        denote best local solution of $(P_{\triangle,\overline{1:K}})$ *and* $(P_{\overline{1:K},\triangle})$ *that is feasible with* $(\mathbf{u}^*, \mathbf{v}^*)$.
        for $k, j = 1 : K$ ***do***
          if *global solutions to* $(P_{\overline{k},\triangle})$ *and* $(P_{\triangle,\overline{j}})$ *are better than* $(\mathbf{u}^*, \mathbf{v}^*)$ ***then***
            solve $(P_{\overline{k},\overline{j}})$.
            update $(\mathbf{u}^*, \mathbf{v}^*)$ *if better and feasible.*
          end if
        end for
      end if
    end if
  end if
else
  solve $(P_{\overline{1:K},\overline{1:K}})$.
  denote best local solution that is feasible with $(\mathbf{u}^*, \mathbf{v}^*)$
  if $(\mathbf{u}^*, \mathbf{v}^*)$ *is not a global solution to* $(P_{\overline{1:K},\overline{1:K}})$ ***then***
    for $k = 1 : K$ ***do***
      if *solution to* $(P_{\overline{k},\overline{1:K}})$ *is better than* $(\mathbf{u}^*, \mathbf{v}^*)$ ***then***
        solve $(P_{\overline{k},\triangle})$.
        update $(\mathbf{u}^*, \mathbf{v}^*)$ *if better and feasible.*
      end if
      if *solution to* $(P_{\overline{1:K},\overline{k}})$ *is better than* $(\mathbf{u}^*, \mathbf{v}^*)$ ***then***
        solve $(P_{\triangle,\overline{k}})$.
        update $(\mathbf{u}^*, \mathbf{v}^*)$ *if better and feasible.*
      end if
    end for
  end if
end if

(a) solver

Figure 3.7: Solver $(P_{1:K,1:K})$ [Maas14]

Each subproblem revolves around minimizing the original REMOS problem (3.12). Yet, because we are using MATLAB's quadprog function, we are obliged to transform it into [Mathworks]:

$$\begin{aligned} &\min_x \tfrac{1}{2} x_T \mathbf{H} \\ &\text{subject to :} \mathbf{A}x \leq b, \mathbf{A}_{eq}x = b_{eq} \end{aligned} \tag{3.20}$$

where $\mathbf{H}$ represents the diagonal matrix and $\mathbf{A}, \mathbf{A_{eq}}$ are matrices belonging to the inequality and the equality constraint, respectively.

The parameters x and and b,b_{eq}, are considered to be vectors.

The matrix H as diagonal-wise combination of 3 random SPD-matrices. These are dense $n \times n$ symmetric, diagonally dominant, positive definite matrices.

```
SPD1= generateSPDmatrix_corrected(2);
SPD2= generateSPDmatrix_corrected(2);
SPD3= generateSPDmatrix_corrected(2);
H = blkdiag(SPD1,SPD2,SPD3);
```

Furthermore, we need a set of uniformly distributed values on the interval [-10,10] :

```
d = round(-10 + (10+10).*rand(N,1));
e = round(-10 + (10+10).*rand(N,1));
f = round(-10 + (10+10).*rand(N,1));
```

which are, later on, used for creating the mean vector.

The constraint matrices, A, A_{eq} are indirectly obtained, depending on the nature of the specific subproblem, from the plane parameter matrix $\mathbf{M}$:

```
for i=1:num_planes
    M(i,:)  = plane_segments(i).planeCoeffs(1:3) * (-1);
    b(i)    = plane_segments(i).planeCoeffs(4);
end
```

with **num_planes** describing the amount of plane segments used for the piecewise planar approximation of the linear constraint.

3.4 Brute force approach

The straight forward method consist of a sequence of iterations over all possible pairs of planes: $\mathbf{G_i}, \mathbf{G_j}$

```
function [v,F,fsb] = solve_brute(A,B,C,d,e,f,M,b)
    H = blkdiag(A,B,C);
    s = size(M);
    h = zeros(6,1);
    mu = [d; e; f];
```

Please note, how the variable h is set to $\vec{0}$ as we do not have a linear term within the initial formula. When transforming the original REMOS equation into a quadprog-eligible statement, there is the necessity of creating the corresponding mean vector $\vec{mu}$.

```
    best_F = 9999.9;
    best_v = [0 0 0 0 0 0 ];
    best_fsb =  -1;
    num_pl = size(M,1);
```

Here, we initialize the parameters for the optimal solution. The pair of points stored in $\vec{best\_v}$ refer to the first and second constraint, respectively.
$best\_F$ denotes the function value F for the optimal minimization. Even if the estimated optimal solution has the lowest score, i.e. the optimal value, we have to check, whether it is indeed a feasible solution. For that reason, we add an auxiliary term $best\_fsb$ to describe, whether the solution is feasible 3.8 in regard to $\mathbf{M}(u_1, v_1, w1)^T - \vec{b} = 0$ and $\mathbf{M}(u_2, v_2, w2)^T - \vec{b} = 0$.

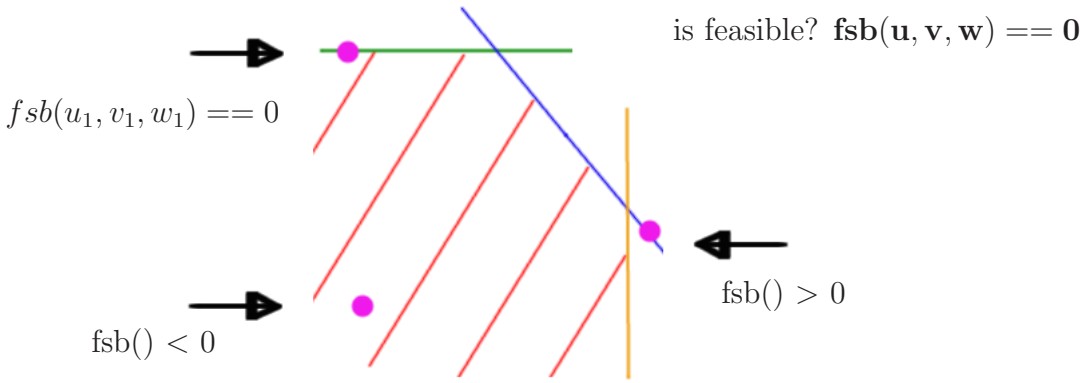

Figure 3.8: Example of a feasible solution $(\mathbf{u_1}, \mathbf{v_1}, \mathbf{w_1})$ provided by the solver.

```
for i=1:num_pl
    for j=1:num_pl
        A_up    = [ M(:,1) zeros(num_planes,1) \
                    M(:,2) zeros(num_planes,1) \
                    M(:,3) zeros(num_planes,1) ] ;
        A_down  = [ zeros(num_planes,1) M(:,1) \
                    zeros(num_planes,1) M(:,2) \
                    zeros(num_planes,1) M(:,3) ] ;
        A = [A_up;A_down];
```

When generating the constraint matrices $\mathbf{A}, \mathbf{A_{eq}}$, the original plane coefficients matrix $\mathbf{M}$ has to be extended, so that, for each dimension of the constraint sequence, only the desired parameters are taken into account. Hence, the first constraint yields the upper

part of the inequality constraint matrix:

$$\begin{pmatrix} M_{1,1} & 0 & M_{1,2} & 0 & M_{1,3} & 0 \\ M_{2,1} & 0 & M_{1,2} & 0 & M_{1,3} & 0 \\ M_{3,1} & 0 & M_{1,2} & 0 & M_{1,3} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{K,1} & 0 & M_{K,2} & 0 & M_{K,3} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \\ w_1 \\ w_2 \end{pmatrix} = \vec{b} \tag{3.21}$$

. In a similar manner, the bottom-part contribution involves only the secondary components:

$$\begin{pmatrix} 0 & M_{1,1} & 0 & M_{1,2} & 0 & M_{1,3} \\ 0 & M_{2,1} & 0 & M_{1,2} & 0 & M_{1,3} \\ 0 & M_{3,1} & 0 & M_{1,2} & 0 & M_{1,3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & M_{K,1} & 0 & M_{K,2} & 0 & M_{K,3} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \\ w_1 \\ w_2 \end{pmatrix} = \vec{b} \tag{3.22}$$

.

```
b = b;
b_ext = [b;b];
Aeq_up = [M(i,1) zeros(1,1) M(i,2) zeros(1,1) M(i,3) \
            zeros(1,1)];
Aeq_low = [zeros(1,1) M(j,1) zeros(1,1) M(j,2) zeros(1,1)\
            M(j,3) ];
Aeq = [Aeq_up;Aeq_low];
```

For our general optimization problem, the equality constraint matrix A_{eq} is required to have at one line equation for each vector component. Furthermore, the extension of **M** applies analogously to how it is done in the case of the inequality constraint. Given

the two planes $\mathbf{G_i}, \mathbf{G_j}$, we define the equality constraint as:

$$\begin{pmatrix} 0 & M_{i,1} & 0 & M_{i,2} & 0 & M_{i,3} \\ M_{j,1} & 0 & M_{j,2} & 0 & M_{j,3} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} b_i \\ b_j \end{pmatrix} \tag{3.23}$$

.

```
beq_up = b(i);
beq_low = b(j);
beq = [beq_up;beq_low];
b_tilde = b_ext - A*mu;
beq_tilde = beq - Aeq*mu;
```

As briefly mentioned before, prior to the actual call of MATLAB's quadprog function, we have to adjust the REMOS-defined parameters $\mathbf{b}, \mathbf{b_{eq}}$ by subtracting $\mathbf{A_{eq}}\tilde{\mu}$. Let us note, that the mean values $\vec{\mu}$ will be added back to the result vector after quadprog finished its estimation routine.

```
opts = optimset('Algorithm','interior-point-convex');
%opts.MaxIter = 1000;
[v,F] = quadprog(H,h,A,b_tilde,Aeq,beq_tilde,[],[],[],opts);

v= v + mu;
fsb = feasible(M,v,b);
```

Should the current pair of planes be feasible and have a better score than the solution, that has been considered best up until this point, then we can update the optimal set.

```
            if fsb == 1 && F < best_F
                best_F = F;
                best_v = v;
                best_fsb = fsb;
            end

        end

    end
    F = best_F;
    v = best_v;
    fsb = best_fsb;
end
```

3.5 $(P_{\triangle,\triangle}), (P_{\triangle,\overline{1:K}}), (P_{\overline{1:K},\triangle}), (P_{\overline{1:K},\overline{1:K},})$

As the overall proceedings are very similar to the first section, there are some substantial perks to the constraint matrix generation $\mathbf{A}, \mathbf{A_{eq}}$ that will prove to have a lower computational load. For instance, the subproblem $(P_{\triangle,\triangle})$:

```
function [v,F,fsb] = solve_ff(A,B,C,d,e,f,M,b)
    H = blkdiag(A,B,C);
    s = size(M);
    h = zeros(6,1);
    A_ext_left = [M(:,1) 0 M(:,2) 0 M(:,3) 0)] ;
    b_left = b;
    A_ext_right = [0 M(:,1) 0 M(:,2) 0 M(:,3)] ;
    b_right = b;
    A_ext = [A_ext_left ; A_ext_right];
    b_ext = [b_left ; b_right];
```

does not contain the equality constraint. This means that, for both components, we only have to consider the hull volumes (3.17) 3.9 with $\mathbf{A_{eq}} = []$, $\mathbf{b_{eq}} = []$.

```
    Aeq = [];
    beq = [];
    mu = [d; e; f];
    b_ext_tilde = b_ext - A_ext*mu;
    opts = optimset('Algorithm','interior-point-convex');
    [v,F] = quadprog(H,h,A_ext,b_ext_tilde,[],[],[],[],[],opts);
    v = v + mu;
    fsb = feasible(M,v,b);
end
```

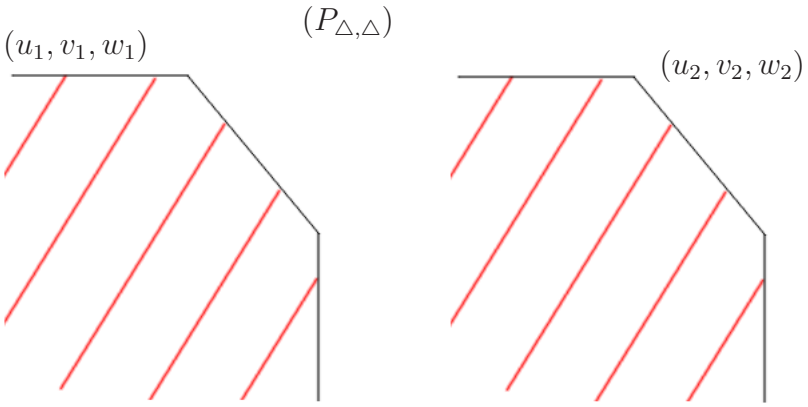

Figure 3.9: sub-solver $(P_{\triangle,\triangle})$

The counterpart to $(P_{\triangle,\triangle})$ would be $(P_{\overline{1:K},\overline{1:K}})$. This variation leaves out the hull volumes and focuses only on the plain pairs $\mathbf{G_i}, \mathbf{G_j}$ referring to the equality contraint 3.10.

```
function [v,F,fsb,res_mat] = solve_ss(A,B,C,d,e,f,M,b)
```

```
H = blkdiag(A,B,C);
h = zeros(6,1);
best_fsb = -1;
best_F = 9999.9;
best_v = [ 0 0 0 0 0 0];
for i=1:s(1)
    for j=1:s(1)
        Aeq_ext_left = [M(i,1) 0 M(i,2) 0 M(i,3) 0] ;
        beq_left = b(i);
        Aeq_ext_right = [0 M(j,1) 0 M(j,2) 0 M(j,3)] ;
        beq_right = b(j);
```

As seen in the case of the brute force approach, the matrix describing the equality constraint may represent a maximum of one equation per vector component (3.23) 3.10

```
        Aeq_ext = [Aeq_ext_left ; Aeq_ext_right];
        beq_ext = [beq_left ; beq_right];
        A = [];
        b = [];
        mu = [d; e; f];
        beq_ext_tilde = beq_ext - Aeq_ext*mu;
        opts = optimset('Algorithm','interior-point-convex');
        [v,F] = quadprog(H,h,A,b,Aeq_ext,beq_ext_tilde,[],[],[],opts);
        v = v + mu;
        fsb = feasible(M,v,b);

        if(fsb == 1 && best_F > F)
            best_F = F;
            best_v = v;
            fsb = 1;
        end

    end
```

```
    end
    v=best_v;
    F=best_F;
    fsb=best_fsb;
end
```

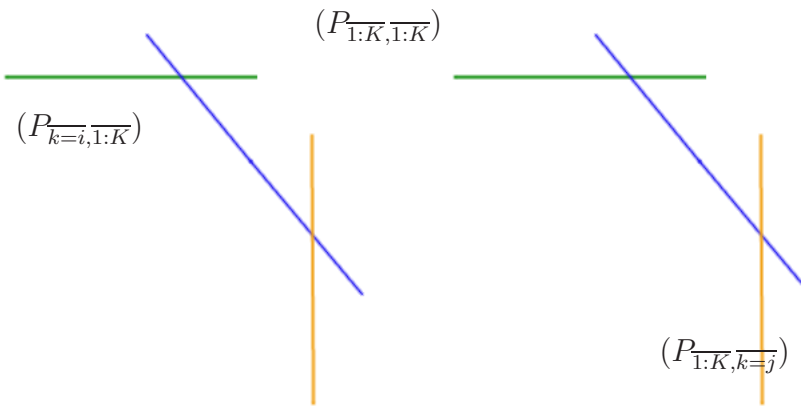

Figure 3.10: sub-solver $(P_{\overline{1:K},\overline{1:K}})$

The other two solvers brought up by the decomposition of $(P_{1:K,1:K})$ are mixtures of equality and inequality constraint based matrices $\mathbf{A}, \mathbf{A_{eq}}$. What keeps them apart, is to which vector component they contribute to. For example, $(P_{\triangle,1:K})$ consideres the first vector component contribution dealing with the hull volume, while the second one going into the equality constraint (3.18) 3.13. As a result, while the inequality ingredient of the first vector component stays constant, we iterate over all planes to find the best constellation for the second constraint 3.12.

```
function [v,F,fsb] = solve_fs(A,B,C,d,e,f,M,b)
    H = blkdiag(A,B,C);
    s = size(M);
    h = zeros(6,1);
```

```
    best_fsb = -1;
    best_F = 9999.9;
    best_v = [ 0 0 0 0 0 0];
    for i=1:s(1)
        A = [M(:,1) zeros(s(1),1) M(:,2) zeros(s(1),1) M(:,3) \
                    zeros(s(1),1)] ;
        b = b;

        Aeq = [0 M(i,1) 0 M(i,2) 0 M(i,3)];
        beq = b(i);

        mu = [d; e; f];
        b_tilde = b - A*mu;
        beq_tilde = beq - Aeq*mu;
        opts = optimset('Algorithm','interior-point-convex');
        [v,F] = quadprog(H,h,A,b_tilde,Aeq,beq_tilde,[],[],[],opts);
        v= v + mu;

        A_cmb = [A; Aeq];
        b_cmb = [b_tilde; beq_tilde];
        fsb = feasible(M,v,b);
        if(fsb == 1 && best_F > F)
            best_F = F;
            best_v = v;
            best_fsb = 1;
        end
    end
    F=best_F;
    v=best_v;
    fsb= best_fsb;

end
```

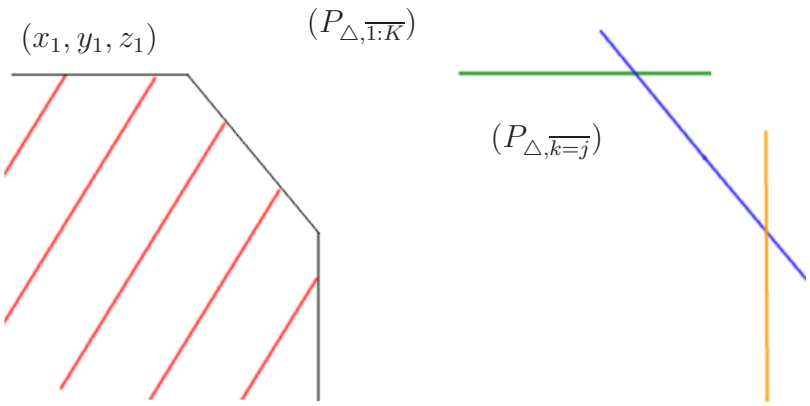

Figure 3.11: sub-solver $(P_{\triangle,\overline{1:K}})$

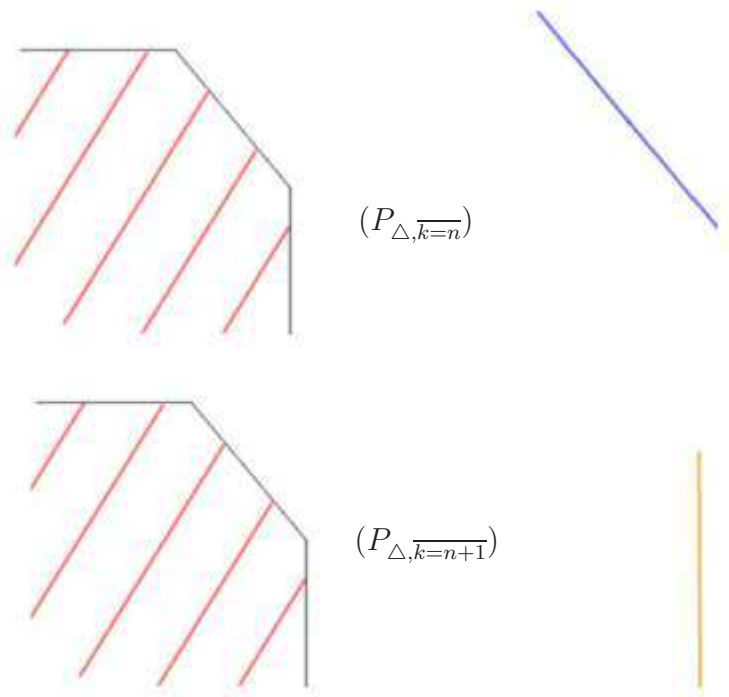

Figure 3.12: sub-solver $(P_{\triangle,\overline{k}})$ while iterating over $\mathbf{k}$.

If we swap the vector constraint components of $(P_{\triangle,\overline{1:K}})$, then the solver for the last sub problem $(P_{\overline{1:K},\triangle})$ emerges 3.13. Here, the first equality constraint is described with the matrix A_{eq} containing, only one plane entry: $\begin{pmatrix} M_{i,1} & 0 & M_{i,2} & 0 & M_{i,3} & 0 \end{pmatrix}$ with $\mathbf{i} \in \mathbf{1} : \mathbf{K}$ the index of the plane segment being currently estimated and $\mathbf{K}$ defining the number of plane segments overall.

```
function [v,F,fsb] = solve_sf(A,B,C,d,e,f,M,b)
    H = blkdiag(A,B,C);
    h = zeros(6,1);
    best_fsb = -1;
    best_F = 9999.9;
    best_v = [ 0 0 0 0 0 0];
    for i=1:num_planess(1)
        A = [zeros(num_planes,1) M(:,1) zeros(num_planes,1) \
                M(:,2) zeros(num_planes,1) M(:,3)] ;
        Aeq = [M(i,1) 0 M(i,2) 0 M(i,3) 0];
        beq = b(i);
        mu = [d; e; f];
        b_tilde = b - A*mu;
        beq_tilde = beq - Aeq*mu;
        opts = optimset('Algorithm','interior-point-convex');
        [v,F] = quadprog(H,h,A,b_tilde,Aeq,beq_tilde,[],[],[],opts);
        v= v + mu;
        A_cmb = [A; Aeq];
        b_cmb = [b_tilde; beq_tilde];
        fsb = feasible(M,v,b);
        if(fsb == 1 && best_F > F)
            best_F = F;
            best_v = v;
            best_fsb = 1;
        end
    end
    v= best_v;
    F= best_F;
    fsb = best_fsb;
end
```

Chapter 4

Evaluation

In order to prove the efficiency of the introduced solver 3.7, a set sequence of tests has been conducted. When compared to the brute-force solution, the optimized solved gained, on average, about 50% in terms of computational speed. An exemplary set of results is show in Tab. 4.1.

	best	average	worst
brute-force	39.7714 sec	45.8117 sec	50.0745 sec
extended solver	27.8301 sec	29.766 sec	31.937 sec

Table 4.1: Runtime comparison between the brute force approach and the extended solver.

Chapter 5

Conclusion

As expected, the proposed solver has proven to have a lower computation time in comparison to the brute force approach while providing reliable results. Even a minor advantage in terms of a single iteration of the solver is a vast improvement from the overall point of view. We have to consider that the optimization problem has to be solved several million times during each step of the Viterbi decoding algorithm. In terms of future work, it is advised to examine the influence of the transformation matrix $\mathbf{S}$ on the computational time. It is still uncertain, whether a specific set of matrix values could prove beneficial. Nevertheless, we expect that with an increase in size of the transformation matrix, the improvement between the brute-force approach and the extended solver, will also increase substantially.

List of Figures

List of Tables

Bibliography

[HRR13] J. R. Hershey, S. J. Rennie, and J. Le Roux. *Factorial models for noise robust speech recognition*. In T. Virtanen, R. Singh, and B. Raj, editors, Techniques for noise robustness in automatic speech recognition, pages 311¿345. John Wiley & Sons, 2013.

[JRW87] Biing-Hwang Juang, L. Rabiner, and J. Wilpon, *On the use of bandpass liftering in speech recognition*. Acoustics, Speech and Signal Processing, IEEE Transactions on, 35(7):947954, 1987.

[Maas14] R. Maas, *Uncertainty Decoding for Reverberation-Robust Speech Recognition*. unpublished PhD thesis draft, University of Erlangen-Nuremberg, 2014.

[Mathworks] http://www.mathworks.de/de/help/optim/ug/quadprog.html

[MWST11] R. Maas, M. Wolf, A. Sehr, C. Nadeu, and W. Kellermann. *Extension ofthe REMOS concept to frequency-filtering-based features for reverberationrobust speech recognition*. In Proceedings IEEE Workshop on Hands-free Speech Communication and Microphone Arrays (HSCMA), pages 13¿18, 2011.

[NHG95] Climent Nadeu, Javier Hernando, and Monica Gorricho. *On the decorrelation of filter-bank energies in speech recognition*. In Eurospeech, volume 95, page 13811384, 1995.

[NMH01] Climent Nadeu, Duvsan Macho, and Javier Hernando. *Time and frequency filtering of filter-bank energies for robust HMM speech recognition*. Speech Communication, 34(1):93114, 2001.

[RNS06] C. K. Raut, T. Nishimoto, and S. Sagayama. *Model adaptation for long convolutional distortion by maximum likelihood based state filtering approach*. In Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 1, 2006.

[Sehr09] A. Sehr, Reverberation modeling for robust distant-talking speech recognition. PhD thesis, University of Erlangen-Nuremberg, 2009.

[SHMK10] A. Sehr, E. A. P. Habets, R. Maas, and W. Kellermann. *Towards a better understanding of the effect of reverberation on speech recognition performance*. In Proceedings International Workshop on Acoustic Echo and Noise Control (IWAENC), page 105106, 2010.

[YSD12] T. Yoshioka, A. Sehr, M. Delcroix, K. Kinoshita, R. Maas, T. Nakatani, and W. Kellermann. *Making machines understand us in reverberant rooms: robustness against reverberation for automatic speech recognition*. IEEE Signal Processing Magazine, 29(6):114¿126, 2012.

SHORT RESUME

Marek Malewicz, born on the 18th of January, 1985, in Nowa Sol (Poland).

$*$2004 Abitur, Gymnasium Annenschule Gymnasium Goerlitz (Germany).

$*$2006 Associate Engineer Degree, Siemens Technical Academy in Erlangen (Germany).