Statistical Signal Processing Techniques for Robust Speech Recognition

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultät II
- Physik und Mechatronik -
der Universität des Saarlandes

von

Friedrich Faubel

Saarbrücken
2013

Tag des Kolloquiums:	19.06.2015
Dekanin/Dekan:	Univ.-Prof. Dr. Georg Frey
Mitglieder des Prüfungsausschusses:	Univ.-Prof. Dr. Dietrich Klakow, Univ.-Prof. Dr. Reinhold Haeb-Umbach, Prof. Steve Renals, Univ.-Prof. Dr. Karsten Kruse (Vorsitz), Dr. Tilman Sauerwald

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Ort, Datum	(Unterschrift)

Abstract

Automatic speech recognition is becoming increasingly more important, with commercial applications such as call steering, dictation or voice-enabled personal assistance systems. Although successful in many respects, the performance of such systems can significantly degrade in noisy environment such as a crowded restaurant. This is due to the fact that noise introduces a mismatch between the clean speech features, which the ASR system has been trained with, and the noisy speech features that are encountered in the operational environment.

This dissertation tries to mitigate the degradation in performance using two principally different approaches: speech feature enhancement (SFE) techniques, which minimize the mismatch between clean and noisy features, and missing feature reconstruction (MFR) techniques, which infer the values of noise-corrupted frequency bins from non-corrupted ones. Particular contributions include (1) a phase-averaged model of how noise corrupts clean speech features, (2) better noise estimation with a Monte Carlo variant of the expectation maximization algorithm, (3) an adaptive level of detail transform that allows for more accurate transformations of Gaussian random variables, and (4) a bounded conditional mean imputation technique.

In addition to the above, it is shown that both SFE and MFR techniques can be derived within the same mathematical framework, just using different models of how noise corrupts clean speech features.

Deutsche Zusammenfassung

Automatische Spracherkennung nimmt einen zusehends wichtigeren Stellenwert ein. Kommerzielle Anwendungen beinhalten Call Steering, Diktieren und sprachgesteuerte Assistenzsysteme. Obwohl derartige Anwendungen durchaus erfolgreich sein können, so leiden sie doch an der Tatsache, dass sich die Spracherkennungsgenauigkeit in geräuschbehafteten Umgebungen verschlechtert. Das rührt daher, dass Hintergrundgeräusche eine Unstimmigkeit zwischen klaren Sprachmerkmalen im Training und geräsuchbehafteten Merkmalen im Einsatz verursachen.

Diese Dissertation untersucht zwei verschiedene Herangehensweisen an dieses Problem: Methoden zur Sprachmerkmalsverstärkung (SMV), welche Unstimmigkeiten zwischen Merkmalen minimieren, und Methoden zur Vervollständigung fehlender Merkmale (VFM), welche stark geräuschgestörte Frequenzen mittels weniger gestörter Frequenzen restaurieren. Spezifische Beiträge umfassen: (1) ein phasengemitteltes Modell dafür, wie Geräusche klare Sprachmerkmale korrumpieren, (2) verbesserte Geräuschschätzung durch einen Monte Carlo Expectation Maximization Algorithmus, (3) genauere Transformationen gaußscher Zufallsvariablen durch einen adaptiven Detailgrad, (4) eine Vervollständigungstechnik, die auf dem beschränkten, bedingten Mittelwert beruht.

Zusätzlich zu obigem wird gezeigt, dass SMV und VFM Methoden sich im gleichen mathematischen Rahmenwerk herleiten lassen, nur eben unter Verwendung verschiedener Modelle für die Korrumpierung von Sprachmerkmalen.

Acknowledgements

In the course of this thesis, I accumulated a debt of gratitude to many people, first and foremost, my advisors Dietrich Klakow and John McDonough who were always helpful and supportive and who grew to be my friends over the last couple of years. Thanks are also due to my colleagues Kenichi Kumatani, Barbara Rauch and Andreas Beschorner, who were fun to work with and who made my stay here an enjoyable experience. Not to forget the countless students I advised in the recent years: Alberto Friedmann, Caroline Ebert, Christian Siegwart, David Turner, Hassan El-Soumsoumani, Jens Hausen, Michael Braun, Munir Georges, Najib Hadir, Raja Hamza, Rahil Mahdian Toroghi, Todd Shore, Youssef Oualil, and Yury Bakanouski. They made me feel a bit like in a large family crowded with smart and curious “kids”.

Apart from the above people, I am indebted to Steve Renals, who kindly agreed to have me in Edinburgh for a four month research stay. To Mark Gales, Hugo van Hamme as well as some anonymous reviewers, whose comments either helped to improve the quality of submitted papers or encouraged new directions of research. Sometimes it was simple suggestions that motivated a new paper, such as “Why do you do this so complicated? Just use a Monte Carlo approach!”.

I am also grateful to my financial sponsors: the German Research Council (DFG), which granted me a PhD scholarship in the framework of the International Research Training Group (IRTG) 715 “Language Technology and Cognitive Systems”; and the Cluster of Excellence for Multimodal Computing and Interaction (MMCI) where I was a research associate over the last few years.

Finally, I would like to thank my parents for their love and support. They were always there when I needed them. And they are greatly responsible for the course my life has taken, teaching me to write Basic language computer programs at the age of 8, showing me how to build a joystick of wood and metal parts, or making me watch protozoa through a microscope. This genuinely ignited my enthusiasm in science and technology.

Contents

1 Introduction

In the last decade, automatic speech recognition (ASR) technology has advanced to the point where economically viable real world applications are possible. Commercially successful products include interactive voice response (IVR) systems for customer self-service or call steering, transcription services for medical records (using e.g. Nuance SpeechMagic^{TM} or M*Modal's SpeechQ) as well as direct voice input (DVI) for high-performance fighter aircraft (e.g. the Eurofighter Typhoon). There are other areas of application such as voice indexing of multimedia files, quality monitoring for call centers, voice search and and dictation. Nonetheless, ASR technology has so far failed to be accepted as a general user interface for computer systems [1, 2]- The reason identified in [1] is that

> *"Speech systems introduce an additional layer between the user and their understanding of the application because of the step of interpreting the acoustic signal as text."*

This can cause confusion on the user-side about how to appropriately convey their intent. If the system does not respond as expected, users are often puzzled whether they should (1) pronounce words more clearly, (2) simplify the sentence to ease interpretation, or (3) give more details in order to specify the intent [3]. Of course there are ways to cope with these problems, such as giving appropriate user feedback [1], making use of context-awareness [2], or letting the user repeat a request when the ASR confidence score is too low [3]. Appropriate training also seems to make a big difference [4]. But it is usually impeded by the fact that home users are not willing to spend up to 50 hours of effort before enjoying the benefits of a new technology. This stands in contrast to professional users, e.g. in the healthcare sector [4, 5].

Nonetheless, ASR technology is now finding its way into home consumer products, in particular since the introduction of *Siri*[1] – Apple's voice-enabled electronic personal assistant that seems to solve some of the above mentioned issues. A remaining problem is that of bad ASR accuracy in noisy environments, which, among other things, triggers *Siri* to give answers like "I'm not sure what you said" or "I don't know what you mean by [...] but I can search the web for you". This worsens if the user does not speak directly into the microphone but speaks from some distance, e.g. in order to see the information shown on the display. According to [3],

> *"Fausto Marasco, CEO of Premier Technologies, indicated that his company's market research showed that half of all complaints about spoken language dialogue systems concerned recognition errors."*

Although Lee and Lai [6] arrived at a much more positive conclusion, only 64% of their users responded that their utterances were understood either very well or somewhat well (at a recognition error rate of 20 to 25%). These numbers can be expected to look worse in a noisy environment, where a recognition error rate of under 1% for clean speech digits can easily increase to 66% at 5 dB car noise [7].

1.1 The Approach Taken in This Thesis

The work in this thesis tries to reduce the error rate in noisy environments to a more reasonable number, ideally below the threshold of 15% where spoken dialogue systems are considered usable [6]. Research in this direction started about thirty years ago, with classical frequency domain noise reduction approaches such as spectral subtraction [8] and Wiener filtering [9, 10], which estimate the clean speech signal in a minimum mean square error fashion. More recent work, such as Acero's codeword-dependent cepstral normalization (CDCN) [11], has tried to optimize noise reduction specifically for speech recognition. Picking up on this idea, this thesis takes a Bayesian approach, which

1. uses prior knowledge of how clean speech "looks like" in order to more accurately suppress the noise.

2. estimates noise based on the clean speech prior, using e.g. the expectation maximization (EM) algorithm or a particle filter.

3. works on log-Mel spectra (see Section 5) in order to perform noise reduction in a domain that is of relevance to speech recognition.

Next to elaborating on the theoretical underpinnings of this approach, including a transformation-centric view of Bayesian state estimation, this work also extends the existing theory. That is accomplished by introducing an adaptive level of detail transform [12, 13] which more

[1] see e.g. `http://en.wikipedia.org/wiki/Siri_(software)`

accurately treats nonlinearities. The probably most intriguing aspect of the work is a unified derivation of two major noise reduction approaches that have so far been considered independently:

1. ***speech feature enhancement techniques***, which try to stochastically map noisy speech to clean speech features (see Section 6).
2. ***missing feature reconstruction techniques***, which try to infer the values of noise corrupted parts of the spectrum from non-corrupted ones (see Section 7).

It is shown in particular that both these approaches can be viewed as a special case of Bayesian estimation theory, using the same equations, just different models of how noise affects clean speech features. The most important practical contributions include the development of a phase-averaged model for how noise corrupts clean speech features [14], improvements to noise estimation algorithms [15, 16] as well as the bounded conditional mean imputation technique from [17].

1.2 Chapter-by-Chapter Review

The following again gives a chapter-by-chapter review of this thesis with a more complete list of the individual contributions.

Chapter 2 starts with a brief introduction to basic concepts of probability theory, including a derivation of the fundamental transformation law of probability, conditional and marginal Gaussian distributions, the expectation maximization algorithm and Monte Carlo methods. Where possible, we also try to cover the historical development.

Chapter 3 forms one of the central chapters of this thesis. It shows how Bayesian state estimation can be viewed as a two step procedure, in which the first step predicts the joint distribution of state and observation, and in which the second step conditions the predicted distribution on the received observation (Section 3.1). As the first step requires a possibly nonlinear transformation of random variables (for which there mostly is no analytical solution), we try to find suitable approximations to such transformations. This leads to new approaches such as the extensive unscented transform [18], which tries to preserve statistical independence of the random variables (see Section 3.6), and the adaptive level of detail transform (ALoDT) [12, 13], which adapts the number of Gaussian components in a mixture to minimize the total linearization error during transformation (see Section 3.7). This involves new techniques for estimating the degree of nonlinearity as well as for splitting Gaussians in regions of high nonlinearity. The performance of the proposed methods is finally evaluated in Section 3.8.

Chapter 4 takes the Bayesian state estimation framework to the sequential level. This naturally leads to tracking algorithms such as Kalman, Gaussian mixture and particle filters, which are here all derived from a transformation-centric point of view (Sections 4.4 - 4.7). Apart from theoretic contributions such as a general principle behind sequential Bayesian estimation (see Section 4.3), the chapter introduces two new nonlinear tracking algorithms, namely: the extensive unscented Kalman filter [18] (4.5) and the adaptive Gaussian mixture filter [19, 16] (Section 4.6.3). Section 4.8 concludes with a performance comparison.

Chapter 5 briefly explains the feature extraction steps of modern ASR systems and then derives a phase-averaged model (interaction function) for the effect of noise to clean speech features [14] (Section 5.2). This allows us to predict noisy speech features in a minimum mean square error (MMSE) fashion and therewith more accurately than with previous models that have been used in the literature (see Section 5.2.5 for an experimental comparison). The chapter closes with a derivation of the inverse phase-averaged model, which essentially provides a noise suppression rule in the feature domain (see Section 5.3).

Chapter 6 uses the Bayesian state estimation framework to suppress background noise in speech features. This is achieved by first constructing the joint distribution of clean and noisy speech, as described in Section 6.3, and then conditioning on the observed noisy speech feature. The result is a minimum mean square error clean speech estimate (see Section 6.3.1), which can efficiently be approximated as explained in Section 6.3.2. The remaining part of the chapter explores different noise estimation techniques with a particular focus on expectation maximization (EM) based approaches. This includes: the derivation of a general EM algorithm for noise estimation [15, 16] (Section 6.4.1), a Gaussian mixture implementation [15] (Section 6.4.4), a Monte-Carlo implementation [16] (Section 6.4.5), as well as theoretical comparisons to other approaches that have been used in the literature (Section 6.4.3). The last section of the chapter presents a particle filter based noise tracking algorithm that avoids stability issues due to the relative phase [14] (Section 6.5.3).

Chapter 7 presents an alternative approach to removing noise from noisy speech features. It is based on the fact that noise actually masks those portions of the clean speech spectrum in which noise is significantly stronger than speech. Consequently, heavily noise-corrupted regions are assumed to be missing, and missing data theory (MDT) is used to reconstruct these regions based on a prior model of how clean speech “looks like”. The chapter starts with a brief review of classical methods, in Section 7.1, and then proceeds with extending them by (1) a bounded reconstruction technique [17] (Section 7.3), which makes use of the fact that the missing part is actually bounded above by the noise, (2) the introduction of box-truncated Gaussian distributions along with approximations to their means and normalizing constants [17] (Section 7.3.1) as well as (3) a particle filter based mask estimation technique [20] (Section 7.4). Section 7.2 shows how MDT techniques are related to the speech feature enhancement approach from Chapter 6.

Chapter 8 evaluates the performance of the proposed feature enhancement and reconstruction techniques by means of automatic speech recognition experiments. This is achieved by (a) extracting speech features from a noisy speech corpus, (b) cleaning the speech features with the noise reduction methods to be evaluated, and then (c) comparing the resulting word error rates (WERs) to those obtained with a baseline recognizer (without noise reduction). The most important outcomes are:

1. under ideal conditions[2], the bounded conditional mean imputation (BCMI) technique from Section 7.3 outperforms all other feature domain noise reduction approaches at a relative improvement of 45% in WER compared to the baseline (see Section 8.4.4).

2. the speech feature enhancement approach from Section 6.3 achieves almost the same performance – i.e. a 42% reduction in WER – without making use of prior knowledge about the noise[3] (see Section 8.4.7).

3. both approaches reduce the total computing time of speech recognition, as they facilitate stronger pruning in the decoder (see Sections 8.4.4. 8.4.6 and 8.4.7).

4. noise estimation with the Monte-Carlo (MC) EM implementation (Section 6.4.5) comes very close to perfectly knowing the noise (see Sections 8.4.6 and 8.4.7).

In addition to these results with artificially added noise, we show that speech feature enhancement also works in a real noise environment, i.e. in a car driving along a highway with up to 55 miles per hour (Section 8.5).

Chapter 9 concludes this thesis by giving a short summary of the main results. This is followed by a brief outline of the remaining problems along with possible extensions.

[2] i.e. when it is perfectly known, which parts of the speech spectrum are masked by noise
[3] meaning this result is much more realistic than the BCMI result from above

2 Basics

As a motivation for the use of probabilistic methods in the framework of this thesis, this chapter starts with the following quote from Pierre Simon Laplace's "A Philosophical Essay on Probabilities" [21]:

> *"The present state of the universe ought to be regarded as the effect of its anterior state and as the cause of the one which is to follow. Hence, given for one instant an intellect which could comprehend all the forces by which nature is animated and the respective situation (state) of all items of which nature is composed – an intellect so vast that it could also submit these data to analysis – it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes."*

The "intellect" of which Laplace is speaking in this paragraph is often referred to as "Laplace's demon". But as mythical creatures seem a little antiquated in the modern world of today, one might rather consider it a giant supercomputer, which has once been initialized to a specific start state – before the big bang – and which is now, ever since, simulating the universe. Interestingly, Laplace does not stop at this point but rather explains why, nevertheless, to us everything seems probabilistic [21]:

> *"All these efforts in the search for truth (discovery of the laws of physics) tend to lead humanity back continually to the vast intellect which we have just mentioned, but from which it will always remain infinitely removed."*

So, Laplace is saying that the limited human mind is incapable of such an intellect; that we perceive only a small fraction of the universe, of which the unseen parts inevitably introduce uncertainty. This gives a very strong motivation for the use of probabilistic models and it puts Laplace not only into the often alleged role as a proponent of determinism, but also into the role of a strong proponent for a probabilistic description of the universe [21]:

> *"Strictly speaking it may even be said that nearly all our knowledge is problematical; ... even in the mathematical sciences themselves, the principal means for ascertaining truth – induction and analogy – are based on probabilities;"*

In particular, Laplace attributes the existence of probabilities not to an intrinsically haphazard universe but to a lack of knowledge in the eye of the beholder. The same, of course, applies to technical devices, such as computers, which motivates the use of statistical signal processing techniques in the framework of this thesis. The details are explained in more detail in the following, starting with an introduction to the basic concepts of probability theory.

2.1 An Introduction to Probability Theory

The following sections give a brief introduction to the concepts of probability theory, starting with basics like random variables and vectors but also treating more advanced concepts such as hidden variables, Markov processes and the fundamental transformation law of probability.

2.1.1 Random Variables and Vectors

A *random variable* X is, as the name hints, a variable. This means X assumes values x from a given domain of definition, in this work the domain of real numbers, $\mathbb{R}$, or a subset thereof. The specifier "random" indicates that there is a likelihood associated with the event of a particular value being assumed. This likelihood is specified by a *probability density function* (pdf) p_X – a non-negative, real-valued function that integrates to 1. The concept of a pdf is useful as it allows for both evaluating the likelihood $p_X(x)$ of the variable X taking a value x and determining the probability

$$P(a \leq X \leq b) = \int_a^b p_X(x)dx \tag{2.1}$$

of X taking a value in an interval $[a,b]$. As the event of a particular value being assumed is usually the outcome of a physical process in which one of many possible alternatives is realized, an instance x is also called a *realization* of the random variable. Besides continuous random variables of the above type, we will occasionally encounter discrete random variables K with values in $\mathbb{N}$. For discrete random variables, the distribution $p_K(k)$ is called a *probability mass function.* In this case, p_K is required to sum to one, i.e. $\sum_{k\in\mathbf{N}} p_K(k) = 1$, and the probability $P(a \leq X \leq b)$ is evaluated by replacing the integral in (2.1) by a sum[1].

[1] Note that a discrete random variable K can be identified with a continuous random variable X, simply by setting $p_X(x) = \sum_{k\in\mathbf{N}} p_K(k)\delta(k-x)$ where δ denotes the Dirac delta function.

With this defined, it is straightforward to move on to random vectors. A *random vector* X is a multivariate, i.e. vector-valued random variable $[X_1 \cdots X_n]$. It takes values $x = [x_1 \cdots x_n]$ in an n-dimensional real vector space, $\mathbb{R}^n$, according to a multivariate probability density function $p_X : \mathbb{R}^n \to (\mathbb{R}^+ \cup \{0\})$ that allows for the calculation of the probability of x being in the area $\mathscr{A}$:

$$P(x \in \mathscr{A}) = \int_{\mathscr{A}} p_X(x)dx.$$

In analogy to the univariate case, $p(x)$ must be ≥ 0 and the integral over the total space $[-\infty, \infty] \times \cdots \times [-\infty, \infty]$ must be 1. In the following, the terms random variable and random vector will be used interchangeably. All random variables will be either real-valued or discrete. In order to unify the notation, both probability density functions and probability mass functions will be called *distributions* as it is often done by engineers.

2.1.2 Marginal and Conditional Random Variables

Marginal and conditional random variables naturally arise when the realization of a multivariate (i.e. vector-valued) random variable is in part observable only. To explain this in a more formal setting, let $Z = [Z_1, \ldots, Z_m]^T$ be an m-dimensional random variable with distribution $p_Z(z) = p_{Z_1,\ldots,Z_m}(z_1, \ldots, z_m)$. Further assume that only a part of the realization $[z_1, \ldots, z_m]^T$ – without loss of generality $[z_{n+1}, \ldots, z_m]^T$ with $n < m$ – is observable, as portrayed below:

$$Z = \Big[\underbrace{\boxed{Z_1 \cdots Z_n}}_{\text{unobservable}} \; \underbrace{\boxed{Z_{n+1} \cdots Z_m}}_{\text{observable}}\Big]^T$$

Then the unobservable part $[Z_1, \ldots, Z_n]^T$ of Z is itself a random variable $X \triangleq [X_1, \ldots, X_n]^T$ with $X_i = Z_i$. Its distribution p_X is obtained by integrating out the observable variables $Z_{n+1}, \ldots, Z_m$:

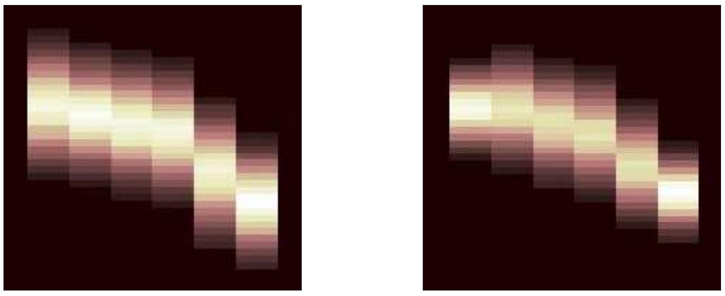
$$p_X(x) = p_{Z_1,\ldots,Z_n}(x_1, \ldots, x_n) = \int \cdots \int p_{Z_1,\ldots,Z_m}(x_1, \ldots, x_n, z_{n+1}, \ldots z_m)\, dz_{n+1} \cdots dz_m \tag{2.2}$$

The process of calculating the integral in (2.2) is called *marginalization*; p_X is called a *marginal distribution* of p_Z. In order to introduce conditional random variables, let Y be the random variable which consists of the observable components of Z, i.e. $Y = [Z_{n+1}, \ldots, Z_m]^T$. Then we obviously have $Z = [X^T \, Y^T]^T$ with both X and Y being marginal random variables of Z. Further assuming that a realization y of Y is available, the distribution of the unobservable part X can be constrained by the knowledge of y. This is achieved by fixing $Y = y$ in the joint distribution $p_{X,Y}(x, y)$ and then normalizing by dividing by $\int p_{X,Y}(x, y)dx = p(y)$:

$$p_{X|y}(x) = \frac{p_{X,Y}(x, y)}{p_Y(y)} = \frac{p_Z\left(x_1, \ldots, x_n, y_1, \ldots, y_{m-n}\right)}{p_Y\left(y_1, \ldots, y_{m-n}\right)}. \tag{2.3}$$

The resulting pdf is called the *conditional distribution* of X given y. The corresponding random variable $X|y$ is referred to as a *conditional random variable*. A special case occurs if X and Y

are statistically independent – that is, $p_{X,Y}(x,y) = p_X(x) \cdot p_Y(y)$. In this case, the conditional distribution $p_{X|y}$ of X given y is obviously equal to the marginal distribution of X, i.e. $p_{X|y} = p_X$. If the variables are dependent, the conditional distribution will be "sharper" (i.e. more concentrated in some area) than the marginal one, as shown in Figure 2.1.



(a) marginal distribution. *(b) conditional distribution.*

Figure 2.1: *The image to the left shows the marginal distribution of the 3-rd to 8-th Mel frequency bins (left to right) of the phoneme "A". The image to the right shows the same distribution conditioned on an observation of the remaining bins (bins 1,2 and 9-30).*

2.1.3 Hidden Variables and Bayes Equation

In the previous section, X was considered to be a variable, which cannot directly be observed but which is statistically tied to an observable variable Y. Such a variable X is, in general, called a *hidden variable*. Its introduction might seem somewhat academic. Nevertheless hidden variables are quite relevant in practice, as physical processes can often only indirectly be observed through related measurements. In the case of a physical process, the hidden variable X corresponds to the system state – that is, the state in which the physical system resides. The associated observation variable Y corresponds to measurements related to that state.

Then, given a realized measurement $Y = y$, the likelihood that the system resides in a particular state x can be obtained by evaluating the conditional distribution $p_{X|y}(x)$. In contrast to expressing $p_{X|y}(x)$ in dependence of $p_{X,Y}(x,y)$ and $p_Y(y)$, as in the previous section, it is here expressed in terms of $p_{Y|x}(y)$ and $p_X(x)$. That is achieved by applying (2.3) in order to obtain $p_{X|y}(x) = p_{X,Y}(x,y)/p_Y(y)$ and $p_{Y|x}(y) = p_{X,Y}(x,y)/p_X(x)$, rewriting the latter equation as $p_{X,Y}(x,y) = p_{Y|x}(y)p_X(x)$, plugging it back into the first equation, and then using marginalization (2.2) to express $p_Y(y)$ in terms of $p_{Y|x}(y)$ and $p_X(x)$. These steps lead to *Bayes equation*,

$$p_{X|y}(x) = \frac{p_{Y|x}(y)p_X(x)}{p_Y(y)} = \frac{p_{Y|x}(y)p_X(x)}{\int p_{Y|x}(y)p_X(x)dx}, \tag{2.4}$$

which has drawn considerable attention in the literature, as it allows for the incorporation of prior knowledge about the distribution $p_X(x)$ of the system state. Funnily enough, it also triggered a lengthy dispute with "frequentist" mathematicians who strongly rejected priors [22].

2.1.4 Stochastic Processes

In order to account for the fact that the state of a physical system usually varies in time, the evolution of the system state might be modeled as a (discrete time) *stochastic process* – that is, a sequence $(X_t)_{t\in\mathbb{N}}$ of random variables X_t where t denotes time. In the context of a physical system, $(X_t)_{t\in\mathbb{N}}$ is called the state process. Its realizations $(x_t)_{t\in\mathbb{N}}$ describe all possible state sequences along with associated likelihoods $p_{X_0,X_1,\ldots}(x_0,x_1,\ldots)$. Corresponding measurement variables Y_t form an associated observation process $(Y_t)_{t\in\mathbb{N}}$, which is interlocked with the state process as portrayed in Figure 2.2.

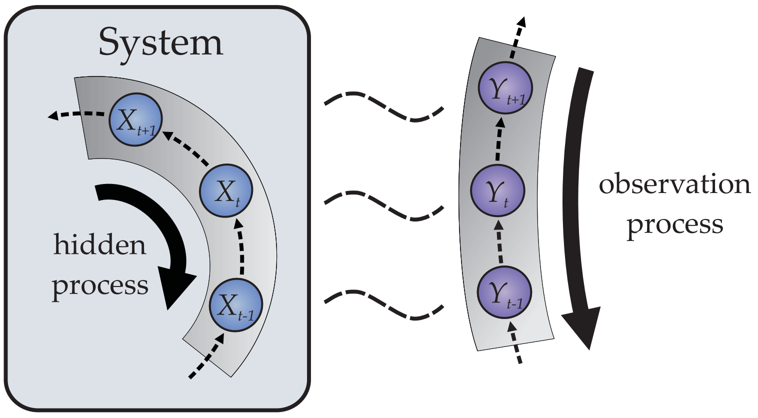


Figure 2.2: *Interlocking of the hidden state process with the observation process.*

The interlocking of the processes can be regarded to be a causal relationship in which system states cause observations. Jointly "pulling" the gray straps in Figure 2.2 in the direction indicated by the thick, black arrows shows how the doubly stochastic process evolves in time. Now given a partial realization $y_{1:\tau} \triangleq [y_0,\ldots,y_\tau]$ of the observation sequence, the likelihood that the corresponding state sequence was $x_{0:\tau} = [x_0,\ldots,x_\tau]$ can be evaluated as

$$p_{X_{0:\tau}|y_{0:\tau}}(x_{0:\tau}) = \frac{p_{X_{0:\tau},Y_{0:\tau}}(x_{0:\tau},y_{0:\tau})}{p_{Y_{0:\tau}}(y_{0:\tau})} = \frac{p_{Y_{0:\tau}|x_{0:\tau}}(y_{0:\tau})p_{X_{0:\tau}}(x_{0:\tau})}{\int\cdots\int p_{Y_{0:\tau}|x_{0:\tau}}(y_{0:\tau})p_{X_{0:\tau}}(x_{0:\tau})dx_0\cdots dx_\tau} \tag{2.5}$$

This is a natural extension of Bayes equation (2.4). Further, making use of the fact that according to (2.3) $p_{X_0,X_1}(x_0,x_1)$ can be written $p_{X_0,X_1}(x_0,x_1) = p_{X_1|x_0}(x_1)p_{X_0}(x_0)$ and then extending this factorization to three variables, $p_{X_0,X_1,X_2}(x_0,x_1,x_2) = p_{X_2|x_0,x_1}(x_2)p_{X_1|x_0}(x_1)p_{X_0}(x_0)$, and so on, it becomes clear that $p_{X_0,\ldots,X_\tau}(x_0,\ldots,x_\tau)$ in the nominator of (2.5) can be written

$$p_{X_{0:\tau}}(x_{0:\tau}) = \prod_{t=0}^{\tau} p_{X_t|x_{0:t-1}}(x_t) \quad \text{with} \quad p_{X_0|x_{0:-1}} \triangleq p_{X_0}.$$

2.1.5 The Hidden Markov Process

General processes are difficult to work with, as all the variables are dependent on each other. Therefore simplifying assumptions are being made, such as the Markov assumption, which consists in the idea that the current state contains all the information necessary for predicting the next all the information necessary for predicting the next state. This implies that the past trajectory is irrelevant. In other words, given the current state x_t the next state x_{t+1} is dependent on x_t only, not on $x_{0:t-1}$. Put into equations this gives $p_{X_{t+1}|x_{0:x_t}}(x_{t+1}) = p_{X_{t+1}|x_t}(x_{t+1})$ or, equivalently,

$$p_{X_{0:\tau}}(x_{0:\tau}) = \prod_{t=0}^{\tau} p_{X_t|x_{t-1}}(x_t), \tag{2.6}$$

for all τ. A process for which (2.6) holds is called a *Markov process*. In order to also simply the relationship $p_{Y_{0:\tau}|x_{0:\tau}}(y_{0:\tau})$ between the state and observation processes, it is common to use the *output independence assumption*. This assumption states that, given the current state x_t, the current observation y_t is independent of all other states and observations:

$$p_{Y_{0:\tau}|x_{0:\tau}}(y_{0:\tau}) = \prod_{t=0}^{\tau} p_{Y_t|x_t}(y_t). \tag{2.7}$$

With this, we can now define the *hidden Markov process* as a Markov process $(X_t)_{t\in\mathbb{N}}$ whose stochastic relationship $p_{Y_{0:\tau}|x_{0:\tau}}(y_{0:\tau})$ to the associated observation process $(Y_t)_{t\in\mathbb{N}}$ follows the output independence assumption (2.7).

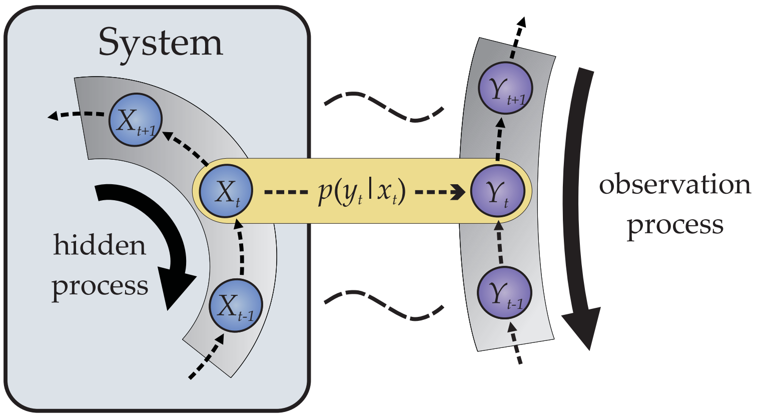


Figure 2.3: *Hidden Markov Process. The output independence assumption is indicated by yellow coloring.*

2.1.6 The Fundamental Transformation Law of Probability

In the context of hidden variables, it can be useful to "transform" random variables. Consider the case, for example, where X is a random variable whose distribution $p_X(x)$ is known and Y is a random variable whose distribution is not known. Then given there is a functional relation $Y = f(X)$ between X and Y the distribution $p_Y(y)$ of Y can be obtained as follows:

$$p_Y(y) = p_X\left(f^{-1}(y)\right)\left|\det\left(\frac{df^{-1}(y)}{dy}\right)\right|. \tag{2.8}$$

In this equation, which is generally referred to as the *fundamental transformation law of probability* [23], $f^{-1}(y)$ is the origin of y and the multiplication by the absolute Jacobian determinant $|\det(df^{-1}(y)/dy)|$ is due to the change of variables, from $x = f^{-1}(y)$ to y. The explanation for why the above constitutes a change of variables comes by going one step further – to the calculation of a probability according to (2.1): $P_Y = \int_{\mathscr{A}} p_Y(y)d_y$. The point is that the pdf has no meaning outside the context of being that function which is integrated in order to calculate a probability. And that is where the integral occurs, due to which we have the change of variables. Things become a bit more complicated if the transformation is not a monotonic function. In this case, f has to be decomposed into a sum of monotonic functions $f(x) = \sum_{i=1}^{n} f_i(x)$ which are defined on disjunct areas $\mathscr{A}_i$: $f_i = f|_{\mathscr{A}_i}$. Then, instead of transforming the variable according to f, it is transformed individually for each f_i, which results in n non-normalized probability density functions $p_Y^{(i)}(y)$, $i = 1,\ldots,n$. In case $f_i^{-1}(y)$ is the empty set, $p_Y^{(i)}(y)$ is set to zero. Subsequently, the total distribution of the transformed random variable can be recovered as the sum over all $p_Y^{(i)}(y)$:

$$p_Y(y) = \sum_{i=1}^{n} p_Y^{(i)}(y) \quad \text{with} \quad p_Y^{(i)}(y) \triangleq \begin{cases} p_X\left(f_i^{-1}(y)\right)\left|\det\left(\frac{df_i^{-1}(y)}{dy}\right)\right|, & f_i^{-1}(y) \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

2.2 Probability Distributions

In order to work with random variables, it is necessary to specify the probability distributions according to which the variables assume their values. Ideally the distributions can be derived from theoretical considerations about the underlying physical process. But even if this not possible the true distribution can be approximated with a parametric standard distribution whose parameters are estimated on a data set. The following sections introduce the distributions, which are used in this thesis, along with the concept of moments - a powerful tool for describing the characteristics of a probability distribution.

2.2.1 Moments and Expectations

It often is of interest to characterize the shape of a probability distribution p_X. This can be achieved by calculating its *moments*, which – for the case of an n-dimensional, multivariate distribution – are defined as

$$\mathscr{M}_X^{(k)}(i_1,\ldots,i_k) = \int \prod_{j=1}^{k} x_{i_j} p_X(x)dx \tag{2.9}$$

for $k = 1,\ldots,\infty$. The k-th moment $\mathscr{M}_X^{(k)}$ consists of all the integrals $\mathscr{M}_X^{(k)}(i_1,\ldots,i_k)$, $i_1,\ldots,i_k \in \{1,\ldots,n\}$, of the distribution p_X multiplied by any possible monomial of order k, $\prod_{i=1}^{n} x_i^{k_i}$ with $k_i \in \mathbb{N}$ and $\sum_{i=1}^{n} k_i = k$. The first moment $\mathscr{M}_X^{(1)}(i_1) = \int x_{i_1} p_X(x)dx$ is called the mean. It indicates the center of the probability distribution; and it is usually written $\mu =$

$\left[\mathscr{M}_X^{(1)}(1) \cdots \mathscr{M}_X^{(1)}(n)\right]^T$. After calculation of the mean, the other moments can be expressed in their centralized form,

$$\tilde{\mathscr{M}}_X^{(k)}(i_1,\ldots,i_k) = \int \prod_{j=1}^{k} \left(x_{i_j} - \mu_{i_j}\right) p_X(x)dx, \tag{2.10}$$

which is invariant under translations of the distribution. The second centralized moment $\tilde{\mathscr{M}}_X^{(2)}$ is called variance for $n = 1$; it is called covariance matrix for $n > 1$; and it specifies the spread of the probability mass around the mean of the distribution as well as the correlation between the dimensions. The name covariance matrix stems from the notation

$$\Sigma = \begin{bmatrix} \mathscr{M}_X^{(2)}(1,1) & \cdots & \mathscr{M}_X^{(2)}(1,n) \\ \vdots & \ddots & \vdots \\ \mathscr{M}_X^{(2)}(n,1) & \cdots & \mathscr{M}_X^{(2)}(n,n) \end{bmatrix},$$

with off-diagonal elements describing covariances $\sigma_{i,j} = \mathscr{M}_X^{(2)}(i,j)$ and with diagonal elements describing variances $\sigma_i^2 = \mathscr{M}_X^{(2)}(i,i)$. In general, the moments $\{M_k | k = 1,\ldots,\infty\}$ do not uniquely determine the distribution, unless the distribution has bounded support [24, 25]. The concept of moments can be generalized to expectations – that is, the expected values $\mathrm{E}_{p_X}\{h(x)\}$ a random variable X has under an arbitrary function $h(x) = \left[h_1(x) \quad \cdots \quad h_m(x)\right]$:

$$\mathrm{E}_{p_X}\{h(x)\} = \int h(x) p_X(x)dx. \tag{2.11}$$

Calculating an expectation $\mathrm{E}_{p_X}\{h(x)\}$ essentially consists of averaging $h_i(x)$ for all possible values x, weighted with the likelihood $p(x)$ that the value is taken.

2.2.2 Empirical and Weighted Empirical Distributions

The most natural way of representing a probability density function probably is to use an the *empirical distribution*. That is because the empirical distribution directly reflects the samples, which have been obtained through the repetition of a physical process, by placing equal probability mass on each of the samples $x^{(i)}$, $i = 1,\ldots,N$:

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} \delta\left(x - x^{(i)}\right) \tag{2.12}$$

with δ denoting the Dirac delta. Obviously, this representation exactly captures the sample distribution. But it does not generalize as a second sample distribution obtained from the same process will “almost certainly” have zero likelihood under the first. Nevertheless, the empirical distribution captures where approximately the probability mass is concentrated; and it is exactly this property which will later be exploited in Monte Carlo methods (see Section 2.5).

Let us now consider the case where we have samples $x^{(i)}$ with associated weights $\omega^{(i)}$ that tell us the likelihood of $x^{(i)}$ being a typical sample. Then we may construct an empirical distribution in which the samples are weighted with the normalized likelihood $\tilde{\omega}^{(i)} = \frac{\omega^{(i)}}{\sum_{j=1}^{N} \omega^{(j)}}$ instead of with $1/N$:

$$p(x) = \sum_{i=1}^{N} \tilde{\omega}^{(i)} \delta\left(x - x^{(i)}\right). \tag{2.13}$$

This distribution is called a *weighted empirical distribution.* The following sections introduce parametric distributions – that is, "off-the-shelf" distributions whose parameters can be fitted to the data. The family or type of distribution used would ideally be chosen out of theoretical considerations about the underlying (generating) physical process. In practice, however, it is mostly determined through "eyeballing".

2.2.3 The Uniform Distribution

If every value of a random variable has the same "chance" of being assumed then the behavior of the random variable is well described by a *uniform distribution.* In the discrete case, the underlying physical process of a uniform distribution can be imagined to be similar to that of throwing a dice, where each of the 6 sides shows up with the same probability. In the continuous case, the uniform distribution assigns an equal likelihood of $1/(b-a)$ to all possible values on a given interval $[a, b]$ and zero likelihood to all other values:

$$\mathscr{U}_{[a,b]}(x) \triangleq \frac{1}{b-a} \begin{cases} 1, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases} \tag{2.14}$$

The uniform distribution is used, for example, to model the phase of an audio signal where every possible phase from the interval $[0, 2\pi]$ occurs with the same likelihood: $1/(2\pi)$. This can be extended to the multivariate case by replacing the interval $[a, b]$ by an area $\mathscr{A}$:

$$\mathscr{U}_{\mathscr{A}}(x) \triangleq \frac{1}{\int_{\mathscr{A}} dx} \begin{cases} 1, & x \in \mathscr{A} \\ 0, & \text{otherwise} \end{cases} \tag{2.15}$$

2.2.4 The Gaussian Distribution

If a random variable describes the outcome of a physical process in which many different influences add up, then the variable can be expected to have a *Gaussian distribution.* The Gaussian distribution was originally discovered by Abraham de Moivre [26], who found it while looking for an approximation to binomial distributions with large sample sizes. Nevertheless, the merit is due to Pierre-Simon de Laplace and Carl Friedrich Gauss for establishing it as the probability distribution which measurement errors typically follow. In his seminal work on the "Theory of the Motion of the Heavenly Bodies" [27, 28], Gauss derived it as the likelihood of deviations that independent observations have from their empirical mean. That directly led him to

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2}, \tag{2.16}$$

which is now known as the Gaussian distribution. In this equation, e denotes Euler's constant. The parameters μ and σ^2 specify the mean and variance of the distribution. Due to its ubiquitousness, the Gaussian distribution is also called the "normal" distribution, which is reflected in the notation by writing $\mathcal{N}(x;\mu,\sigma)$ for (2.16). The multivariate case is obtained by replacing the quadratic form $(x-\mu)^2/\sigma^2$ in the exponent by the vector quadratic form $(x-\mu)^T\Sigma^{-1}(x-\mu)$:

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}. \tag{2.17}$$

Here, x is a vector value, μ is the mean vector and Σ is the positive-definite covariance matrix. Using the eigen decomposition $U\Lambda U^T$ of Σ with a unitary matrix U and a diagonal matrix Λ, it can be shown that the multivariate Gaussian distribution is a rotated and translated version of the axis-parallel product of one-dimensional Gaussians:

$$\frac{1}{\sqrt{(2\pi)^n \prod_{i=1}^n \Lambda_{i,i}}} \exp\left(-\frac{1}{2}\sum_{i=1}^n x_i \Lambda_{i,i}^{-1} x_i\right) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\Lambda_{i,i}}} \exp\left(-\frac{1}{2} x_i^2/\Lambda_{i,i}\right). \tag{2.18}$$

While Λ specifies the variance $\Lambda_{i,i}$ in each direction, U specifies the rotation. In the special case where all off-diagonal elements of the covariance matrix are zero, Λ is identical to Σ and U is the identity matrix. In this case, the distribution is called a *diagonal Gaussian distribution* and (2.17) can be written as a product of univariate Gaussians as in (2.18).

2.2.5 Conditional and Marginal Gaussian Distributions

If $Z = (X, Y)$ is a joint Gaussian random variable of which $Y = y$ is observed then the unobserved part $X|y$ has a *conditional Gaussian distribution.* In order to show this, let X and Y be and n_X and n_Y dimensional random vectors with joint distribution

$$p_{X,Y}(x,y) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}\begin{bmatrix} x-\mu_X \\ y-\mu_Y \end{bmatrix}^T \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}^{-1} \begin{bmatrix} x-\mu_X \\ y-\mu_Y \end{bmatrix}\right), \tag{2.19}$$

where $n = n_X + n_Y$, Σ_{XX} is the covariance matrix of X, Σ_{YY} the covariance matrix of Y and $\Sigma_{XY} = \Sigma_{YX}^T$ is the cross-covariance matrix between X and Y. Then the main idea of the following proof can be formulated as to factorize the joint distribution $p_{X,Y}(x,y)$ into $p_{X|y}(x)\cdot p_Y(y)$, by forcing the lower left and upper right blocks of the covariance matrix in 2.19 to zero without modifying Σ_{YY}. This is achieved by first subtracting from the first column the second column multiplied by $\Sigma_{YY}^{-1}\Sigma_{YX}$ from the right (which can be written as a matrix multiplication):

$$\underbrace{\begin{bmatrix}\Sigma_{XX} & \Sigma_{XY}\\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}}_{\triangleq\Sigma}\cdot\underbrace{\begin{bmatrix}I & 0\\ -\Sigma_{YY}^{-1}\Sigma_{YX} & I\end{bmatrix}}_{\triangleq A}=\begin{bmatrix}\Sigma_{XX}-\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX} & \Sigma_{XY}\\ 0 & \Sigma_{YY}\end{bmatrix}.$$

Subsequently, the upper right part of the resulting matrix is forced to zero by subtracting from the first row the second row multiplied by $A^T = \Sigma_{XY}\Sigma_{YY}^{-1}$ from the left:

$$\underbrace{\begin{bmatrix}I & -\Sigma_{XY}\Sigma_{YY}^{-1}\\ 0 & I\end{bmatrix}}_{=A^T}\cdot\begin{bmatrix}\Sigma_{XX}-\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX} & \Sigma_{XY}\\ 0 & \Sigma_{YY}\end{bmatrix}^T=\underbrace{\begin{bmatrix}\Sigma_{XX}-\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX} & 0\\ 0 & \Sigma_{YY}\end{bmatrix}}_{\triangleq\tilde{\Sigma}}$$

Combining these two multiplications gives $A^T\Sigma A=\tilde{\Sigma}$ with $\tilde{\Sigma}$ having block-diagonal form. The matrix A is invertible as it obviously has full rank, no matter what values $\left(-\Sigma_{YY}^{-1}\Sigma_{YX}\right)$ assumes. Thus, the joint covariance matrix Σ can be written $\Sigma = A^{-T}\tilde{\Sigma}A^{-1}$, which when substituted into (2.19) gives:

$$\begin{aligned}p_{X,Y}(x,y) &= \frac{1}{\sqrt{(2\pi)^n\det(\Sigma)}}\exp\left(-\frac{1}{2}\begin{bmatrix}x-\mu_X\\ y-\mu_Y\end{bmatrix}^T A\tilde{\Sigma}^{-1}A^T\begin{bmatrix}x-\mu_X\\ y-\mu_Y\end{bmatrix}\right)\\ &= \frac{1}{\sqrt{(2\pi)^n|\Sigma|}}\exp\left(-\frac{1}{2}\underbrace{\left(A^T\begin{bmatrix}x-\mu_X\\ y-\mu_Y\end{bmatrix}\right)^T}_{=\tilde{z}^T}\underbrace{\begin{bmatrix}\Sigma_{XX|y}^{-1} & 0\\ 0 & \Sigma_{YY}^{-1}\end{bmatrix}}_{=\tilde{\Sigma}^{-1}}\underbrace{\left(A^T\begin{bmatrix}x-\mu_X\\ y-\mu_Y\end{bmatrix}\right)}_{=\tilde{z}}\right),\end{aligned}$$

where $\Sigma_{XX|y}\triangleq\Sigma_{XX}-\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}$ and where the inverse of the block diagonal matrix $\tilde{\Sigma}$ was constructed as the block diagonal matrix consisting of the inverse individual blocks. Now expanding $\tilde{z}$, it is found that:

$$\tilde{z}=A^T\begin{bmatrix}x-\mu_X\\ y-\mu_Y\end{bmatrix}=\begin{bmatrix}I & -\Sigma_{XY}\Sigma_{YY}^{-1}\\ 0 & I\end{bmatrix}\cdot\begin{bmatrix}x-\mu_X\\ y-\mu_Y\end{bmatrix}=\begin{bmatrix}x-\mu_X-\Sigma_{XY}\Sigma_{YY}^{-1}(y-\mu_Y)\\ y-\mu_Y\end{bmatrix}$$

Defining $\mu_{X|y}\triangleq\mu_X+\Sigma_{XY}\Sigma_{YY}^{-1}(y-\mu_Y)$ and making use of the block diagonal structure of $\tilde{\Sigma}^{-1}$ as well as the fact that $\det(\Sigma)=\det(A^T)^{-1}\det(\tilde{\Sigma})\det(A)^{-1}=\det(\tilde{\Sigma})=\det(\Sigma_{XX|y})\det(\Sigma_{YY})$, it becomes clear that the joint distribution from (2.19) can be written:

$$\begin{aligned}p_{X,Y}(x,y) &= \frac{1}{\sqrt{(2\pi)^n\det(\Sigma)}}\exp\left(-\frac{1}{2}(x-\mu_{X|y})^T\Sigma_{XX|y}^{-1}(x-\mu_{X|y})-\frac{1}{2}(y-\mu_Y)^T\Sigma_{YY}^{-1}(y-\mu_Y)\right)\\ &= \frac{\exp\left(-\frac{1}{2}(x-\mu_{X|y})^T\Sigma_{XX|y}^{-1}(x-\mu_{X|y})\right)}{\sqrt{(2\pi)^{n_X}\det(\Sigma_{XX|y})}}\cdot\frac{\exp\left(-\frac{1}{2}(y-\mu_Y)^T\Sigma_{YY}^{-1}(y-\mu_Y)\right)}{\sqrt{(2\pi)^{n_Y}\det(\Sigma_{YY})}}\qquad(2.20)\end{aligned}$$

So, we have arrived at the factorization $p_{X,Y}(x,y) = p_{X|y}(y) \cdot p_Y(y)$ that we have been looking for. In particular, the factorization is the product of two Gaussians:

$$p_{X,Y}(x,y) = \mathcal{N}(x;\mu_{X|y},\Sigma_{XX|y}) \cdot \mathcal{N}(y;\mu_Y,\Sigma_{YY}). \tag{2.21}$$

Marginalizing (2.21) over x shows that Y has a Gaussian distribution with $p_Y(y) = \mathcal{N}(y;\mu_Y,\Sigma_{YY})$, as the integral over $p_{X|y}$ is obviously 1. Dividing (2.21) by $p_Y(y)$ shows that the conditional random variable of $X|y$ has a conditional Gaussian distribution with $p_{X|y}(x) = \mathcal{N}(x;\mu_{X|y},\Sigma_{XX|y})$, as summarized again in the following:

Conditional Gaussian distribution

Let X, Y have a joint Gaussian distribution with parameters specified as in (2.19). Then the conditional distribution of X given y is $p_{X|y}(x) = \mathcal{N}(x;\mu_{X|y},\Sigma_{XX|y})$ with

$$\mu_{X|y} = \mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(y-\mu_Y), \quad \Sigma_{XX|y} = \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}. \tag{2.22}$$

2.2.6 Linearly Transformed Gaussian Distributions

If a Gaussian random variable is linearly transformed then the result is again a Gaussian random variable. In order to show this, let A be a full-rank $n \times n$ matrix, such that A is invertible. Furthermore, let X be an n-dimensional Gaussian random variable with distribution $p_X(x) = \mathcal{N}(x;\mu,\Sigma)$. Then the distribution of the transformed random variable $Y = A \cdot X$ can be obtained by applying the fundamental transformation law of probability from Section 2.1.6:

$$\begin{aligned}
p_Y(y) &= p_X\left(A^{-1}y\right) \cdot \left|\det\left(A^{-1}\right)\right| \\
&= \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(A^{-1}y-\mu)^T\Sigma^{-1}(A^{-1}y-\mu)\right) \cdot \frac{1}{|\det(A)|} \\
&= \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(y-A\mu)^T A^{-T}\Sigma^{-1}A^{-1}(y-A\mu)\right) \cdot \frac{1}{\sqrt{|\det(A)||\det(A^T)|}} \\
&= \frac{1}{\sqrt{(2\pi)^n \det(A\Sigma A^T)}} \exp\left(-\frac{1}{2}(y-A\mu)^T(A\Sigma A^T)^{-1}(y-A\mu)\right),
\end{aligned}$$

In this equation, it was used that $\det\left(A^{-1}\right) = 1/\det(A)$, $\det\left(A^T\right) = \det(A)$ and $\det\left(A\Sigma A^T\right) = \det(A) \cdot \det(\Sigma)\det\left(A^T\right)$. So, we can conclude that the linearly transformed random variable $Y = A \cdot X$ has a multivariate Gaussian distribution with

$$p_Y(y) = \mathcal{N}\left(y;A\mu;A\Sigma A^T\right). \tag{2.23}$$

2.2.7 Gaussian Mixture Distributions

Gaussian mixture distributions are typically used if the exact parametric form of a distribution cannot be derived from theory but samples of the random variable are available. In this case, the samples can be used for estimating a mixture of Gaussians that approximates the true distribution of the random variable. Formally, a Gaussian mixture is defined as a weighted sum of Gaussians:

$$p(x) = \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}\left(x; \mu_X^{(k)}, \Sigma_X^{(k)}\right)}_{=p_{X|k}(x)} \tag{2.24}$$

where c_k, $\mu_X^{(k)}$ and $\Sigma_X^{(k)}$ are the weight, mean and covariance matrix of the k-th Gaussian component, respectively. As the weight c_k actually corresponds to the prior probability $p_K(k)$ that a sample originated from the k-th Gaussian, the sum of the weights has to be one, i.e. $\sum_{k=1}^{K} c_k = 1$, in order for (2.24) to be a valid distribution. The widespread use of Gaussian mixture distributions is due to the fact that Gaussian mixtures can in principle – with the number of components approaching infinity – approximate arbitrary probability density functions with an arbitrary degree of accuracy. In most cases, however, a relatively small number of components[2] is sufficient for a "reasonable" quality of approximation.

2.2.8 Transformations of Gaussian Mixture Distributions

Transforming a Gaussian mixture variable X according to a function f reduces to transforming the individual Gaussian components. In order to show this, the Gaussian mixture distribution from (2.24) may be rewritten as follows:

$$p_X(x) = \sum_{k=1}^{K} \underbrace{p_K(k)}_{=c_k} p_{X|k}(x) = \sum_{k=1}^{K} p_{X,K}(x,k)$$

Then using the fundamental transformation law of probability, it is found that the distribution p_Y of the transformed random variable $Y = f(X)$ can be expressed as:

$$\begin{aligned} p_Y(y) &= p_X\left(f^{-1}(y)\right)\left|\det\left(\frac{df^{-1}(y)}{dy}\right)\right| \\ &= \sum_{k=1}^{K} p_{X,K}\left(f^{-1}(y),k\right)\left|\det\left(\frac{df^{-1}(y)}{dy}\right)\right| \\ &= \sum_{k=1}^{K} \underbrace{p_K(k)}_{=c_k} \underbrace{p_{X|k}\left(f^{-1}(y)\right)\left|\det\left(\frac{df^{-1}(y)}{dy}\right)\right|}_{=p_{Y|k}(y)}. \end{aligned} \tag{2.25}$$

Hence, the transformation of a Gaussian mixture variable can be accomplished by transforming the individual Gaussian components. In the particular case where the transformation is linear, each single $p_{Y|k}$ can be obtained as described in Section 2.2.6 and the result is again a Gaussian mixture distribution.

[2] up to a couple of thousands

2.3 Parameter Estimation

The use of parametric distributions consists in choosing a distribution prototype whose parameters are then fitted to the data. In order to formalize the latter, the parameters to be estimated are here denoted by θ. The data samples that are used for estimation are denoted by $y \triangleq \left\{y^{(1)}, \ldots, y^{(N)}\right\}$. Then, an estimator δ of the parameters is defined as a function $\hat{\theta} = \delta(y)$ of the samples, which is chosen in such a way that the parameters are optimal with respect to some criterion.

2.3.1 Maximum Likelihood Parameter Estimation

The most common criterion in parameter estimation is that of *maximum likelihood*. As the name hints, this criterion consists in choosing that parameter θ, which maximizes the data likelihood $\mathscr{L}(\theta; y) = p_{Y|\theta}(y)$. This procedure goes back to Johann Heinrich Lambert (1760) and Daniel Bernoulli (1778), although it was also used by Gauss [27] and later formalized by Edgeworth [29, 30]. The resulting estimator is formally described as:

$$\hat{\theta} = \delta_{ML}(y) \triangleq \underset{\theta}{\operatorname{argmax}}\, p_{Y|\theta}(y). \tag{2.26}$$

It is worth noting that this estimate does not change if the logarithm is taken on the right hand side, simply because the application of a monotonic function does not change the maximum. Further assuming the samples $y^{(1)}, \ldots, y^{(N)}$ are independent and identically distributed (i.i.d.), the total data likelihood can be written as the product of individual likelihoods. The application of the logarithm converts this product into a sum:

$$\underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p_{Y|\theta}\left(y^{(i)}\right) = \underset{\theta}{\operatorname{argmax}}\, \log\left(\prod_{i=1}^{N} p_{Y|\theta}\left(y^{(i)}\right)\right) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{Y|\theta}\left(y^{(i)}\right).$$

This was well-known even at Gauss's time and it was implicitly used in [27].

2.3.2 The Expectation Maximization Algorithm

Unfortunately, the maximum likelihood method cannot directly be applied if the parameters are dependent on hidden variables. This problem occurs in the context of hidden Markov processes where observations are dependent on the underlying hidden states, but also in the context of Gaussian mixture distributions where the component index needs to be regarded as a hidden variable on which the parameters of the individual Gaussians are dependent. In such cases, an approximate maximum likelihood solution can be found by iterating between estimating the values of the hidden variables and estimating the distribution parameters. A first proof of convergence of this method was given by Baum et al. [31], at the example of hidden Markov processes. Nevertheless, the merit is due to Dempster et al. [32] for (1) recognizing the general applicability of this solution to maximum likelihood estimation with hidden variables,

(2) formalizing the procedure as a simple algorithm and, finally, (3) giving it the name "expectation maximization (EM) algorithm". The main idea behind this algorithm is that the estimation problem would be easier to solve if, in addition to the observed samples $y = \{y^{(1)}, \ldots, y^{(N)}\}$, corresponding samples $h = \{h^{(1)}, \ldots, h^{(N)}\}$ of the hidden variables were available. Then, declaring the augmented set $\{y, h\}$ "complete" and the original set y "incomplete", an approximate solution to the estimation problem can be constructed by iterating between two steps:

(A) an expectation step, in which the auxiliary function $\mathcal{Q}(\theta|\theta^{(l)})$ is constructed as the expectation $\mathcal{E}_{p_{H|y,\theta^{(l)}}}\{\log p_{Y,H|\theta}(y,h)\}$ of the log likelihood of θ under the complete data $\{y, h\}$, given the incomplete data y as well as the current parameter estimate $\theta^{(l)}$:

$$\mathcal{Q}(\theta|\theta^{(l)}) = \int \log\big(p_{Y,H|\theta}(y,h)\big) \cdot p_{H|y,\theta^{(l)}}(h)dh \tag{2.27}$$

(B) a maximization step in which $\theta^{(l+1)}$ is chosen to be a value θ that maximizes the auxiliary function:

$$\theta^{(l+1)} = \underset{\theta}{\operatorname{argmax}}\, \mathcal{Q}(\theta|\theta^{(l)}) \tag{2.28}$$

The heuristic idea behind this procedure is to calculate an approximation to the log likelihood function $\log p_{Y|\theta}(y)$, based on a previous parameter estimate $\theta^{(l)}$. This is done in the expectation step. In the maximization step, the obtained approximation to $\log p(y|\theta)$ is maximized in order to refine the parameter estimate. Iterating these steps causes the parameter θ to converge to a local maximum of the likelihood function [32]. In the special case where $\mathcal{Q}$ is convex, it is even guaranteed that the global maximum is found.

2.4 Minimum Mean Square Error Estimation

In statistics, it often is of interest to estimate the value of a hidden variable based on a corresponding observation. This is typically done in such a fashion that the estimate is optimal with respect to a certain criterion, such as minimization of the *mean squared error* (MSE). In order to formalize the estimation framework, let X bet the hidden variable to be estimated and let $Y = y$ be the observation on which we would like to base our estimation. Furthermore, let $\hat{x} = \delta(y)$ be an arbitrary estimator for x given y, i.e. a function that maps from Y to X. Then, the expected mean squared error which is introduced by using the estimate $\hat{x} = \delta(y)$ instead of the true x is

$$\begin{aligned} MSE\{\delta|y\} &\triangleq \mathcal{E}_{p_{X|y}(x)}\left\{\left\|\delta(y)-x\right\|^2\right\} = \int \left\|\delta(y)-x\right\|^2 \cdot p_{X|y}(x)dx \\ &= \underbrace{\left\|\delta(y)\right\|^2 \int p_{X|y}(x)dx}_{\delta(y)^T\delta(y)} + \underbrace{\int \|x\|^2 \cdot p_{X|y}(x)dx}_{\mathcal{E}_{p_{X|y}(x)}\{x^T x\}} - \underbrace{2\delta(y)^T \int x \cdot p(x|y)dx}_{2\delta(y)^T\mathcal{E}_{p_{X|y}(x)}\{x\}} \end{aligned}$$

where it was used that $||\delta(y)-x||^2 = (\delta(y)-x)^T \cdot (\delta(y)-x)$. Now, the mean squared error may be minimized by taking the derivative with respect to $\delta(y)$ and then equating the result to zero. Using the above expansion, we obviously get $\frac{dMSE\{\delta|y\}}{d\delta(y)} = 2\delta(y) - 2\mathcal{E}_{p_{X|y}(x)}\{x\}$, which, when equated to zero and solved for $\delta(y)$, gives the general *minimum mean squared error* (MMSE) estimate:

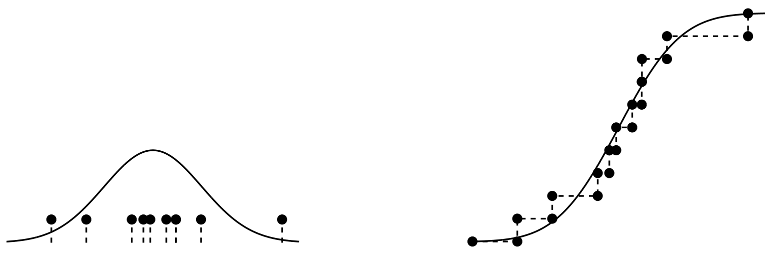
$$\delta_{MMSE}(y) = \mathcal{E}_{p_{X|y}(x)}\{x\}. \tag{2.29}$$

As practical implementations of (2.29) require knowledge of $p_{X|y}$, the following chapters will put quite some effort into constructing this distribution. This will be done based on a prior distribution of X as well as a statistical model for how the hidden variable X is related to observations (see Chapter 3 for the details).

2.5 Monte Carlo Methods

For some problems that occur in statistics, closed form analytic solutions are either unwieldy or simply not available. After transformation of a random variable, for example, an originally parametric distribution might no longer have a parametric form. Similarly, if expectations of a random variable are to be calculated, it might happen that there is no analytic solution to the occurring integrals. In such cases, it can be useful to work with empirical distributions rather than with parametric ones, as then all integrals are converted into sums and the distributions of transformed random variables can simply be obtained by transforming samples. This is the principal idea behind *Monte Carlo* (MC) methods, whose history began in 1946 in Los Alamos with a secret project in which the use of such methods was investigated during the development of the thermonuclear bomb. Incidentally, the code name of the project was "Monte Carlo", which leaves little room for speculation as to where the name stems from. According to Metropolis [33], "the work was triggered by Stanislav Ulam recognizing that the ENIAC – a computer that was originally conceived for calculating artillery firing table solutions – could be used to resuscitate statistical sampling techniques. These techniques had been used before, but they had fallen into desuetude due to the length and tediousness of the calculations".

Ulam's idea consisted in generating the required samples on a computer, which can be performed much more efficiently than real random experiments. Figure 2.4-(a) depicts sampling from a Gaussian distribution. The dashed lines with black, filled circles on top indicate the Dirac deltas with which the empirical distribution places discrete probability mass at the locations of the samples. The solid black line shows the continuous Gaussian density function for comparison. A visual comparison might not reveal that the empirical distribution provides a good approximation to the parametric one (and for calculating likelihoods it surely is not). Nevertheless, the empirical distribution approximately captures where the probability mass is concentrated, as samples are more likely to be drawn from regions of high probability mass. This becomes clear by looking at the cumulative density functions, $P(X \leq x) = \int_{-\infty}^{x} p_X(x')dx'$, which are shown in Figure 2.4-(b).



(a) probability density functions *(b) cumulative density functions*

Figure 2.4: *Empirical versus parametric distributions at the example of a Gaussian distribution.*

2.5.1 Monte Carlo Integration

The probably most well known Monte Carlo method is Monte Carlo integration (MCI). This method uses the empirical distribution of a random variable in order to approximate the expectation $\mathrm{E}_{p_X}\{h(x)\}$ under a function h. This is achieved by (1) simulating samples $x^{(1)},\ldots,x^{(N)}$ form the parametric distribution p_X of the random variable; (2) constructing the corresponding empirical distribution

$$\hat{p}_X(x) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(x - x^{(i)}\right);$$

and then (3) replacing the parametric distribution in the integral $\mathrm{E}_{p_X}\{h(x)\}$ by its empirical counterpart. As a result, the expectation integral $\mathrm{E}_{p_X}\{h(x)\}$ is turned into a sum:

$$\int h(x)\cdot p_X(x)dx \quad \approx \quad \int h(x)\hat{p}_X(x)dx \quad = \quad \frac{1}{N}\sum_{j=1}^{N} h(x^{(j)}). \tag{2.30}$$

The error of this approximation in general decreases with N. By the strong law of large numbers, it is even guaranteed that $\mathrm{E}_{\hat{p}_X}\{h(x)\} = \frac{1}{N}\sum_{j=1}^{N} h(x^{(j)})$ "almost surely" converges to $\mathrm{E}_{p_X}\{h(x)\}$ [23]. According to [33], Enrico Fermi, a world-renowned physicist and Nobel prize laureate, reportedly used this technique to astonish his colleagues with remarkably accurate "too-good-to-believe" predictions of experimental results. In particular, it is said that he performed the necessary computations with a small mechanical adding machine only [33]; and that was long before Stanislav Ulam actually published the method.

2.5.2 Monte Carlo Transformation

In the Monte Carlo integration technique from the previous section, expectations were approximated by evaluating a function at simulated values of a random variable. Monte Carlo transformation, in contrast, tries to approximate the distribution of a transformed random variable $Y = f(X)$ by transforming simulated values (samples) $x^{(1)},\ldots,x^{(N)}$ of the variable X. This can be regarded as the Monte Carlo equivalent to the fundamental transformation law of probability, and it results in the following (empirical) approximation of p_Y:

$$\hat{p}_Y(y) = \frac{1}{N}\sum_{j=1}^{N}\delta\left(y - y^{(j)}\right) \quad \text{with} \quad y^{(j)} = f\left(x^{(j)}\right). \tag{2.31}$$

A heuristic argument for the validity of this approximation is that the samples $x^{(1)},\ldots,x^{(N)}$ form p_X capture where the probability mass of X is concentrated. Hence, transforming the samples moves the probability mass according to the function f and consequently simulates the distribution of $f(X)$. This can be formally proven by considering expectations $\mathrm{E}_{p_X}\left\{h(f(x))\right\}$ of X under functions h of $f(x)$. Then, performing a change of variables from x to $y = f(x)$ yields:

$$\mathrm{E}_{p_X}\left\{h(f(x))\right\} = \int h\big(f(x)\big)p_X(x)dx = \int h(y)\underbrace{p_X\left(f^{-1}(y)\right)\left|\det\left(\frac{df^{-1}(y)}{dy}\right)\right|}_{=p_Y(y)}dy = \mathrm{E}_{p_Y}\left\{h(y)\right\}.$$

This shows that the moments of p_Y are identical to those of $f(X)$. Now using $\mathrm{E}_{p_Y}\left\{h(y)\right\} = \mathrm{E}_{p_X}\left\{h(f(x))\right\}$ and further approximating $\mathrm{E}_{p_X}\left\{h(f(x))\right\}$ by Monte Carlo integration, it follows that

$$\mathrm{E}_{p_Y}\left\{h\big(y\big)\right\} = \mathrm{E}_{p_X}\left\{h\big(f(x)\big)\right\} \approx \mathrm{E}_{\hat{p}_X}\left\{h\big(f(x)\big)\right\} = \frac{1}{N}\sum_{j=1}^{N}h\left(f\left(x^{(j)}\right)\right) = \mathrm{E}_{\hat{p}_Y}\left\{h(y)\right\}.$$

So, we have shown that $\hat{p}_Y$ approximates all moments of p_Y, from which it follows that $\hat{p}_Y$ approximates the distribution of the transformed random variable $Y = f(X)$. For completeness, it should be noted that the latter actually only holds if p_X and p_Y have bounded support [24, 25]. But this does not mater in practice, as the support can always be truncated at some remote point of the tail (as an arbitrarily accurate approximation of p_X and p_Y with bounded support).

2.5.3 Sample Generation

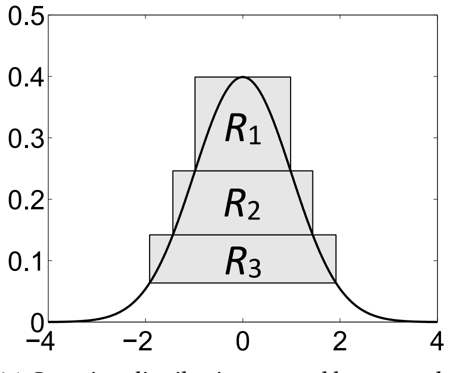
Monte Carlo methods require generating a larger number of samples from the random variables of interest. But this is complicated by the fact that computers are inherently deterministic machines that can by no means generate true random numbers of whatever kind. Hence, we have to resort to deterministic algorithms that produce seemingly unpredictable sequences of so-called *pseudo-random numbers.* Such algorithms are called *random number generators* and they are typically designed in an engineering-like fashion, by starting with a simple standard generator to produce uniformly distributed samples and then applying inversion or rejection methods [34] in order to obtain samples of the desired type. A popular choice for a standard generator is George Marsaglia's KISS [35] whose C source code is shown in Algorithm 2.1. The KISS algorithm is based on simultaneously running one congruential generator in the variable i as well as two shift register generators in the variables j and k, whose outputs are subsequently combined. The C code uses 32-bit unsigned integer variables i, j and k as an internal representation. The output is a floating point number simulating a sample from the standard uniform distribution on the interval $[0,1]$.

```
float kiss32() {
  i = (69069*i) + 23606797;
  j = j ^ (j << 17);
  k = (k ^ (k << 18)) & 0x7FFFFFFF;
  j = j ^ (j >> 15);
  k = k ^ (k >> 13);
  return (float)(i + j + k) / MAX_INT;
}
```

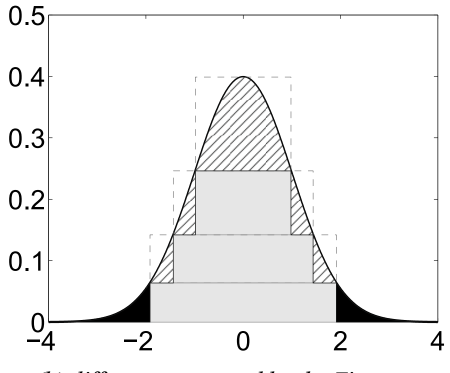
Algorithm 2.1: "C" code of the KISS generator which generates uniformly distributed samples.

2.5.3.1 Sampling from the Standard Normal Distribution

The probably most efficient way to generate a sample from the standard normal distribution is to use Marsaglia and Tsang's *Ziggurat* method [36]. It covers the normal pdf with a set of equal-area rectangles plus a remainder consisting of a smaller rectangle as well as the tail. Figure 2.5-(a) shows this at the example of 3 rectangles, R_1, R_2 and R_3. The remainder R_4 covers the same area as each of these rectangles. Hence, a sample of the distribution can be simulated by first selecting (by random, with equal probability) a region R_i to sample from and then generating a sample from that particular region.



(a) Gaussian distribution covered by rectangles



(b) different areas used by the Ziggurat

Figure 2.5: *The Zigurrat. The picture to the left shows where the method draws its name from: terraced step pyramids of ancient Mesopotamia such as the Neo-Sumerian Great Ziggurat of Ur.*

Rejection Sampling

If the region selected for sampling is one of the rectangles, *rejection sampling* is performed. In this case, a sample u is drawn from the uniform distribution and scaled to fit the width $w(R_i)$ of the rectangle: $x = w(R_i)(u-\frac{1}{2})$. Consequently, x is uniformly distributed along the x-direction of R_i. But we would like it to follow the Gaussian curve. Hence, we reject x with probability

$$\mathscr{P}_r \ = \ 1 - \frac{\min\{y_2(R_i), f(x)\} - y_1(R_i)}{y_2(R_i) - y_1(R_i)}$$

where $y_2(R_i)$ is the upper y-coordinate of the rectangle, $y_1(R_i)$ is the lower y-coordinate and

f is the Gaussian density function: $f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$. The rejection procedure causes that samples x for which $f(x)$ is close to $y_1(R_i)$ will be rejected with high probability. Samples for which $f(x)$ is close to $y_2(R_i)$ will mostly be accepted. This is indicated through the dashed areas in Figure 2.5-(b), which show the percentage of accepted samples for each rectangle in dependence of the x-coordinate. The solid gray areas mark regions where samples can directly be accepted due to $\min\left\{y_2(R_i), f(x)\right\} = y_2(R_i)$. These areas are precomputed in order to avoid the evaluation[3] of f at runtime (and it is exactly this, which makes the Ziggurat so efficient).

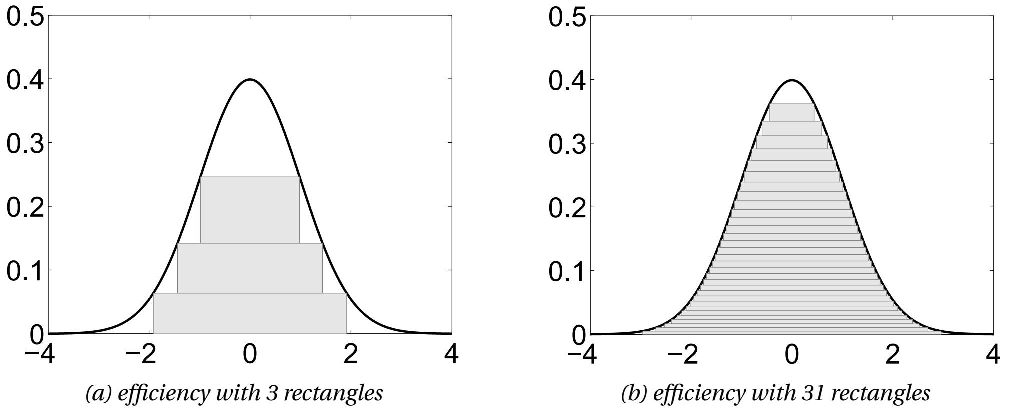


Figure 2.6: *Efficiency of the Zigurrat. The smaller the white area between the Gaussian curve and the pile of gray rectangles the fewer samples are rejected. With 255 rectangles 99.33% of the samples reside in the solid gray areas and can directly be accepted [36].*

Simulating from the Remainder

In case the remainder was selected for sampling, we have to figure out whether to draw a sample from the gray area sitting on the x-axis of Figure 2.5-(b) or whether to draw a sample from the tail, which is marked in solid black. This is done by first selecting one of the two areas by random (with a probability corresponding to the relative probability mass of each area) and then either drawing a sample from the gray area as described above or generating a sample from the tail as described in more detail in [37, 36].

2.5.3.2 Sampling from Multivariate Gaussian Distributions

Sampling from a univariate Gaussian distribution can easily be extended to the multivariate case. For that, let $p_Y(y) = \mathcal{N}(y;\mu,\Sigma)$ denote the n-dimensional Gaussian distribution from which we would like to sample; and let $\Sigma = R^T R$ be the Cholesky decomposition of its covariance matrix with an upper triangular matrix R. Then, a sample y from p_Y can be simulated by drawing x from $p_X(x) = \mathcal{N}(x;\mathbf{0},\mathbf{I})$ and subsequently transforming x according to:

$$X \longmapsto Y = f(X) \quad \text{with} \quad y = f(x) = R^T x + \mu. \tag{2.32}$$

[3] Note that the evaluation of the Gaussian density function $f(x)$ is computationally expensive as it requires the calculation of the exponential function.

The claim that the resulting samples y follow the desired Gaussian distribution is easily verified through application of Sections 2.2.6 and 2.5.2 because, according to these, the transformed samples y have a Gaussian distribution with

$$p_Y(y) = \mathcal{N}\left(x;\, \mu + R^T \cdot \mathbf{0},\, R^T \mathbf{I} R\right)$$

where $\mathbf{I}$ is the identity matrix and where $R^T \mathbf{I} R = R^T R$ is by definition equal to Σ. Due to the choice of $\mathbf{I}$ as a covariance matrix, samples x from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ can easily be simulated by independently drawing the individual x_i from the standard normal distribution $\mathcal{N}(0,1)$, for example through use of the Ziggurat method from Section 2.5.3.1.

2.5.3.3 Sampling from Gaussian Mixture Distributions

Sampling from a Gaussian mixture distribution is achieved by (1) selecting one of the Gaussian components, with a probability corresponding to the component weight c_k (see Section 2.2.7), and then (2) generating a sample from that Gaussian. In order to do this, let us first of all calculate the cumulative weights

$$C_0 \triangleq 0, \qquad C_k \triangleq \sum_{i=1}^{k} c_i, \quad k = 1, \ldots, K.$$

Then, the probability $\mathcal{P}(C_{k-1} \leq u < C_k)$ that a sample u from the standard uniform distribution is located in the k-th interval $I_k \triangleq [C_{k-1} \quad C_k]$ is equal to $C_k - C_{k-1} = c_k$. Hence, we draw a sample u from $\mathcal{U}_{[0,1]}$, select the k-th Gaussian if $u \in I_k$ and then draw a sample from this Gaussian as described in Section 2.5.3.2.

2.5.4 Importance Sampling

In some cases it is not possible to directly simulate samples from the distribution of a random variable. Such a case occurs, for example, if $X|y$ is a conditional random variable whose distribution $p_{X|y}(x)$ was obtained from Bayes' equation,

$$p_{X|y}(x) = \frac{p_{Y|x}(y) p_X(x)}{p_Y(y)} = \frac{p_{Y|x}(y) p_X(x)}{\int p_{Y|x}(y) p_X(x) dx},$$

with only $p_{Y|x}(y)$ and $p_X(x)$ being known. In this case, generating a sample from $X|y$ becomes problematic if the integral in the denominator, i.e. $p_Y(y) = \int p_{Y|x}(y) p_X(x) dx$, cannot be calculated analytically. The idea behind importance sampling is to resolve this issue by

(1) simulating samples $x^{(1)}, \ldots, x^{(N)}$ from p_X.

(2) evaluating $p_{Y|x^{(j)}}(y)$ for every sample.

(3) weighting the samples by $\omega^{(j)} \triangleq \frac{p_{Y|x^{(j)}}(y)}{p_Y(y)}$, respectively.

The weighting in the last step essentially matches the sample distribution of the $X^{(j)}$ to that of $p_{X|y}$. This does not solve the problem of $p_Y(y)$ being non-analytic. But it allows us to approximate $p_Y(y)$ by Monte Carlo integration: $\hat{p}_Y(y) \approx \frac{1}{N}\sum_{i=1}^{N} p_{Y|x^{(j)}}(y)$. The result is a weighted empirical distribution of $X|y$:

$$\tilde{p}_{X|y} = \sum_{j=1}^{N} \tilde{\omega}^{(j)} x^{(j)} \quad \text{with} \quad \tilde{\omega}^{(j)} \triangleq \frac{p_{Y|x^{(j)}}(y)}{\sum_{i=1}^{N} p_{Y|x^{(i)}}(y)}$$

where the normalized weights $\tilde{\omega}^{(j)}$ are equal to $\frac{1}{N}\omega^{(j)}$ as the sum over the $\omega^{(j)} = \frac{p_{Y|x^{(j)}}(y)}{\hat{p}_Y(y)}$ is N for the above approximation of $p(y)$. In order to generalize this procedure, let p_X be a distribution that is to be sampled from; let $\pi(x)$ be an arbitrary proposal or importance distribution whose support includes the support[4] of p_X; and assume that π is easy to sample from, so that samples $x^{(j)}$ from π are available. Then making use of the *importance sampling fundamental identity* [23],

$$p_X(x) = \underbrace{\frac{p_X(x)}{\pi(x)}}_{\triangleq\omega(x)} \pi(x) = \omega(x)\pi(x), \tag{2.33}$$

it becomes clear that the distribution of X can be approximated by samples $x^{(j)}$ from π if these samples are weighted with $\omega^{(j)} = p_X\left(x^{(j)}\right)/\pi\left(x^{(j)}\right)$. This procedure is called importance sampling. The resulting weighted empirical distribution $\tilde{p}_X = \sum_{i=1}^{N} \tilde{\omega}^{(j)} x^{(j)}$ can be used in the context of Monte Carlo methods if the concerned methods are modified. This is straight forward: simply replace p_X by $\tilde{p}_X$ instead of replacing it by $\hat{p}_X$ (as it has been done in Sections 2.5.1 and 2.5.2). The validity can be shown in analogy to the proofs for empirical distributions.

2.5.5 Importance Resampling

Any weighted empirical density can be converted to an empirical density by simply sampling from it. This is of particular interest when importance sampling is performed in an iterative fashion, as it is the case in a particle filter (see chapter 4.7). In such a case, it may happen that some of the weights get increasingly smaller over time, as a consequence of which the samples corresponding to these weights effectively become irrelevant. To alleviate this problem, it has been proposed to occasionally convert the weighted empirical distribution to an equally-weighted empirical distribution. This is achieved by drawing (with replacement) samples $\tilde{x}^{(k)}$, $k = 1, \ldots, N$ from the weighted empirical density

$$\tilde{p}(x) = \sum_{i=1}^{N} \tilde{\omega}^{(i)} x^{(i)}$$

This is performed as described in Algorithm 2.2. In order to verify that this algorithm returns $\tilde{x}^{(k)} = x^{(i)}$ with probability $\tilde{\omega}^{(i)}$, it should be noted that the probability of a sample u from the

[4] i.e. it is required that $\pi(x) \neq 0$ when $p(x) \neq 0$

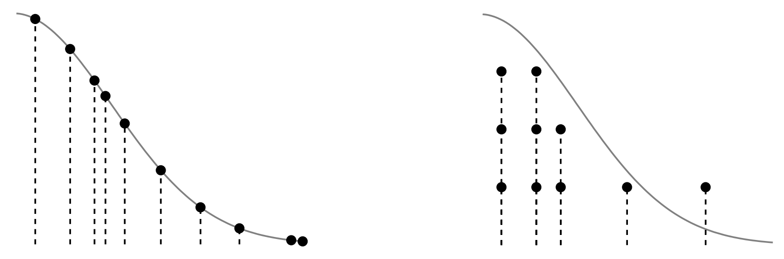
standard uniform distribution being located in the interval $[C_{i-1}, C_i]$ is equal to $C_i - C_{i-1} = \omega^{(i)}$. Every sample u is located in one of the intervals, as $C_0 = 0$ and $C_N = 1$ (the latter holds due to the fact that the sum over the normalized weights $\tilde{\omega}^{(i)}$ is 1).

```
Importance Resampling

1. draw a sample $u$ from the standard uniform distribution $\mathcal{U}_{[0,1]}$
2. look up in which interval $[C_{i-1}, C_i]$, $i \in 1, \ldots, N$ with $C_i = \sum_{j=1}^{i} \tilde{\omega}_j$ it is located
3. return the corresponding $x^{(i)}$ as $\tilde{x}^{(k)}$.
```

Algorithm 2.2: Importance Resampling

As depicted in Figure 2.7, the algorithm essentially multiplies samples with a high relative weight and removes samples with a low relative weight. The name "importance resampling" stems from the facts that (1) sampling is performed from a sample distribution and (2) that the samples are drawn with the relative frequency of their normalized importance weights. Interestingly, Rubin [38] originally proposed this method in a comment to Tanner and Wong's data augmentation algorithm [39], although in retrospect it would have surely deserved a stand-alone publication.



(a) weighted empirical density *(b) resampled empirical density*

Figure 2.7: *Importance Resampling. The picture to the left shows the original weighted empirical density. The picture to the right shows how importance resampling mimics weighted samples by putting several equally-weighted samples in the same place.*

3 Bayesian State Estimation

This chapter deals with the problem of estimating the state of a physical system based on a corresponding observation. To have a concrete scenario, consider the system state to be the location of a speaker in a room. Further, consider the observation to be the time delay which emitted sound waves introduce at an array of microphones (as portrayed in Figure 3.1). Then, the estimation problem can be described as localizing the speaker based on measured time delays. In order to account for uncertainties in measurements as well as the physical model, the

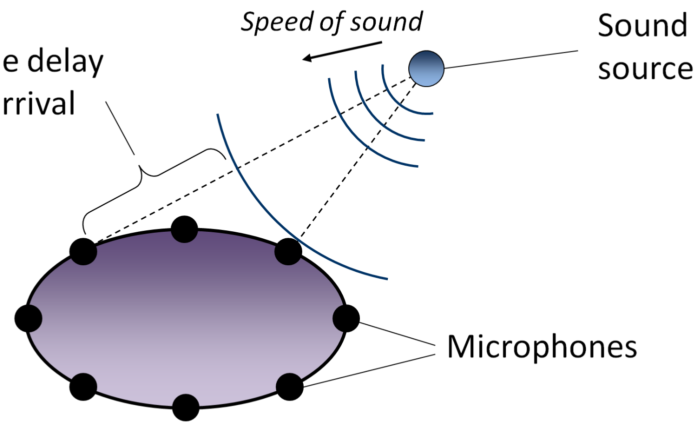


Figure 3.1: *Speaker Localization.*

system states and observations may be modeled as random variables X and Y. Subsequently, the estimation problem can be formulated as constructing the conditional distribution $p_{X|y}$ of the state given the realized (i.e. measured) observation $Y = y$. Particular estimates $\hat{x} = \delta(y)$ are obtained by applying a criterion of optimality, such as minimum mean squared error (MMSE) or maximum a-posteriori (MAP):

$$\delta_{MMSE}(y) = \mathscr{E}_{p_{X|y}(x)}\{x\}, \qquad \delta_{MAP}(y) = \underset{x}{\operatorname{argmax}}\, p_{X|y}(x). \tag{3.1}$$

The remaining part of the chapter is concerned with the central aspect of this estimation framework: constructing $p_{X|y}$. This will be done in a Bayesian fashion, through use of a prior distribution of the system state.

3.1 The Bayesian Approach

Using a prior distribution for the state to be estimated is generally referred to as the “Bayesian approach”. That is, because the prior p_X can be factored into the conditional distribution $p_{X|y}$ by making use of Bayes rule:

$$p_{X|y}(x) = \frac{p_{Y|x}(y)p_X(x)}{\int p_{Y|x}(y)p_X(x)dx}. \tag{3.2}$$

Following the developments in [18], the Bayesian approach is here viewed from a slightly different perspective: as a two step procedure, which

(1) constructs the joint distribution $p_{X,Y}$ of state X and observation Y, based on the prior distribution of the state as well as a model for how states and observations are related.

(2) conditions the joint distribution $p_{X,Y}$ on a realization $Y = y$ of the observation variable in order to obtain the posterior distribution $p_{X|y}$.

The intuition behind this formulation is that the joint distribution $p_{X,Y}$ contains all statistical knowledge about the relationship between the variables X and Y. Conditioning $p_{X,Y}$ on y can be interpreted as using this statistical relationship to constrain the prior distribution p_X of the state by the knowledge obtained through reception of $Y = y$.

3.1.1 Observation Model

In order to develop a formal observation model that describes the relationship between states and observations, let us revisit the example of localizing a speaker in a room. In this case, the relationship between speaker position x and measured time delay y_i at the i-th microphone is the distance of the speaker from the microphone, divided by the speed of sound:

$$y_i = g_i(x) \triangleq \frac{\|x - m_i\|}{c}$$

The m_i in this equation denotes the position of the i-th microphone; c denotes the speed of sound. In order to account for uncertainties in the measurements, the observation y_i might be made dependent not only on the system state X but also on random influences U during measurement. This leads to the following observation model:

$$y_i = g_i(x) + u, \quad u \sim \mathcal{N}(\mu_U, \Sigma_U)$$

where u denotes a sample from the Gaussian distribution of random influences, $p_U(u) = \mathcal{N}(u; \mu_U, \Sigma_U)$. This example can easily be extended to a general observation model that is fully specified by the following two items:

General Observation Model

- a prior distribution p_U of random influences during observation
- an observation function $y = g(x, u)$

3.1.2 A General Method for Constructing the Joint

After introduction of this observation model, the joint distribution of state and observation can be constructed by transforming the random variables X and U according to:

$$\begin{bmatrix} X \\ U \end{bmatrix} \longmapsto \begin{bmatrix} X \\ Y \end{bmatrix} = \tilde{g}\left(\begin{bmatrix} X \\ U \end{bmatrix}\right) \quad \text{with} \quad \tilde{g}\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) = \begin{bmatrix} x \\ g(x,u) \end{bmatrix}. \tag{3.3}$$

In this equation, $\tilde{g}$ is called the *augmented observation function*; g denotes the observation function from Section 3.1.1; and U denotes the variable of random influences during observation. While the transformation in 3.3 can generally be performed with the fundamental transformation law of probability, practical implementations are given in Sections 3.3 - 3.7. In the following, it is in particular assumed that the state is statistically independent of random influences during observation, as a consequence of which: $p_{X,U}(x,u) = p_X(x) \cdot p_U(u)$.

3.1.3 A Transformation-Centric View

The above developments reveal that Bayesian state estimation can be viewed as a two step approach, in which

1. the joint distribution $p_{X,Y}$ of state and observation is constructed by transforming the random variables X and U according to the augmented observation function $\tilde{g}$.
2. the posterior distribution $p_{X|y}$ is obtained by conditioning $p_{X,Y}$ on the measured observation $Y = y$.

This formulation is transformation-centric, as it solves the construction of $p_{X,Y}$ through a general transformation of random variables. The following section (Section 3.2) shows practical implementations for cases in which the joint distribution is Gaussian (Section 3.2.1), a Gaussian mixture (Section 3.2.2) or a semi-empirical distribution (Section 3.2.3). The remaining part of the chapter (Sections 3.3 - 3.7) derives particular transformations for constructing $p_{X,Y}$. This includes standard methods such as linear-Gaussian transforms (Sections 3.3 and 3.4) as well as the unscented transform (Section 3.5). But it also introduces novel methods which more accurately treat nonlinearities in the observation function, namely: the extensive unscented transform (Section 3.6) and the adaptive level of detail transform (Section 3.7).

3.2 Particular Implementations

3.2.1 The Gaussian Case

The most well-known implementation of Bayesian estimation is based on the assumption that the distribution of X and Y is jointly Gaussian. This approach can be applied whenever both the distributions of state X and random influences U during observation are Gaussian. In this case, $p_{X,Y}$ may be constructed as described in Sections 3.3 - 3.5 and it may be written:

$$p_{X,Y}(x,y) = \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}\right). \tag{3.4}$$

Then applying Section 2.2.5, it becomes clear that the conditional distribution $p_{X|y}$ of X given y is a conditional Gaussian distribution:

$$p_{X|y}(x) = \mathcal{N}(x; \mu_{X|y}, \Sigma_{XX|y}) \tag{3.5}$$

with $\mu_{X|y}$ and $\Sigma_{XX|y}$ being defined as:

$$\mu_{X|y} = \mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(y - \mu_Y), \quad \Sigma_{XX|y} = \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}. \tag{3.6}$$

This is what is used in the Kalman filter [40, 41] [42, 43, 44] but also in other estimation approaches such as the Vector Taylor Series approach for Environment-Independent Speech Recognition [45, 46].

3.2.2 The Gaussian Mixture Case

Another popular implementation of Bayesian state estimation is based on the assumption that the joint distribution of X and Y can be approximated as a Gaussian mixture. This is of interest in the context of nonlinear or non-Gaussian estimation problems for which the true joint distribution is usually difficult to obtain. In these cases, a Gaussian mixture approximation

$$p_{X,Y}(x,y) = \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_{X|k} \\ \mu_{Y|k} \end{bmatrix}, \begin{bmatrix} \Sigma_{XX|k} & \Sigma_{XY|k} \\ \Sigma_{YX|k} & \Sigma_{YY|k} \end{bmatrix}\right)}_{=p_{X,Y|k}(x,y)} \tag{3.7}$$

of the joint distribution may be obtained as described in Section 3.7. Subsequently, the conditional distribution of the state X given the realized observation y can be calculated by dividing

$p_{X,Y}(x,y) = \sum_{k=1}^{\kappa} p_{X,Y,K}(x,y,k)$ by the observation likelihood $p_Y(y)$:

$$p_{X|y}(x) = \frac{\sum_{k=1}^{\kappa} p_{X,Y,K}(x,y,k)}{p_Y(y)} = \sum_{k=1}^{\kappa} p_{X,K|y}(x,k) = \sum_{k=1}^{\kappa} p_{X|y,k}(x) p_{K|y}(k).$$

This is obviously a mixture of conditional Gaussian distributions:

$$p_{X|y}(x) = \sum_{k=1}^{\kappa} c_k^+ \underbrace{\mathcal{N}\left(x; \mu_{X|y,k}, \Sigma_{X|y,k}\right)}_{=p_{X|y,k}(x)} \tag{3.8}$$

where the c_k^+ denote the posterior probabilities $p_{K|y}(k)$ of the individual Gaussian components,

$$p_{K|y}(k) = \frac{c_k p_{Y|k}(y)}{\sum_{k'=1}^{\kappa} c'_k p_{Y|k'}(y)} = \frac{c_k \mathcal{N}\left(y; \mu_{Y|k}, \Sigma_{Y|k}\right)}{\sum_{k'=1}^{\kappa} c_{k'} \mathcal{N}\left(y; \mu_{Y|k'}, \Sigma_{Y|k'}\right)}, \tag{3.9}$$

and where the $\mu_{X|y,k}$ and $\Sigma_{X|y,k}$ denote their conditional means and covariance matrices, respectively, which can be calculated in analogy to (3.6):

$$\mu_{X|y,k} = \mu_{X|k} + \Sigma_{XY|k} \Sigma_{YY|k}^{-1} (y - \mu_{Y|k}), \quad \Sigma_{XX|y,k} = \Sigma_{XX|k} - \Sigma_{XY|k} \Sigma_{YY|k}^{-1} \Sigma_{YX|k}. \tag{3.10}$$

This implementation forms the basis of Alspach and Sorenson's Gaussian mixture filters [47, 48] as well as the speech feature enhancement approaches in [49, 50].

3.2.3 The Semi-Empirical Case

Yet another way to approach nonlinear and non-Gaussian estimation problems is to approximate the prior distribution of the state by samples. This is done in the context of Monte Carlo methods, such as particle filters (see Section 4.7), which approximate the distribution of X as an empirical distribution

$$p_X(x) = \frac{1}{N} \sum_{j=1}^{N} \delta\left(x - x^{(j)}\right), \tag{3.11}$$

with N samples $\left\{x^{(1)}, \ldots, x^{(N)}\right\}$. This is advantageous, as it imposes no restrictions on the estimation problem other than that it must be possible to sample from X. Now using this approximation, the joint distribution $p_{X,Y}$ of state and observation can be constructed according to:

$$p_{X,Y}(x,y) = \underbrace{\left(\frac{1}{N} \sum_{j=1}^{N} \delta\left(x - x^{(j)}\right)\right)}_{=p_X(x)} p_{Y|x}(y) = \frac{1}{N} \sum_{j=1}^{N} \delta\left(x - x^{(j)}\right) p_{Y|x^{(j)}}(y) \tag{3.12}$$

where the observation density $p_{Y|x^{(j)}}(y)$ of the j-th sample $x^{(j)}$ is obtained by transforming the variable U of random influences according to $Y|x^{(j)} = g\left(x^{(j)}, U\right)$. This is achieved by using the fundamental transformation law of probabilities from Section 2.1.6. The resulting distribution

(3.12) is semi-empirical, as it is empirical in X and arbitrary in Y. Hence, the name of this section. Now, given the joint distribution from (3.12), the conditional distribution of X given y can be obtained by fixing $Y = y$ and then dividing the result by $p_Y(y)$:

$$p_{X|y}(x) \;=\; \frac{p_{X,Y}(x,y)}{p_Y(y)} \;=\; \frac{1}{N}\sum_{j=1}^{N}\delta\left(x-x^{(j)}\right)\frac{p_{Y|x^{(j)}}(y)}{p_Y(y)}. \tag{3.13}$$

This can be further simplified by calculating the normalizing constant $p_Y(y)$ as a marginal distribution of $p_{X,Y}(x,y)$:

$$p_Y(y) \;=\; \int \underbrace{\frac{1}{N}\sum_{j=1}^{N}\delta\left(x-x^{(j)}\right)p_{Y|x^{(j)}}(y)}_{=p_{X,Y}(x,y)}dx \;=\; \frac{1}{N}\sum_{j=1}^{N}p_{Y|x^{(j)}}(y). \tag{3.14}$$

Finally plugging (3.14) back into (3.13), it turns out that the conditional distribution $p_{X|y}(y)$ can be approximated as a weighted empirical distribution:

$$\tilde{p}_{X|y} \;=\; \frac{1}{N}\sum_{j=1}^{N}\left(\frac{p_{Y|x^{(j)}}(y)}{\frac{1}{N}\sum_{i=1}^{N}p_{Y|x^{(i)}}(y)}\right)\delta\left(x-x^{(j)}\right) \;=\; \sum_{j=1}^{N}\tilde{\omega}^{(j)}\delta\left(x-x^{(j)}\right) \tag{3.15}$$

where the weights $\tilde{\omega}^{(j)}$ are defined as

$$\tilde{\omega}^{(j)} \triangleq \frac{p_{Y|x^{(j)}}(y)}{\sum_{i=1}^{N}p_{Y|x^{(i)}}(y)}. \tag{3.16}$$

3.3 The Kalman-Type Linear Transform

As stressed at the start of this chapter, all Bayesian estimation approaches require constructing the joint distribution $p_{X,Y}$. This can generally be achieved by transforming the prior distribution $p_{X,U}$ of X and U according to the augmented observation function $\tilde{g}$ from (3.3). This section treats the particular case in which

(1) the observation function g is a linear function of the form $g(x,u) = Bx + Cu$ with an arbitrary $n \times n$ matrix B and a non-singular $m \times m$ matrix C

(2) the variables X and U are n- and m-dimensional Gaussian random variables that have a joint Gaussian distribution

$$p_{X,U}(x,u) = \mathcal{N}\left(\begin{bmatrix} x \\ u \end{bmatrix}; \begin{bmatrix} \mu_X \\ \mu_U \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XU} \\ \Sigma_{UX} & \Sigma_{UU} \end{bmatrix}\right)$$

This class of transformations has been established through Kalman's seminal work on linear filtering and prediction problems [40]. Hence, it is here called the *Kalman-type linear transform* [18]. In order to show that the above restrictions lead to a joint Gaussian distribution of state and observation, let Z be the random variable, which results from the transformation of X, U according to the augmented observation function:

$$Z = \tilde{g}\left(\begin{bmatrix} X \\ U \end{bmatrix}\right) = \begin{bmatrix} X \\ g(X,U) \end{bmatrix} = \underbrace{\begin{bmatrix} I_{n,n} & 0_{n,m} \\ B & C \end{bmatrix}}_{\triangleq A} \begin{bmatrix} X \\ U \end{bmatrix}. \tag{3.17}$$

Here $I_{n,n}$ denotes a $n \times n$ identity matrix; $0_{n,m}$ denotes a $n \times m$ zero matrix. As a consequence of C being non-singular, the matrix A defined in (3.17) has full rank. Hence, the distribution of Z can be obtained according to Section 2.2.6:

$$p_Z(z) = \mathcal{N}\left(z; A \begin{bmatrix} \mu_X \\ \mu_U \end{bmatrix}, A \begin{bmatrix} \Sigma_{XX} & \Sigma_{XU} \\ \Sigma_{UX} & \Sigma_{UU} \end{bmatrix} A^T \right).$$

In particular, Z again has a Gaussian distribution. Further expanding A according to its definition in (3.17), it is found that the mean and covariance of this distribution can be written:

$$\begin{aligned} \mu_Z &= \begin{bmatrix} I_{n,n} & 0_{n,m} \\ B & C \end{bmatrix} \begin{bmatrix} \mu_X \\ \mu_U \end{bmatrix} = \begin{bmatrix} \mu_X \\ B\mu_X + C\mu_U \end{bmatrix}, \\ \Sigma_Z &= \begin{bmatrix} I_{n,n} & 0_{n,m} \\ B & C \end{bmatrix} \begin{bmatrix} \Sigma_{XX} & \Sigma_{XU} \\ \Sigma_{UX} & \Sigma_{UU} \end{bmatrix} \begin{bmatrix} I_{n,n} & B^T \\ 0_{n,m} & C^T \end{bmatrix} \\ &= \begin{bmatrix} \Sigma_{XX} & \Sigma_{XX}B^T + \Sigma_{XU}C^T \\ B\Sigma_{XX} + C\Sigma_{UX} & B\Sigma_{XX}B^T + B\Sigma_{XU}C^T + C\Sigma_{UX}B^T + C\Sigma_{UU}C^T \end{bmatrix}. \end{aligned}$$

The joint distribution of X and Y is finally obtained by defining $Y \triangleq g(X,U)$ and then rewriting $p_Z(z)$ as

$$p_{X,Y}(x,y) = \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}\right) \tag{3.18}$$

with $\Sigma_{X,Y} = \Sigma_{Y,X}^T$ and

$$\begin{gathered} \mu_Y = B\mu_X + C\mu_U, \quad \Sigma_{Y,X} = B\Sigma_{XX} + C\Sigma_{UX}, \\ \Sigma_{Y,Y} = B\Sigma_{XX}B^T + B\Sigma_{XU}C^T + C\Sigma_{UX}B^T + C\Sigma_{UU}C^T. \end{gathered} \tag{3.19}$$

This is the most general form of the Kalman-type linear transform. In most practical applications, the variables X and U can be assumed to be statistically independent, as a consequence of which the cross covariance matrices Σ_{XU} and Σ_{UX} are 0 and (3.19) simplifies to:

$$\mu_Y = B\mu_X + C\mu_U, \quad \Sigma_{Y,X} = B\Sigma_{XX}, \quad \Sigma_{Y,Y} = B\Sigma_{XX}B^T + C\Sigma_{UU}C^T. \tag{3.20}$$

3.4 Local Linearization

As the Gaussian case results in a highly desirable closed-form solution to the Bayesian estimation problem, it is of interest to reduce the limitations brought about by the Kalman-type linear transform. Hence, this section investigates how the results from the previous section can be extended to nonlinear functions $g(x,u)$ by locally linearizing g around the means of X and U. For that, let $\nabla_x g(x,u)$ and $\nabla_u g(x,u)$ denote the Jacobians of g with respect to x and u, respectively:

$$\nabla_x g(x,u) = \begin{bmatrix} \frac{dg_1(x,u)}{dx_1} & \cdots & \frac{dg_1(x,u)}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{dg_m(x,u)}{dx_1} & \cdots & \frac{dg_m(x,u)}{dx_n} \end{bmatrix}, \quad \nabla_u g(x,u) = \begin{bmatrix} \frac{dg_1(x,u)}{du_1} & \cdots & \frac{dg_1(x,u)}{du_m} \\ \vdots & \ddots & \vdots \\ \frac{dg_m(x,u)}{du_1} & \cdots & \frac{dg_m(x,u)}{du_m} \end{bmatrix} \tag{3.21}$$

Then $g(x,u)$ can be linearized around μ_X, μ_U by evaluating $\nabla_x g(x,u)$ and $\nabla_u g(x,u)$ at (μ_X, μ_U) and then forcing a plane with slope $B \triangleq \nabla_x g(\mu_X, \mu_U)$ in x-direction and slope $C \triangleq \nabla_u g(\mu_X, \mu_U)$ in u-direction through the point $g(\mu_X, \mu_U)$:

$$g(x,u) \approx Bx + Cu + \underbrace{g(\mu_X, \mu_U) - B\mu_X - C\mu_U}_{\triangleq b}. \tag{3.22}$$

This is equivalent to a first-order Taylor series expansion around (μ_X, μ_U). After subtraction of b, the Kalman-type linear transform can be used to approximate the transformation of X, U according to g. Hence, the total transformation can be written in analogy to (3.17),

$$Z = \underbrace{\begin{bmatrix} I_{n,n} & 0_{n,m} \\ B & C \end{bmatrix}}_{\triangleq A} \begin{bmatrix} X \\ U \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix}. \tag{3.23}$$

This shows that the joint distribution of X and Y can be calculated according to (3.18) and (3.19) but with μ_Y being translated by b: $\mu_Y = B\mu_X + C\mu_U + b = g(\mu_X, \mu_U)$. Doing so results in a joint Gaussian approximation of $p_{X,Y}(x,y)$:

$$p_{X,Y}(x,y) \approx \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}\right) \tag{3.24}$$

where μ_Y, Σ_{YY} and $\Sigma_{YX} = \Sigma_{XY}^T$ are calculated according to:

$$\begin{gathered} \mu_Y = g(\mu_X, \mu_U), \quad \Sigma_{YX} = B\Sigma_{XX} + C\Sigma_{UX}, \\ \Sigma_{YY} = B\Sigma_{XX}B^T + B\Sigma_{XU}C^T + C\Sigma_{UX}B^T + C\Sigma_{UU}C^T \end{gathered} \tag{3.25}$$

and where $B = \nabla_x g(\mu_X, \mu_U)$ and $C = \nabla_u g(\mu_X, \mu_U)$. It is important to note that this procedure becomes problematic if the Jacobian of g with respect to u becomes singular at (μ_X, μ_U).

3.5 The Unscented Transform

As an alternative to approximating the nonlinear transformation of a Gaussian random variable through local linearization, the transformation can be captured by using Julier and Uhlmann's *unscented transform* (UT) [51]. This transformation essentially

(1) represents the initial Gaussian distribution by a finite number of points, which are chosen in such a way that they have the same mean and covariance as the original distribution.

(2) transforms each of the points according to the nonlinear function.

(3) re-estimates the mean and covariance matrix from the transformed points in order to reobtain a Gaussian fit.

This makes the UT similar in principle to the Monte Carlo transformation from Section 2.5.2, except that (a) the UT is an inherently deterministic method and (b) the re-estimation step converts the point mass representation back to a continuous distribution. In order to give a more formal description, let X be an n-dimensional Gaussian random variable with mean μ_X and covariance matrix Σ_X, such that

$$p_X(x) = \mathcal{N}(x; \mu_X, \Sigma_X).$$

Moreover, let $R^T R$ be the Cholesky decomposition of Σ_X. Then, the point set used by the unscented transform can be described as a weighted empirical distribution

$$\tilde{p}_X(x) = \sum_{i=0}^{2n} W_i \delta(x - \mathcal{X}_i) \tag{3.26}$$

where δ denotes the Dirac delta and where the points $\mathcal{X}_i$ and weights W_i are given by

$$\begin{array}{lcl|lcl} \mathcal{X}_0 & = & \mu_X & W_0 & = & \kappa/\lambda \\ \mathcal{X}_{2i+1} & = & \mu_X + \sqrt{\lambda} R_i & W_{2i+1} & = & 1/(2\lambda) \\ \mathcal{X}_{2i+2} & = & \mu_X - \sqrt{\lambda} R_i & W_{2i+2} & = & 1/(2\lambda) \end{array} \tag{3.27}$$

for $i = 0, \ldots, (n-1)$. In this equation, R_i denotes the i-th row of R, λ is defined as $\lambda \triangleq n + \kappa$, and $\kappa \in \mathbb{R}$ specifies how much weight is placed on the mean, $\mathcal{X}_0$. Choosing a value of $1/2$ results in an equal weight for each of the points. Setting κ to $3 - n$ minimizes the error in the fourth moment [52]. The resulting point set is shown in Figure 3.2 at the example of a two-dimensional Gaussian distribution.

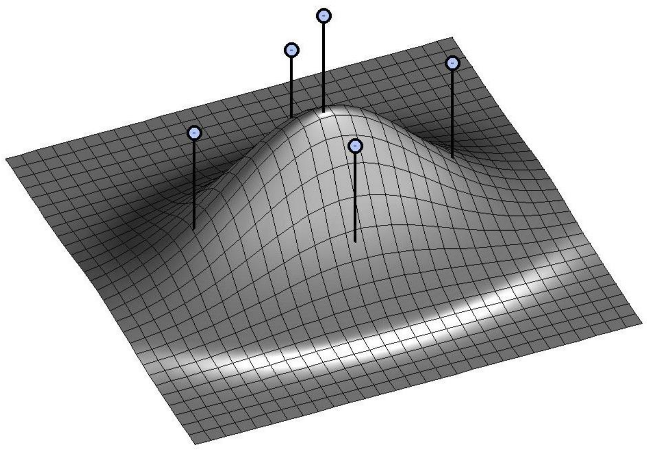


Figure 3.2: Point mass representation used by the unscented transform.

Given this point set, we can now proceed with the next step – transforming the points according to the function $X \underset{f}{\longmapsto} Y = f(X)$:

$$\mathcal{Y}_i = f(\mathcal{X}_i),$$

which can be interpreted as moving the weight W_i placed at $\mathcal{X}_i$ to $f(\mathcal{X}_i) = \mathcal{Y}_i$. Consequently, the distribution of Y is approximated as the weighted empirical distribution:

$$\tilde{p}_Y(y) = \sum_{i=0}^{2n} W_i \delta(y - \mathcal{Y}_i). \tag{3.28}$$

The last step of the unscented transform consists in calculating the mean and covariance of $\tilde{p}_Y(y)$ in order to reobtain a Gaussian approximation $\hat{p}_Y(y) = \mathcal{N}(Y; \hat{\mu}_Y, \hat{\Sigma}_Y)$ of p_Y with

$$\hat{\mu}_Y = \sum_{i=0}^{2n} W_i \mathcal{Y}_i, \quad \hat{\Sigma}_Y = \sum_{i=0}^{2n} W_i (\mathcal{Y}_i - \mu_Y)(\mathcal{Y}_i - \mu_Y)^T. \tag{3.29}$$

These estimates are accurate up to the second order term of the Taylor series expansion [52]. Hence, for linear transformations the unscented transform is exact – i.e. the Gaussian fit it provides is not an approximation but the true distribution of the transformed variable. For nonlinear transformations, the unscented transform is at least as accurate[1] as a first order Taylor series approximation (see Section 3.4), without necessitating the computation of Jacobians and Hessians [52, 53, 54] and without increasing the computational cost [53]. Apart from these properties, the unscented transform can be interpreted as weighted statistical linear regression [55]; and for a certain choice of κ it can even be shown to performs partial Gauss-Hermite

[1] even if second order correction terms are used

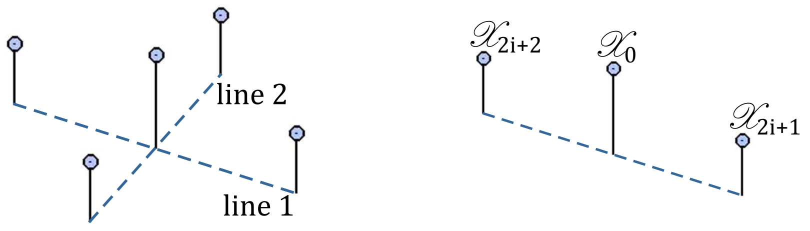
quadrature [56] of the expectation integrals:

$$\mu_Y = \int y p_Y(y) dy, \quad \Sigma_{YY} = \int y y^T p_Y(y) dy. \tag{3.30}$$

In [56] it was further shown that full Gauss-Hermite quadrature gives slightly better results than the unscented transform. This was, however, achieved at an exponential computational expense with regard to the dimension of state space.

3.5.1 Estimating the Degree of Nonlinearity

Regarding the unscented transform, it should be noted that only linear transformations of a Gaussian random variable result in a Gaussian random variable. Nonlinear transformations inevitably lead to non-Gaussian distributions, which cause approximation errors in the Gaussian fits. This triggered the idea of developing a novel measure [19, 12] for the appropriateness of the unscented transform based on the approximation error of the Gaussian fits. This measure is here called the *"degree of nonlinearity"*; and it is based on the fact that, in the unscented transform, each triple $\{\mathcal{X}_{2i+1}, \mathcal{X}_0, \mathcal{X}_{2i+2}\}$ forms a set of equidistant points on a line, as portrayed below:



If the transformation $Y = g(X)$ is linear, the triple $\{\mathcal{Y}_{2i+1}, \mathcal{Y}_0, \mathcal{Y}_{2i+2}\}$ of transformed points will also form a set of equidistant points on a line. This observation motivates the idea of determining the degree of nonlinearity by

(1) fitting to each $\mathcal{Y}^{(i)} = \{\mathcal{Y}_{2i+1}, \mathcal{Y}_0, \mathcal{Y}_{2i+2}\}$ a set $\mathcal{Z}^{(i)} = \{\mathcal{Z}_{2i+1}, \mathcal{Z}_0, \mathcal{Z}_{2i+2}\}$ of equidistant points on a line.

(2) calculating the degree of nonlinearity of each triple as the squared error that is introduced by the linear fit.

(3) summing the degrees of nonlinearities of the individual triples.

In order to obtain an equidistant linear fit, the points of each linearized triple $\{\mathcal{Z}_{2i+1}, \mathcal{Z}_0, \mathcal{Z}_{2i+2}\}$ may (without loss of generality) be written:

$$\mathcal{Z}_0 = b, \quad \mathcal{Z}_{2i+1} = b + a_i, \quad \mathcal{Z}_{2i+2} = b - a_i. \tag{3.31}$$

Then the linearized set $\mathscr{X}$ of points can be determined by minimizing the squared error between the two sets, $\mathscr{Y}^{(i)}$ and $\mathscr{X}^{(i)}$. This is achieved by taking the derivatives of the squared error

$$\begin{aligned} \mathrm{ERR}\big(\mathscr{X}^{(i)},\mathscr{Y}^{(i)}\big) &\triangleq (\mathscr{X}_0-\mathscr{Y}_0)^2+(\mathscr{X}_{2i+1}-\mathscr{Y}_{2i+1})^2+(\mathscr{X}_{2i+2}-\mathscr{Y}_{2i+2})^2 \\ &= (b-\mathscr{Y}_0)^2+(b+a_i-\mathscr{Y}_{2i+1})^2+(b-a_i-\mathscr{Y}_{2i+2})^2 \end{aligned}$$

with respect to a_i and b and then equating the result to zero. Doing so gives $a_i=\frac{1}{2}(\mathscr{Y}_{2i+1}-\mathscr{Y}_{2i+2})$ and $b=\overline{\mathscr{Y}^{(i)}}\triangleq\frac{1}{3}(\mathscr{Y}_0+\mathscr{Y}_{2i+1}+\mathscr{Y}_{2i+2})$, which when plugged back into 3.31 yields:

$$\begin{aligned} \mathscr{X}_0 &= \overline{\mathscr{Y}^{(i)}} \\ \mathscr{X}_{2i+1} &= \overline{\mathscr{Y}^{(i)}}+\tfrac{1}{2}(\mathscr{Y}_{2i+1}-\mathscr{Y}_{2i+2}) \\ \mathscr{X}_{2i+2} &= \overline{\mathscr{Y}^{(i)}}-\tfrac{1}{2}(\mathscr{Y}_{2i+1}-\mathscr{Y}_{2i+2}) \end{aligned} \tag{3.32}$$

for $i=0,\ldots,(n-1)$. Now defining the degree of nonlinearity η_i of this triple $\mathscr{X}^{(i)}=\{\mathscr{X}_0,\mathscr{X}_{2i+1},\mathscr{X}_{2i+2}\}$ as the linearization error $\eta_i\triangleq\mathrm{ERR}\big(\mathscr{X}^{(i)},\mathscr{Y}^{(i)}\big)$ between $\mathscr{X}^{(i)}$ and $\mathscr{Y}^{(i)}$, we arrive at:

$$\eta_i = \left\|\mathscr{Y}_0-\overline{\mathscr{Y}^{(i)}}\right\|^2+\frac{1}{2}\left\|\mathscr{Y}_{2i+1}+\mathscr{Y}_{2i+2}-2\overline{\mathscr{Y}^{(i)}}\right\|^2 = \frac{1}{6}\left\|\mathscr{Y}_{2i+1}+\mathscr{Y}_{2i+2}-2\mathscr{Y}_0\right\|^2. \tag{3.33}$$

Further normalizing this term by the empirical variance of $\mathscr{Y}^{(i)}=\{\mathscr{X}_0,\mathscr{X}_{2i+1},\mathscr{X}_{2i+2}\}$ yields a normalized measure for the degree of nonlinearity:

$$\eta_i'=\frac{\frac{1}{6}\left\|\mathscr{Y}_{2i+1}+\mathscr{Y}_{2i+2}-2\mathscr{Y}_0\right\|^2}{\left\|\mathscr{Y}_0-\overline{\mathscr{Y}^{(i)}}\right\|^2+\left\|\mathscr{Y}_{2i+1}-\overline{\mathscr{Y}^{(i)}}\right\|^2+\left\|\mathscr{Y}_{2i+2}-\overline{\mathscr{Y}^{(i)}}\right\|^2}, \tag{3.34}$$

The nominator of this equation accounts for the variance or modeling error introduced by the linear fit. The denominator normalizes that term by the variance of the points. It can be shown in particular that (3.34) takes values in the range $[0,1]$, as it has the following relationship to the R^2 measure[2] from linear regression: $\eta_i'=1-R^2$. Subsequently, the total degree of nonlinearity is obtained by averaging the η_i or η_i' for $i=0,\ldots,(n-1)$:

$$\eta\triangleq\frac{1}{n}\sum_{i=0}^{n-1}\eta_i. \tag{3.35}$$

If this value is close to zero the transformation is approximately linear and the Gaussian approximation of $p_Y(y)$ is justified. For larger values, the parametric Gaussian fit might not well represent the true distribution.

[2] The R^2 measure is a measure for the "goodness of fit". It is commonly used in linear regression; it is also known as coefficient of determination; and it takes values between zero and one [57].

Extension to Other Transformations

Note that both the normalized and non-normalized measure for the degree of nonlinearity can easily be extended to related approaches such as the extensive unscented transform (see Section 3.6) or Gauss-Hermite quadrature [56]. Just consider lines in one variable or coefficient and then average over all possible values that the other variables or coefficients can assume. This is demonstrated in Section 3.6.3 at the example of the extensive unscented transform.

3.5.2 Stacking of Variables / Augmentation

In order to use the unscented transform for Bayesian state estimation, the UT needs to be extended to the transformation of several Gaussian random variables at a time. Let $X^{(1)},\ldots,X^{(m)}$ denote statistically independent variables that are to be transformed, with

$$p_{X^{(i)}}\left(x^{(i)}\right)=\mathcal{N}\left(x^{(i)};\mu_{X^{(i)}},\Sigma_{X^{(i)}}\right),$$

and let $Y=f(X^{(1)},\ldots,X^{(m)})$ denote the transformation. Then, the individual $X^{(i)}$ can be written as a single, joint Gaussian random variable $Z\triangleq\begin{bmatrix}X^{(1)T} & \cdots & X^{(m)T}\end{bmatrix}^T$ with distribution

$$p_Z\left(\begin{bmatrix}x^{(1)}\\x^{(2)}\\\vdots\\x^{(m)}\end{bmatrix}\right)=\mathcal{N}\left(\begin{bmatrix}x^{(1)}\\x^{(2)}\\\vdots\\x^{(m)}\end{bmatrix};\begin{bmatrix}\mu_{X^{(1)}}\\\mu_{X^{(2)}}\\\vdots\\\mu_{X^{(m)}}\end{bmatrix},\begin{bmatrix}\Sigma_{X^{(1)}} & 0 & \cdots & 0\\0 & \Sigma_{X^{(2)}} & \ddots & \vdots\\\vdots & \ddots & \ddots & 0\\0 & \cdots & 0 & \Sigma_{X^{(m)}}\end{bmatrix}\right). \tag{3.36}$$

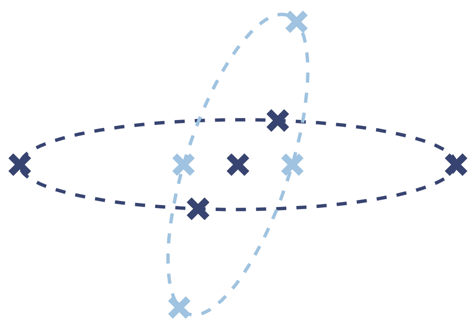
After *"stacking"* the variables as described above, the unscented transform can be applied as in the case of a single variable. This can be used for Bayesian state estimation, in order to construct the joint distribution of state and observation through transformation of X,U according to $\tilde{g}$ (see Section 3.1.2). Figure 3.3 depicts the resulting point set at the example of two 2-dimensional Gaussian random variables $X^{(1)}$ and $X^{(2)}$. Each of the 5 points chosen for $X^{(1)}$ is augmented with the mean of $X^{(2)}$; conversely, each of the 5 points chosen for $X^{(2)}$ is augmented with the mean of $X^{(1)}$. Following the naming convention in [18], the unscented transform of stacked variables is here called *augmented unscented transform*[3] (AUT).

3.5.3 Estimating the Degree of Nonlinearity for Stacked Variables

When the unscented transform is applied to a stacked variable, as described in the previous section, the degree of nonlinearity may be calculated individually for each of the variables [12]. In order to explain this, let $X^{(j)}$ be an n_j-dimensional Gaussian random variable with distribution

$$p(x^{(j)})=\mathcal{N}(x^{(j)},\mu_{X^{(j)}},\Sigma_{X^{(j)}}),$$

[3] The inspiration for this name was that Julier and Uhlman [51, 52] call the procedure of stacking "augmentation".



$$\begin{aligned}
\mathscr{X}_0 &= \left[\mu_U^T \quad \mu_{\mathbf{V}}^T\right]^T \\
\mathscr{X}_{2i+1} &= \left[(\mu_U + \sqrt{\lambda} A_i)^T \quad \mu_V^T\right]^T \\
\mathscr{X}_{2i+2} &= \left[(\mu_U - \sqrt{\lambda} A_i)^T \quad \mu_V^T\right]^T \\
\mathscr{X}_{2(n+j)+1} &= \left[\mu_U^T \quad (\mu_V + \sqrt{\lambda} B_j)^T\right]^T \\
\mathscr{X}_{2(n+j)+2} &= \left[\mu_U^T \quad (\mu_V - \sqrt{\lambda} B_j)^T\right]^T
\end{aligned}$$

Figure 3.3: *Points (crosses) used by the augmented unscented transforms along with rescaled covariance ellipses of the corresponding Gaussians (dashed lines). Note that the originally 4-dimensional picture has been projected to a 2-dimensional one. The dark blue crosses indicate the points chosen for $U = X^{(1)}$, augmented with the mean of $V = X^{(2)}$. The light blue crosses indicate the points chosen for $V = X^{(2)}$, augmented with the mean of $U = X^{(1)}$, μ_U. A and B denote the Cholesky factors of Σ_U and Σ_V.*

for $j = 1, \ldots, m$. Furthermore, let the $X^{(j)}$ be statistically independent and let Z denote the stacked variable $Z \triangleq \left[X^{(1)T} \quad \cdots \quad X^{(m)T}\right]^T$. Then the joint covariance matrix Σ_Z of the variables has block-diagonal form, with the blocks being the covariance matrices $\Sigma_{X^{(j)}}$ of the individual variables:

$$\Sigma_Z = \begin{bmatrix} \Sigma_{X^{(1)}} & 0 & \cdots & 0 \\ 0 & \Sigma_{X^{(2)}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \Sigma_{X^{(m)}} \end{bmatrix}. \tag{3.37}$$

Consequently, the Cholesky factorization R of Σ_Z has block-diagonal form, with the blocks being the right Cholesky factors $R^{(j)}$ of the individual $\Sigma_{X^{(j)}}$. This means the points considered for $X^{(j)}$,

$$\{\mathscr{X}_{2a_j-1}, \ldots, \mathscr{X}_{2b_j}\}, \text{ with } a_j = 1 + \sum_{i=1}^{j-1} n_i, \; b_j = \sum_{i=1}^{j} n_i,$$

differ only in the coordinates a_j to b_j. All the other points $\mathscr{X}_k$, $k \notin [2a_j - 1, 2b_j]$, have these coordinates fixed to $\mu_{X^{(j)}}$. Hence, the degree of nonlinearity contributed by $X^{(j)}$ may be calculated as

$$\eta^{(j)} = \frac{1}{n} \sum_{i=a_j}^{b_j} \eta_i. \tag{3.38}$$

3.6 The Extensive Unscented Transform

As an alternative to the augmented unscented transform from Section 3.5.2, the transformation of several statistically independent random variables may be performed by representing the variables as separate point-sets and then considering all possible combinations[4] thereof. The motivation for this procedure is to preserve statistical independence of the variables. This stands in contrast to the augmented unscented transform (where the independence is notably not preserved) [18]. The name *extensive unscented transform* (XUT) [18] refers to the fact that the procedure uses all possible combinations of points from different variables.

3.6.1 Description of the Transform

Let X and U be n and m dimensional Gaussian random variables as in the Kalman-type linear transform from section 3.3. In particular, let X and U be statistically independent. Then, the distributions of X and U can both be approximated with an unscented point-mass representation:

$$\tilde{p}_X(x) = \sum_{i=0}^{2n} W_i^{(X)} \delta(x - \mathcal{X}_i), \quad \tilde{p}_U(u) = \sum_{j=0}^{2m} W_j^{(U)} \delta(u - \mathcal{U}_j)$$

where the points and weights, $\mathcal{X}_i$, $W_i^{(X)}, i = 1, \dots, (2n+1)$ and $\mathcal{U}_j$, $W_j^{(U)}, j = 1, \dots, (2m+1)$, are chosen as in (3.27), respectively. These individual point mass representations of X and U can obviously be used to approximate the joint distribution $p_{X,U}(x,u) = P_X(x) p_U(u)$ as the product of $\tilde{p}_X(x)$ and $\tilde{p}_U(u)$:

$$\tilde{p}_{X,U}(x,u) = \sum_{i=0}^{2n} \sum_{j=0}^{2m} W_i^{(X)} W_j^{(U)} \delta(x - \mathcal{X}_i) \delta(u - \mathcal{U}_j). \tag{3.39}$$

Consequently, the joint distribution of X and Y can be approximated by transforming the points $\begin{bmatrix} \mathcal{X}_i^T & \mathcal{U}_j^T \end{bmatrix}^T$ according to the augmented observation function $\tilde{g}(x,u) = \begin{bmatrix} x^T & g(x,u)^T \end{bmatrix}^T$ from Section 3.1. This leads to the weighted empirical distribution

$$\tilde{p}_{X,Y}(x,y) = \sum_{i=0}^{2n} \sum_{j=0}^{2m} W_{i,j}^{(Y)} \delta(x - \mathcal{X}_i) \delta(y - \mathcal{Y}_{i,j})$$

where $\mathcal{Y}_{i,j} = g(\mathcal{X}_i, \mathcal{U}_j)$ and $W_{i,j}^{(Y)} = W_i^{(X)} W_j^{(U)}$. Estimating the mean $\hat{\mu}_{X,Y}$ and covariance $\hat{\Sigma}_{X,Y}$ of the transformed points finally gives a Gaussian approximation $\hat{p}_{X,Y}$ [13] with

$$\hat{\mu}_{X,Y} = \sum_{i=0}^{2n} \sum_{j=0}^{2m} W_{i,j}^{(Y)} \begin{bmatrix} \mathcal{X}_i \\ \mathcal{Y}_{i,j} \end{bmatrix}, \qquad \hat{\Sigma}_{X,Y} = \sum_{i=0}^{2n} \sum_{j=0}^{2m} W_{i,j}^{(Y)} \begin{bmatrix} \mathcal{X}_i^T & \mathcal{Y}_{i,j}^T \end{bmatrix} \begin{bmatrix} \mathcal{X}_i \\ \mathcal{Y}_{i,j} \end{bmatrix}. \tag{3.40}$$

3.6.2 Comparison to the Augmented Unscented Transform

Figure 3.4 shows the difference between the augmented and extensive unscented transforms. The AUT augments each of the $(2n+1)$ points chosen for X with the mean of U and, conversely, each of the $(2m+1)$ points chosen for U with the mean of X. This results in the points on the

[4] i.e. the Cartesian product of the point sets

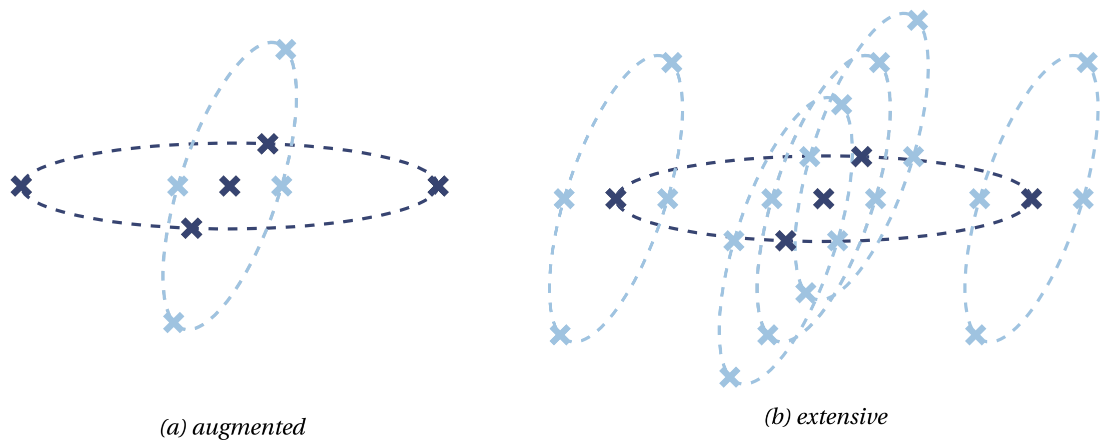


Figure 3.4: *Augmented versus Extensive Unscented Transform. The pictures to the left and to the right show the points (crosses) used by the augmented and extensive unscented transforms along with covariance ellipses (dashed lines). The dark blue crosses indicate the $2n+1=5$ points chosen for X, augmented with the mean of U. The light blue crosses indicate the points chosen for U, augmented with the mean of X (left) or with each of the points chosen for X (right), respectively.*

two ellipses in Figure 3.4-(a). The XUT uses all possible combinations of points that the AUT considers for X and U individually. Hence, the XUT considers $(2n+1)\cdot(2m+1)$ points instead of the $2(n+m)+1$ points used by the AUT. This increase in the number of points causes a growth of the computational expense to roughly $\mathcal{O}\left(\max\{n,m\}^4\right)$, up from $\mathcal{O}\left(\max\{n,m\}^3\right)$. But this increase is still low compared to full Gauss-Hermite quadrature [56] for which the computational expense grows exponentially with the dimension.

Statistical Independence

On the first look it might be a puzzling why the augmented unscented transform should violate the statistical independence of variables. The stacking procedure of the augmented unscented transform sets all the cross-variable covariance terms of the joint covariance matrix to zero (see (3.36)), which obviously preserves the uncorrelatedness of the variables. In the Gaussian case, this even implies statistical independence. The point where things go awry is the point selection mechanism of the unscented transform. As shown in Appendix II of [52], this mechanism changes the higher-order moments between the variables. In particular, for $n = m = 1$ and even integers $k, l \geq 2$, the higher order expectations $\mathcal{E}_{p_{X,U}(x,u)}\left\{x^k u^l\right\}$ are approximated as

$$\mathcal{E}_{\tilde{p}_{X,U}(x,u)}\left\{x^k u^l\right\} = W_0 \mu_X^k \mu_U^l + \underbrace{\textstyle\sum_{i=1}^{2n} W_i \mathcal{X}_i^k}_{=\mathcal{E}_{\tilde{p}_X(x)}\{x^k\}} \mu_U^l + \mu_X^k \underbrace{\textstyle\sum_{i=2n+1}^{2(n+m)} W_i \mathcal{U}_i^l}_{=\mathcal{E}_{\tilde{p}_U(u)}\{u^l\}}, \tag{3.41}$$

This clearly violates $\mathcal{E}_{\tilde{p}_{X,U}(x,u)}\left\{x^k u^l\right\} = \mathcal{E}_{\tilde{p}_X(x)}\left\{x^k\right\}\mathcal{E}_{\tilde{p}_U(u)}\left\{u^l\right\}$, which we should have in the case of independent variables. The extensive unscented transform does not have this problem as its points set is designed to get these moments right by definition. This means that approximation errors stay confined to the individual variables. The higher order moments between the variables are not changed [18].

Other Properties

Apart from preserving the statistical independence of the point-sets, the extensive unscented transform correctly captures the mean and covariance of $p_{X,U}(x,u)$. Hence, it is exact up to the second order term of the Taylor series expansion, just as the UT is [52]. In the particular case where $n = m = 1$ and κ from Section 3.5 is set to 2, the XUT coincides with the 3-point Gauss-Hermite quadrature rule [56]. For higher dimensional cases, its accuracy is somewhere between the unscented transform and full Gauss Hermite quadrature [18].

3.6.3 Estimating the Degree of Nonlinearity

As mentioned in Section 3.5.1, the degree of nonlinearity can be extended to the extensive unscented transform, by simply considering lines in one variable and then averaging the degree of nonlinearity over all possible values that the other variables can assume. This gives the following estimates $\nu^{(X)}$ and $\nu^{(U)}$ for X and U:

$$\nu^{(X)} = \frac{1}{n\cdot(2m+1)}\sum_{i=1}^{n}\sum_{j=0}^{2m}\underbrace{\frac{1}{6}\left\|\mathcal{Y}_{2i+1,j}+\mathcal{Y}_{2i+2,j}-\mathcal{Y}_{0,j}\right\|}_{\nu_{i,j}^{(X)}}, \tag{3.42}$$

$$\nu^{(U)} = \frac{1}{(2n+1)\cdot m}\sum_{i=0}^{2n}\sum_{j=1}^{m}\underbrace{\frac{1}{6}\left\|\mathcal{Y}_{i,2j+1}+\mathcal{Y}_{i,2j+2}-\mathcal{Y}_{i,0}\right\|}_{\nu_{i,j}^{(U)}}, \tag{3.43}$$

if the degree of nonlinearity is calculated individually for each of the variables (see Section 3.5.2). The total degree of nonlinearity is obtained by averaging $\nu^{(X)}$ and $\nu^{(U)}$.

3.7 The Adaptive Level of Detail Transform

The transformation methods considered so far were all based on the assumption that the distribution of a transformed random variable can be well approximated by a Gaussian. This assumption is perfectly reasonable for linear and approximately linear transformations. But it may be very inappropriate in the presence of considerable nonlinearities, especially if the true transformed distribution is multimodal. Hence, Alspach and Sorenson [48] proposed to approximate the distribution of the variable to be transformed by a mixture of Gaussians, under the rationale that this way the variances might be chosen small enough for the local linearizations to be valid for each of the Gaussians. Consequently, a more accurate approximation of the transformed distribution was obtained by individually transforming the Gaussian components and then recombining the transformed distributions to a Gaussian mixture.

In the framework of this thesis, that approach has been extended by a method which adapts the level of detail of the Gaussian mixture distribution to the nonlinearities present in the transformation [19, 12, 13]. "Adapting the level of detail", in this context, means keeping more Gaussians (with smaller variances) in regions where the linearization error is high and fewer Gaussians (with larger variances) in regions where it is low. This concept is explained in more detail in the following, starting with Alspach and Sorenson's approach.

3.7.1 Alspach and Sorenson's Approach

In Alspach and Sorenson's approach [48], the original distribution $p_X(x) = \mathcal{N}(x, \mu_X, \Sigma_X)$ of the random variable, X, to be transformed is approximated as a mixture of Gaussian distributions:

$$p_X(x) \approx \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}(x; \mu_X^{(k)}, \Sigma_X^{(k)})}_{=p_{X|k}(x)} \tag{3.44}$$

where c_k, $\mu_X^{(k)}$ and $\Sigma_X^{(k)}$ denote the weights, means and covariance matrices of the Gaussians. With this representation, a transformation $Y = f(X)$ of the variable X can be approximated by individually transforming the mixture components $X|k$ and then recombining the resulting distributions to a Gaussian mixture. In the original work [48], each of these individual transformations

$$X|k \underset{f}{\longmapsto} Y|k = f(X|k)$$

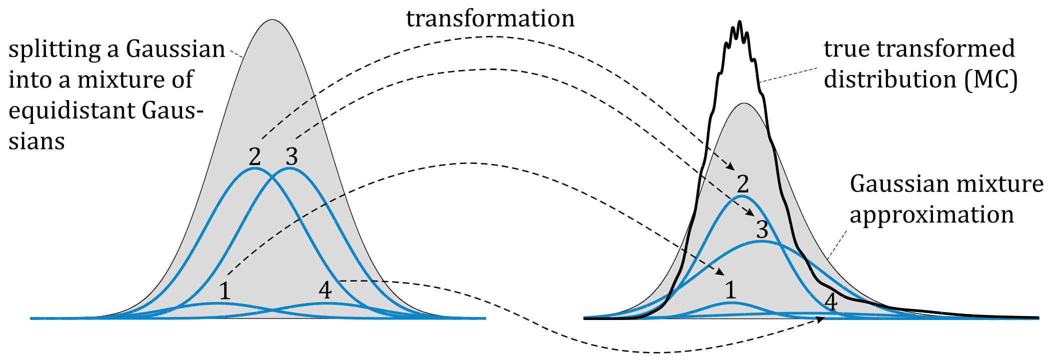
was approximated with the method of local linearization. This thesis uses the unscented transform [19, 12, 13] for which the resulting Gaussian mixture approximation of Y may be written

$$p_Y(y) \approx \sum_{k=1}^{K} c_k \underbrace{\mathrm{UT}\{p_{X|k}, f\}(y)}_{=p_{Y|k}(y)} \tag{3.45}$$

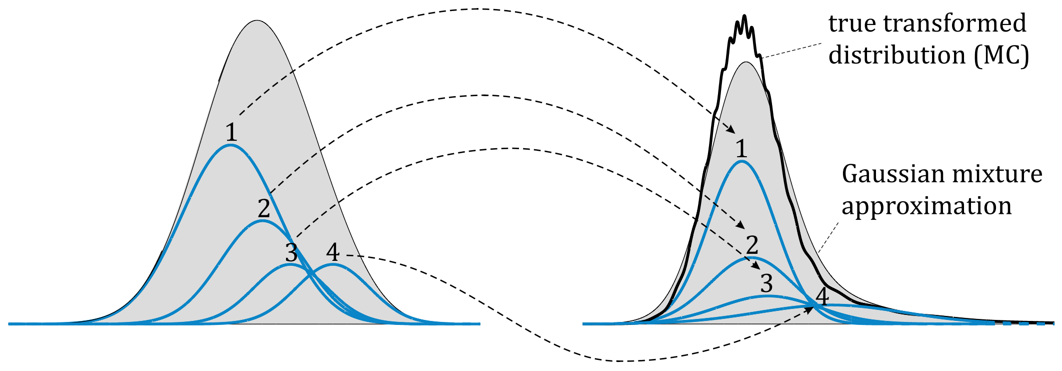
where $\mathrm{UT}\{p_{X|k}, f\}$ denotes the unscented transform of the Gaussian random variable $X|k$ with respect to the function f. This can be used to construct the joint Gaussian mixture distribution for the Bayesian state estimation approach from Section 3.2.2. Simply choose f to be the augmented observation function $\tilde{g}$ from Section 3.1.2 and use it to transform the stacked variable of X and U.

3.7.2 Adapting the Level of Detail

As mentioned before, the idea behind Alspach and Sorenson's approach is to use a sufficiently large number of mixture components such that the individual variances can be chosen small enough for f to be approximately linear for each of the transformations. In order to achieve this, the Gaussians are typically arranged on an equidistant grid with equal covariance matrices and the mixture weights are optimized so as to minimize the mean squared error to the true distribution p_X [48]. Such a replacement of a Gaussian by an equidistant grid of Gaussians can be regarded as increasing the level of detail in a uniform fashion:


transformation
splitting a Gaussian into a mixture of equidistant Gaussians
true transformed distribution (MC)
Gaussian mixture approximation
1
2
3
4

This is beneficial in nonlinear regions as it decreases the variances and thereby the nonlinearities to which the Gaussians are subject during transformation. In linear regions, however, the Gaussian mixture approach cannot improve over a single unscented transform, as the unscented transform is exact in the linear Gaussian case. Hence, the aim should be to keep fewer Gaussians in relatively linear regions where the transformation is accurate and more Gaussians in nonlinear regions where the linearization error is higher:


true transformed distribution (MC)
Gaussian mixture approximation
1
2
3
4

The *adaptive level of detail transform* (ALoDT) [12] tries to achieve this in an iterative fashion, by starting with a single Gaussian and then iteratively splitting that Gaussian component which is subject to the highest degree of nonlinearity. This procedure is visualized in Figure 3.5 and and more formally described in Algorithm 3.1. The algorithm shown there accepts, as an input argument, the Gaussian distribution p_X of the random variable to be transformed as well as the function f, which specifies the transformation. It returns a Gaussian mixture approximation p_Y of the distribution of the transformed random variable $Y = f(X)$. $\text{UT}\{p_X, f\}$ again denotes the unscented transform of the distribution p_X according to f, and $\text{dnl}\{p_X, f\}$ denotes the corresponding degree of nonlinearity, which is calculated according to (3.35).

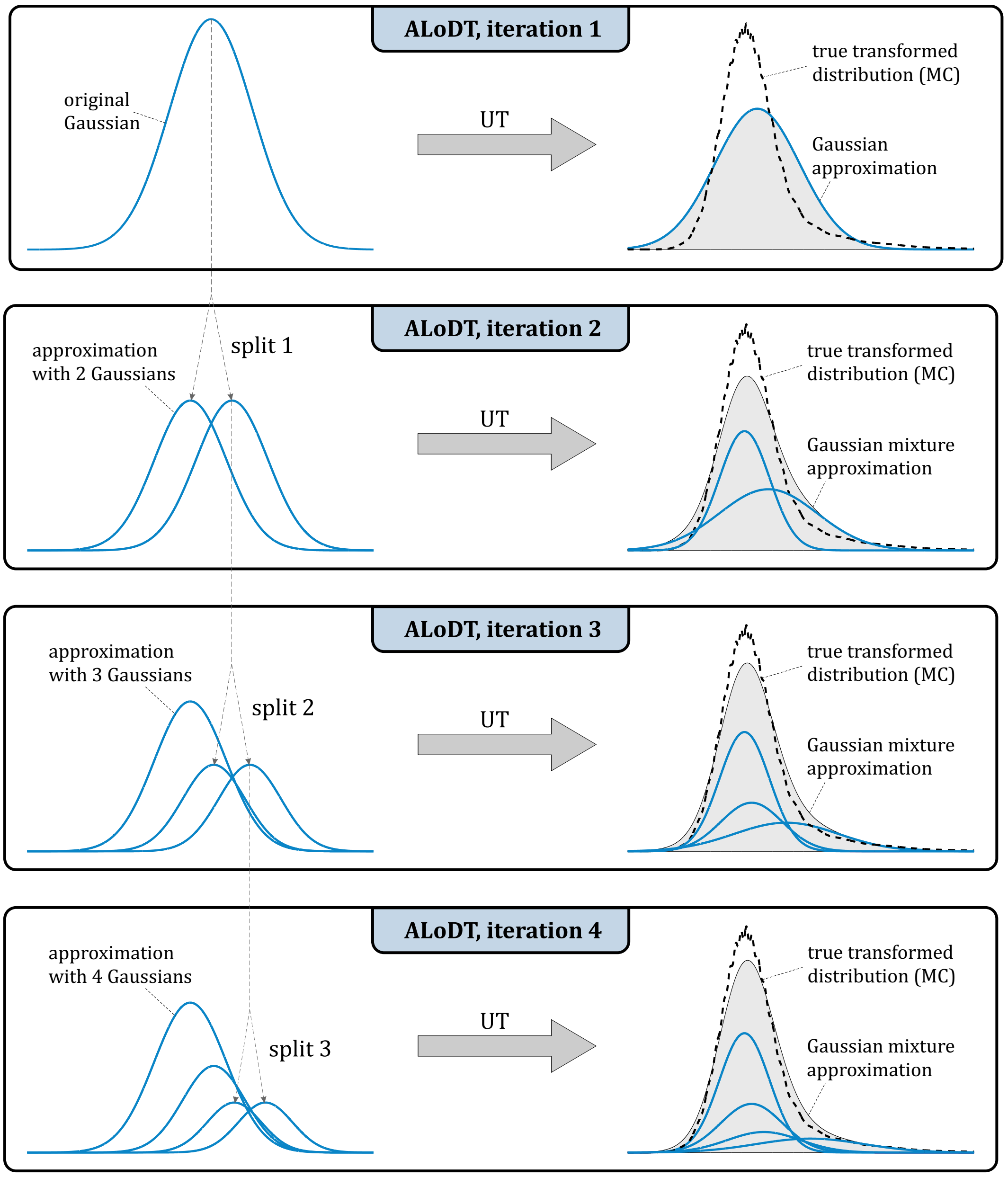

ALoDT, iteration 1
original Gaussian
UT
true transformed distribution (MC)
Gaussian approximation
ALoDT, iteration 2
approximation with 2 Gaussians
split 1
UT
true transformed distribution (MC)
Gaussian mixture approximation
ALoDT, iteration 3
approximation with 3 Gaussians
split 2
UT
true transformed distribution (MC)
Gaussian mixture approximation
ALoDT, iteration 4
approximation with 4 Gaussians
split 3
UT
true transformed distribution (MC)
Gaussian mixture approximation

Figure 3.5: *Visualization of the adaptive level of detail transform (ALoDT). The transformation starts with a single Gaussian and then refines the result by iteratively splitting that Gaussian for which the linearization error (degree of nonlinearity) is largest. The name ALoDT is motivated by the fact that the above procedure can be viewed as adapting the level of detail of the Gaussian mixture representation to the degree of nonlinearity of the transformation.*

The Adaptive Level of Detail Transform – $p_Y = \text{ALoDT}(p_X, f)$

1. initialize the algorithm with the original Gaussian distribution p_X of X by
   (a) setting the list $\mathscr{S}_X$ of non-transformed Gaussians to $\mathscr{S}_X = \{p_X\}$
   (b) setting the list $\mathscr{S}_c$ of weights to $\mathscr{S}_c = \{1\}$
   (c) setting the list $\mathscr{S}_Y$ of transformed Gaussians to $\mathscr{S}_Y = \left\{\text{UT}\{p_X, f\}\right\}$
   (d) setting the list $\mathscr{S}_\eta$ of splitting priorities to $\mathscr{S}_\eta = \{\text{dnl}\{p_X, f\}\}$
2. identify the Gaussian component with the highest degree of nonlinearity by:
   (a) calculating $i = \text{argmax}_{j\in\{1,\dots,\text{length}(\mathscr{S}_\eta)\}} \mathscr{S}_\eta[j]$
   (b) setting g_i to the i-th component of $\mathscr{S}_X$: $g_i = \mathscr{S}_X[i]$
   (c) setting c_i to the i-th component of $\mathscr{S}_c$: $c_i = \mathscr{S}_c[i]$
   (d) setting η_i to the i-th component of $\mathscr{S}_\eta$: $\eta_i = \mathscr{S}_\eta[i]$
3. remove the i-th element from $\mathscr{S}_X$, $\mathscr{S}_c$, $\mathscr{S}_Y$ and $\mathscr{S}_\eta$
4. split g_i into n Gaussians $g_{i,1}, \dots, g_{i,n}$ with associated weights $c_{i,1}, \dots, c_{i,n}$ as described in Section 3.7.3
5. for each $j \in \{1, \dots, n\}$:
   (a) compute the new weight $c'_{i,j} = c_i \cdot c_{i,j}$
   (b) compute the transformed distribution $g'_{i,j} = \text{UT}\{g_{i,j}, f\}$
   (c) compute the degree of nonlinearity $\eta_{i,j} = \text{dnl}\{g_{i,j}, f\}$
   (d) add $g_{i,j}$, $c'_{i,j}$, $g'_{i,j}$ and $\eta_{i,j}$ to $\mathscr{S}_X$, $\mathscr{S}_c$, $\mathscr{S}_Y$ and $\mathscr{S}_\eta$, respectively
6. while the number of Gaussians, i.e. length($\mathscr{S}_X$), is smaller than N and the maximum degree of nonlinearity η_i is larger than a threshold τ: go back to step 2
7. construct the Gaussian mixture approximation of Y as $p_Y(y) = \sum_{i=1}^{\text{length}(\mathscr{S}_X)} \mathscr{S}_c[i]\mathscr{S}_Y[i](y)$

Algorithm 3.1: Adaptive Level of Detail Transform

Modified Splitting for Stacked Variables

4. split g_i into n Gaussians $g_{i,1}, \dots, g_{i,n}$ by
   (a) un-stacking X by separating the stacked distribution g_i of $X^{(1)}, \dots, X^{(m)}$ into variable-dependent distributions $g_i^{(1)}, \dots, g_i^{(m)}$, which is possible if the variables are statistically independent
   (b) setting k_{max} to the index of the variable $X^{(k)}$ for which the degree of nonlinearity is highest: $k_{max} = \text{argmax}_k \, \eta_i^{(k)}$ (see (3.38))
   (c) splitting the variable $X^{(k_{max})}$ with the highest degree of nonlinearity by splitting $g_i^{(k_{max})}$ into n Gaussians $g_{i,1}^{(k_{max})}, \dots, g_{i,n}^{(k_{max})}$ with associated weights $c_{i,1}, \dots, c_{i,n}$ as described in Section 3.7.3
   (d) re-stacking X by setting $g_{i,j} = \text{stack}\left\{g_i^{(1)}, \dots, g_i^{(k_{max}-1)}, g_{i,j}^{(k_{max})}, g_i^{(k_{max}+1)}, \dots, g_i^{(m)}\right\}$, where the stack-operator creates a joint Gaussian distribution by stacking the means and building the canonical block-diagonal covariance matrix as in Section 3.5.2

Algorithm 3.2: ALoDT - Modified Splitting Procedure for Stacked Variables

3.7.2.1 Splitting Priority

Selecting the mixture component to be split based only on its degree of nonlinearity can result in repeated splits of components whose weights are getting smaller and smaller. This is not desirable, as components with a very low weight represent only a small amount of probability mass and thereby do not contribute much to the transformation. Hence, in this thesis it has been proposed to replace the splitting criterion from the previous section – the component's degree of nonlinearity $\mathrm{dnl}(g,f)$ – by the *splitting priority* $\mathrm{spp}(g,f,c)$ [12]:

$$\mathrm{spp}(g,f,c) \triangleq c \cdot \mathrm{dnl}(g,f) \tag{3.46}$$

where c denotes the mixture weight of the Gaussian component g. The advantage of this criterion is that it captures the total linearization error which a component contributes to the transformation, rather than just the linearization error of the individual Gaussian. Its integration into the adaptive level of detail transform (Algorithm 3.1) requires the following changes:

(1) in step 1d, the degree of nonlinearity, $\mathrm{dnl}\{p_X,f\}$ needs to be replaced by $\mathrm{spp}(p_X,f,1)$

(2) in step 5c, $\mathrm{dnl}\{g_{i,j},f\}$ needs to be replaced by $\mathrm{spp}(g_{i,j},f,c'_{i,j})$.

3.7.2.2 Treatment of Stacked Variables

For the transformation of m statistically independent random variables $X^{(1)},\dots,X^{(m)}$, Algorithm 3.1 needs to be modified. Firstly, the variables have to be stacked before transformation, as described in Section 3.5.2; and the degree of nonlinearity in step 1d has to be calculated individually for each of the variables (according to (3.38)). Consequently, the scalar degree of nonlinearity, η, is replaced by a vector of degrees of nonlinearity with one component for each variable:

$$\eta = \begin{bmatrix} \eta^{(1)} & \eta^{(m)} \end{bmatrix}^T .$$

The splitting step (step 4 of Algorithm 3.1) is modified to split only that variable $X^{(k_{\max})}$ which has the highest degree of nonlinearity: $k_{\max} = \mathrm{argmax}_k\, \eta_i^{(k)}$. This is done as described in Algorithm 3.2. All the other variables $X^{(k)}$, $k \neq k_{\max}$ remain unchanged. Regarding the comparison in the "while" instruction of Step 6, the scalar degree of nonlinearity, η_i, is replaced with the maximum of the vector η_i: $\max_k \eta_i^{(k)}$.

3.7.2.3 Other Extensions

The adaptive level of detail transform can easily be extended to transforming Gaussian mixture random variables, simply by initializing Algorithm 3.1 with a Gaussian mixture instead of a single Gaussian. For that, each individual Gaussian is transformed with the unscented transform and its degree of nonlinearity is calculated. After the transformation, the complexity of the Gaussian mixture approximation can be reduced with Gaussian mixture reduction techniques

[58, 59, 60, 61]. This is of particular interest in sequential Bayesian estimation (see Chapter 4) where the number of mixture components increases exponentially in time.

3.7.3 Splitting Gaussian Distributions

Another important implementation aspect of the adaptive level of detail transform concerns how multivariate Gaussians can be "split" into a mixture of Gaussians with smaller variances. In contrast to earlier approaches, which have been based on the use of splitting libraries [62, 63] or a slight displacement of the means [64, 62], this work considers a moment matching based approach that makes use of the symmetry of the Gaussian distribution [19, 13]. In order to avoid problems with indefinite covariance matrices, splitting is considered in the direction of eigenvectors only. The direction in which the Gaussian distribution is split might be given by the eigenvector corresponding to the largest eigenvalue – that is, the direction of the largest variance – or by the eigenvector to which the direction of nonlinearity is most similar (see Section 3.7.4 for details). This thesis proceeds by first showing how the standard normal distribution can be split into mixtures of two and three components, and by then extending this approach to splitting general multivariate Gaussian distributions.

3.7.3.1 Splitting into Two Components

In order to split the standard normal distribution $\mathcal{N}(x;0,1)$ into two components, $g_1(x)$ and $g_2(x)$, let us first of all use its symmetry. The symmetry tells us that if we displace one of the Gaussians by ν from the origin then the other Gaussian must be placed at $-\nu$. For the same reason, the two components must have the same mixture weight α and the same variance σ^2, which constrains the parameter optimization problem to finding the displacement as well as the variance of the two components,

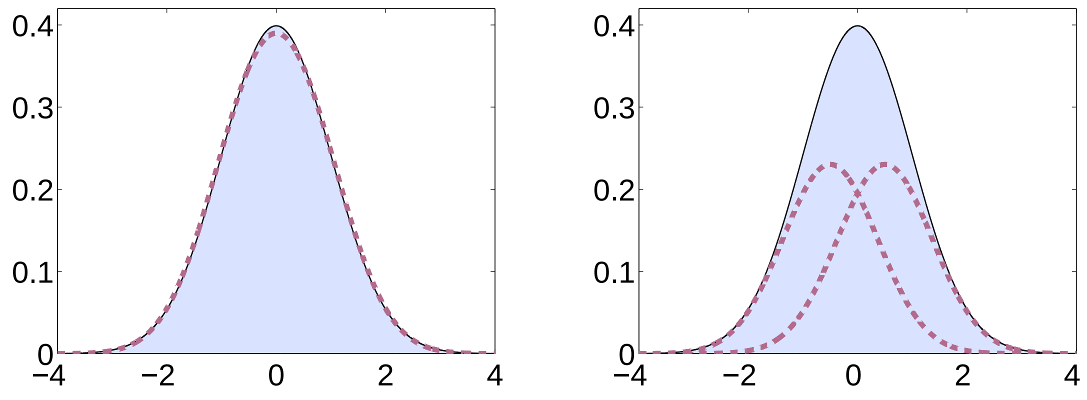
$$g_1(x) = \mathcal{N}(x;\nu,\sigma^2) \quad \text{and} \quad g_2(x) = \mathcal{N}(x;-\nu,\sigma^2). \tag{3.47}$$

From the law of total probability it is clear that the mixture weights must be one half. As a consequence, splitting the normal distributions is tantamount to replacing it by the mixture

$$m(x) = 0.5g_1(x) + 0.5g_2(x). \tag{3.48}$$

The second moment $\mathcal{E}_m\left\{x^2\right\} = \int x^2 m(x)dx$ of this mixture is obtained by first using the linearity property of integration, in order to get separate integrals over $g_1(x)$ and $g_2(x)$, and then performing a change of variables from x to $y = x - \nu$ and $y = x + \nu$, respectively. This yields: $\mathcal{E}_m\left\{x^2\right\} = \nu^2 + \sigma^2$. Now matching the second moment of the mixture to that of the normal distribution (i.e. 1) the variance can be expressed in dependence of the displacement:

$$\sigma^2 = 1 - \nu^2. \tag{3.49}$$



(a) mixture *(b) components*

Figure 3.6: *Splitting into two Gaussians with a displacement of 0.5. The picture to the left shows the original distribution (solid line, highlighted area) along with the mixture of split components (dashed line). The picture to the right shows the individual components.*

In order for this equation to be valid ν must be in the range $[-1,1]$. Further, it can be shown that the absolute (L^1) error in the fourth moment is $2\nu^4$, which is clearly minimal for the trivial solution $\nu = 0$ and which monotonically increases with $|\nu|$ until it takes its maximum, 2, at $|\nu| = 1$. This means ν should be kept as a design parameter, as there is no point in optimizing it based on the other moments. A value of 0.5 seemed to give a good trade-off between displacement and accuracy of approximation [19, 13].

3.7.3.2 Splitting into Three Components

The splitting approach from the previous section can easily be extended to the case of splitting a Gaussian into three components, $g_1(x)$, $g_2(x)$ and $g_3(x)$ [13]. Making use of the symmetry of the normal distribution, the first Gaussian is again displaced by ν, the second one by $-\nu$. The third Gaussian is centered at zero as portrayed in Figure 3.7-b. Then choosing a weight $\alpha \leq 0.5$ for each of the displaced components uniquely determines the weight of the center component, as $1-2\alpha$. Hence the mixture can be written

$$m(x) = \alpha g_1(x) + \alpha g_2(x) + (1-2\alpha) g_3(x). \tag{3.50}$$

The Gaussians in this mixture are further parameterized by the variance σ^2 of the displaced components, as well as the variance τ^2 of the center component. In the following, it will be assumed that τ is equal to σ, which greatly simplifies the optimization problem in that it does not require matching the sixth moment of the distributions. With this simplification, the mixture

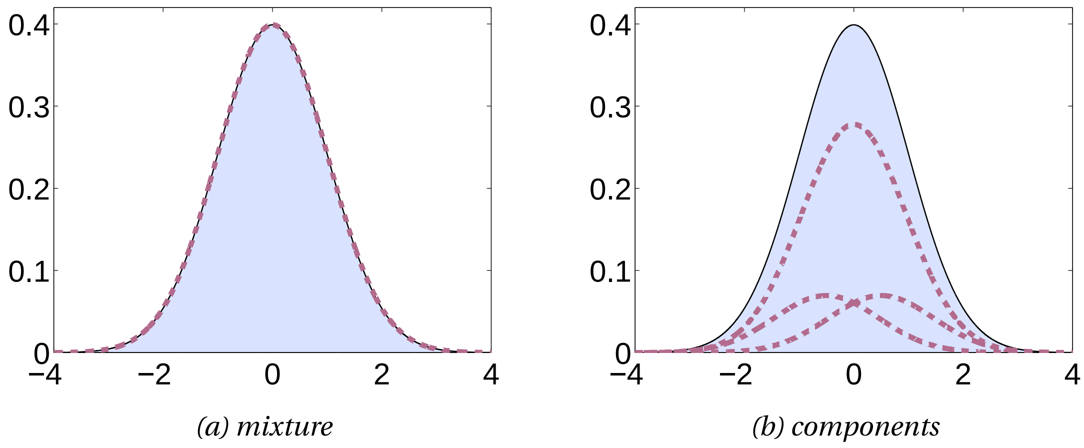


Figure 3.7: *Splitting into three Gaussians with a displacement of 0.5. The picture to the left shows the original distribution (solid line, highlighted area) along with the mixture of split components (dashed line). The picture to the right shows the individual components.*

components can be specified as follows:

$$g_1(x) = \mathcal{N}(x;\, \nu, \sigma^2), \quad g_2(x) = \mathcal{N}(x; -\nu, \sigma^2), \\ g_3(x) = \mathcal{N}(x; 0, \sigma^2). \tag{3.51}$$

Similar to the case of splitting a Gaussian into two components, α and σ^2 may be expressed in dependence of the displacement ν. For that, let us match the second and fourth moment of the mixture to those of the normal distribution:

$$\begin{aligned} \mathcal{E}_m\left\{x^2\right\} = 2\alpha\nu^2 + \sigma^2 &= 1 \\ \mathcal{E}_m\left\{x^4\right\} = 2\alpha\nu^4 + 12\alpha\sigma^2\nu^2 + 3\sigma^4 &= 3 \end{aligned}$$

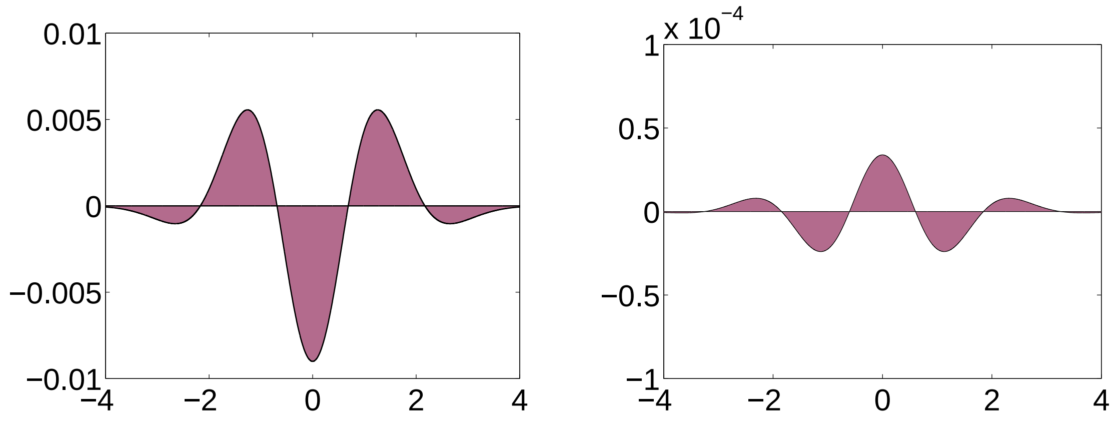
Then solving this system of equations and discarding the trivial solution $\sigma^2 = 1$ yields

$$\alpha = \frac{1}{6}, \quad \sigma^2 = 1 - \frac{1}{3}\nu^2. \tag{3.52}$$

These equations are valid for displacements ν in the range $[-\sqrt{3}, \sqrt{3}]$. As for the case of two Gaussians, the splitting introduces errors in the higher order moments – for the sixth moment, it is $\frac{2}{9}\nu^6$ [13]. Figure 3.8-(b) shows the deviation of the Gaussian mixture approximation from the original Gaussian distribution. Figure 3.8-(a) gives a comparison to the "two component" approach from the previous section.

3.7.3.3 Multivariate Splitting

This section shows how splitting a multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}; \mu; \Sigma)$ in direction of an eigenvector can be reduced to splitting a standard normal distribution. For that, let $U^T \Lambda U$



(a) 2 Gaussians *(b) 3 Gaussians*

Figure 3.8: *Difference between original and split distributions for a displacement of 0.5. Not that the scale of the image to the right is one hundredth of that of the image to the left.*

be the eigen decomposition of the covariance matrix Σ, with a diagonal matrix Λ containing the eigenvalues λ_i and a unitary matrix U containing the corresponding eigenvectors $\mathbf{u}_i$:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}, \quad U = \begin{bmatrix} \boxed{\quad \mathbf{u}_1^T \quad} \\ \vdots \\ \boxed{\quad \mathbf{u}_n^T \quad} \end{bmatrix}.$$

Using the exponentiation identity $\exp(x+y) = \exp(x)\cdot\exp(y)$ and the fact that the determinant of Σ can be factored as $\det(\Sigma) = \prod_i \lambda_i$, the probability density function can be written

$$p(\mathbf{x}) = \prod_{i=1}^{n} \underbrace{\mathcal{N}\left(\mathbf{u}_i^T\mathbf{x};\, \mathbf{u}_i^T\mu,\, \lambda_i\right)}_{\triangleq f_i(\mathbf{x})}. \tag{3.53}$$

Now let $\tilde{g}_k(x) = \mathcal{N}(x; \tilde{\mu}_k, \tilde{\sigma}_k^2)$, $k = 1, \ldots, K$, be the components resulting from a split of the standard normal distribution. Then the multivariate distribution $p(\mathbf{x})$ can be split in direction of the j-th eigenvector by simply performing the following steps:

(1) Scaling the components, $\tilde{g}_k(x)$, by $1/\sqrt{\lambda_j}$ in x-direction in order to match the variance λ_j of $p(\mathbf{x})$ in direction of the eigenvector $\mathbf{u}_j$.

(2) Projecting the rescaled components with distribution $\bar{g}_k(x) = \mathcal{N}(x; \sqrt{\lambda_j}\tilde{\mu}_k, \lambda_j\tilde{\sigma}_k^2)$ onto $\mathbf{u}_j$ and then adding the mean $\mathbf{u}_j^T\mu$, which gives:

$$\tilde{f}_{j,k}(\mathbf{x}) \triangleq \mathcal{N}\left(\mathbf{u}_j^T\mathbf{x};\, \mathbf{u}_j^T\mu + \sqrt{\lambda_j}\tilde{\mu}_k,\, \lambda_j\tilde{\sigma}_k^2\right).$$

Splitting into Two Gaussians		
$\omega_1 = \frac{1}{2}$	$\mu_1 = \mu + \nu\sqrt{\lambda}\mathbf{u}$	$\Sigma_1 = \Sigma - \nu^2\lambda\mathbf{u}\mathbf{u}^T$
$\omega_2 = \frac{1}{2}$	$\mu_2 = \mu - \nu\sqrt{\lambda}\mathbf{u}$	$\Sigma_2 = \Sigma - \nu^2\lambda\mathbf{u}\mathbf{u}^T$

Splitting into Three Gaussians		
$\omega_1 = \frac{1}{6}$	$\mu_1 = \mu + \nu\sqrt{\lambda}\mathbf{u}$	$\Sigma_1 = \Sigma - \frac{1}{3}\nu^2\lambda\mathbf{u}\mathbf{u}^T$
$\omega_2 = \frac{1}{6}$	$\mu_2 = \mu - \nu\sqrt{\lambda}\mathbf{u}$	$\Sigma_2 = \Sigma - \frac{1}{3}\nu^2\lambda\mathbf{u}\mathbf{u}^T$
$\omega_3 = \frac{4}{6}$	$\mu_3 = \mu$	$\Sigma_3 = \Sigma - \frac{1}{3}\nu^2\lambda\mathbf{u}\mathbf{u}^T$

Table 3.1: *Mixture parameters for splitting $\mathcal{N}(\mu, \Sigma)$ into two and three Gaussians with displacement ν in the direction of eigenvector $\mathbf{u}$ with corresponding eigenvalue λ.*

(3) Replacing $f_j(\mathbf{x})$ in (3.53) by $\tilde{f}_{j,k}(\mathbf{x})$ for $k = 1, \ldots, K$ in order to obtain the split components of the multivariate distribution:

$$g_k(\mathbf{x}) = \left(\prod_{\substack{i=1 \\ i\neq j}}^{n} f_i(\mathbf{x})\right)\tilde{f}_{j,k}(\mathbf{x}), \tag{3.54}$$

From this, the mean μ_k of the k-th component g_k can be recovered as

$$\mu_k = \sum_{\substack{i=1 \\ i\neq j}}^{n} \mathbf{u}_i\left(\mathbf{u}_i^T\mu\right) + \mathbf{u}_j\left(\mathbf{u}_j^T\mu + \sqrt{\lambda_j}\tilde{\mu}_k\right) = \mu + \sqrt{\lambda_j}\tilde{\mu}_k\mathbf{u}_j. \tag{3.55}$$

The corresponding covariance matrix Σ_k can be expressed by means of its eigenvectors and eigenvalues:

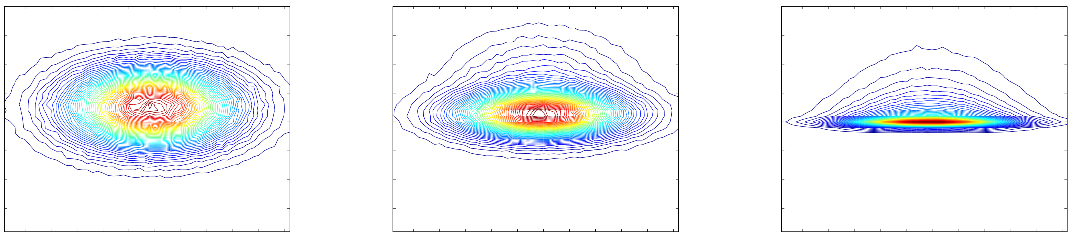
$$\Sigma_k = \sum_{\substack{i=1 \\ i\neq j}}^{n} \lambda_i\mathbf{u}_i\mathbf{u}_i^T + \tilde{\sigma}_k^2\lambda_j\mathbf{u}_j\mathbf{u}_j^T = \underbrace{\sum_{i=1}^{n} \lambda_i\mathbf{u}_i\mathbf{u}_i^T}_{=\Sigma} - \left(1-\tilde{\sigma}_k^2\right)\lambda_j\mathbf{u}_j\mathbf{u}_j^T, \tag{3.56}$$

which is a simple rank-1 downdate[5]. Table 3.1 concludes this section by explicitly giving the mixture weights ω_k, means μ_k and covariance matrices Σ_k for splitting $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ into two and three Gaussian components.

3.7.4 Splitting in Direction of the Nonlinearity

The previous section considered splitting Gaussian distributions in the direction of an eigenvalue. Using the eigenvector corresponding to the largest eigenvalue of the covariance matrix obviously gives the greatest reduction in variance [63, 19, 12]. As motivated in [13], however,

[5] This means the Cholesky factor of Σ_k can efficiently be obtained through a Cholesky downdate if the Cholesky factor of Σ is available, as it is the case for the square root implementation of the unscented transform [53].



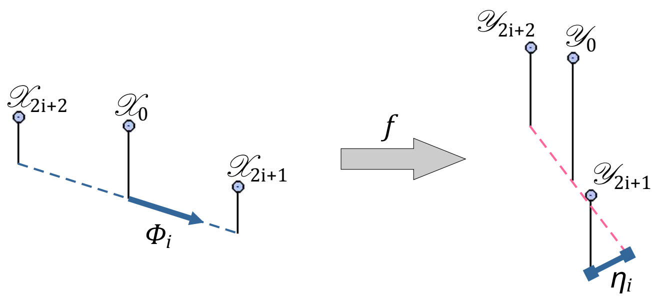
(a) ALoDT-4[v] *(b) ALoDT-4[n]* *(c) true density*

Figure 3.9: *Contour plots of the transformed distributions obtained with the ALoDT using 4 Gaussians, for splitting in direction of the largest variance [v] and splitting in direction of the nonlinearity [n].*

this approach is suboptimal if we regard the fact that the reason for splitting is actually the nonlinearity to which the distribution is subject during transformation. To illustrate this issue, consider the following example of a transformation $f(X)$ of a Gaussian random variable X with distribution $p_X(x) = \mathcal{N}(x;\mu_X,\Sigma_X)$:

$$\mu_X = \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \quad \Sigma_X = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}, \quad f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2^2 \end{bmatrix}.$$

For this example, the adaptive level of detail transform from Section 3.7.2 cannot improve over the unscented transform – at least not in the first couple of iterations. That is because the distribution of X is split in x_1-direction where the variance is largest but where f is linear. In this case, splitting increases the approximation error. But, it does not provide a better approximation of the transformation. This becomes clear by looking at Figure 3.9, which shows the transformed distributions for the case of splitting into the direction of the largest variance and the direction of nonlinearity, respectively. For the largest variance, the distribution is almost identical to the one obtained with a single unscented transform. For the direction of nonlinearity, it is obviously closer to the true density from Figure 3.9-(c). Hence, in the framework of this thesis [13] it has been proposed to split the distribution in direction of the nonlinearity. Bearing in mind that in the unscented transform the i-th triple $\{\mathcal{X}_{2i+1}, \mathcal{X}_0, \mathcal{X}_{2i+2}\}$ of points forms a set of equidistant points on a line, the nonlinearity in direction of this triple can obviously be approximated by the degree of nonlinearity, η_i, associated with these points.



This means, if we determine the direction of the i-th triple as

$$\phi_i = \frac{\mathscr{X}_{2i+1} - \mathscr{X}_0}{\|\mathscr{X}_{2i+1} - \mathscr{X}_0\|}, \tag{3.57}$$

the *direction of nonlinearity* can be defined as the eigenvector ψ corresponding to the largest eigenvalue of the covariance matrix Ψ that is described by the vectors ϕ_i, weighted with the η_i [13]:

$$\Psi = \sum_{i=0}^{n-1} \eta_i \phi_i \phi_i^T. \tag{3.58}$$

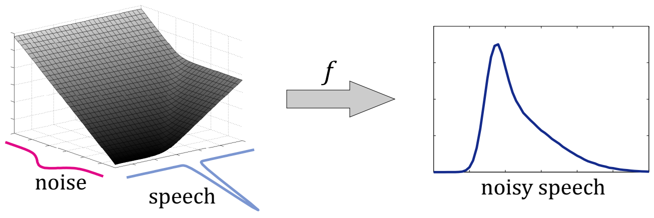
The resulting eigenvector obviously indicates the direction in which the nonlinearity is strongest. As a computationally less demanding alternative, the direction of nonlinearity may be approximated as the average over the ϕ_i, weighted with the corresponding η_i [13]:

$$\psi' = \frac{\sum_{i=0}^{n-1} \eta_i \phi_i}{\left\| \sum_{i=0}^{n-1} \eta_i \phi_i \right\|}. \tag{3.59}$$

Now, ψ or ψ' may be used for splitting the Gaussian to be transformed in direction of the nonlinearity. When implementing this, it should be noted that splitting a Gaussian distribution in an arbitrary direction ψ turns out to be difficult, unless ψ coincides with one of the principal axes of the covariance matrix. Hence, it is here split in the direction of that eigenvector $\mathbf{u}_i$ which ψ is most similar to. That is the one for which $\mathbf{u}_i^T \psi$ is maximal: $i = \mathrm{argmax}_j \left(\mathbf{u}_j^T \psi\right)$ [13].

3.8 Performance Evaluation of Nonlinear Transforms

This section compares the performance of the proposed transformation methods – namely, the adaptive level of detail transform [12, 13] and the extensive unscented transform [18] – to that of state of the art methods such as the unscented transform. For this comparison, we chose the nonlinear transformation problem which arises if clean speech features are contaminated by additive noise [45, 46, 50].



This transformation requires approximating the distribution $p_Y(y)$ of noisy speech Y, given the distribution $\mathscr{N}(x;\mu_X,\Sigma_X)$ of clean speech X, the distribution $\mathscr{N}(n;\mu_N,\Sigma_N)$ of noise N as well as the following interaction function:

$$y = \underbrace{\log(\exp(x) + \exp(n))}_{=f(x,n)}, \tag{3.60}$$

which specifies the relationship between these variables in the speech feature (logarithmic Mel spectra) domain. Motivated by the fact that the frequency bands can be treated independently if the Gaussians have[6] diagonal covariance matrices, the transformation $Y = f(X, N)$ is here simulated for one dimension only.

3.8.1 Experimental Setup

For the simulation, it was assumed that clean speech has a Gaussian distribution with a mean of 8.9 and a standard deviation of 0.6. The noise was assumed to have a Gaussian distribution with mean 6.3 and standard deviation 3.0. In order to have a reference for the transformed distribution, we generated 10 million samples from the speech and noise distributions and then transformed these samples according to (3.60). The resulting empirical distribution was used to "train" a mixture distribution of 20 Gaussian components, whose weights, means and variances were found by performing 50 iterations of the EM algorithm [65]. The means for the first iteration were initialized with the k-Means algorithm [65]. In order to calculate the Kullback-Leibler divergence (KLD) between this reference distribution $p_Y^{(ref)}$ and a given approximation $p_Y^{(app)}$, we used the following Monte Carlo approximation:

$$D_{KL}\left(p_Y^{(ref)} \middle\| p_Y^{(app)}\right) \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{p_X^{(ref)}\left(y^{(i)}\right)}{p_Y^{(app)}\left(y^{(i)}\right)}\right)$$

with $N = 10$ million samples $y^{(i)}$ drawn from $p_Y^{(ref)}$. Note that this approximation is necessary as the KLD between Gaussian mixtures cannot be calculated in an analytic fashion [66].

3.8.2 Results

Figure 3.10 shows the Kullback-Leibler divergence (KLD) between the reference and approximations obtained with the adaptive level of detail transform. The plot to the left compares different splitting criteria – namely, component weight (weight), degree of nonlinearity (dnl), the splitting priority from Section 3.7.2.1 (spp) and the normalized splitting priority[7] (spp(n)). The plot to the right compares the augmented (AUT) and extensive unscented transforms (XUT). For the considered transformation problem, the latter is an XUT with $n = m = 1$ and $\kappa = 2$. Hence, the plot actually shows a comparison to Gauss-Hermite quadrature. The exact numbers are given in Tables 3.2 and 3.3.

[6] This is commonly assumed in the area of automatic speech recognition.

[7] That is the splitting priority using the normalized measure for the degree of nonlinearity.

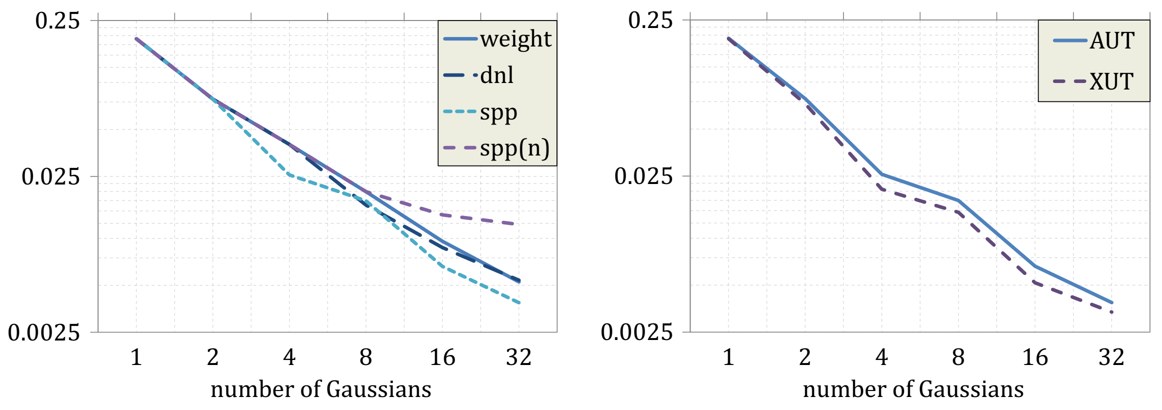


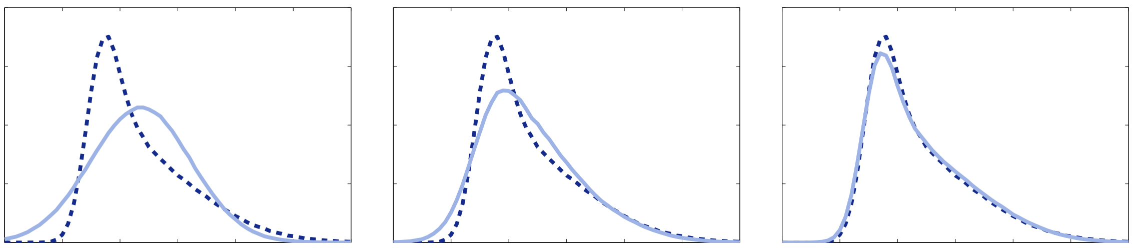
Figure 3.10: *Kullback-Leibler divergence between the reference distribution and Gaussian mixture approximations obtained with the adaptive level of detail transform. The plot to the left shows results for different splitting criteria. The plot to the right gives a comparison between the augmented (AUT) and extensive unscented transform (XUT) under use of the splitting priority.*

In general, an increase in the number of Gaussians led to a decrease of the approximation error[8]. Among the splitting criteria, the splitting priority gave the best results. The normalized splitting priority performed the worst. Regarding the comparison between augmented and extensive unscented transform, the XUT (Gauss Hermite quadrature) consistently outperformed the AUT. Note that Tables 3.2 and 3.3 also give a comparison to the standard UT, as the ALoDT with one Gaussian (ALoDT-1) is identical to a single unscented transform. So, the ALoDT with 16 Gaussians reduced the Kullback-Leibler divergence of the UT by a factor of 28. With 128 Gaussians it was a factor of 75.

Table 3.5 shows results for the Monte Carlo (MC) approach, which was used in Section 3.8.1 in order to obtain the reference for the true transformed distribution. This approach is similar in principle to the one proposed in [54]. For the results given here, we used a lower number of samples (between 100 and 1 million instead of 10 million) and performed only 10 and 20 iterations, respectively, of clustering and EM training. As can be seen, the approximation error decreased with the number of samples. After 20 iterations with 100,000 samples, the approximation error was 0.00127, which compares to a value of 0.00238 for the ALoDT-128.

The computation times in Tables 3.4 and 3.6 reveal the major advantage of the ALoDT over the Monte Carlo approach: computational efficiency. For a reasonable accuracy of approximation – i.e. a KLD of 0.00518 with 16 Gaussians — the ALoDT took 0.8329 milliseconds to compute. The Monte Carlo approach necessitated 10 iterations of EM training with 10,000 samples in order to get a comparable number. Its computational expense was 1.1 seconds, which is about 1300 times that of the ALoDT. Even with 128 Gaussians, the ALoDT took only 4 to 8 milliseconds to compute. This compares to 10 milliseconds for the Monte Carlo approach with 100 samples, for which the approximation error is 200 times as high.

[8] at least for a displacement of $\nu = 0.5$ during splitting (see Section 3.7.3)



(a) ALoDT-1 *(b) ALoDT-4* *(c) ALoDT-16*

Figure 3.11: *Transformed Distribution. The dashed curve shows the true distribution. The solid curves show approximations obtained with the adaptive level of detail transform (ALoDT) with 1, 4 and 16 Gaussians, using the splitting priority (spp) as a splitting criterion.*

splitting	number of Gaussians							
criterion	1	2	4	8	16	32	64	128
weight	0.19079	0.07825	0.04009	0.01997	0.00958	0.00526	0.00344	0.00273
dnl	0.19079	0.07825	0.02556	0.01637	0.00874	0.00539	0.00323	0.00290
spp	0.19079	0.07825	0.02556	0.01744	0.00661	0.00387	0.00310	0.00253
spp(n)	0.19079	0.07825	0.04009	0.01997	0.01416	0.01228	0.00811	0.00606

Table 3.2: *Approximation error (in KLD) for the adaptive level of detail transform. The rows show results for different splitting criteria: component weight (weight), degree of nonlinearity (dnl), splitting priority (spp), normalized splitting priority (spp(n)).*

splitting	number of Gaussians							
criterion	1	2	4	8	16	32	64	128
AUT	0.19079	0.07825	0.02556	0.01744	0.00661	0.00387	0.00310	0.00253
XUT	0.18677	0.07265	0.02059	0.01464	0.00518	0.00336	0.00287	0.00238

Table 3.3: *KLD for two versions of the ALoDT. One is using the augmented unscented transform (AUT) for transforming the individual Gaussian components as well as for calculating their degrees of nonlinearity. The other is using the extensive unscented transform (XUT).*

splitting	number of Gaussians							
criterion	1	2	4	8	16	32	64	128
AUT	0.0148	0.0368	0.0792	0.1691	0.3590	0.7590	1.6635	4.0592
XUT	0.0297	0.0830	0.1893	0.3989	0.8329	1.7346	3.6538	7.9937

Table 3.4: *Computation times in milliseconds for the ALoDT using the augmented (AUT) and extended unscented transform (AUT). The increase is approximately linear with respect to the number of Gaussians.*

#(iterations EM)	number of samples				
	100	1,000	10,000	100,000	1,000,000
10	0.49325	0.02692	0.00499	0.00342	0.00318
20	0.53045	0.03207	0.00346	0.00127	0.00157

Table 3.5: *Approximation error (in KLD) for the Monte Carlo transformation approach with EM-based Gaussian mixture recovery. This is essentially the approach taken in [54].*

#(iterations EM)	number of samples				
	100	1,000	10,000	100,000	1,000,000
10	0.01	0.11	1.10	11.19	110.35
20	0.02	0.24	2.26	22.58	233.70

Table 3.6: *Computation times in seconds for the Monte Carlo transformation approach with EM-based Gaussian mixture recovery. When comparing these results to Table 3.4, note that the units here are seconds whereas those used in Table 3.4 are milliseconds.*

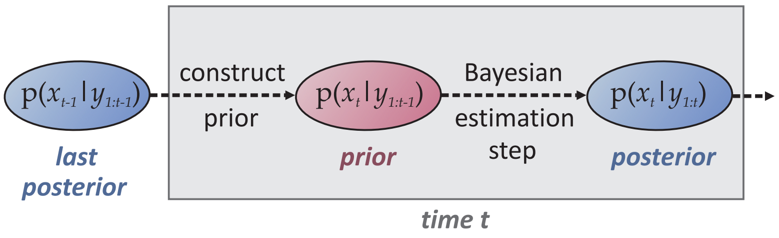
3.9 Contributions of this Chapter

The following list again gives an overview of the individual contributions of this thesis to Bayesian state estimation:

1. A unified view of Bayesian state estimation approaches based on a simple but general transformation principle (Section 3.1).
2. The extensive unscented transform – a modification of the unscented transform which preserves the statistical independence of the variables (Section 3.6) [18].
3. The degree of nonlinearity – a measure for estimating the linearization error of the unscented transform (Sections 3.5.1 and 3.5.3) [19, 12].
4. The adaptive level of detail transform with its refinements (Section 3.7) [19, 12, 13].
5. Splitting Gaussians in direction of the nonlinearity (Section 3.7.4) [13].

4 Sequential Bayesian Estimation

The previous chapter dealt with estimating the state of a physical system based on a corresponding observation. That was done in a Bayesian fashion, by (a) considering a prior distribution for the state, (b) using this prior to construct the joint distribution of state and observation, and (c) calculating the posterior by conditioning on the realized (measured) observation. This chapter extends that approach to estimating a time-varying system state x_t, $t \in \mathbb{N}$, based on a sequence $y_{1:t} = \{y_1, \dots, y_t\}$ of corresponding observations. The main idea is to (1) use the last last posterior $p_{X_{t-1}|y_{1:t-1}}$ for constructing a prior distribution $p_{X_t|y_{1:t-1}}$ at time t and to then (2) perform a Bayesian estimation step in order to obtain the new posterior $p_{X_t|y_{1:t}}$.



While the Bayesian estimation step can be performed with any of the methods from Chapter 3, the construction of the prior distribution necessitates further explanation. In order to get a start on this, let us first of all assume that the state has not changed from time $t-1$ to time t. Then the last posterior (i.e. the distribution of the "state estimate" under the measurements observed so far) could be used as a prior for the state at time t: $p_{X_t|y_{1:t-1}}(x_t) = p_{X_{t-1}|y_{1:t-1}}(x_{t-1})$. This is the simplest possible case. A more accurate prior is obtained by predicting $p_{X_t|y_{1:t-1}}$ from $p_{X_{t-1}|y_{1:t-1}}$ based on a process model that describes the evolution in time of the physical system.

4.1 Example of a Process Model

To exemplify the use of a process model, consider the case of a football being kicked as portrayed in Figure 4.1-(a). Let the system state $x_t = [p_{t,1}\ \ p_{t,2}\ \ v_{t,1}\ \ v_{t,2}]^T$ describe the ball's current position $p_t = [p_{t,1}\ \ p_{t,2}]^T$ as well as its velocity $v_t = [v_{t,1}\ \ v_{2,t}]^T$.

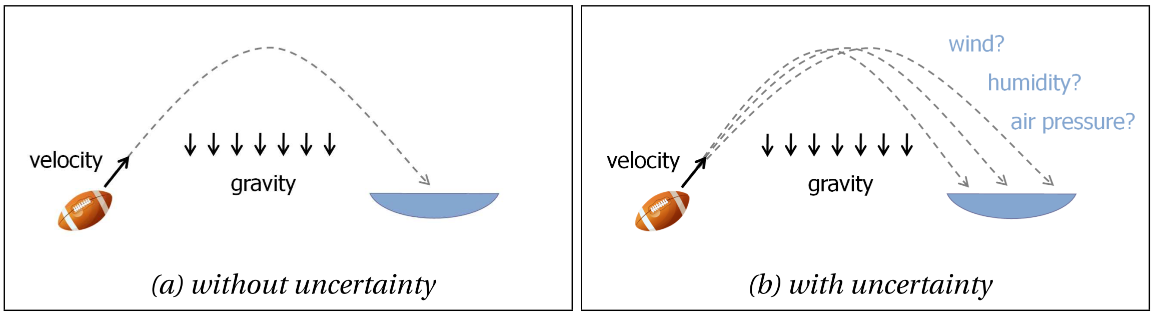


Figure 4.1: *A simple process model. Flightpath of a football with and without uncertainty.*

Let us further denote the difference in time between the t-th and $(t+1)$-st measurements by Δ_t. Then each increment of t decreases the velocity $v_{t,2}$ in direction of the height by the acceleration g of gravity times Δ_t. Similarly, each increment of t changes the position of the ball by Δ_t times the current velocity. Hence, we have:

$$x_t = \begin{bmatrix} p_{t,1} \\ p_{t,2} \\ v_{t,1} \\ v_{t,2} \end{bmatrix} \longmapsto x_{t+1} = \begin{bmatrix} p_{t,1} + \Delta_t v_{t,1} \\ p_{t,2} + \Delta_t v_{t,2} \\ v_{t,1} \\ v_{t,2} - \Delta_t g \end{bmatrix}, \tag{4.1}$$

This model gives a deterministic description of the flightpath. The variables that were not taken into consideration – such as the speed of wind, variations in air pressure and gravity, etc. – however introduce uncertainties, i.e. "random effects" that would more accurately be described probabilistically. Hence, (4.1) may be extended by an additive Gaussian noise[1] term $w_t = \left[w_{t,1}, \dots, w_{t,4}\right]^T \sim \mathcal{N}(\mu_{W_t}, \Sigma_{W_t})$, which describes these uncertainties:

$$x_t = \begin{bmatrix} p_{t,1} \\ p_{t,2} \\ v_{t,1} \\ v_{t,2} \end{bmatrix} \longmapsto x_{t+1} = \begin{bmatrix} p_{t,1} + \Delta_t v_{t,1} \\ p_{t,2} + \Delta_t v_{t,2} \\ v_{t,1} \\ v_{t,2} - \Delta_t g \end{bmatrix} + \begin{bmatrix} w_{t,1} \\ w_{t,2} \\ w_{t,3} \\ w_{t,4} \end{bmatrix}. \tag{4.2}$$

This idea of "explicitly accounting for uncertainties in a physical system" is now generalized to dynamic state space models.

[1] Note that Gaussian noise is a good choice as uncertainties in macroscopic systems tend to consists of many different influences that add up (see the *central limit theorem* [67]).

4.2 Dynamic State Space Models

A *dynamic state space model* (DSSM) describes states and observations as a *doubly stochastic system* $(X_t, Y_t)_{t\in\mathbb{N}}$, i.e. a system which consists of two processes:

1. a hidden state process $(X_t)_{t\in\mathbb{N}}$ that describes the evolution of the systems state.
2. a related measurement process $(Y_t)_{t\in\mathbb{N}}$ that describes the corresponding observations.

For sequential Bayesian estimation, the state process X_t is considered to be a Markov process. The observations Y_t are considered to follow the output independence assumption (Section 2.1.5). Hence, (X_t, Y_t) describes a hidden Markov model, which is fully specified by giving a *process model* along with a corresponding *measurement model*.

Process Model

The process model consists of a prior distribution $p_{W_t}(w_t)$, which describes random influences in the physical system, as well as a process equation which describes the transition of the state from time $(t-1)$ to time t:

$$x_t = f(x_{t-1}, w_t). \tag{4.3}$$

Measurement Model

The measurement model consists of a prior distribution $p_{V_t}(v_t)$, which describes random influences during measurement, as well as a measurement equation which describes the relationship between states and observations:

$$y_t = h(x_t, v_t). \tag{4.4}$$

In the above context, the term w_t is typically referred to as "*process noise*". The term v_t is referred to as "*measurement noise*". In particular, the w_t and v_t are statistically independent at all times (due to the Markov assumption as well as the output independence assumption). After the specification of a DSSM, sequential Bayesian estimation can be performed by (1) using the process equation in order to construct a prior $p_{X_t|y_{1:t-1}}$ at time t, and then (2) performing a Bayesian state estimation step which uses the measurement equation as an an observation model (see Section 3.1.1). This is explained in more detail in the following.

4.3 The General Principle Behind Sequential Bayesian Estimation

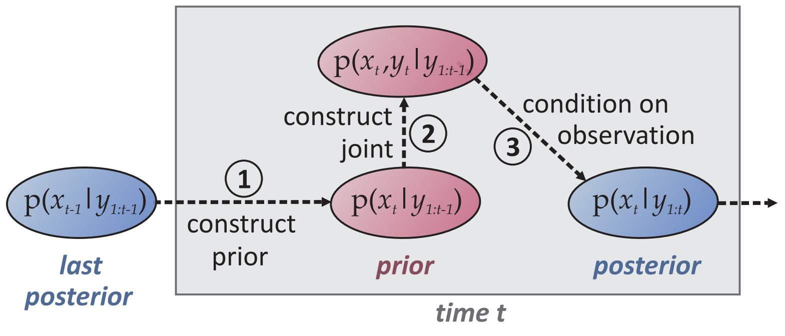
Given a particular DSSM – specified by a process model along with an associated measurement model – the prior distribution $p_{X_t|y_{1:t-1}}$ can be constructed from the last posterior $p_{X_{t-1}|y_{1:t-1}}$. That is achieved by transforming the joint random variable of $X_{t-1}|y_{1:t-1}$ and W_t according to the process equation (4.3):

$$X_t|y_{1:t-1} = f\left(X_{t-1}|y_{1:t-1}, W_t\right). \tag{4.5}$$

This prediction step is followed by a Bayesian state estimation step in which the joint predictive distribution $p_{X_t,Y_t|y_{1:t-1}}$ of state $X_t|y_{1:t-1}$ and observation Y_t is constructed by transforming $X_t|y_{1:t-1}$ and V_t according to the augmented measurement equation:

$$X_t, Y_t|y_{1:t-1} = \tilde{h}\left(X_t|y_{1:t-1}, V_t\right) \quad \text{with} \quad \tilde{h}(x_t, v_t) \triangleq \begin{bmatrix} x_t \\ h(x_t, v_t) \end{bmatrix}. \tag{4.6}$$

The new posterior $p_{X_t|y_{1:t}}$, which is also referred to as *filtering density*, is subsequently obtained by conditioning $p_{X_t,Y_t|y_{1:t-1}}$ on the realized observation $Y_t = y_t$, in analogy to Section 3.1. After the Bayesian estimation step, a particular estimate $\hat{x}_t|y_{1:t}$ of the state can be calculated by applying a criterion of optimality, such as the minimum mean squared error criterion for which $\hat{x}_t|y_{1:t} = \mathcal{E}_{p_{X_t|y_{1:t}}(x_t)}\{x_t\}$. This is the principle behind sequential Bayesian estimation, which is again summarized in the following figure and formalized in Algorithm 4.1.



A General Algorithm for Sequential Bayesian Estimation

1. Predict the distribution $p_{X_t|y_{1:t-1}}(x_t)$ of the next state from $p_{X_{t-1}|y_{1:t-1}}(x_{t-1})$, based on the process model.

2. Construct the joint predictive distribution $p_{X_t,Y_t|y_{1:t-1}}(x_t, y_t)$ of state and observation according to the measurement model.

3. Condition the joint distribution of X_t and Y_t on the realized observation $Y_t = y_t$ in order to update the filtering density to time t: $p_{X_t|y_{1:t}}(x_t)$.

4. Calculate the current state estimate $\hat{x}_t|y_{1:t}$. Then increase t, wait for the arrival of a new measurement y_t and subsequently go back to step 1.

Algorithm 4.1: Sequential Bayesian Estimation (SBE)

The following again explains this algorithm in more detail: The first step creates a prior by predicting the distribution of the state from the last posterior. This generally "widens" the distribution due to an increase in uncertainty. The second step constructs the joint distribution of state and observation. This (1) predicts the likelihood for receiving particular observations and (2) captures the statistical relationship that states and observations have according to the measurement model. The third step uses this relationship by conditioning the joint distribution on the realized observation. This can be interpreted as constraining (or "correcting") the predicted distribution of the state based on the received observation.

The remaining part of the chapter considers several implementations of this general procedure. This includes most of the common tracking algorithms, such as the Kalman filter (Section 4.4), the unscented Kalman filter (Section 4.5), Gaussian mixture filters (Section 4.6) and particle filters (Section 4.7). Section 4.6.3 introduces a novel Gaussian mixture filter, which adapts the number of Gaussians based on the degree of nonlinearity. Section 4.8 closes the chapter with a numerical comparison of the described tracking algorithms.

4.4 The Kalman Filter

The first and probably most well known implementation of sequential Bayesian estimation was developed by Rudolph E. Kalman [40]. His implementation, which is nowadays known as the Kalman filter, is based on the assumption that the filtering density is Gaussian at all times. This requires firstly, that the functions f and h are linear, and secondly, that the process and measurement noise distributions are Gaussian:

$$\begin{aligned} p_{W_t}(w_t) &= \mathcal{N}(w_t; \mu_{W_t}, \Sigma_{W_t W_t}), \quad &(4.7)\\ p_{V_t}(v_t) &= \mathcal{N}(v_t; \mu_{V_t}, \Sigma_{V_t V_t}), \quad &(4.8)\end{aligned}$$

Here, μ_{W_t} and $\Sigma_{W_t W_t}$ denote the mean and covariance matrix of W_t; μ_{V_t} and $\Sigma_{V_t V_t}$ denote the mean and covariance matrix of V_t, respectively. Now making use of the first requirement – i.e. the linearity of f and g – the process and measurement equations can be written

$$f(x_{t-1}, w_t) = F x_{t-1} + w_t \quad \text{and} \quad h(x_t, v_t) = H x_t + v_t$$

where F and H are matrix representations of f and h. This in combination with the second requirement – i.e. Gaussianity – implies that all the required transformations of random variables can be performed with the Kalman-type linear transform (see Section 3.3), as described in more detail in the following.

4.4.1 Prediction (Step 1)

Following the developments in Section 4.3, sequential Bayesian estimation starts with predicting the distribution $p_{X_t|y_{1:t-1}}(x_t)$ of the next state. This is in general achieved by transforming the random variables $X_{t-1}|y_{1:t-1}$ and W_t according to (4.5): $X_t|y_{1:t-1} = f\left(X_{t-1}|y_{1:t-1}, W_t\right)$. In the particular case of the Kalman filter, both $X_{t-1}|y_{1:t-1}$ and W_t are Gaussian Random variables. Hence, the Kalman-type linear transform can be used in order to obtain the joint Gaussian distribution p_{X_{t-1},X_t} of last and current state under the observation history $y_{1:t-1}$:

$$p_{X_{t-1},X_t|y_{1:t-1}}(x_{t-1},x_t) = \mathcal{N}\left(\begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix}; \begin{bmatrix} \mu_{X_{t-1}} \\ \mu^-_{X_t} \end{bmatrix}, \begin{bmatrix} \Sigma_{X_{t-1}X_{t-1}} & \Sigma^-_{X_{t-1}X_t} \\ \Sigma^-_{X_tX_{t-1}} & \Sigma^-_{X_tX_t} \end{bmatrix}\right). \tag{4.9}$$

This just requires us to apply Section 3.3 with $X = X_{t-1}|y_{1:t-1}$, $U = W_t$, $Y = X_t|y_{1:t}$, $g = f$ and $B = F$. The $\mu_{X_{t-1}}$ and $\Sigma_{X_{t-1}X_{t-1}}$ in (4.9) denote the mean and covariance matrix of the last posterior, $p_{X_{t-1}|y_{1:t-1}}$. The other parameters ($\mu^-_{X_t}$, $\Sigma^-_{X_tX_t}$ and $\Sigma^-_{X_tX_{t-1}}$) are calculated according to (3.19), under the assumption that the X_{t-1} and W_t are uncorrelated:

$$\mu^-_{X_t} = F\mu_{X_{t-1}} + \mu_{W_t}, \quad \Sigma^-_{X_tX_{t-1}} = F\Sigma_{X_{t-1}X_{t-1}}, \quad \Sigma^-_{X_tX_t} = F\Sigma_{X_{t-1}X_{t-1}}F^T + \Sigma_{W_tW_t}. \tag{4.10}$$

Now given the joint distribution from (4.9), the predicted distribution $p_{X_t|y_{1:t-1}}$ of the state can be obtained by marginalizing over x_{t-1}. This yields:

$$p_{X_t|y_{1:t-1}}(x_t) = \mathcal{N}(x_t; \mu^-_{X_t}, \Sigma^-_{X_tX_t}) \tag{4.11}$$

where $\mu^-_{X_t}$ and $\Sigma^-_{X_tX_t}$ are calculated according to (4.10) and where the superscript "-" indicates that the parameters concern a prior distribution.

4.4.2 Constructing the Joint (Step 2)

After prediction according to (4.11), we need to construct the joint Gaussian distribution of state and observation $p_{X_t,Y_t|y_{1:t-1}}$ by transforming $X_t|y_{1:t-1}$ and V_t according to the augmented measurement equation (4.6): $X_t, Y_t|y_{1:t-1} = \tilde{h}\left(X_t|y_{1:t-1}, V_t\right)$. In case of the Kalman filter, both $X_t|y_{1:t-1}$ and V_t are Gaussian random variables. Hence, the transformation can again be performed with the Kalman-type linear transform – just this time with $X = X_t|y_{1:t-1}$, $U = V_t$, $Y = Y_t$, $g = h$ and $B = H$. This gives:

$$p_{X_t,Y_t|y_{1:t-1}}(x_t, y_t) = \mathcal{N}\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}; \begin{bmatrix} \mu^-_{X_t} \\ \mu_{Y_t} \end{bmatrix}, \begin{bmatrix} \Sigma^-_{X_tX_t} & \Sigma_{X_tY_t} \\ \Sigma_{Y_tX_t} & \Sigma_{Y_tY_t} \end{bmatrix}\right) \tag{4.12}$$

where $\mu^-_{X_t}$ and $\Sigma^-_{X_tX_t}$ are as in (4.11). Assuming uncorrelated process and measurement noise, i.e. $\Sigma_{X_tV_t} = 0$ (as required in Section 4.2 by asking for w_t and v_t to be statistically independent), the other distribution parameters of (4.12) can be calculated according to:

$$\mu_{Y_t} = H\mu^-_{X_t} + \mu_{V_t}, \quad \Sigma_{X_tY_t} = \Sigma^-_{X_tX_t}H^T, \quad \Sigma_{Y_tY_t} = H\Sigma^-_{X_tX_t}H^T + \Sigma_{V_tV_t}. \tag{4.13}$$

The term $\Sigma_{Y_tX_t}$ does not need to be calculated as it can be obtained by transposition:

$$\Sigma_{Y_tX_t} = \left(\Sigma_{X_tY_t}\right)^T.$$

4.4.3 Conditioning on the Observation (Step 3)

The posterior distribution $p_{X_t|y_{1:t}}$ at time t is finally obtained by conditioning the joint predictive distribution $p_{X_t,Y_t|y_{1:t-1}}$ of state and observation on the realized (i.e. measured) observation $Y_t = y_t$. In case of the Kalman filter, this step consists in calculating the parameters of a conditional Gaussian distribution from a joint Gaussian distribution, as described in Section 3.2.1. This gives

$$p_{X_t|y_{1:t}}(x_t) = \mathcal{N}(x_t; \mu^+_{X_t}, \Sigma^+_{X_tX_t}) \tag{4.14}$$

where the conditional mean $\mu^+_{X_t}$ and covariance $\Sigma^+_{X_tX_t}$ are calculated according to

$$\begin{aligned} \mu^+_{X_t} &= \mu^-_{X_t} + \Sigma_{X_tY_t}\Sigma^{-1}_{Y_tY_t}(y_t - \mu_{Y_t}), \\ \Sigma^+_{X_tX_t} &= \Sigma^-_{X_tX_t} - \Sigma_{X_tY_t}\Sigma^{-1}_{Y_tY_t}\Sigma_{Y_tX_t}. \end{aligned} \tag{4.15}$$

The terms $\mu^-_{X_t}$, $\Sigma^-_{X_tX_t}$, $\Sigma_{X_tY_t}$, $\Sigma_{Y_tY_t}$ and μ_{Y_t} are obtained from (4.10) and (4.13); and superscript "+" indicates that the parameters concern a posterior distribution (i.e. a distribution that is conditioned on $y_{1:t}$). Regarding the above, it is interesting to note that the equations in (4.15) are generally called the Kalman filter update equations. But they are typically expressed by means of the Kalman gain $K_t \triangleq \Sigma_{X_tY_t}\Sigma^{-1}_{Y_tY_t}$ [42, 68, 43, 65, 44], which gives:

$$\begin{aligned} \mu^+_{X_t} &= \mu^-_{X_t} + K_t(y_t - \mu_{Y_t}), \\ \Sigma^+_{X_tX_t} &= \Sigma^-_{X_tX_t} - K_t\Sigma_{Y_tY_t}K_t^T. \end{aligned} \tag{4.16}$$

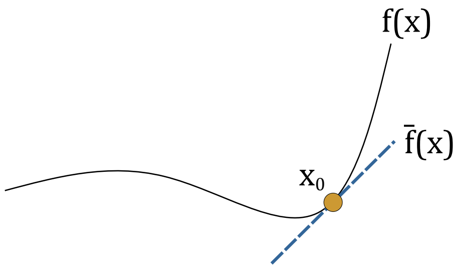
After this conditioning step, the minimum mean squared error estimate of the system state can be obtained as the mean of the posterior distribution. In case of the Kalman filter, this is just: $\hat{x}_t|y_{1:t} = \mu^+_{X_t}$.

4.4.4 Kalman's Contribution and the Bayesian Interpretation

The origin of the Kalman filter update equations from (4.16) is without doubt Rudolph E. Kalman seminal work on linear filtering and prediction problems [40]. But, as Kalman himself acknowledges [40], the statistical literature of that time [69] was well aware of the fact that the Wiener Problem (MMSE estimation) could be approached from the point of view of conditional distributions and expectations. This included knowledge of the advantages that the use of Gaussian processes brings to least square error estimation [69]. In this light, Kalman's main contribution can be regarded to consist in (1) making the theory of conditional Gaussian estimation accessible to engineers and in (2) combining it with the concept of dynamic state space models, which was at that time developed in control systems theory [70, 71, 72]. Just four years after Kalman's original publication, Ho and Lee [73] showed that the Kalman filter can be interpreted in a Bayesian fashion. This *Bayesian interpretation* is explained in more detail in [74], and it forms the basis of the transformation-centric view that has been taken in this work.

4.5 Kalman Filter Extensions

The Kalman filter (KF) provides a very attractive and, particularly important, tractable formulation to sequential Bayesian estimation. However, it also requires the process and measurement functions to be linear, which severely limits the practical applicability of the KF. That motivated the development of extensions, such as the *extended Kalman filter* (EKF) [41], which "locally" linearizes f and h around the current state estimate, as described in Section 3.4 and again portrayed in the graph below.



In this graph, $\bar{f}(x)$ denotes a linearized version of f which has been obtain with a first order Taylor series approximation around x_0. Apart from the linearization, the EKF works exactly like the original Kalman filter: it constructs a joint Gaussian distribution of $X_t, Y_t|y_{1:t-1}$ and then conditions this joint distribution on the new observation y_t. This works surprisingly well in most practical situations, although the EKF also has its shortcomings:

(1) the local linearizations can produce highly unstable filters if the assumption of local linearity is violated.

(2) the first-order Taylor series expansion requires the calculation of Jacobian matrices, which are often tedious to implement.

"These reasons", as some guys form the British defense industry put it in a personal communication, "led some Americans to believe that the extended Kalman filter stinks". The Americans mentioned were, of course, no one less than Julier and Uhlman who came up with the idea of an "unscented" Kalman filter (UKF) [51]. This filter supposedly avoids the above mentioned problems by replacing the Kalman-type linear transforms with the augmented unscented transform (AUT) from Section 3.5.2. That is done as indicated in Algorithm 4.2. The required augmented unscented transform of two Gaussian random variables is performed as described in Algorithm 4.3. As an alternative to the AUT, the extensive unscented transform (Section 3.6) may be used [18]. In both cases, the resulting filter is at least as accurate[2] as a second order EKF [75], without the need for computing Jacobians or Hessians [54, 52, 53] and without greatly increasing the computational cost [53].

[2] That is because both the augmented and extensive unscented transforms are accurate at least up to the second order term of the Taylor series expansion.

The Unscented Kalman Filter

1. **Predict State**: Use the augmented unscented transform from section 3.5.2 in order to approximate the predicted distribution $p_{X_t|y_{1:t-1}}(x_t)$ of the next state according to (4.5): $p_{X_t|y_{1:t-1}}(x_t) = \mathrm{AUT}_2\left\{p_{X_{t-1}|y_{1:t-1}}, p_W; f\right\}$, where AUT_2 is given by Algorithm 4.3.

2. **Construct the Joint**: Use the augmented unscented transform in order to construct the joint distribution $p_{X_t,Y_t|y_{1:t}}$ of state and observation according to (4.6): $p_{X_t,Y_t|y_{1:t-1}}(x_t) = \mathrm{AUT}_2\left\{p_{X_t|y_{1:t-1}}, p_V; \tilde{h}\right\}$. This gives the joint Gaussian distribution

$$p_{X_t,Y_t|y_{1:t-1}}(x_t) = \mathcal{N}\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}; \begin{bmatrix} \mu^-_{X_t} \\ \mu_{Y_t} \end{bmatrix}, \begin{bmatrix} \Sigma^-_{X_tX_t} & \Sigma_{X_tY_t} \\ \Sigma_{Y_tX_t} & \Sigma_{Y_tY_t} \end{bmatrix}\right).$$

3. **Condition (Update)**: As in the Kalman filter, calculate $\mu^+_{X_t} = \mu^-_{X_t} + \Sigma_{X_t,Y_t}\Sigma^{-1}_{Y_t,Y_t}(y_t - \mu_{Y_t})$ and $\Sigma^+_{X_tX_t} = \Sigma^-_{X_tX_t} - \Sigma_{X_tY_t}\Sigma^{-1}_{Y_tY_t}\Sigma_{Y_tX_t}$. Then set $p_{X_t|y_{1:t}} = \mathcal{N}(\mu^+_{X_t}, \Sigma^+_{X_t,X_t})$.

Algorithm 4.2: Unscented Kalman Filter

$p_Y = \mathrm{AUT}_2\left\{p_X, p_V; f\right\}$:

1. Calculate the right Cholesky factors A and B of the covariance matrices Σ_X and Σ_V of the n_X- and n_V-dimensional Gaussian input distributions.

2. Generate points $\left[\mathcal{X}_k^T \; \mathcal{V}_k^T\right]^T$ and weights W_k, $k = 0, \ldots, N-1$ with $N = 2(n_X + n_W) + 1$, in order to approximate the joint distribution of X and V. This is done according to (3.27):

$$\begin{array}{l|l|l} \mathcal{X}_0 = \mu_X & \mathcal{V}_0 = \mu_V & W_0 = \kappa/\lambda \\ \mathcal{X}_{2i+1} = \mu_X + \sqrt{\lambda}A_i & \mathcal{V}_{2i+1} = \mu_V & W_{2i+1} = 1/(2\lambda) \\ \mathcal{X}_{2i+2} = \mu_X - \sqrt{\lambda}A_i & \mathcal{V}_{2i+2} = \mu_V & W_{2i+2} = 1/(2\lambda) \\ \mathcal{X}_{2(n+j)+1} = \mu_X & \mathcal{V}_{2(n+j)+1} = \mu_V + \sqrt{\lambda}B_j & W_{2(n+j)+1} = 1/(2\lambda) \\ \mathcal{X}_{2(n+j)+2} = \mu_X & \mathcal{V}_{2(n+j)+2} = \mu_V - \sqrt{\lambda}B_j & W_{2(n+j)+2} = 1/(2\lambda) \end{array} \tag{4.17}$$

for $i = 0, \ldots, (n_X - 1)$, $j = 0, \ldots, (n_V - 1)$. The A_i and B_j denote the i-th and j-th rows of A and B, respectively, and $\lambda = n_X + n_W + \kappa$.

3. Transform the points $\left[\mathcal{X}_k^T \; \mathcal{V}_k^T\right]^T$ according to (3.28): $\mathcal{Y}_k = f(\mathcal{X}_k, \mathcal{V}_k)$

4. Estimate the mean and covariance matrix of the transformed random variable Y according to: $\mu_Y = \sum_{k=0}^{N-1} W_k \mathcal{Y}_k$ and $\Sigma_Y = \sum_{k=0}^{N-1} W_k \left(\mathcal{Y}_k - \mu_Y\right)\left(\mathcal{Y}_k - \mu_Y\right)^T$.

5. Return $p_Y(y) = \mathcal{N}(y; \mu_Y, \Sigma_Y)$

Algorithm 4.3: Augmented Unscented Transform for 2 independent Variables

4.6 Gaussian Mixture Filters

The Kalman filter extensions from the previous section provide a suitable solution if the assumption of local linearity is approximately valid. But the Gaussian fits of the EKF and UKF may no longer provide a reasonable approximation of the true filtering density if there are considerable nonlinearities or if there is non-Gaussian process or measurement noise. The idea behind Gaussian mixture filters is to treat such cases (of nonlinear and non-Gaussian tracking problems) by approximating the filtering density as a Gaussian mixture. This is beneficial as a sufficiently large number of mixture components can approximate any density of interest with an arbitrary accuracy. The original implementation by Alspach and Sorenson [47, 48] achieved that by maintaining a bank $\mathscr{K}_t^1, \ldots, \mathscr{K}_t^{N_t}$ of Kalman filters in order to

(1) treat Gaussian mixture process and measurement noise by operating one unscented Kalman filter per Gaussian noise component [47].

(2) treat nonlinear tracking problems by choosing the variances of the individual Kalman filters small enough for the local linearizations to be approximately valid for each of the filters [48] (also see Section 3.7).

In both cases, propagating the Kalman filters as well as their posterior probabilities through time gives a Gaussian mixture filtering density, where the filters' individual densities are Gaussian components and where the posterior probabilities are their weights. In order to formally derive this, let $p_{X_t|\mathscr{K}_t^n, y_{1:t}}(x_t)$ denote the Gaussian filtering density of the n-th Kalman filter $\mathscr{K}_t^n$ at time t, with $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$. Then the overall filtering density $p_{X_t|y_{1:t}}(x_t)$ can be expressed as marginal density of $p_{X_t, \mathscr{K}_t|y_{1:t}}(x_t, \mathscr{K}_t^n)$:

$$p_{X_t|y_{1:t}}(x_t) = \sum_{n=1}^{N_t} \underbrace{p_{X_t|\mathscr{K}_t^n, y_{1:t}}(x_t) \cdot p_{\mathscr{K}_t|y_{1:t}}(\mathscr{K}_t^n)}_{=p_{X_t,\mathscr{K}_t|y_{1:t}}(x_t,\mathscr{K}_t^n)}. \tag{4.18}$$

Here, $p_{X_t|\mathscr{K}_t^n, y_{1:t}}(x_t)$ is a Gaussian component and $p_{\mathscr{K}_t|y_{1:t}}(\mathscr{K}_t^n)$ is the corresponding posterior probability (or weight), which can be calculated according to Bayes' rule:

$$p(\mathscr{K}_t^n|y_{1:t}) = \frac{p(y_t|\mathscr{K}_t^n, y_{1:t-1}) \cdot p(\mathscr{K}_t^n|y_{1:t-1})}{\sum_{n'=1}^{N_t} p(y_t|\mathscr{K}_t^{n'}, y_{1:t-1}) \cdot p(\mathscr{K}_t^{n'}|y_{1:t-1})}. \tag{4.19}$$

This is the main principle behind Gaussian mixture filters. For the sake of notational simplicity, the dependency on $y_{1:t}$ will be dropped in the following. So, we will write $p(\mathscr{K}_t^n)$ instead of $p(\mathscr{K}_t^n|y_{1:t})$ and $p_{X_t|\mathscr{K}_t^n}(x_t)$ instead of $p_{X_t|\mathscr{K}_t^n, y_{1:t}}(x_t)$. Also note that (4.18) and (4.19) are the sequential equivalents to (3.8) and (3.9) from the Gaussian mixture implementation of Bayesian estimation (see Section 3.2.2). Just replace the component index k by the filter index $\mathscr{K}_t^n$ and replace the observation y by the set $y_{1:t}$ of past observations. This result will be used in the upcoming sections in order to

(A) construct an unscented Gaussian mixture filter for non-Gaussian process and measurement noise (Section 4.6.1) [19].

(B) develop a Gaussian mixture filter for nonlinear tracking problems based on the adaptive level of detail transform (Section 4.6.3) [13].

As, for these filters, the number of mixture components grows exponentially in time, Section 4.6.2 introduces Gaussian mixture reduction techniques.

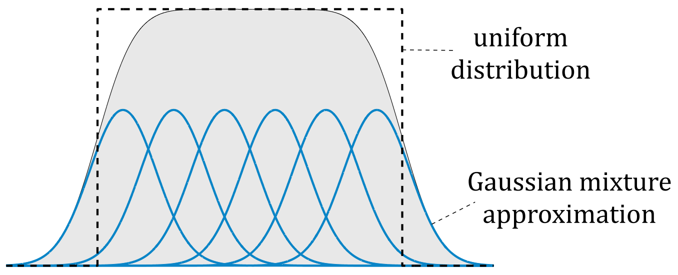
4.6.1 The Unscented Gaussian Mixture Filter for non-Gaussian Noise

For the derivation of the Kalman filter and its extensions, the process and measurement noise distributions were assumed to be Gaussian. This allowed for an analytically tractable *conjugate prior* solution in which both the prior and posterior distributions remained Gaussian at all times. Although this Gaussian assumption is supported by the fairly generally accepted fact that the errors of macroscopic systems tend to have a Gaussian distribution [40], there are practical scenarios in which the noise distributions are non-Gaussian. In acoustics, for example, it is well known that the relative phase between two independent signals has a uniform distribution on the interval $[0, 2\pi]$ (see [76] for example). In such a case, the process and measurement noise distributions may be approximated as Gaussian mixtures [47]:

$$p_{W_t}(w_t) \approx \sum_{k=1}^{K} c_W^{(k)} \mathcal{N}(w_t; \mu_W^{(k)}, \Sigma_W^{(k)}), \tag{4.20}$$

$$p_{V_t}(v_t) \approx \sum_{l=1}^{L} c_V^{(l)} \mathcal{N}(v_t; \mu_V^{(l)}, \Sigma_V^{(l)}), \tag{4.21}$$

where $\mathcal{N}$ denotes the Gaussian distribution, and where $c_V^{(l)}$, $\mu_V^{(l)}$, $\Sigma_V^{(l)}$ and $c_W^{(k)}$, $\mu_W^{(k)}$, $\Sigma_W^{(k)}$ denote the prior probability, mean, and covariance, respectively, of the l-th and k-th Gaussian components.



These process and measurement noise approximations can now be treated within the Gaussian mixture filter framework by operating one unscented Kalman filter per Gaussian noise component. Algorithm 4.4 describes this at the example of the *unscented Gaussian mixture filter* (UGMF), which achieves the above by first splitting each Kalman filter into K and L filters, respectively, and then assigning one filter to each of the Gaussian noise components, as visualized in Figure 4.2. After the splitting step, all the filters are updated with the current observation y_t and their posterior probabilities are calculated according to (4.19).

The Unscented Gaussian Mixture Filter

1. **Predict State**: Split each filter $\mathcal{K}_{t-1}^{n}$ into K identical filters $\mathcal{K}_{t-1}^{n,1},\dots,\mathcal{K}_{t-1}^{n,K}$. Assign each $\mathcal{K}_{t-1}^{n,k}$ to the k-th process noise component $p_{W_t|k}(w_t) = \mathcal{N}(w_t;\mu_W^{(k)},\Sigma_W^{(k)})$. Then predict each filter's state distribution with the augmented unscented transform: $p_{X_t|\mathcal{K}_{t-1}^{n,k}}(x_t) = \mathrm{AUT}_2\left\{p_{X_{t-1}|\mathcal{K}_{t-1}^{n}}, p_{W_t|k}; f\right\}$, where f is the state transition function from (4.3). Finally, update each filter's probability by multiplying it by the probability of the assigned noise component: $p_{\mathcal{K}_{t-1}}(\mathcal{K}_{t-1}^{n,k}) = c_W^{(k)} p_{\mathcal{K}_{t-1}}(\mathcal{K}_{t-1}^{n})$.

2. **Construct the Joint**: In analogy to state prediction, split each filter $\mathcal{K}_{t-1}^{n,k}$ into L filters $\mathcal{K}_{t-1}^{n,k,1},\dots,\mathcal{K}_{t-1}^{n,k,L}$ and assign $\mathcal{K}_{t-1}^{n,k,l}$ to the l-th observation noise component. Proceed by predicting the joint distribution of state and observation with the augmented unscented transform: $p_{X_t,Y_t|\mathcal{K}_{t-1}^{n,k,l}}(x_t,y_t) = \mathrm{AUT}_2\left\{p_{X_t|\mathcal{K}_{t-1}^{n,k}}, p_{V_t|l}; \tilde{h}\right\}$, where $p_{V_t|l}(v_t) = \mathcal{N}(v_t;\mu_V^{(l)},\Sigma_V^{(l)})$ and where $\tilde{h}$ is the augmented observation function from (4.6). Then update the corresponding probabilities $p_{\mathcal{K}_{t-1}}(\mathcal{K}_{t-1}^{n,k,l}) = c_V^{(l)} p_{\mathcal{K}_{t-1}}(\mathcal{K}_{t-1}^{n,k})$.

3. **Condition (Update)**: Update the filtering density of each filter $\mathcal{K}_{t-1}^{n,k,l}$ by conditioning the joint distribution on the realized observation, as in the update step of the unscented Kalman filter (Algorithm 4.2, step 3). Then calculate the filters' posterior probabilities according to (4.19):

$$p_{\mathcal{K}_t}(\mathcal{K}_t^{n,k,l}) = \frac{p_{Y_t|\mathcal{K}_{t-1}^{n,k,l}}(y_t)\, p_{\mathcal{K}_{t-1}}(\mathcal{K}_{t-1}^{n,k,l})}{\sum_{n'=1}^{N_t}\sum_{k'=1}^{K}\sum_{l'=1}^{L} p_{Y_t|\mathcal{K}_{t-1}^{n,k,l}}(y_t)\, p_{\mathcal{K}_{t-1}}(\mathcal{K}_{t-1}^{n',k',l'})}.$$

Algorithm 4.4: Unscented Gaussian Mixture Filter

Gaussian Mixture Reduction

1. Select the filter $\mathcal{K}_t^{n}$ with the lowest posterior probability $p_{\mathcal{K}_t}(\mathcal{K}_t^{n})$ and determine its similarity $\rho\left(p_{X_t|\mathcal{K}_t^{m}}, p_{X_t|\mathcal{K}_t^{n}}\right)$ to all other filters $\mathcal{K}_t^{m}$, $m \neq n$, as described in Section 4.6.2.2.

2. Merge $\mathcal{K}_t^{n}$ with the most similar filter $\mathcal{K}_t^{m}$ by

   (a) merging the Gaussian filtering densities of $\mathcal{K}_t^{n}$ and $\mathcal{K}_t^{m}$ as described in Section 4.6.2.1, with

   $$\begin{aligned} g_1 &= p_{X_t|\mathcal{K}_t^{n}}(x_t), & c_1 &= p_{\mathcal{K}_t}(\mathcal{K}_t^{n})/\left((p_{\mathcal{K}_t^{n}}(\mathcal{K}_t^{n}) + p_{\mathcal{K}_t^{n}}(\mathcal{K}_t^{m})\right),\\ g_2 &= p_{X_t|\mathcal{K}_t^{m}}, & c_2 &= p_{\mathcal{K}_t}(\mathcal{K}_t^{m})/\left((p_{\mathcal{K}_t^{n}}(\mathcal{K}_t^{n}) + p_{\mathcal{K}_t^{n}}(\mathcal{K}_t^{m})\right). \end{aligned}$$

   (b) initializing $\mathcal{K}_t^{n}$ with the merged filtering density, setting the corresponding posterior probability $p_{\mathcal{K}_t}(\mathcal{K}_t^{n})$ to $p_{\mathcal{K}_t}(\mathcal{K}_t^{n}) + p_{\mathcal{K}_t}(\mathcal{K}_t^{m})$, and then discarding $\mathcal{K}_t^{m}$ by first swapping $\mathcal{K}_t^{m}$ with $\mathcal{K}_t^{N_t}$ and subsequently decreasing the number of filters (N_t) by one.

3. If $N_t > M$ go back to step 1.

Algorithm 4.5: Gaussian Mixture Reduction

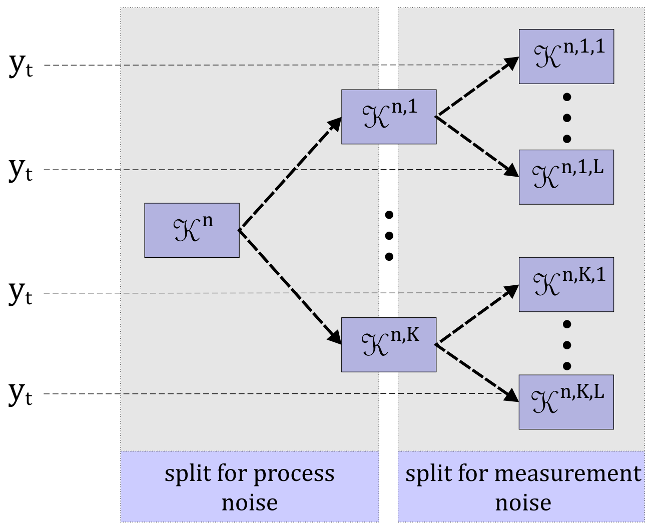


Figure 4.2: *Schematic diagram of the unscented Gaussian mixture filter. In the first splitting step, the Kalman filters are split for each process noise component. In the second splitting step, they are split for each measurement noise component.*

At the end of each iteration, the filter indices are relabeled by setting $\mathcal{K}_t^{n'} = \mathcal{K}^{n,k,l}$ with $n' = nKL + kL + l$, for $n = 1,\dots,N_{t-1}$, $k = 1,\dots,K$, $l = 1,\dots,L$ and then updating the number of filters to $N_t = N_{t-1}KL$. Subsequently, the minimum mean squared error estimate $\hat{x}_t|y_{1:t}$ of the state can be obtained as the conditional mean:

$$\mu_{X_t} \quad = \quad \int \sum_{n=1}^{N_t} p_{\mathcal{K}_t}(\mathcal{K}_t^n) p_{X_t|\mathcal{K}_t^n, y_{1:t}}(x_t) d x_t \quad = \quad \sum_{n=1}^{N_t} c_{\mathcal{K}_t^n} \mu_{X_t|\mathcal{K}_t^n}, \tag{4.22}$$

with $\mu_{X_t|\mathcal{K}_t^n}$ denoting the mean of the n-th Kalman filter and with $c_{\mathcal{K}_t^n}$ denoting the posterior probability $p_{\mathcal{K}_t}(\mathcal{K}_t^n)$ of the n-the Kalman filter at time t.

4.6.2 Gaussian Mixture Reduction

The relabeling step at the end of the previous paragraph shows that the number of filters increases by a factor of $K \cdot L$ at every time instant t. This leads to an exponential growth in the number of filters, which quickly becomes intractable from a computational point of view. Hence, Alspach and Sorenson [47] proposed to reduce the number of filters after each iteration by removing components with low posterior probability and merging components with similar means [47, 48]. This approach was further advanced by Salmond who investigated Gaussian mixture reduction techniques in the context of multi-target tracking in clutter [58, 59] (where the problem of an exponentially growing number of filters arises due to the data association problem[3]). But this topic is not a main focus of this thesis. Hence, the reader is referred to [60, 61, 77] for an in-depth discussion of more recent techniques.

[3] that is, uncertainty regarding from which target a specific observation originated.

In this thesis, mixture reduction is performed as described in Algorithm 4.5. The algorithm given there is similar to Salmond's joining algorithm [59]. It merges components successively in pairs until a predefined number of M components has been reached. In every iteration, the component with the smallest weight is selected and then merged with that Gaussian which it is most similar to. That is achieved by calculating the similarity as the cosine between the Gaussians as described in Section 4.6.2.2. Merging is performed with the moment matching based approach from Section 4.6.2.1. As an alternative to Section 4.6.2.2, the similarity may be determined with the proximate Kullback-Leibler divergence from Section 4.6.2.3. Preliminary experiments, however, indicated that this gives inaccurate results. This was later confirmed in a personal communication with the authors of [66].

4.6.2.1 Merging Two Gaussians

Let $g_1(x) = \mathcal{N}(x;\mu_1,\Sigma_1)$ and $g_2(x) = \mathcal{N}(x;\mu_2,\Sigma_2)$ be Gaussian distributions with weights c_1 and c_2 such that $c_1 + c_2 = 1$. Then g_1 and g_2 can be merged to a single Gaussian $\mathcal{N}(x;\tilde{\mu},\tilde{\Sigma})$ by estimating the mean $\tilde{\mu}$ and covariance $\tilde{\Sigma}$ of their mixture $m(x) = c_1 g_1(x) + c_2 g_2(x)$ [59, 18]. That is achieved by calculating of the expectation integrals $\mathcal{E}_{m(x)}\{x\}$ and $\mathcal{E}_{m(x)}\left\{(x-\mu)(x-\mu)^T\right\}$:

$$\begin{aligned}\tilde{\mu} &= c_1\mu_1 + c_2\mu_2, && (4.23)\\ \tilde{\Sigma} &= c_1\left(\Sigma_1 + \mu_1\mu_1^T\right) + c_2\left(\Sigma_2 + \mu_2\mu_2^T\right) - \tilde{\mu}\tilde{\mu}^T. && (4.24)\end{aligned}$$

4.6.2.2 Similarity Measure Based on Cosine

In this work, the similarity of two Gaussians $g_1(x) = \mathcal{N}(x;\mu_1,\Sigma_1)$ and $g_2(x) = \mathcal{N}(x;\mu_2,\Sigma_2)$ is determined by calculating the cosine between g_1 and g_2:

$$\rho(g_1,g_2) \triangleq \frac{\int g_1(x)g_2(x)dx}{\sqrt{\int g_1(x)g_1(x)dx \int g_2(x)g_2(x)dx}}. \tag{4.25}$$

To simplify (4.25), let us make use of the fact that the product of two Gaussians, g_i and g_j, is a scaled single Gaussian: $g_i(x)g_j(x) = \alpha_{i,j}\mathcal{N}(x;\mu_{i,j},\Sigma_{i,j})$ with

$$\begin{gathered}\alpha_{i,j} = \frac{\mathcal{N}(0;\mu_i,\Sigma_i)\mathcal{N}(0;\mu_j,\Sigma_j)}{\mathcal{N}(0;\mu_{i,j},\Sigma_{i,j})},\\ \mu_{i,j} = \Sigma_{i,j}\left(\Sigma_i^{-1}\mu_i + \Sigma_j^{-1}\mu_j\right), \quad \Sigma_{i,j} = \left(\Sigma_i^{-1} + \Sigma_j^{-1}\right)^{-1}.\end{gathered} \tag{4.26}$$

Using this representation, it is clear that the integrals $\int g_i(x)g_j(x)dx$ in the nominator and denominator of (4.25) evaluate to $\alpha_{i,j}$, as

$$\int g_i(x)g_j(x)dx = \int \alpha_{i,j}\mathcal{N}(x;\mu_{i,j},\Sigma_{i,j})dx = \alpha_{i,j}\underbrace{\int \mathcal{N}(x;\mu_{i,j},\Sigma_{i,j})dx}_{=1} = \alpha_{i,j}.$$

Especially, for $i = j$ we have $\alpha_{i,i} = \mathcal{N}(0;\mu_i,\Sigma_i)^2/\mathcal{N}\left(0;\mu_i,\frac{1}{2}\Sigma_i\right)$. Hence, the similarity of $g_1(x)$ and $g_2(x)$ can be calculated as

$$\rho(g_1,g_2) = \frac{\sqrt{\mathcal{N}\left(0;\mu_1,\frac{1}{2}\Sigma_1\right)\mathcal{N}\left(0;\mu_2,\frac{1}{2}\Sigma_2\right)}}{\mathcal{N}(0;\mu_{1,2},\Sigma_{1,2})} \tag{4.27}$$

with $\mu_{1,2}$ and $\Sigma_{1,2}$ being defined as in (4.26) [18].

4.6.2.3 Similarity Measure Based on Proximate Kullback-Leibler Divergence

As an alternative to the above, the similarity between two Gaussians g_1 and g_2 with associated weights c_1 and $c_2 = 1 - c_1$ can be calculated as the Kullback-Leibler (KL) divergence $D\left(m\|\tilde{g}\right)$ between the mixture $m(x) = c_1 g_1(x) + c_2 g_2(x)$ and the Gaussian $\tilde{g}(x) = \mathcal{N}\left(x;\tilde{\mu},\tilde{\Sigma}\right)$ that would result from merging g_1 and g_2 according to Section 4.6.2.1. This approach is complicated by the fact that there is no analytic solution to the occurring integrals. Nevertheless, the KL divergence can be approximated by using the point mass representation from the unscented transform [78, 79]. This works as follows. Let $f(x)$ be a Gaussian mixture distribution,

$$f(x) = \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}(x;\mu_k,\Sigma_k)}_{\hat{=} f(x|k)},$$

and let $g(x)$ be an arbitrary distribution. Then the KL divergence $D(f\|g)$ of f and g can be written

$$D(f\|g) = \int f(x)\log\left\{\frac{f(x)}{g(x)}\right\}dx = \sum_{k=1}^{K} c_k \int f(x|k)\log\left\{\frac{f(x)}{g(x)}\right\}dx \tag{4.28}$$

and each of the integrals in (4.28) can be approximated by replacing $f(x|k)$ with its unscented point mass representation:

$$\int f(x|k)\log\left\{\frac{f(x)}{g(x)}\right\}dx \approx \sum_{i=0}^{2n} W_i \log\left\{\frac{f\left(\mathcal{X}_i^{(k)}\right)}{g\left(\mathcal{X}_i^{(k)}\right)}\right\} \tag{4.29}$$

where the $W_0,\ldots,W_{2n}$ as well as the $\mathcal{X}_0^{(k)},\ldots,\mathcal{X}_{2n}^{(k)}$ are defined as specified in Section 3.5.

4.6.3 The Adaptive Gaussian Mixture Filter for Nonlinear Tracking

Now that the problem of Gaussian mixture reduction has been addressed, let us revisit the Gaussian mixture filter from Section 4.6.1. The idea of that filter was to handle non-Gaussian process and measurement noise by first approximating the noise distributions as Gaussian mixtures and then operating one Kalman filter for each of the Gaussian noise components. Alspach and Sorenson [48] discovered that this principle can also be used to cope with nonlinear process and measurement equations. That is achieved by

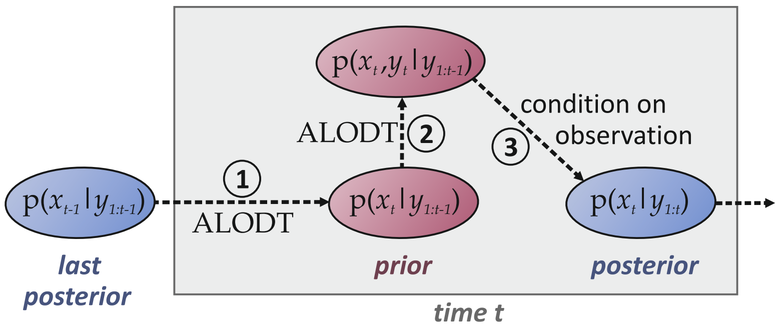
(1) approximating the Gaussian noise distributions as Gaussian mixtures.

(2) treating potential nonlinearities by choosing the variances of the individual Gaussians small enough for the local linearizations to be valid for each of the Kalman filters.

This procedure gives more accurate results for nonlinear tracking problems. But it also introduces density approximation errors, which may be harmful in regions where the linearization error is smaller than the error of approximating a Gaussian by a Gaussian mixture. In particular, the procedure does not guarantee that the variances of the filtering densities remain small enough for the local linearizations to be valid throughout time. Hence, this thesis discards the idea of running a bank of Kalman filters and, instead, focuses on approximating the joint predictive distribution of state and observation as accurately as possible. That is achieved by using the adaptive level of detail transform from Section 3.7 for both predicting the state and constructing the joint distribution of state and observation:

$$\begin{bmatrix} X_{t-1}|y_{1:t-1} \\ W_t \end{bmatrix} \xrightarrow[f]{ALODT} X_t|y_{1:t-1} \quad \text{and} \quad \begin{bmatrix} X_t|y_{1:t-1} \\ V_t \end{bmatrix} \xrightarrow[\tilde{h}]{ALODT} \begin{bmatrix} X_t|y_{1:t-1} \\ Y_t|y_{1:t-1} \end{bmatrix}.$$

This effectively approximates each involved random variable – i.e. $X_{t-1}|y_{1:t-1}$, W_t and V_t – by a Gaussian mixture distribution, however, in such a fashion that the linearization error is minimized during transformation. The result is a Gaussian mixture approximation of $p_{X_t,Y_t|y_{1:t-1}}$, from which the new posterior can be extracted by conditioning on the realized observation.



Putting these steps together yields the adaptive Gaussian mixture filter [13], which is described in Algorithm 4.6. It was developed in the framework of this thesis, starting with [19] where the idea was to split Kalman filters in likely regions of state space and to merge them in unlikely regions. The motivation for this approach was to adapt the level of detail of the filtering density according to the posterior probability of the modes, similar as it is done in the resampling stage of particle filters by multiplying and removing samples. Unfortunately, this approach cannot improve the results in linear regions of state space, where the Kalman filter is optimal[4]. Hence, in the framework of this thesis [19], it has been proposed to use a split control technique that prevents filters from being split if they operate in relatively linear regions of state space. This

[4] under the assumption of Gaussian process and measurement noise

The Adaptive Gaussian Mixture Filter

1. **Predict State**: Use the adaptive level of detail transform from section 3.7 in order to approximate the predicted distribution $p_{X_t|y_{1:t-1}}$ of the next state according to (4.5):

$$p_{X_t|y_{1:t-1}} = \mathrm{ALoDT}_2\left\{p_{X_{t-1}|y_{1:t-1}}, p_W; f\right\}.$$

   Here, ALoDT$_2$ is the adaptive level of detail transform of two stacked variables.

2. **Construct the Joint**: Use the adaptive level of detail transform in order to construct the joint distribution $p_{X_t,Y_t|y_{1:t}}$ of state and observation according to (4.6):

$$p_{X_t,Y_t|y_{1:t-1}} = \mathrm{ALoDT}_2\left\{p_{X_t|y_{1:t-1}}, p_V; \tilde{h}\right\}.$$

   This gives a joint Gaussian mixture distribution as in Section 3.2.2:

$$p_{X_t,Y_t|y_{1:t-1}}(x_t, y_t) = \sum_{k=1}^{\kappa} c_k \mathcal{N}\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}; \begin{bmatrix} \mu^-_{X_t|k} \\ \mu_{Y_t|k} \end{bmatrix}, \begin{bmatrix} \Sigma^-_{X_t X_t|k} & \Sigma_{X_t Y_t|k} \\ \Sigma_{Y_t X_t|k} & \Sigma_{Y_t Y_t|k} \end{bmatrix}\right).$$

3. **Condition (Update)**: As in Section 3.2.2, condition the joint Gaussian mixture distribution of X_t and Y_t on the realized observation y_t. This is done in analogy to (3.8):

$$p_{X_t|y_{1:t}}(x_t) = \sum_{k=1}^{\kappa} c_k^+ \mathcal{N}\left(x_t; \mu^+_{X_t|k}, \Sigma^+_{X_t X_t|k}\right)$$

   where $\mu^+_{X_t|k} = \mu^-_{X_t|k} + \Sigma_{X_t,Y_t|k}\Sigma^{-1}_{Y_t,Y_t|k}(y_t - \mu_{Y_t|k})$ and $\Sigma^+_{X_t X_t|k} = \Sigma^-_{X_t X_t|k} - \Sigma_{X_t Y_t|k}\Sigma^{-1}_{Y_t Y_t|k}\Sigma_{Y_t X_t|k}$ and where c_k^+ is calculated according to (3.9):

$$c_k^+ = \frac{c_k p_{Y|k}(y)}{\sum_{k'=1}^{\kappa} c_{k'} p_{Y|k'}(y)} = \frac{c_k \mathcal{N}\left(y_t; \mu_{Y_t|k}, \Sigma_{Y_t Y_t|k}\right)}{\sum_{k'=1}^{\kappa} c_{k'} \mathcal{N}\left(y_t; \mu_{Y_t|k'}, \Sigma_{Y_t Y_t|k'}\right)}.$$

4. **Simplify**: Simplify the filtering density through Gaussian mixture reduction, as described in Section 4.6.2.

Algorithm 4.6: Adaptive Gaussian Mixture Filter

approach was further extended in [12], which introduced a splitting criterion that considers both the component's mixture weight and its degree of nonlinearity. And it eventually led to the splitting priority from Section 3.7.2.1, which constitutes a suitable splitting criterion, as it reflects the total linearization error that a component contributes to the transformation. The work in [13] further refined this approach by splitting Gaussians in the direction of nonlinearity instead of splitting them in the direction of the largest variance (see Section 3.7.4).

4.6.4 Other Gaussian Mixture Filters

Over the recent years, there has been a surge of interest in Gaussian mixture filters. The local linearizations of Alspach and Sorenson's original implementation [48] have been replaced by numerical integration techniques such as the unscented transform [80], Gauss-Hermite quadrature [56, 81], cubature [82] as well as Monte Carlo integration [83]. There have been innovative approaches like progressive Bayes [62, 84, 85] where Gaussians are split based on their L^2 distance measure to the true density, and the sliced Gaussian mixture filter [86], which uses a Dirac mixture in the nonlinear subspace and a Gaussian mixture in the linear subspace. Aside from these theoretical extensions, Gaussian mixture filters have successfully been applied to radar target tracking [87], integrated INS / GPS navigation [80], simultaneous localization and mapping (SLAM) [88] as well as terrain aided navigation [89]. Also, it has been shown that Gaussian mixture filters can outperform particle filters on certain tasks [19, 89], even when compared to state of the art methods such as the unscented particle filter [90].

4.7 Particle Filters

The sequential Bayesian estimation approaches discussed so far were all based on analytic, continuous density function approximations of the posterior distribution. As an alternative to these, the posterior may by approximated as an empirical, sample based distribution. This has the advantage that (1) prediction can easily be handled through the transformation of samples, and (2) the conditioning step can be performed by simply reweighing the transformed samples in dependence of the realized observation. These two steps form the basis of Gordon et al.'s *bootstrap filter* [91], which is described in more detail in Section 4.7.1. Its advantages are that it imposes no restrictions on the process and measurement noise distributions[5]; nor does it impose restrictions on potential nonlinearities in the process and measurement equations. On the downside, it may, however, require maintaining a large number of samples to achieve a good accuracy of approximation, especially if

(a) the process and measurement equations are not "well-behaved".

(b) the observation likelihood $p_{Y_t|x_t}$ is peaked.

[5] as long as it is possible to sample from these distributions

(c) the tails of the distributions do matter.

(d) the measurement noise is low.

These issues triggered the combination of the bootstrap filter with analytic filtering techniques [92, 90], the theoretical basis for which relies on the fact that the sampling stage of the bootstrap filter can be extended to importance sampling from arbitrary distributions. This leads to the generalized *particle filtering* framework, which is described in Section 4.7.2 and discussed in more detail in the monographs by Liu [93] and Doucet [94] as well as Arulampalam et al.'s IEEE tutorial on particle filters [95].

4.7.1 The Bootstrap Filter

The main idea of the bootstrap filter [91] consists in representing the filtering density as an empirical distribution of N state samples $x_{t-1}^{(j)}$, $j = 1, \ldots, N$:

$$p_{X_{t-1}|y_{1:t-1}}(x_{t-1}) = \frac{1}{N} \sum_{j=1}^{N} \delta\left(x_{t-1} - x_{t-1}^{(j)}\right)$$

These samples are propagated in time by iterating steps 1 - 4 of Algorithm 4.7. In step 1, the empirical distribution of the state is predicted by first drawing N noise samples $w_t^{(j)}$ from the process noise distribution p_{W_t} and then transforming each tuple $\left(x_{t-1}^{(j)}, w_t^{(j)}\right)$ according to the state transition function (4.3). This effectively approximates the prior state distribution through Monte Carlo transformation (see Section 2.5.2):

$$p_{X_t|y_{1:t-1}}(x_t) = \frac{1}{N} \sum_{j=1}^{N} \delta\left(x_t - x_t^{(j)}\right) \quad \text{with} \quad x_t^{(j)} \triangleq f\left(x_{t-1}^{(j)}, w_t^{(j)}\right). \tag{4.31}$$

In step 2, the joint distribution of state and observation is constructed by predicting the observation density $p_{Y_t|x_t^{(j)}}$ of each state sample $x_t^{(j)}$. That is achieved by first transforming[6] the tuples $\left(x_t^{(j)}, V_t\right)$ according to the measurement equation (4.4),

$$Y_t|x_t^{(j)} = h\left(x_t^{(j)}, V_t\right), \tag{4.32}$$

and then multiplying each $\delta\left(x_t - x_t^{(j)}\right)$ by the observation distribution that results from (4.32). This gives the semi-empirical density (see Section 3.2.3)

$$p_{X_t,Y_t|y_{1:t-1}}(x_t, y_t) = \frac{1}{N} \sum_{j=1}^{N} \delta\left(x_t - x_t^{(j)}\right) p_{Y_t|x_t^{(j)}}(y_t), \tag{4.33}$$

where the samples $x_t^{(j)}$ can be regarded to be possible state hypotheses and where the $p_{Y_t|x_t^{(j)}}$ are predicted observation densities that correspond to the $x_t^{(j)}$. Step 3 of the bootstrap filter con-

[6] This transformation is again achieved with the fundamental transformation law of probability.

The Bootstrap Filter

1. **Predict State**: Construct the prior distribution for the estimation problem at time t by first simulating N samples $w_t^1, \ldots, w_t^N$ from the process noise distribution and then transforming the empirical distribution $p_{X_{t-1}|y_{1:t-1}}(x_{t-1}) = \frac{1}{N}\sum_{j=1}^N \delta\left(x_{t-1} - x_{t-1}^{(j)}\right)$ of the last posterior according to the process equation (4.3). This gives:

$$p_{X_t|y_{1:t-1}}(x_t) = \frac{1}{N}\sum_{j=1}^N \delta\left(x_t - x_t^{(j)}\right) \quad \text{with} \quad x_t^{(j)} \triangleq f\left(x_{t-1}^{(j)}, w_t^{(j)}\right).$$

2. **Construct the Joint**: Construct the joint distribution $p_{X_t,Y_t|y_{1:t}}$ of state and observation by first predicting the observation distribution $p_{Y_t|x_t^{(j)},y_{1:t-1}}$ of each state sample $x_t^{(j)}$ and then multiplying each $\delta\left(x_t - x_t^{(j)}\right)$ by the corresponding $p_{Y_t|x_t^{(j)},y_{1:t-1}}$. This gives the semi-empirical distribution (see Section 3.2.3)

$$p_{X_t,Y_t|y_{1:t-1}}(x_t, y_t) = \frac{1}{N}\sum_{j=1}^N \delta\left(x_t - x_t^{(j)}\right) p_{Y_t|x_t^{(j)}}(y_t). \tag{4.30}$$

   In the above, the prediction of $p_{Y_t|x_t^{(j)}}$ is achieved by fixing $X_t = x_t^{(j)}$ and then transforming $\left(x_t^{(j)}, V_t\right)$ according to the measurement equation (4.4):

$$p_{Y_t|x_t^{(j)},y_{1:t-1}}(y_t) = h\left(x_t^{(j)}, V_t\right).$$

3. **Condition (Update)**: Condition the semi-empirical density from (4.30) on the realized observation y_t, as described in Section 3.2.3:

$$p_{X_t|y_{1:t}}(x_t) = \sum_{j=1}^N \omega_t^{(j)} \delta\left(x_t - x_t^{(j)}\right) \quad \text{with} \quad \omega_t^{(j)} \triangleq \frac{p_{Y_t|x_t^{(j)}}(y_t)}{\sum_{i=1}^N p_{Y_t|x_t^{(i)}}(y_t)}.$$

4. **Resample**: Use importance resampling (Section 2.5.5) in order to convert the weighted empirical distribution from Step 3 to an (equally-weighted) empirical distribution:

$$\hat{p}_{X_t|y_{1:t}}(x_t) = \frac{1}{N}\sum_{i=1}^N \delta\left(x_t - \tilde{x}_t^{(i)}\right).$$

   Here, $\tilde{x}_t^{(i)}$ denotes a sample drawn from $x_t^{(j)}$ with probability $\omega_t^{(j)}$, $j = 1, \ldots N$.

Algorithm 4.7: The Bootstrap Filter

ditions the joint distribution from (4.33) on the realized observation. That is done according to Section 3.2.3, by first evaluating all the observation densities $p_{Y_t|x_t^{(j)}}$ at the realized observation $Y_t = y_t$, and then weighting each sample $x_t^{(j)}$ by its relative observation likelihood $w_t^{(j)}$:

$$p_{X_t|y_{1:t}}(x_t) = \sum_{j=1}^{N} \omega_t^{(j)} \delta\left(x_t - x_t^{(j)}\right) \quad \text{with} \quad \omega_t^{(j)} \triangleq \frac{p_{Y_t|x_t^{(j)}}(y_t)}{\sum_{i=1}^{N} p_{Y_t|x_t^{(i)}}(y_t)}. \tag{4.34}$$

The last step of the bootstrap filter consists in converting this weighted empirical approximation of the filtering density back to an (equally-weighted) empirical distribution. That is achieved by means of an importance resampling step (step 4 of Algorithm 4.7) which effectively multiplies samples that have a high relative weight and which removes samples that have a low relative weight. This resampling step greatly increases the efficiency of the filter, as it causes that the state space is explored more thoroughly in likely regions and less thoroughly in unlikely regions. A similar effect can be achieved by using the accept / reject procedure from [96] or the clustering approach from [97], which have both been proposed before the bootstrap filter.

4.7.2 Generalized Particle Filtering

The Bootstrap filter is based on first drawing samples $x_t^{(j)}$ from the predicted state variable $X_t|x_t^{(j-1)} = f\left(x_{t-1}^{(j)}, W_t\right)$ and then weighting each of these samples with $\omega_t^{(j)}$. This procedure can be interpreted as importance sampling (see Section 2.5.4) with $p_{X_t|x_t^{(j-1)}}$ as a proposal distribution [94, 93]. Using this interpretation, the idea behind generalized particle filtering can be formulated as extending the bootstrap filter to importance sampling from arbitrary distributions. That may be achieved as follows. Let

$$p_{X_{t-1}|y_{1:t-1}}(x_t) = \frac{1}{N} \sum_{j=1}^{N} \delta\left(x_{t-1} - x_{t-1}^{(j)}\right)$$

be the empirical approximation of the last posterior distribution and let $\pi_t^{(j)}$ be a proposal distribution for the sample $x_t^{(j)}$. Then the predictive distribution $p_{X_t|y_{1:t-1}}$ of the state can be approximated by first drawing one $x_t^{(j)}$ from $\pi_t^{(j)}$, for $j = 1\ldots, N$, and then re-weighting each sample by its corresponding importance weight:

$$\breve{w}_t^{(j)} \triangleq \frac{1}{N} \frac{p_{X_t|x_{t-1}^{(j)}}(x_t^{(j)})}{\pi_t^{(j)}(x_t^{(j)})}.$$

Normalizing these weights gives a weighted empirical approximation of the prior distribution. Hence, we can now perform a Bayesian estimation step, in analogy to steps 2-4 of the bootstrap filter. This gives the generalized particle filter that is described in more detail in Algorithm 4.8.

Generalized Particle Filtering

1. **Predict State**: Construct a proposal distribution $\pi_t^{(j)}$ for each sample $x_{t-1}^{(j)}$ of the last posterior distribution. Then draw one sample $x_t^{(j)}$ from each $\pi_t^{(j)}$. Weight the resulting samples with normalized importance weights in order to match their empirical distribution to the distribution of X_t:

$$p_{X_t|y_{1:t-1}}(x_t) = \sum_{j=1}^{N} \underbrace{\frac{\breve{w}_t^{(j)}}{\sum_{i=1}^{N} \breve{w}_t^{(i)}}}_{\tilde{w}_t^{(j)}} \delta\left(x_t - x_t^{(j)}\right) \quad \text{with} \quad \breve{w}_t^{(j)} \triangleq \frac{1}{N} \frac{p_{X_t|x_{t-1}^{(j)}}\left(x_t^{(j)}\right)}{\pi_t^{(j)}\left(x_t^{(j)}\right)}. \tag{4.35}$$

Here, $p_{X_t|x_{t-1}^{(j)}}\left(x_t^{(j)}\right)$ denotes the likelihood of a transition from $x_{t-1}^{(j)}$ to $x_t^{(j)}$ according to the process equation.

2. **Construct the Joint**: Construct the joint distribution $p_{X_t,Y_t|y_{1:t}}$ of state and observation by first predicting the observation distribution $p_{Y_t|x_t^{(j)},y_{1:t-1}}$ of each state sample $x_t^{(j)}$, and then multiplying each $\left(x_t - x_t^{(j)}\right)$ by its corresponding $p_{Y_t|x_t^{(j)},y_{1:t-1}}$. This gives:

$$p_{X_t,Y_t|y_{1:t-1}}(x_t, y_t) = \sum_{j=1}^{N} \tilde{w}_t^{(j)} \delta\left(x_t - x_t^{(j)}\right) p_{Y_t|x_t^{(j)}}(y_t). \tag{4.36}$$

3. **Condition (Update)**: Condition the semi-empirical density from (4.36) on the realized observation y_t, as described in Section 3.2.3:

$$p_{X_t|y_{1:t}}(x_t) = \sum_{j=1}^{N} \omega_t^{(j)} \delta\left(x_t - x_t^{(j)}\right) \quad \text{with} \quad \omega_t^{(j)} \triangleq \frac{\tilde{w}_t^{(j)} p_{Y_t|x_t^{(j)}}(y_t)}{\sum_{i=1}^{N} \tilde{w}_t^{(i)} p_{Y_t|x_t^{(i)}}(y_t)}.$$

4. **Resample**: Use importance resampling in order to convert the weighted empirical distribution from Step 3 to an (equally-weighted) empirical distribution:

$$\hat{p}_{X_t|y_{1:t}}(x_t) = \frac{1}{N} \sum_{i=1}^{N} \delta\left(x_t - \tilde{x}_t^{(i)}\right).$$

Here, $\tilde{x}_t^{(i)}$ denotes a sample drawn from $x_t^{(j)}$ with probability $\omega_t^{(j)}$, $j = 1, \dots N$.

Algorithm 4.8: Generalized Particle Filtering

4.7.2.1 Choice of the Proposal Distribution

The proposal distributions $\pi_t^{(j)}$ are simply distributions that the samples $x_t^{(j)}$ are drawn from. They can be chosen almost arbitrarily, with the only restriction being that their support

$$\sup\left\{\pi_t^{(j)}\right\} \triangleq \left\{x_t \,\middle|\, \pi_t^{(j)}(x_t) > 0\right\}$$

includes the support of the predicted state density $p_{X_t|x_{t-1}^{(j)}}$. Choosing $\pi_t^{(j)} = p_{X_t|x_{t-1}^{(j)}}$ yields the bootstrap filter from the Section 4.7.1. Other proposal distributions that have been discussed in the literature are based on

- local linearization of the state space model [98].
- using the posterior distribution of an extended Kalman filter [92, 99].
- using the posterior distribution of an unscented Kalman filter [90].

The last two approaches show certain similarities to Gaussian mixture filters, which raises the question whether the sampling framework is really necessary. And it is exactly this question which motivated the development of the adaptive Gaussian mixture filter in the framework of this thesis [19, 12, 13], which tries to achieve the same with analytical approximations.

4.8 Performance Evaluation of Nonlinear Filters

This section compares the performance of different nonlinear filters at the example of tracking a maneuvering object based on polar measurements. This is done in form of a simulation in which an object moves along the synthetic trajectory which is portrayed in Figure 4.3. The trajectory is formally described by:

$$x_t = 10\Big[\sin(s_t)+1 \quad (\cos(s_t)+1)\sin(\tfrac{s_t}{2}) \quad \cos(2s_t)s_t\Big]^T \quad \text{with} \quad s_t = \frac{4\pi t}{500}$$

for $t = 1, \ldots, 500$. The object moving along it is observed by virtual sensors, which are are located at $[0\;0\;0]^T$ and which provide measurements in polar coordinates. Additive measurement noise is simulated from a Gaussian whose covariance matrix is chosen at random – once for each of the 50 experiments that were performed. This gives a standard deviation of 0.27 on average for the distance and an angular accuracy of 3.8°.

Process Model: As a process model, this section uses a zeroth-order linear dynamic model, $x_t = x_{t-1} + w_t$, with zero-mean Gaussian process noise w_t. The process noise covariance Σ_W was estimated on the synthetic trajectory and further scaled by a factor of two in order to increase stability. At time 0, the filters were initialized with a Gaussian distribution around the true state, x_1, with process noise covariance.

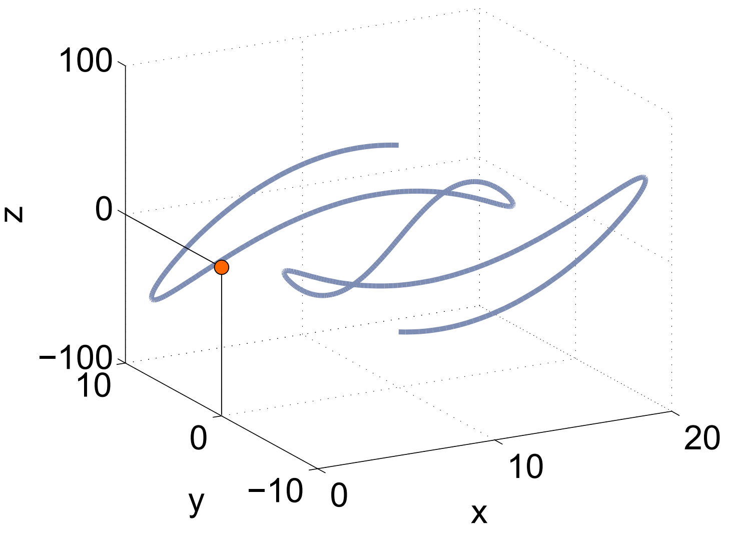


Figure 4.3: *Plot of the trajectory, which was used in the simulations (solid blue line). The location of the sensor is indicated by an orange-filled circle. Its minimum distance from the trajectory is* 3.99 *at* $t = 156$.

Measurement Model: The polar measurement model used in the experiments is formally specified by the measurement equation $y_t = h(x_t) + v_t$ where v_t denotes additive Gaussian measurement noise and where h denotes the function which converts Cartesian to polar coordinates:

$$h\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} \theta \\ \phi \\ r \end{bmatrix} \quad \text{with} \quad \begin{array}{rcl} \theta & = & \text{atan2}(x_2/x_1) \\ \phi & = & \text{atan2}(\cos(\theta)\cdot x_3/x_1) \\ r & = & \sqrt{x_1^2 + x_2^2 + x_3^2} \end{array} .$$

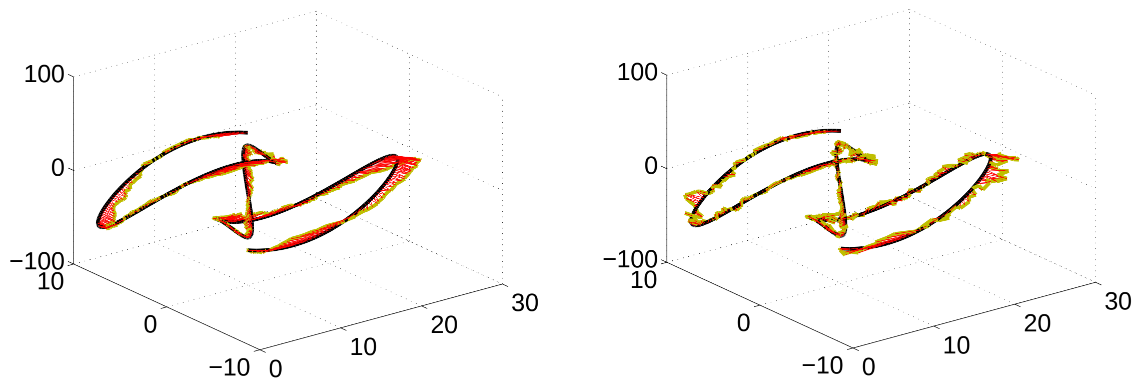
The terms θ, ϕ and r denote the azimuth, elevation and distance, respectively. The measurement noise covariance Σ_V during filtering was the same as in the simulated measurements.

4.8.1 Results for the Adaptive Gaussian Mixture Filter

Table 4.1 shows the mean squared errors (MSEs) that have been obtained with the adaptive Gaussian mixture filter from Sections 4.6.3. The numbers are averaged over 50 simulations and correspond to 500 point trajectories. The first row of the table shows results for splitting in direction of the largest variance. The other rows show the mean squared errors that were obtained by splitting in direction of the nonlinearity. In row two, the direction of nonlinearity was estimated as the eigenvector corresponding to the largest eigenvalue of (3.58). In row three, it was estimated according to (3.59). All the results are shown in dependence of the number of filters (#Gaussians). Splitting was always performed with the “two component” approach from Section 3.7.3.1. The best result was obtained in row 3, with 64 filters. In this case, splitting was performed in the direction of nonlinearity. The MSE was 573, which is 40% lower than that

	Mean Square Error						Computation Time					
direction	#Gaussians						#Gaussians					
of split	2	4	8	16	32	64	2	4	8	16	32	64
variance	779	774	768	765	764	762	0.13	0.27	0.61	1.50	4.24	12.9
nonlinearity#1	935	770	688	632	621	608	0.12	0.26	0.57	1.37	3.59	11.1
nonlinearity#2	1104	825	703	628	595	573	0.11	0.23	0.51	1.26	3.40	10.1

Table 4.1: *Mean square error (MSE) and computation times in seconds of the adaptive Gaussian mixture filter. The numbers correspond to 500 point tracks. Splitting was either performed in the direction of the largest variance or in the direction of nonlinearity. The given results compare to an MSE of 927 for a single unscented Kalman filter, at a computation time of 0.02 seconds.*



(a) Unscented Kalman Filter *(b) Adaptive Gausisan Mixture Filter*

Figure 4.4: *Deviation from the true trajectory for the unscented Kalman filter as well as the adaptive Gaussian mixture filter with 32 Gaussians. The black curve shows the true trajectory. The yellow curve shows the tracking result. The red lines indicate the deviation.*

of the unscented Kalman filter (the UKF had an MSE of 927) and roughly 25% lower than for splitting in the direction of the largest variance (in this case the MSE was 762). The right-hand side of Table 4.1 shows computation times in seconds for the adaptive Gaussian mixture filter (AGMF). With 64 Gaussians it took between 10 and 13 seconds to process a 500 point track. This compares to 0.11 - 0.13 seconds for the AGMF with 2 Gaussians and to 0.02 seconds for a single unscented Kalman filter.

4.8.2 Results for the Bootstrap Filter

Table 4.2 shows the results that have been obtained with the bootstrap (particle) filter from Section 4.7.1. The numbers are again averaged over 50 simulations; they correspond to 500 point tracks; and they are given in dependence of the number of particles (#particles). With 1,000 (1K) particles, the mean squared error was 70% higher than that of the unscented Kalman filter (UKF). At the same time, the computation time was comparable to that of an adaptive Gaussian mixture filter (AGMF) with 12 filters. With 100,000 (100K) particles, the MSE was similar to that

Mean Square Error						Computation Time					
#Particles						#Particles					
1K	5K	10K	50K	100K	500K	1K	5K	10K	50K	100K	500K
1610	1139	1068	967	945	926	0.93	8.93	19.47	111.7	239.5	1477

Table 4.2: *Mean square error (MSE) and computation times in seconds for the bootstrap (particle) filter from Section 4.7.1. The results are shown in dependence of the number of particles.*

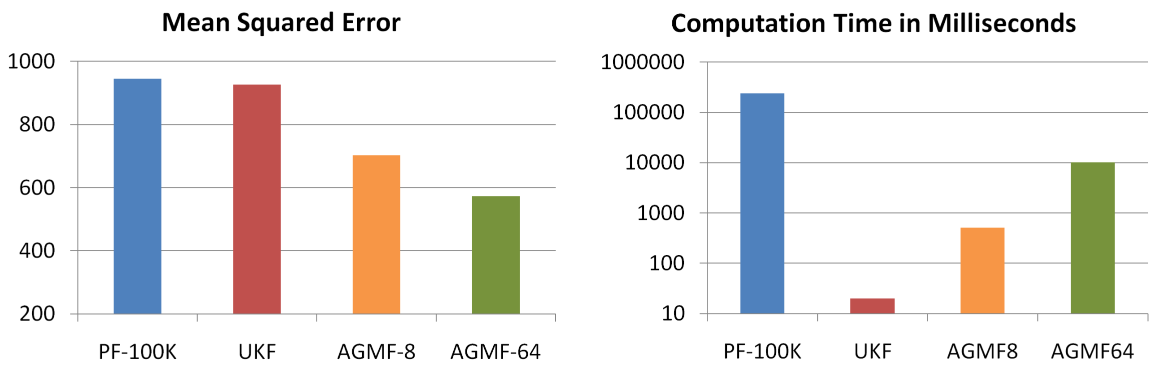


Figure 4.5: *Comparison between the bootstrap filter (PF) with 100,000 particles, the unscented Kalman filter (UKF) as well as the adaptive Gaussian mixture filter (AGMF) with 8 and 64 Gaussians, respectively.*

of the UKF. But the computation time was 20 times higher than for the AGMF with 64 filters. Figure 4.5 finally gives a graphical comparison between the different filters.

4.9 Contributions of this Chapter

The following list again gives an overview of the individual contributions of this thesis to sequential Bayesian estimation:

1. A transformation-centric view on sequential Bayesian estimation (Section 4.3).
2. Derivations for all commonly used tracking algorithms – Kalman, Gaussian mixture and particle filters – based on the above point of view (Sections 4.4, 4.5, 4.6.3, 4.7 and [18]).
3. A variant of the unscented Kalman filter that makes use of the extensive unscented transform [18] (Section 4.5).
4. A novel Gaussian mixture filter which adapts the level of detail (i.e. the number of Gaussians) of all concerned probability distributions (predicted state, process noise and measurement noise) in order to minimize the linearization error [19, 13] (Section 4.6.3).

5
ASR Basics & Speech Features

In order to give the necessary background for the upcoming chapters, this chapter briefly sketches the basic architecture of automatic speech recognition (ASR) systems and then investigates the effect that noise has on clean speech features. Turning to the former, an ASR system may be described as a device that converts an acoustic speech signal to text. Such a device typically consist of two components: a front end and a decoder, which process the speech signal as portrayed in the architectural overview in Figure 5.1.

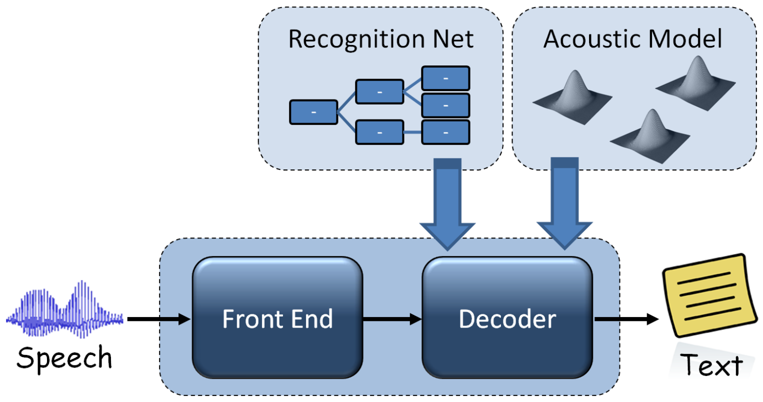


Figure 5.1: *Overview of a typical ASR System.*

While the front-end extracts speech features from the audio signal, the decoder performs a Viterbi "search" [100] in order to find that sentence[1] which is the most likely explanation for the

[1] i.e. path through a network of acoustical states

observed sequence of speech features. This requires two stochastic models, the combination of which defines a hidden Markov process:

1. a recognition network that contains all possible sentences as sequences of acoustical states (s_t) along with associated transition probabilities $p_{S_t|s_{t-1}}(s_t)$.
2. an acoustic model that specifies the distributions $p_{X_t|s_t}(x_t)$ that the extracted speech features (x_t) have in the acoustical states.

Practical (i.e. computationally viable) implementations approximate the full Viterbi search by dropping unlikely state sequences at run time. This is done in relation to the best hypothesis, in a procedure which is generally referred to as beam search [101]. More detailed descriptions of state of the art ASR systems can be found in [101, 102].

5.1 Speech Feature Extraction

The features of modern ASR systems are mostly based on *Mel frequency cepstral coefficients* (MFCCs). Hence, feature extraction typically starts with calculating MFCCs according to the processing chain in Figure 5.2. In this chain, the digitized 16 kHz audio signal is firstly cut

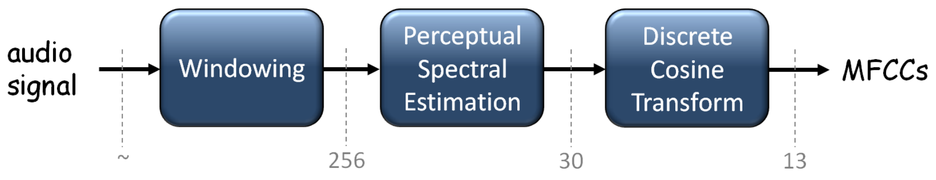


Figure 5.2: *Processing chain for Mel frequency cepstral coefficients.*

into overlapping frames of 256 samples. The samples of each frame are further multiplied by a Hamming window in order to reduce leakage of spectral energy into neighboring frequency bands. After the windowing step, perceptual spectral estimation is performed according to the processing chain in Figure 5.3. This chain starts with calculating the *power spectrum* by first

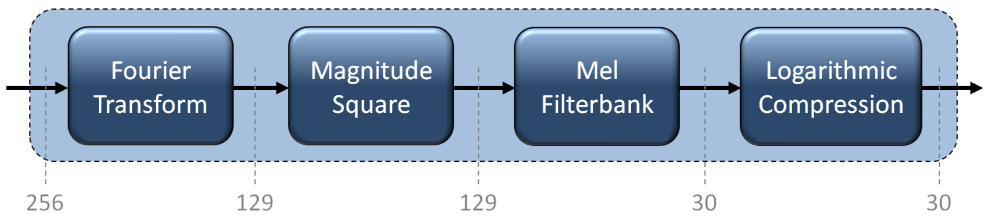
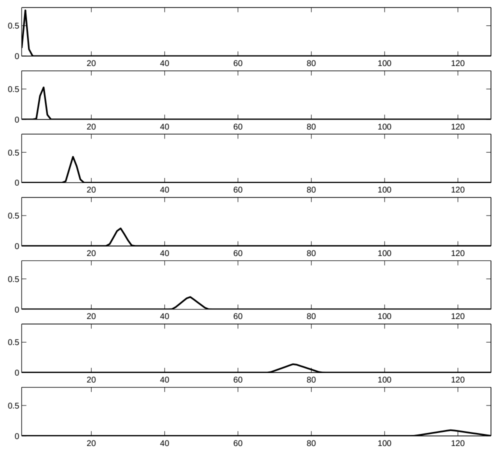


Figure 5.3: *Processing chain for perceptual spectral estimation.*

performing a Fourier transform and then taking the magnitude square $|\cdot|^2$ of each frequency bin. Due to the fact that the power spectrum is symmetric around the 129-th frequency bin, it is sufficient to keep the lower half of the power spectrum. In the next step, the logarithmic frequency perception of the human auditory systems is emulated by mapping the frequencies of the power spectrum to the Mel scale. This is achieved by multiplying the power spectrum with a 30×129 Mel filterbank matrix W whose individual rows are triangular shaped as shown in Figure 5.4-(a). Figure 5.4-(b) shows all the rows in one figure. After applying the Mel filterbank,


(a) *Individual filters.*

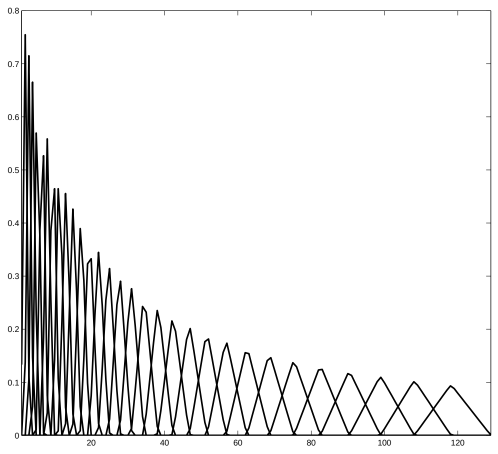

(b) *All filters in one figure.*

Figure 5.4: *Frequency Response of the Mel Filterbank*

the logarithm of each individual frequency bin is taken in order to also mimic the logarithmic intensity perception of the human auditory system. This results in a so-called *log-Mel* spectrum Multiplying it by a 30×30 discrete cosine transform (DCT-II) matrix and then taking the first 13 elements of the resulting vector finally gives MFCC features. The DCT-II matrix used in this work is shown in Figure 5.5. Its elements $c_{n,k}$ are given by $c_{n,k} = cos\left[\frac{\pi}{30}\left(n+\frac{1}{2}\right)k\right]$.

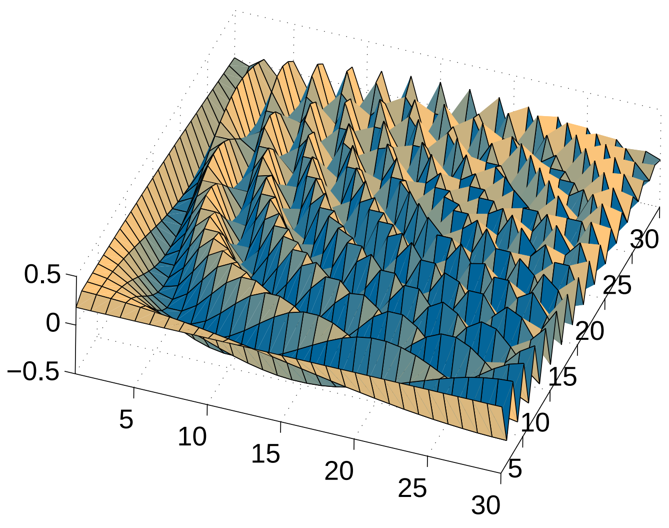


Figure 5.5: *Discrete Cosine Transform.*

MFCCs are perceptually relevant, dimensionally reduced and approximately decorrelated. They do not, however, capture the dynamic properties of speech. Hence, MFCC features are typically augmented with so-called delta and delta-delta features [101], which are essentially their first and second order derivatives with respect to time [103, 104]. Another way to incorporate the dynamic properties of speech is to use LDA features [105, 106, 107]. These are obtained by stacking 15 adjacent MFCC features – that is, combining the current MFCC vector with 7 preceding and 7 succeeding MFCC vectors, as shown in Figure 5.6 – and then performing a linear

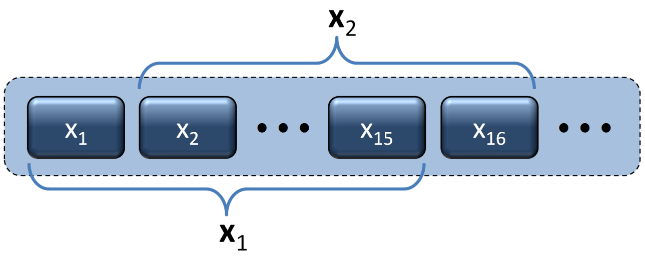


Figure 5.6: *Stacking Adjacent MFCCs.*

discriminant analysis (LDA) [108] on these super-features by first calculating an LDA matrix on a training corpus; and then multiplying the extracted and stacked MFCC vectors by this matrix [105, 106, 107]. For completeness, it should be mentioned that the rows of the LDA matrix are actually eigenvectors which are found as the solution of a generalized eigenvalue problem [108, 107]. Hence, truncating the LDA matrix to the largest eigenvectors (i.e. the ones which correspond to the largest eigenvalues) reduces the dimension without sacrificing too much on discriminability. The ASR system used in this thesis performs cepstral mean and variance normalization [109, 110, 111] before calculating LDA features. This decreases the variability between different speakers and microphones and, hence, increases robustness. As the LDA matrix is truncated to 42 eigenvectors, the final features are 42-dimensional.

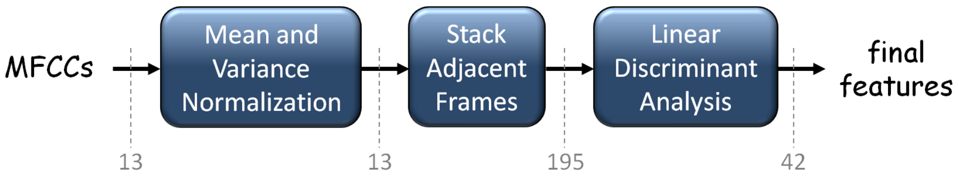
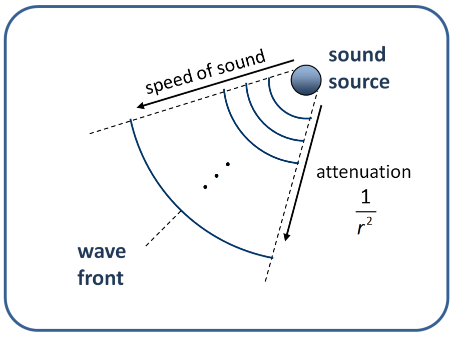


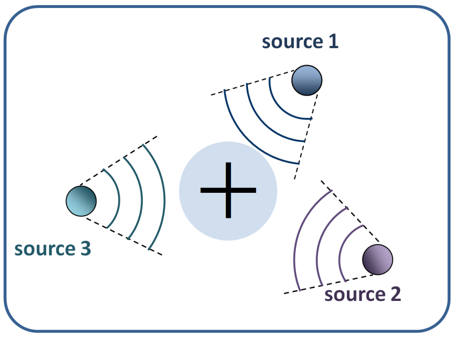
Figure 5.7: *Dynamic Features through Linear Discriminant Analysis.*

5.2 The Effect of Noise to Speech Features

This section studies the effect that noise has on clean speech features, starting with the relationship of speech and noise at the wave level. At this level, sound is a disturbance of air molecules propagating as a wave front of compressed air followed by decompressed air. Waves from different sources add up at each point of the space (superposition principle), as portrayed in Figure 5.8.



(a) *Wave Propagation.*



(b) *Superposition Principle.*

Figure 5.8: *Sound Propagation*

This implies that, whenever a microphone is used to pick up the sound from a particular source, the resulting audio signal will also contain sound from other sources, such as environmental noise or reverberation (due to reflections off the walls). This is formally captured by modeling the received signal $y(t)$ as:

$$y(t) = h(t) * x(t) + n(t) \tag{5.1}$$

where $x(t)$ denotes the original speech signal, $n(t)$ denotes the noise signal and $h(t)$ denotes the room impulse response, which accounts for reflections off the walls. The "$*$" symbol denotes the convolution operator. To emphasize the fact that the propagation of the audio signal from the speaker to the microphone can be described as a communication channel, the function $h(t)$ is also referred to as "channel".

5.2.1 The Effect of Noise in the log-Mel Domain

We would now like to bring the relationship from (5.1) to the log-Mel domain, i.e. the domain of logarithmic frequency / logarithmic intensity spectra. This is achieved by translating (5.1) through all of the individual feature extraction steps, starting with the translation from the time (5.1) to the frequency domain:

$$Y(\omega) = H(\omega) \cdot X(\omega) + N(\omega) \tag{5.2}$$

where $Y(\omega)$, $H(\omega)$, $X(\omega)$ and $N(\omega)$ denote the short-time Fourier transforms of $y(t)$, $h(t)$, $x(t)$ and $n(t)$. Moving to the power spectral domain, by taking the magnitude square on both sides, gives:

$$\begin{aligned}
|Y(\omega)|^2 &= (H(\omega)X(\omega) + N(\omega)) \cdot (H(\omega)X(\omega) + N(\omega))^* \\
&= H(\omega)H(\omega)^* X(\omega)X(\omega)^* + N(\omega)N(\omega)^* + H(\omega)X(\omega)N(\omega)^* + N(\omega)H(\omega)^* X(\omega)^* \\
&= |H(\omega)|^2 |X(\omega)|^2 + |N(\omega)|^2 + \underbrace{H(\omega)X(\omega)N(\omega)^* + (H(\omega)X(\omega)N(\omega)^*)^*}_{\triangleq \mathrm{err}(\omega)}
\end{aligned}$$

where $|Y(\omega)| = \sqrt{Y(\omega)Y(\omega)^*}$ denotes the spectral magnitude. The error term err(ω) consists of a complex value plus the conjugate of the same complex value. Hence, it is 2 times the real part of $H(\omega)X(\omega)N(\omega)^*$, which can be further simplified through use of the cosine:

$$\text{err}(\omega) = 2\mathscr{R}\left\{H(\omega)X(\omega)N(\omega)^*\right\} = 2cos(\theta_\omega)|H(\omega)||X(\omega)||N(\omega)|$$

where θ_ω denotes the phase or "argument" of $H(\omega)X(\omega)N(\omega)^*$. Given this equation, we can conclude that the relationship between speech $x(t)$, channel $h(t)$ and noise $n(t)$ in the power spectral domain is:

$$|Y(\omega)|^2 = |H(\omega)|^2|X(\omega)|^2 + |N(\omega)|^2 + 2cos(\theta_\omega)|H(\omega)||X(\omega)||N(\omega)|. \tag{5.3}$$

The angle θ_ω corresponds to the relative phase between speech $|H(\omega)X(\omega)|$ and noise $|N(\omega)|$. Depending on the value of θ_ω, speech is either amplified or attenuated as indicated below:

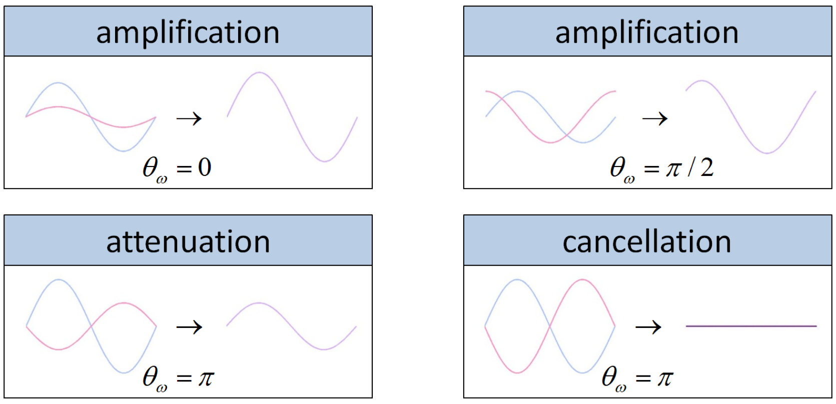


Figure 5.9: *Effect of the Relative Phase. The blue and red curves on the left-hand side of the arrow show speech and noise signals, respectively. The violet curves on the right-hand side show the resulting noisy speech signals.*

Now discretizing the signal and performing a windowed Fourier transforms, as it is done in speech feature extraction, we obtain the following approximation of (5.3):

$$|Y_k|^2 = |H_k|^2|X_k|^2 + |N_k|^2 + 2cos(\theta_k)|H_k||X_k||N_k|, \tag{5.4}$$

where the $k = 1, ..., d_s$ denote discrete frequencies and where d_s is the spectral dimension. The use of a windowed Fourier transform, however, truncates the room impulse response $h(t)$ to the window length. This essentially cuts the reverberation tails and thereby makes (5.4) unsuitable for reverberant environments [112, 113, 114]. We nevertheless proceed by denoting the Mel spectrum of clean speech $x(t)$ by

$$\tilde{X}_k = \underbrace{\begin{bmatrix} W_{k,1} & \cdots & W_{k,d_s} \end{bmatrix}}_{\triangleq W_k} \cdot \underbrace{\begin{bmatrix} |X_1|^2 & \ldots & |X_{d_s}|^2 \end{bmatrix}^T}_{\triangleq |X|^2},$$

$k = 1,\dots,d_m$, where W_k denotes the k-th row of the Mel Matrix W, i.e. the k-th Mel filter. Then the relationship in the power spectral domain can be translated to the Mel domain by multiplying both sides of (5.4) by the Mel matrix W. This gives:

$$\tilde{Y}_k = \tilde{H}_k \tilde{X}_k + \tilde{N}_k + 2\alpha_k \sqrt{\tilde{H}_k \tilde{X}_k \tilde{N}_k} \tag{5.5}$$

with $\tilde{Y}_k = W_k \cdot |Y|^2$, $\tilde{H}_k = W_k \cdot |H|^2$, $\tilde{X}_k = W_k \cdot |X|^2$, $\tilde{N}_k = W_k \cdot |N|^2$ and with α_k denoting the so-called phase factor [76],

$$\alpha_k = \frac{\sum_{i=1}^{d_s} W_{k,i} cos(\theta_i)|H_i||X_i||N_i|}{\sqrt{\tilde{H}_k \tilde{X}_k \tilde{N}_k}}, \tag{5.6}$$

which assumes values in the range $[-1,1]$. It is interesting to note that α_k is introduced simply because it allows us to write (5.5) in terms of the variables $\tilde{H}_k$, $\tilde{X}_k$ and $\tilde{N}_k$. This becomes clear by verifying that the nominator of (5.6) is just the multiplication of the k-th row of the Mel matrix with the vector of cross terms:

$$\begin{bmatrix} W_{k,1} & \cdots & W_{k,d_s} \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_1)|H_1||X_1||N_1| \\ \dots \\ \cos(\theta_{d_s})|H_{d_s}||X_{d_s}||N_{d_s}| \end{bmatrix}.$$

So, the nominator of (5.6) is the actual cross term in the Mel domain and the denominator of (5.6) is only introduced in order to cancel the $\sqrt{\tilde{H}_k \tilde{X}_k \tilde{N}_k}$ in (5.5). Aside from this, it should be mentioned that (5.5) is actually an approximation unless the frequency response $|H_k|$ of the channel is 1 for all k. That is because $\tilde{H} \odot \tilde{X} = \left(W|H|^2\right) \odot \left(W|X|^2\right) \neq W\left(|H|^2 \odot |X|^2\right)$, where $\odot$ operator denotes the component-wise multiplication of the vectors. Further defining $y_k = \log\left(\tilde{Y}_k\right)$, $x_k = \log\left(\tilde{X}_k\right)$, $h_k = \log\left(\tilde{H}_k\right)$ and $n_k = \log\left(\tilde{N}_k\right)$, equation (5.5) can be written in dependency of the log-Mel spectra y, x, h and n:

$$\begin{aligned} e^{y_k} &= e^{x_k} e^{h_k} + e^{n_k} + 2\alpha_k \sqrt{e^{x_k} e^{h_k} e^{n_k}} \\ &= e^{x_k + h_k}\left(1 + e^{n_k - x_k - h_k} + 2\alpha_k \sqrt{e^{n_k - x_k - h_k}}\right) \end{aligned}$$

Taking the logarithm on both sides finally gives the following relationship in the log-Mel domain [115, 76]:

$$y_k = \underbrace{x_k + h_k + log\left(1 + e^{n_k - x_k - h_k} + 2\alpha_k \sqrt{e^{n_k - x_k - h_k}}\right)}_{\triangleq f(x_k, h_k, n_k, \alpha_k)}. \tag{5.7}$$

This is the so-called interaction function [116]. It allows for the calculation of noisy speech y, given clean speech s, noise n and channel h (all in the log-Mel domain) along with a vector of phase factors $\alpha = \left[\alpha_1, \dots, \alpha_{d_m}\right]$. Unfortunately, the α_k are difficult to estimate in practice as they are volatile and almost uncorrelated in different frequency bins. Hence, Deng et al. [115, 76] proposed to keep the phase factors as random terms in order to explicitly account for their uncertainty. This was done under the assumption that the α_k follow a Gaussian distribution. Deviating from that approach, this work takes a Monte Carlo approach which averages the interaction function from (5.7) with respect to the true distribution of the α_k [14]. This is done in Section 5.2.3, right after the distribution of the phase factors has been determined.

5.2.2 Distribution of the Phase Factors

In Deng et al. [76], it was assumed that the phase factors follow a zero-mean Gaussian distribution. This assumption was based on the reasoning that the relative phases θ_k are uniformly distributed on $[-\pi, \pi]$ and that, consequently, the phase factor of each Mel filter is a weighted sum of cosines of the uniform distribution. This means, in higher Mel frequencies bins many of the $\cos(\theta_k)$ are added, as can be seen from Figure 5.4. The authors of [76] deduced from this that the central limit theorem can be applied and that, consequently, it would be reasonable to approximate the distribution of the phase factors by a Gaussian. Unfortunately, this is not entirely true for lower Mel frequency bins, especially not for the lowermost 3-4 bins where the distribution actually becomes bimodal [14]. Hence, this thesis uses the empirical phase factor distribution (see Figure 5.10). It is obtained by (1) adding known speech and noise signals and then (2) computing the phase factors of the corresponding Mel spectra according to:

$$\alpha_k = \frac{\tilde{Y}_k - \tilde{X}_k - \tilde{N}_k}{2\sqrt{\tilde{X}_k \tilde{N}_k}}. \tag{5.8}$$

In this equation, which is essentially (5.5) solved for α_k, $\tilde{Y}$ denotes the Mel spectrum which has been calculated from the noisy speech signal; $\tilde{X}$ and $\tilde{N}$ denote the Mel spectra which have been calculated from the individual clean speech and noise signals. Figure 5.10 shows the resulting phase factor distribution at the example of the multi-channel Wall Street Journal audio visual (MC-WSJ-AV) corpus [117] with added factory noise from the NOISEX-92 database [118].

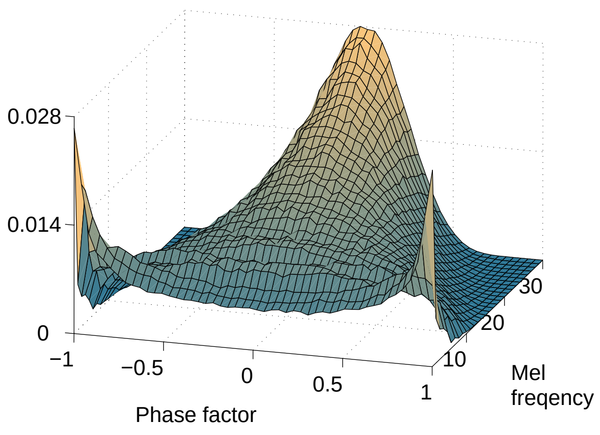


Figure 5.10: *Empirical distribution of the phase factors.*

As an alternative to experimentally determining the α_k on a training corpus, the phase factors can also be obtained through simulation [14]. For this, it is assumed that the relative phase is uniformly and independently distributed in the power spectral bins. Then, the vector θ of relative phases can be simulated by (1) drawing samples $\theta_k^{(j)}$ from a uniform distribution on $[0, 2\pi]$ for $k = 1, \dots, d_s$, $j = 1, \dots, N$, and then (2) converting these samples by first taking the cosine of each $\theta_k^{(j)}$ and then multiplying the $\cos\left(\theta^{(j)}\right)$ by the Mel filterbank matrix W:

$$\alpha_i^{(j)} \approx \sum_k W_{i,k} \cos\left(\theta_k^{(j)}\right). \tag{5.9}$$

This is an approximation, as (5.9) implicitly assumes that the spectra of speech and noise are constant over the spectral bins covered by each of the individual Mel filters. Deviating from this theoretical result, better results were obtained in practice by first multiplying the $\theta^{(j)}$ with W and then taking the cosine of the components, respectively:

$$\alpha_i^{(j)} \approx \cos\left(\sum_k W_{i,k} \theta_k^{(j)}\right). \tag{5.10}$$

Figure 5.11 shows the variance and kurtosis[2] of the resulting phase factor distribution in comparison to the experimentally determined one, where the latter was obtained from the MC-WSJ-AV corpus with added tank (leopard) and factory noise.

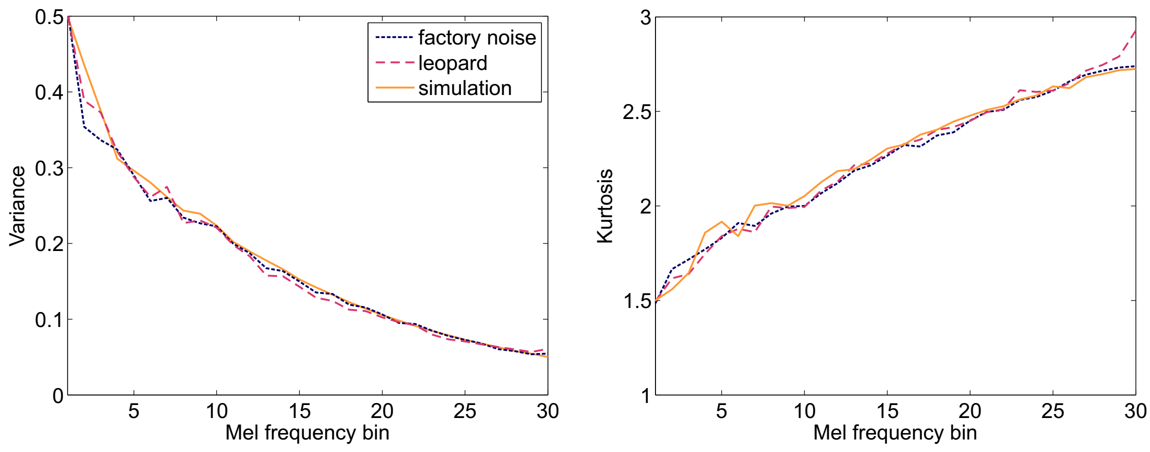


Figure 5.11: *Variance and kurtosis of the phase factors distribution, for the simulation-based approach (solid line)* (5.10) *as well as for the experimental approach* (5.8) *with added tank (dashed) and factory noise (dotted).*

Interestingly, Figure 5.11 also reveals that the empirical distribution does not strongly vary for different noise types. In general, the variance seems to be decreasing with the Mel frequency bins whereas the kurtosis seems to be increasing. The phase factors are bimodal in the lowest four bins, with a kurtosis smaller than 1.8 (that is the kurtosis of the uniform distribu-

[2] Note that we compute the kurtosis as the fourth central moment divided by the variance square.

tion). The phase factors are approximately Gaussian in the highest five bins where the kurtosis approaches 3.0. These experimental results have been confirmed in more recent work [119], through an analytic approximation of the moments of the phase factor distribution.

5.2.3 A Phase-Averaged Model for the Effect of Noise

Now that the phase factor distributions is known, the noisy speech spectrum in the log-Mel domain can be modeled as a conditional random variable $Y_k|x_k, h_k, n_k$ where the randomness is due to uncertainty about which values the phase factors assume (in particular note that Y_k here denotes a random variable and not a Fourier coefficient). Using this model, the minimum mean square error (MMSE) estimate of noisy speech is the expectation $\mathscr{E}_{p_{Y_k|x_k,h_k,n_k}(y_k)}\{y_k\}$ of y_k under the distribution $p_{Y_k|x_k,h_k,n_k}$. In order to calculate it, let us rewrite equation (5.7) as

$$y_k = x_k + h_k + \tilde{f}_{\alpha_k}(z_k) \tag{5.11}$$

where the function $\tilde{f}_{\alpha_k}(z_k)$ is defined as $\log\left(1 + e^{z_k} + 2\alpha_k\sqrt{e^{z_k}}\right)$ with $z_k \triangleq (n_k - x_k - h_k)$. Then, the expectation $\mathscr{E}_{p_{Y_k|x_k,h_k,n_k}(y_k)}\{y_k\}$ can be calculated by averaging $\tilde{f}_{\alpha_k}(z_k)$ with respect to α_k:

$$\begin{aligned}\mathscr{E}_{p_{Y_k|x_k,h_k,n_k}(y_k)}\{y_k\} &= x_k + h_k + \mathscr{E}_{p_{A_k}(\alpha_k)}\{\tilde{f}_{\alpha_k}(n_k - x_k - h_k)\} \\ &= x_k + h_k + \int \tilde{f}_{\alpha_k}(n_k - x_k - h_k) p_{A_k}(\alpha_k)\, d\alpha_k \end{aligned}\tag{5.12}$$

where $p_{A_k}(\alpha_k)$ denotes the phase factor distribution. Unfortunately, the integral in (5.12) cannot be calculated analytically. But it can be approximated by Monte Carlo integration [14] or by means of a Taylor series expansion [119]. The former has been proposed in the framework of this thesis [14]; and it approximates the expectation with a set $\mathscr{M}$ of phase factor samples $\alpha^{(j)} = \begin{bmatrix}\alpha_1^{(j)} & \cdots & \alpha_{d_m}^{(j)}\end{bmatrix}^T$:

$$\mathscr{E}_{p_{Y_k|x_k,h_k,n_k}(y_k)}\{y_k\} \approx x_k + h_k + \frac{1}{|\mathscr{M}|}\sum_{\alpha^{(j)}\in\mathscr{M}} \tilde{f}(z_k, \alpha_k^{(j)}). \tag{5.13}$$

This gives a phase-averaged model for the effect of noise in the log-Mel domain [14]. In particular, the expectation $\mathscr{E}_{p_{A_k}(\alpha_k)}\{\tilde{f}_{\alpha_k}(z_k)\} \approx \frac{1}{|\mathscr{M}|}\sum_{\alpha^{(j)}\in\mathscr{M}} \tilde{f}(z_k, \alpha_k^{(j)})$ in (5.13) is dependent on

```
Phase Averaged Interaction Function: $y_k = \text{interact}(x_k, h_k, n_k)$
if $z_k$ < -50dB
    return $y_k = x_k + h_k$;
if $z_k$ > 50dB
    return $y_k = n_k$;
return $y_k = x_k + h_k + \mathscr{E}_{p_{A_k}(\alpha_k)}\{\tilde{f}_{\alpha_k}(z_k)\}$;
```

Algorithm 5.1: Evaluating the Phase Averaged Interaction Function

$z_k = (n_k - x_k - h_k)$ only. For small z_k, it is approximately 0. For large z_k it is approximately $\mathrm{id}(z_k)$, as shown in Figure 5.12. Hence, (5.13) can be calculated by tabulating $\mathscr{E}_{p_{A_k}(\alpha_k)}\{\tilde{f}_{\alpha_k}(z_k)\}$ in a relatively small area (e.g. [−50,50] dB) and then calculating the phase-averaged interaction function according to Algorithm 5.1.

5.2.4 The Zero-Phase Factor Model

Deviating from the above, most authors [120, 121, 45, 122, 123, 124, 125, 126, 127] average the relative phase in the power spectral domain. This has its origin in power spectral subtraction [120, 128], and it results in the following relationship between clean and noisy speech:

$$y_k = x_k + h_k + \underbrace{\log\left(1 + e^{n_k - x_k - h_k}\right)}_{=\tilde{f}_0(z_k)}. \tag{5.14}$$

Equation (5.14) was originally derived by Van Compernolle [120], in 1987, and it was later used in a large variety of approaches, starting with Acero's codeword dependent cepstral normalization [11] and Moreno's vector Taylor series approach [45]. Due to the fact that (5.14) is obtained by setting α_k in (5.11) to zero, it is here called the *zero-phase factor model*. Figure 5.12 shows

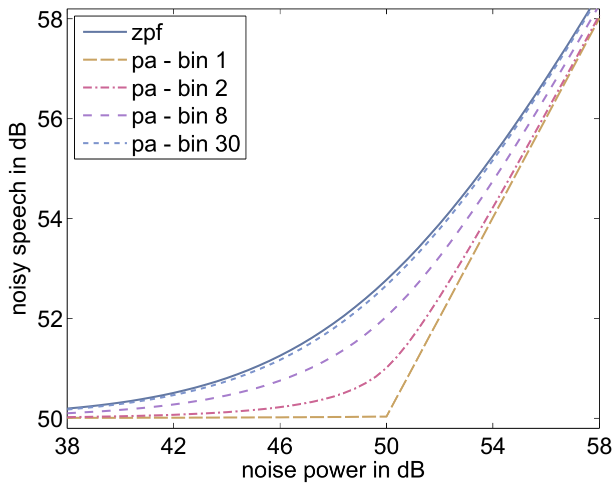


Figure 5.12: *Noisy speech, y, as a function of the noise power, n, in the log-Mel domain. The plot shows the interaction function of the zero-phase factor (zpf) model* (5.14) *in comparison to the phase-averaged (pa) model* (5.13)*, which is dependent on the frequency bin. The clean speech power has been fixed to 50dB.*

this model in comparison to the minimum mean square error (MMSE) solution from the previous section. Obviously, (5.14) provides a good approximation in higher Mel frequency bins, not however in lower bins where the MMSE solution is closer to the log-max approximation [129, 130, 131, 132],

$$y_k = \max(x_k + h_k, n_k), \tag{5.15}$$

which is used in missing feature reconstruction (see Chapter 7).

5.2.5 Performance Comparison

This section compares the performance of the phase-averaged and zero-phase factor models. That is achieved by adding known speech and noise signals in order to simulate noisy speech. After individually extracting the log-Mel spectrograms of clean speech x, noise n and noisy speech y, the models from Sections 5.2.3 and 5.2.4 are used to predict the noisy speech spectrogram $\hat{y}$ from x and n. The quality of prediction is evaluated as the mean square error (MSE) between y and $\hat{y}$:

$$MSE\left\{y-\hat{y}\right\} = \sum_{t=1}^{\tau}\sum_{k=1}^{K}\left(y_{t,k}-\hat{y}_{t,k}\right)^2 .$$

Figure 5.13 shows the result for the zero-phase factor (zpf) model in comparison to the phase averaged model (pa) from Section 5.2.3. In the latter case, the error is shown both for the closest table entry (without interpolation) and for linear interpolation (+li) between table entries. The error of the phase-averaged model obviously drops with the table size and it saturates at a value of around 300. With a table size of 250, the phase-averaged model outperforms the zero-phase factor model with a relative improvement of up to 7% (see Table 5.1). In the lowest frequency (62.5 Hz), the relative improvement is 16% and hence twice as large as the average.

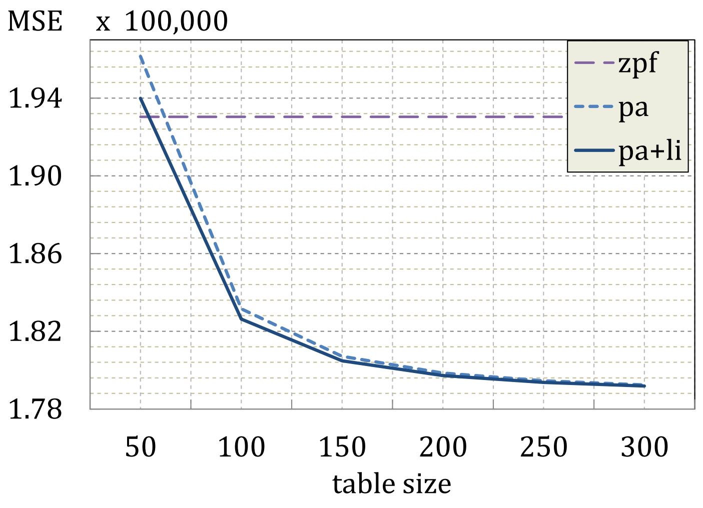


Figure 5.13: *Error for predicting noisy speech with the zero-phase factor and phase-averaged models. The results are shown in dependency of the table size, with and without interpolation.*

interpolation	table size									
	50	100	150	200	250	300	350	400	450	500
none	-0.49	5.39	6.51	6.90	7.08	7.18	7.24	7.27	7.30	7.32
linear	-1.61	5.12	6.38	6.83	7.04	7.15	7.21	7.26	7.29	7.31

Table 5.1: *Relative improvement of the phase-averaged model over the zero-phase factor model. Note that the improvements are measured as the relative reduction in MSE when predicting noisy speech spectrograms from clean speech and noise spectrograms.*

5.3 Reverting the Effect of Noise

The previous section was concerned with developing models for the effect that noise has on clean speech features. This is useful for simulating noisy speech. In the context of speech feature enhancement, however, it might be of interest to also perform the opposite, i.e. revert the effect of noise in order to get back to clean speech features. That is done in this section by deriving the inverses of the models from Section 5.2.

5.3.1 The Inverse Interaction Function

Starting with the inversion of the interaction function from (5.7), let us firstly rewrite the relationship (5.5) between noisy speech, clean speech and noise in the Mel domain by using the method of "completion of the squares":

$$\begin{aligned}\tilde{Y}_k &= \tilde{H}_k\tilde{X}_k+\tilde{N}_k+2\alpha_k\sqrt{\tilde{H}_k\tilde{X}_k\tilde{N}_k}\\ &= \left(\sqrt{\tilde{H}_k\tilde{X}_k}+\alpha_k\sqrt{\tilde{N}_k}\right)^2+(1-\alpha_k^2)\tilde{N}_k.\end{aligned}$$

Then subtracting $(1-\alpha_k^2)\tilde{N}_k$ on both sides and subsequently taking the square root gives:

$$\sqrt{\tilde{H}_k\tilde{X}_k}+\alpha_k\sqrt{\tilde{N}_k} = \pm\sqrt{\tilde{Y}_k-(1-\alpha_k^2)\tilde{N}_k}.$$

This can be solved for clean speech $\tilde{X}_k$ by first subtracting $\alpha_k\sqrt{\tilde{N}_k}$, again squaring both sides and then dividing by $\tilde{H}_k$. Doing so yields

$$\tilde{X}_k=\frac{\left(\pm\sqrt{\tilde{Y}_k+(\alpha_k^2-1)\tilde{N}_k}-\alpha_k\sqrt{\tilde{N}_k}\right)^2}{\tilde{H}_k},$$

which can be translated to the log-Mel domain by replacing $\tilde{X}_k$, $\tilde{Y}_k$, $\tilde{N}_k$ and $\tilde{H}_k$ by e^{x_k}, e^{y_k}, e^{n_k} and e^{h_k}, respectively:

$$\begin{aligned}e^{x_k} &= \left(\pm\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}-\alpha_k\sqrt{e^{n_k}}\right)^2 e^{-h_k}\\ &= e^{y_k-h_k}\left(\sqrt{1+(\alpha_k^2-1)e^{n_k-y_k}}\pm\alpha_k\sqrt{e^{n_k-y_k}}\right)^2.\end{aligned}$$

Taking the logarithm on both sides finally gives the inverse of the interaction function from (5.7):

$$x_k^{\pm}=\underbrace{y_k-h_k+\log\left(\sqrt{1+(\alpha_k^2-1)e^{n_k-y_k}}\pm\alpha_k\sqrt{e^{n_k-y_k}}\right)^2}_{\triangleq g_k^{\pm}(y_k,h_k,n_k,\alpha_k)}. \tag{5.16}$$

This solution was firstly given in equation (5) of [14]. But in that work (5.16) was rewritten by means of $u_k\triangleq 1+(\alpha_k^2-1)e^{n_k-y_k}$ and $v_k\triangleq\alpha_k\sqrt{e^{n_k-y_k}}$:

$$x_k^{\pm}=y_k-h_k+\log\left(\sqrt{u_k}\pm v_k\right)^2. \tag{5.17}$$

It is also important to note that there is a typing error in [14], which consists in the "±" appearing in front of the logarithm instead of in front of $\sqrt{v_k}$. This typo does not, however, concern the experiments in [14], as (5.17) has been implemented correctly. Having a closer look at (5.17), it becomes clear that the solutions x_k^+ and x_k^- exist only if $u_k \geq 0$ and, in case of x_k^-, if $(\sqrt{u_k} - v_k) \neq 0$. Otherwise y_k, h_k, n_k and α_k form a physically impossible configuration for which there exists no clean speech solution x_k (see [14, 133]).

5.3.2 The Inverse Phase-Averaged Model

In order to calculate the inverse of the phase-averaged model from Section 5.2.3, let us rewrite the log term in (5.16) as a function of $z_k = n_k - y_k$:

$$x_k^{\pm} = \underbrace{y_k - h_k + \underbrace{\log\left(\sqrt{1 + (\alpha_k^2 - 1)e^{n_k - y_k}} \pm \alpha_k \sqrt{e^{n_k - y_k}}\right)^2}_{\triangleq \tilde{g}_k^{\pm}(z_k, \alpha_k)}}_{= g_k^{\pm}(y_k, h_k, n_k, \alpha_k)}. \tag{5.18}$$

Then, the minimum mean square error (MMSE) clean speech estimate $\hat{x}_k^{\pm}$ is obtained by calculating the expectation $\mathcal{E}_{p_{A_k}(\alpha_k)}\left\{g_k^{\pm}(y_k, h_k, n_k, \alpha_k)\right\} = y_k - h_k + \mathcal{E}_{p_{A_k}(\alpha_k)}\left\{\tilde{g}_k^{\pm}(z_k, \alpha_k)\right\}$ of (5.18) in analogy to (5.12) and (5.13). During Monte Carlo integration, however, only those samples α_k are to be considered which result in valid solutions of (5.17). This means, a different subset $\mathcal{M}_k^{\pm} \subseteq \mathcal{M}$ of samples needs to be considered for each considered Mel frequency bin k. Consequently, we get the following approximation [14]:

$$\hat{x}_k^{\pm} = y_k - h_k + \frac{1}{\left|\mathcal{M}_k^+\right| + \left|\mathcal{M}_k^-\right|} \left(\sum_{\alpha_k \in \mathcal{M}_k^+} \tilde{g}_k^+(z_k, \alpha_k) \; + \sum_{\alpha_k \in \mathcal{M}_k^-} \tilde{g}_k^-(z_k, \alpha_k) \right) \tag{5.19}$$

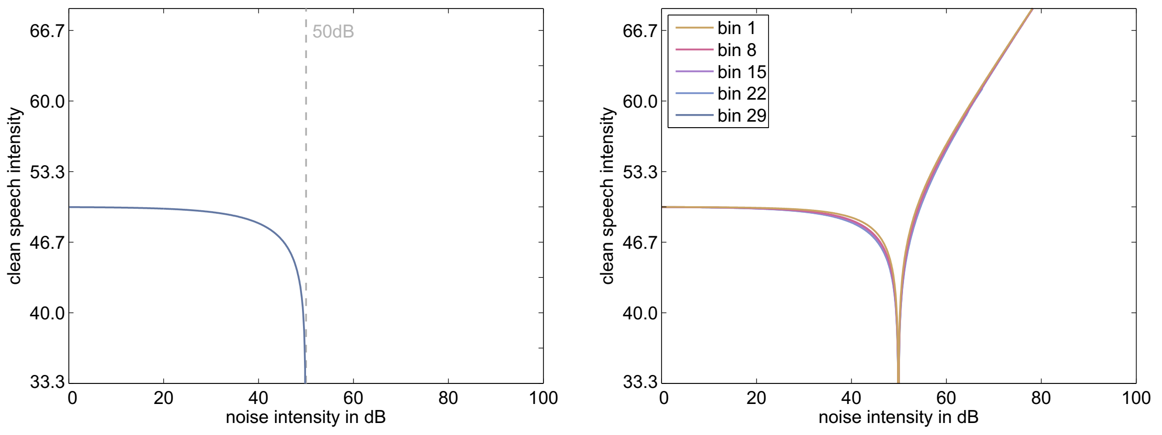
As the expectation $\mathcal{E}_{p_{A_k}(\alpha_k)}\left\{\tilde{g}_k^{\pm}(z_k, \alpha_k)\right\}$ calculated in the two sums of (5.19) is dependent on z_k only, it may be tabulated in analogy to section 5.2.3.

5.3.3 The Inverse Zero-Phase Factor Model

The inverse of the zero-phase factor model from Section 5.2.4 is obtained by either solving (5.14) for x_k or by setting the phase factor α_k in (5.16) to 0. In both cases, the result is:

$$\hat{x}_k = \underbrace{y_k - h_k + \log\left(1 - e^{n_k - y_k}\right)}_{= g_k^{\pm}(y_k, h_k, n_k, 0)}. \tag{5.20}$$

But this solution exists only if $n_k < y_k$. Hence, the zero-phase factor model does not allow noise hypotheses to exceed the observed noisy speech spectrum [133, 134]. This issue will be discussed in more detail in the following, at the hand of Figure 5.14-(a) which shows a plot of (5.20) in dependency of the noise intensity n_k.



(a) inverse of the zero-phase factor model *(b) inverse of the phase averaged model*

Figure 5.14: *Inverse interaction functions at a constant noisy speech power y_k of 50dB.*

With the noise intensity approaching noisy speech y_k (from below), the clean speech estimate $\hat{x}_k$ approaches minus infinity as the term in the logarithm of (5.20) tends towards zero. For $n_k > y_k$, we have a negative value in the logarithm. So, $\hat{x}_k$ does not exist. This is due to the fact that the zero-phase factor model ignores the relative phase between speech and noise [133, 134] and therewith ignores the fact that noise can attenuate or cancel certain portions of the speech signal (see Figure 5.9). Figure 5.14-(b) shows the phase-averaged model for comparison. Although the estimated clean speech intensity tends towards minus infinity with n_k approaching y_k, $\hat{x}_k$ again rises after this point. This can be explained by the fact that the intensity of the noise can only exceed that of noisy speech when attenuation occurs. But this requires (1) that the clean speech and noise intensities are comparable, and (2) that the phase factor is closer to minus one. The first point is responsible for the rise in clean speech intensity. The second point means with n_k increasing there will be fewer and fewer phase factors α_k, which give a valid solution according to (5.17). The statistical effect of the latter is compensated in the upcoming section.

5.3.4 Jacobian of the Inverse Phase-Averaged Model

Some speech feature enhancement approaches [135, 127, 133, 136, 134] necessitate calculating the Jacobian of the inverse interaction function with respect to y_k. In order to do so, let us rewrite (5.16) by pulling y_k back into the square roots:

$$g_k^{\pm}(y_k, h_k, n_k, \alpha_k) = -h_k + \log\left(\sqrt{e^{y_k} + (\alpha_k^2 - 1)e^{n_k}} \pm \alpha_k \sqrt{e^{n_k}}\right)^2.$$

Then the derivative of g_k^+ with respect to y_k can be calculated by first using the chain rule and then extracting an e^{y_k} from each additive term in the denominator:

$$\begin{aligned}
\frac{dg_k^+(y_k,h_k,n_k,\alpha_k)}{dy_k} &= \frac{2\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}+\alpha_k\sqrt{e^{n_k}}}{\left(\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}+\alpha_k\sqrt{e^{n_k}}\right)^2}\cdot\left(\frac{e^{y_k}}{2\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}}\right)\\
&= \frac{e^{y_k}}{\left(\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}+\alpha_k\sqrt{e^{n_k}}\right)\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}}\\
&= \frac{e^{y_k}}{e^{y_k}+(\alpha_k^2-1)e^{n_k}+\alpha_k\sqrt{e^{n_k}}\sqrt{e^{y_k}+(\alpha_k^2-1)e^{n_k}}}\\
&= \frac{1}{1+(\alpha_k^2-1)e^{n_k-y_k}+\alpha_k\sqrt{e^{n_k-y_k}}\sqrt{1+(\alpha_k^2-1)e^{n_k-y_k}}}.
\end{aligned}$$

Now calculating the derivative of g_k^- in analogy to the above, we find that $dg_k^\pm/dy_k$ can be expressed in a single equation:

$$\frac{dg_k^\pm(y_k,h_k,n_k,\alpha_k)}{dy_k}=\gamma_k^\pm(z_k,\alpha_k)\triangleq\frac{1}{\underbrace{1+(\alpha_k^2-1)e^{z_k}}_{u_k}\pm\underbrace{\alpha_k\sqrt{e^{z_k}}}_{v_k}\underbrace{\sqrt{1+(\alpha_k^2-1)e^{z_k}}}_{\sqrt{u_k}}} \tag{5.21}$$

with $z_k=n_k-y_k$ and with u_k and v_k being defined as in Section 5.3.2. The derivatives with respect to other variables, y_l, $l\neq k$, are 0. Hence, the Jacobian of $g=\begin{bmatrix}g_1 & \cdots & g_{d_m}\end{bmatrix}^T$ with respect to $y=\begin{bmatrix}y_1 & \cdots & y_{d_m}\end{bmatrix}^T$ is a diagonal matrix:

$$J_\alpha(y,h,n)\triangleq\begin{bmatrix}\gamma_1^\pm(n_1-y_1,\alpha_1) & 0 & \cdots & 0\\ 0 & \gamma_2^\pm(n_2-y_2,\alpha_2) & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & \gamma_{d_m}^\pm(n_{d_m}-y_{d_m},\alpha_{d_m})\end{bmatrix}. \tag{5.22}$$

The Jacobian $J(y,h,n)$ of the inverse phase-averaged model (5.19) is obtained by calculating the expectation of $J_\alpha(y,h,n)$ with respect α:

$$J(y,h,n)=\operatorname{diag}\left(\left[\mathscr{E}_{p_{A_1}(\alpha_1)}\{\gamma_1^\pm(n_1-y_1,\alpha_1)\}\ \cdots\ \mathscr{E}_{p_{A_{d_m}}(\alpha_{d_m})}\left\{\gamma_{d_m}^\pm(n_{d_m}-y_{d_m},\alpha_{d_m})\right\}\right]^T\right) \tag{5.23}$$

where diag(·) denotes the operator which translates a vector to a diagonal matrix. The individual $\mathscr{E}_{p_{A_k}(\alpha_k)}\{\gamma_k^\pm(z_k,\alpha_k)\}$ are approximated with Monte Carlo integration,

$$\mathscr{E}_{p_{A_k}(\alpha_k)}\{\gamma_k^\pm(z_k,\alpha_k)\}\approx\frac{1}{|\mathscr{M}_k^+|+|\mathscr{M}_k^-|}\left(\sum_{\alpha_k\in\mathscr{M}_k^+}\gamma_k^+(z_k,\alpha_k)+\sum_{\alpha_k\in\mathscr{M}_k^-}\gamma_k^-(z_k,\alpha_k)\right), \tag{5.24}$$

in analogy to Section 5.3.2. If the Jacobian is calculated in order to evaluate the likelihood of a noisy speech spectrum through use of the fundamental transformation law of probability,

$$p_Y(y) = p_X\left(\mathcal{E}_{p_A(\alpha)}\left\{g^{\pm}(y,h,n,\alpha)\right\}\right)\left|\det J(y,h,n)\right|,$$

(as in [135, 127, 133, 136, 134]) then the Jacobian needs to be further multiplied by the likelihood of obtaining a valid clean speech solution (see Section 5.3.2):

$$\beta_k(z_k) \triangleq \frac{|\mathcal{M}_k^+| + |\mathcal{M}_k^-|}{2|\mathcal{M}|} \tag{5.25}$$

It is interesting to note that this multiplication by $\beta_k(z_k)$ is equivalent to setting $\gamma_k^{\pm}(z_k,\alpha_k) = 0$ for invalid α_k. That is consistent with previous work in which the Jacobian of invalid arguments was set to zero [133, 134]. In order to efficiently implement the evaluation of $\det J(y,h,n)$, the $\mathcal{E}_{p_{A_k}(\alpha_k)}\left\{\gamma_k^{\pm}(z_k,\alpha_k)\right\}$ and $\beta_k(z_k)$ may be stored in a table. Subsequently, the appropriate values may be retrieved at runtime and $\det J(y,h,n)$ may be calculated according to:

$$\det J(y,h,n) = \prod_{k=1}^{d_m} \mathcal{E}_{p_{A_k}(\alpha_k)}\left\{\gamma_k^{\pm}(n_k - y_k,\alpha_k)\right\}\beta_k(n_k - y_k). \tag{5.26}$$

5.3.5 Jacobian of the Inverse Zero-Phase Factor Model

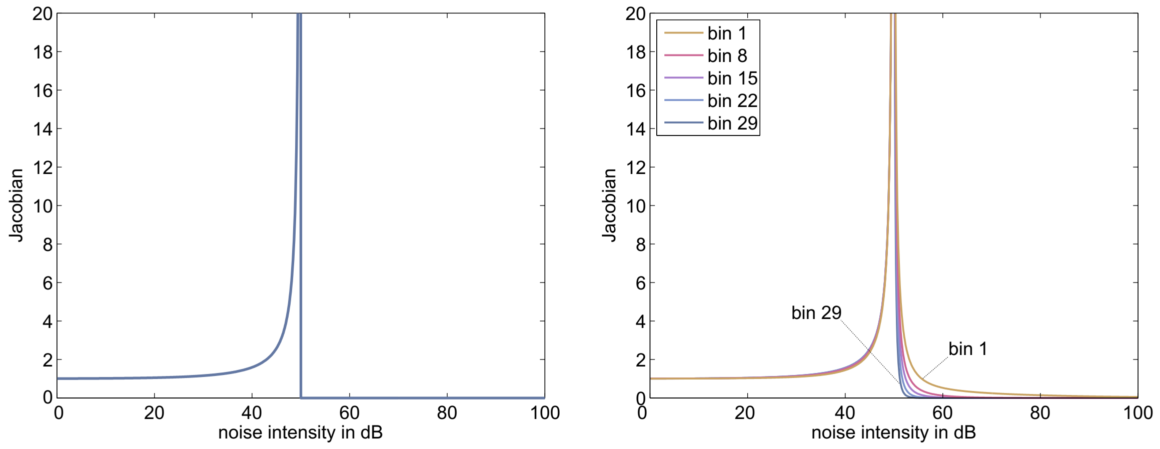
The Jacobian of the inverse zero-phase factor model is obtained by taking the derivative of (5.20) with respect to y. Alternatively, the α_k in (5.21) and (5.23) may be set to 0. In both cases we get:

$$\frac{dg_k^{\pm}(y_k,h_k,n_k,0)}{dy_k} = \frac{1}{1-e^{n_k-y_k}} = \gamma_k^{\pm}(n_k - y_k,0) \tag{5.27}$$

and $dg_k(y_k,h_k,n_k)/dy_l = 0$ for all $l \neq k$. Following [133], (5.27) is further set to zero if $n_k \geq y_k$. This is justifiable by the fact that the inverse zero-phase factor model does not allow the noise to exceed the noisy speech power (see [133, 137, 136] and Section 5.3.3). So, the total Jacobian may be written:

$$\frac{dg(y,h,n)}{dy} = \begin{bmatrix} \gamma_1^{\pm}(z_1,0)\cdot\mathbf{1}_{\{z_1<0\}} & 0 & \cdots & 0 \\ 0 & \gamma_2^{\pm}(z_2,0)\cdot\mathbf{1}_{\{z_2<0\}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \gamma_{d_m}^{\pm}(z_{d_m},0)\cdot\mathbf{1}_{\{z_{d_m}<0\}} \end{bmatrix} \tag{5.28}$$

where $z_k = n_k - y_k$, $k = 1,\ldots,d_m$, and where $\mathbf{1}$ denotes the indicator function. Figure 5.15-(a) shows a plot of $\gamma_k^{\pm}(z_k,0)\cdot\mathbf{1}_{\{z_k<0\}}$ in dependence of the noise intensity n_k. The curve obviously tends towards infinity with n_k approaching y_k. But it sharply drops to zero when n_k exceeds y_k. This can cause severe stability issues [133, 134] with speech feature approaches that are based on [135]. Hence, in the framework of this thesis [14] it has been proposed to replace the zero-phase factor model by the phase-averaged model. This alleviates the problem by allowing noise to exceed the noisy speech power. Figure 5.15-(b) shows the resultant Jacobian in dependence of the Mel frequency bin.



(a) zero-phase factor model *(b) phase averaged model*

Figure 5.15: *Jacobian of the inverse interaction functions for a noisy speech power of 50dB. The plot to the left shows $\gamma_k^{\pm}(z_k,0)\cdot\mathbf{1}_{\{z_k<0\}}$ in dependence of the noise power. The plot to the right shows $\mathscr{E}_{p_{A_k}(\alpha_k)}\left\{\gamma_k^{\pm}(z_k,\alpha_k)\right\}\cdot\beta_k(z_k)$ for different frequency bins k.*

5.4 Contributions

The following list again gives an overview of the individual contributions of this chapter:

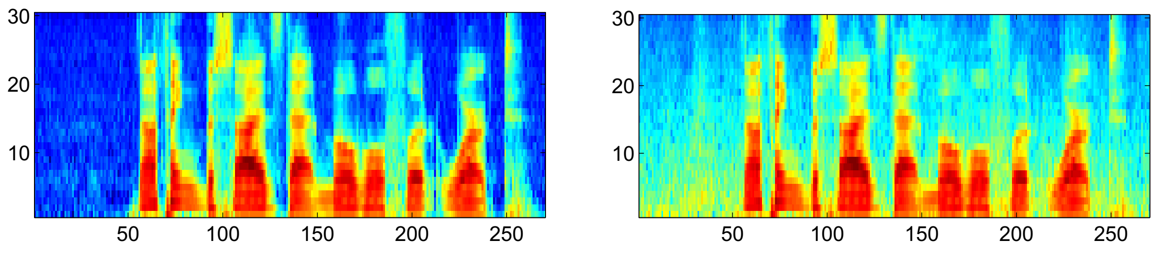
1. Derivation of a phase-averaged (minimum mean square error) model for the effect of noise to clean speech features [14] (Section 5.2).
2. Calculation of the inverse of the phase-averaged model plus its Jacobian [14] (Section 5.3).
3. Efficient implementations through use of tables.
4. A numerical comparison between the zero-phase factor and phase-averaged models (Section 5.2.5).

6 Speech Feature Enhancement

This is the first of two applications chapters which demonstrate the effectiveness of Bayesian estimation theory in the area of robust speech recognition. The chapter starts with a brief motivation for speech feature enhancement, including a discussion of possible alternatives. This is followed by a rough outline of the approach taken in this thesis plus the actual content sections.

6.1 Motivation

As shown in Section 5.2, environmental noise and reverberation change the distribution of extracted speech features. Figure 6.1 portrays this at the example of added factory noise. While high-energy regions appear to remain intact, spectral valleys are obviously "filled" with noise.



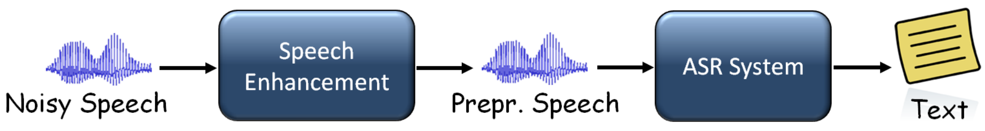
(a) clean speech (headset) *(b) with added factory noise*

Figure 6.1: *Log-Mel spectrograms of clean and noisy speech. The y-axis values represent Mel frequency bins (1-30). The x-axis represents time. Dark-blue areas indicate spectral valleys, i.e. regions of low energy whereas red areas indicate spectral peaks, i.e. regions of high energy. Green and yellow indicate medium-low and medium-high energies.*

This causes a mismatch between the clean speech features that the acoustic models were trained with and the noisy speech features that are encountered in the operational environment. This mismatch eventually leads to a degradation of the speech recognition performance. To give an example of just how bad it can get: the word error rate (WER) for connected digit recognition is below 1% for clean speech; it is 66% when car noise is added at a signal-to-noise ratio of 5 decibels (dB) [7]. In the first case, most (99%) of the digits are recognized correctly. In the second case, the automatic speech recognition (ASR) system only gets every 3-rd digit right. This discrepancy gives a strong motivation for techniques that reduce the mismatch between training and testing conditions [11]. Such techniques are discussed in the upcoming sections and they can be roughly grouped into the following categories:

6.1.1 Speech Enhancement

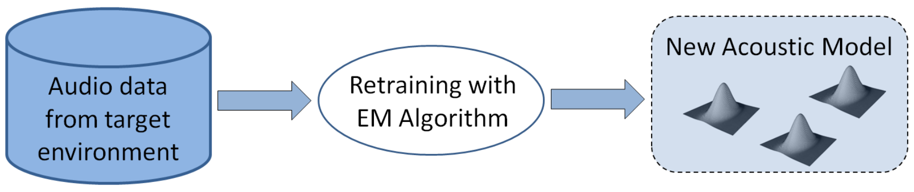
The most straightforward approach to reduce the mismatch between clean and noisy speech features consists in preprocessing the speech signal with standard speech enhancement techniques such as spectral subtraction [138, 139, 140, 8] or Wiener filtering [141, 9, 10]. Such approaches began to be of interest in the early 1980th, starting with Neben, McAulay and Weinstein [142] as well as Porter and Boll [143, 144]. More recent work employs a two-stage Wiener filter [145] or Ephraim and Malah's log-spectral amplitude estimator [146, 147]. The average noise spectrum (which needs to be known for these enhancement methods) can be estimated with minimum statistics [148, 149], histograms [150] or energy based voice activity detectors [147]. In general, the improvements obtained through speech enhancement can be substantial.



Nevertheless, there is a fundamental point of criticism with preprocessing-based approaches [11]: speech enhancement methods have primarily been developed with the aim of improving the subjective quality and intelligibility of speech rather than with the aim of achieving optimal speech recognition performance. This claim is substantiated through [151], which gives an in-depth analysis of the issues with spectral subtraction, including an experimental evaluation of the effect that these issues have on the word error rate.

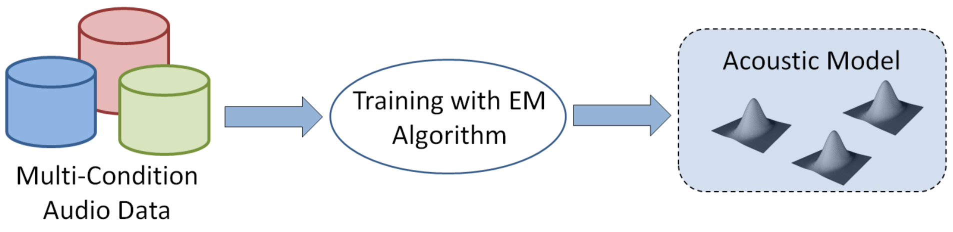
6.1.2 Retraining the Acoustic Model

Another approach to reduce the mismatch between training and testing conditions consists in retraining the acoustic models with data that has been collected "in the same acoustic environment as that of the anticipated operating (testing) conditions" [152]. This approach has been investigated in early studies on voice control interfaces for military cockpits (with helicopter cockpit noise) [153, 154] as discussed in more detail in a 1984 National Research Council report by Flanagan et al. [152]. The Flanagan report in particular suggests that, although retraining has the potential of significantly improving the speech recognition performance, it is impractical to retrain a system for each specific noise condition [152]. This especially holds for state-of-the-art large vocabulary systems, which require a large amount of training data for estimating the distribution parameters of a couple of thousand acoustic states.


Audio data from target environment
Retraining with EM Algorithm
New Acoustic Model

6.1.3 Multi-Condition Training

As an alternative to retraining the acoustic model for each particular speaking style or environmental condition, Lippmann et al. [155, 156] proposed to train the ASR system with a mix of speech data that has been collected under different conditions. This type of training makes the system less sensitive to changes of the environmental conditions. But it cannot achieve the same accuracy as retraining for a particular condition [11]. Another disadvantage of multi-condition training is that it tries to capture the large variability of noise in the acoustic model [11].

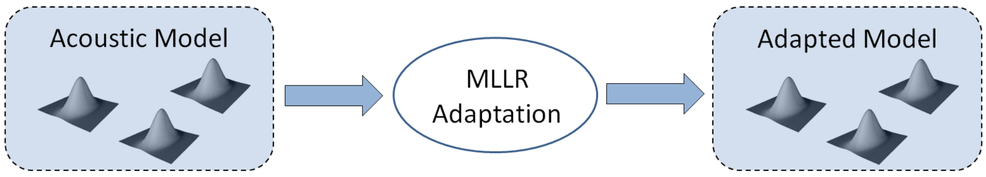


This might work for relatively small, specific scenarios. But it cannot be considered a generic solution because the variability of noise in different environments is some magnitudes higher than that of speech. A non-exhaustive list of noise types includes noise from engines, factories, trains and air conditioning systems, fan noise from computers, wind, rain, waves at the sea, scratching, clicking, rustling, dripping, boiling, footsteps, doors slamming, glass breaking, dishes clanking, water dripping, metal sounds, ringing, beeping, birds singing, shots, explosions, different categories of musical instruments, just to name a few. The problem of variability is worsened by the fact that noise can mask the speech spectrum[1] if the power of the noise is significantly stronger than that of speech (see Chapter 7).

[1] At moderate signal-to-noise ratios this might concerns some spectral regions only. But these regions may vary in time (due the non-stationarity of speech).

6.1.4 MLLR Adaptation

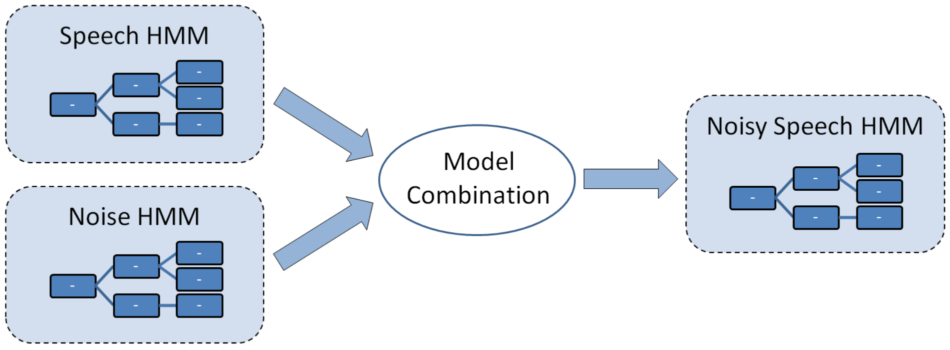
Another popular approach consists in estimating an affine, linear transform $f(x) = Ax + b$ that minimizes the mismatch between features of the operating (testing) and training environments. This approach was originally developed for speaker adaptation [157], in the early 1980th. But it would not become popular until the minimum mean square error (MMSE) estimate from [157] was replaced by maximum likelihood (ML) estimation, with one [158] or multiple [159, 160, 161] regression classes. The latter also coined the term under which the method is known today: *maximum likelihood linear regression* (MLLR) [160].



As firstly mentioned by Gales [161] and later demonstrated by Parikh, Raj and Stern [162], MLLR adaptation is not only useful for adapting to a particular speaker but also for adapting to a noisy acoustic environment. This result was confirmed in other work [163, 164], partly under substantial improvements of the speech recognition performance. Nevertheless, the usefulness of MLLR adaptation is limited in noisy environments because noise has a non-linear effect on clean speech features (see Section 5.2 or [120, 165, 121, 11]) while MLLR adaptation can only compensate for linear distortions.

6.1.5 Decoder Modification and Model Combination

An entirely different way to handle the effect of noise is to modify the decoder. Early approaches in this direction were the works of Klatt [129] and Bridle et al. [130] who found that noise actually tends to mask the spectral valleys of speech. Based on this observation, the main idea behind [129, 130] can be described as "telling the decoder to ignore the masked spectral bins". Holmes and Sedgwick [131] formulated this idea more concisely as "marginalizing over those spectral bins which are masked by noise". In addition to this, Holmes and Sedgwick proposed to use the observed noise as an upper bound. This laid the groundwork for the *bounded marginalization* technique, which would later be investigated in more detail by Josifovski et al. [166]. Nadas et al. [132] took a similar approach by using the log-max approximation (see (5.15) in Section 5.2.4) as a model for how noise corrupts clean speech features. And, instead of using masks for identifying "missing parts of the spectrum", they proposed to approximate noise as a Gaussian distribution in the log-spectral domain. This led to a solution which adequately treats uncertainties in the mask estimates [132]. Just one year after the publication of [132], Varga and Moore generalized Nadas's work in their *hidden Markov model (HMM) decomposition* [167] of speech and noise. The central idea of this approach is to use separate HMMs for speech and and noise. The decoder is modified to search the joint state space[2]. The output distribution of each joint (speech/noise) state is calculated with a *combination operator* that approximates the distribution of noisy speech. Gales and Young [168, 169, 170] developed a particular implementation of this general procedure. It goes by the name of *parallel model combination*; it works in the cepstral domain rather than on log-Mel spectra; and it uses the log-normal approximation for combining speech and noise distributions.



Although this approach drew quite some attention, it has been shown [123] that better results can be obtained by replacing the log-normal approximation [168, 169, 170] with the vector Taylor series approach by Moreno, Raj and Stern [45]. Alternatively, a Monte Carlo approximation may be used [171, 172].

6.1.6 Speech Feature Enhancement

Regarding the previous section, it should be mentioned that masking-based techniques such as [129, 130, 131, 166] work with log-spectral features only. That is because the masking principle cannot easily be translated to Mel frequency cepstral coefficients (i.e. the features which are used in modern ASR systems) [173]. Model combination approaches [132, 167, 170, 123, 172] do not suffer from this problem. But they are computationally expensive as they require transforming each and every Gaussian of the acoustic model. This can be detrimental for large vocabulary systems which tend to use some ten to a couple of hundred thousand Gaussians. One way to avoid transforming all the Gaussians is to project the transformation back to the feature space. And this is exactly what all speech feature enhancement approaches do in principle.

The credit for the first such approach is probably due to Van Compernolle, who translated the spectral subtraction rule to logarithmic filterbank energies [120] and then devised a minimum mean square error estimate for clean speech features [165]. This was done under the simplifying assumption that clean speech has a uniform distribution over the frequency bins. Erell and Weintraub [174] extended that approach by modeling the clean speech distribution as a Gaussian mixture. In addition to this, they proposed to jointly minimize the mean squared error in the frequency bands rather than individually optimizing each band.

[2] i.e. the Cartesian product of the speech and noise states from the individual HMMs

Just one year later, Acero and Stern used elements of both these approaches ([120] and [174]) to develop a *codeword-dependent cepstral normalization* (CDCN) technique [121, 11]. This technique introduced a number of important novelties. In particular: (a) the spectral subtraction rule was translated to the cepstral domain; (b) the authors added a term for the channel in order to account for differences in the acoustic transfer function; and (c) the parameters of the Gaussian noise distribution as well as the channel were jointly estimated in the framework of the expectation maximization algorithm [32]. This essentially laid the foundation for modern speech feature enhancement approaches, such as Moreno, Raj and Stern's vector Taylor series (VTS) [45] approach as well as various extensions thereof: Frey et al.'s ALGONQUIN [175], Deng, Droppo and Acero's phase-sensitive model [76], Stouten, Van Hamme and Wambacq's Higher Order VTS [176], Leutnant and Haeb-Umbach's work [119] just to name a few.

6.2 The Approach Taken in this Chapter

The approach taken in this chapter is a direct extension of Moreno's vector Taylor series approach [45]. It uses prior knowledge of how clean speech looks like in order to more accurately suppress the noise; it estimates the distribution of noise with an expectation maximization algorithm; and it works in the log-Mel domain in order to perform noise reduction in a domain that is of relevance to speech recognition. More specifically, the approach taken here is a Bayesian state estimation approach which

- models the distribution of clean speech as a Gaussian mixture.
- models the distribution of noise as a single Gaussian.
- uses the interaction function from Section 5.2 in order to predict the effect of noise to clean speech features.

Speech feature enhancement is performed by:

1. constructing the joint distribution of clean and noisy speech features as described in Section 6.3.3.
2. conditioning the resulting distribution on an observed noisy speech spectrum as described at the start of Section 6.3.
3. calculating the minimum mean square error (MMSE) estimate of clean speech as the mean of this distribution, as described in Section 6.3.1.

Deviating from [45], the distribution of noisy speech is predicted with the unscented transform rather than with a Taylor series expansion. This closely follows the work by Shinohara [50], which is here extended by a computationally efficient approximation of the MMSE clean speech estimate (in Section 6.3.2).

The novel aspects in this thesis mostly concern noise estimation. Particular contributions include:

- the derivation of a general EM algorithm for estimating noise from noisy speech features (Section 6.4.1).
- two new EM implementations which are based on the unscented transform (Sections 6.4.2 and 6.4.4) and Monte Carlo transformation (Section 6.4.5), respectively.
- a detailed theoretical comparison to the vector Taylor series implementations by Kim [124] and Li et al. [177].

Section 6.5 investigates an alternative approach, which tracks the noise with a particle filter and then estimates clean speech with the inverse model from Section 5.3. The usefulness of the proposed methods is finally evaluated by means of ASR experiments, in Chapter 8.

6.3 The Bayesian Solution to Speech Feature Enhancement

In order to explain Bayesian speech feature enhancement in more detail, let X_t denote the random variable of the hidden clean speech spectrum at time t. Furthermore, let Y_t denote the random variable of the corresponding observation (i.e. the observed noisy speech spectrum). Then the joint distribution p_{X_t,Y_t} contains all statistical knowledge about the relationship between X_t and Y_t. Following [178, 179, 49, 180, 50], this relationship is here modeled as a Gaussian mixture,

$$p_{X_t,Y_t}(x_t,y_t) = \sum_{k=1}^{K} c_k \ \mathcal{N}\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}; \begin{bmatrix} \mu_{X|k} \\ \mu_{Y|k} \end{bmatrix}, \begin{bmatrix} \Sigma_{XX|k} & \Sigma_{XY|k} \\ \Sigma_{YX|k} & \Sigma_{YY|k} \end{bmatrix}\right), \tag{6.1}$$

whose parameters are either learned from a joint clean / noisy speech corpus as in SPLICE [181] and stereo-based stochastic mapping [49, 180]; or constructed from the prior distributions of clean speech, channel and noise [50, 45], as described in Section 6.3.3. The posterior distribution $p_{X_t|y_t}$ of X_t given y_t is easily obtained according to Section 3.2.2. The result is a mixture of conditional Gaussian distributions:

$$p_{X_t|y_t}(x_t) = \sum_{k=1}^{K} c_{k|y_t} \mathcal{N}\left(x_t; \mu_{X_t|y_t,k}, \Sigma_{X_t|y_t,k}\right) \tag{6.2}$$

where $\mu_{X_t|y_t,k}$ and $\Sigma_{X_t|y_t,k}$ are the conditional mean and covariance matrix of the k-th Gaussian,

$$\mu_{X_t|y_t,k} = \mu_{X|k} + \Sigma_{XY|k}\Sigma_{YY|k}^{-1}(y_t - \mu_{Y|k}) \quad \text{and} \quad \Sigma_{X_t|y_t,k} = \Sigma_{XX|k} - \Sigma_{XY|k}\Sigma_{YY|k}^{-1}\Sigma_{YX|k}, \tag{6.3}$$

and where $c_{k|y_t}$ is its posterior probability

$$c_{k|y_t} \triangleq p_{K_t|y_t}(k) = \frac{c_k \mathcal{N}\left(y_t; \mu_{Y|k}, \Sigma_{YY|k}\right)}{\sum_{k'=1}^{K} c_{k'} \mathcal{N}\left(y_t; \mu_{Y|k'}, \Sigma_{YY|k'}\right)}. \tag{6.4}$$

Particular clean speech estimates are obtained by applying a criterion of optimality, such as the maximum a posteriori (MAP) criterion for which the optimal estimate consists in selecting that clean speech spectrum x_t which maximizes the likelihood function: $\hat{x}_t = \operatorname{argmax}_{x_t}\left(p_{X_t|y_t}(x_t)\right)$ [180]; or the minimum mean square error criterion, which is discussed in more detail in the following.

6.3.1 The Minimum Mean Square Error Solution

The minimum mean square error (MMSE) clean speech estimate consists in calculating the conditional mean $\hat{x}_t = \mathscr{E}_{p_{X_t|y_t}(x_t)}\{x_t\}$ of the hidden variable X_t given a realization y_t of the observable variable Y_t (see Section 2.4). So, given the joint Gaussian mixture distribution p_{X_t,Y_t} from (6.1) as well as a realized noisy speech spectrum y_t, the MMSE estimate of the clean speech spectrum x_t can be calculated according to:

$$\begin{aligned}\hat{x}_t &= \int x_t \sum_{k=1}^{K} c_{k|y_t} \mathcal{N}\left(x_t;\mu_{X_t|y_t,k},\Sigma_{X_t|y_t,k}\right) dx_t \\ &= \sum_{k=1}^{K} c_{k|y_t} \underbrace{\int x_t \mathcal{N}\left(x_t;\mu_{X_t|y_t,k},\Sigma_{X_t|y_t,k}\right) dx_t}_{\mu_{X_t|y_t,k}}\end{aligned} \tag{6.5}$$

where the posterior $c_{k|y_t}$ of the k-th Gaussian mode is obtained according to (6.3) and where the conditional mean $\mu_{X_t|y_t,k}$ is calculated according to (6.4). This solution has been used in a large variety of approaches, starting with Acero's codeword dependent cepstral normalization (CDCN) [121], Moreno's vector Taylor series (VTS) approach [45] and Kim's statistical linear approximation (SLA) [122]. Its popularity is due to its analytically tractability, in contrast to the MAP approach from [180] which requires an approximation with the EM algorithm.

6.3.2 Mode-Dependent Bias Correction

Some approaches, such as stereo-based piecewise linear compensation for environments (SPLICE) [181, 182], approximate the above MMSE solution by

$$\hat{x}_t \approx \sum_{k=1}^{K} c_{k|y_t}\left(\mu_{X|k} + y_t - \mu_{Y|k}\right) = y_t - \sum_{k=1}^{K} c_{k|y_t} \underbrace{\left(\mu_{Y|k} - \mu_{X|k}\right)}_{\triangleq \Delta_k}, \tag{6.6}$$

where Δ_k denotes the bias which the noise introduces to the k-th Gausssian mode. This estimate is equivalent to the 0-th order Taylor series expansion form [121, 45], but it can also be interpreted[3] as approximating $\Sigma_{XY|k}$ by $\Sigma_{YY|k}$ [180]. Following [13], (6.6) is here referred to as *mode dependent bias correction* (MDBC), as it corrects (i.e. subtracts) the bias which is introduced at each mode, weighted with the probability of being in that mode.

[3] simply compare $(y - \Delta_k)$ to the true conditional mean from (6.3).

6.3.3 Constructing the Joint

Both the general Bayesian approach to speech feature enhancement and its MMSE implementation from Section 6.3.1 require a Gaussian mixture approximation of $p_{X,Y}$. This section explains how such an approximation can be constructed by transforming the prior distributions of clean speech, channel and noise according to the interaction function from Section 5.2. That is done under the assumption [121, 45] that clean speech features (in the log-Mel domain) can be modeled as a Gaussian mixture random variable X, whose prior probabilities c_k, means $\mu_{X|k}$ and covariance matrices $\Sigma_{XX|k}$ have been learned on a clean speech training corpus:

$$p_X(x) = \sum_{k=1}^{\kappa} c_k \mathcal{N}(x; \mu_{X|k}, \Sigma_{XX|k}). \tag{6.7}$$

Regarding the choice of κ, it should be mentioned that a relatively small number (about 128 Gaussians) is usually sufficient for speech feature enhancement. Following Rose [183] and Moreno [45], the noise is modeled as a Gaussian random variable N_t (in the log-Mel domain) with mean[4] μ_N and covariance matrix Σ_{NN}:

$$p_N(n) = \mathcal{N}(n; \mu_N, \Sigma_{NN}) \tag{6.8}$$

In order to also account for convolutive distortions of speech spectra, Acero [121] and Moreno [45] proposed to model the channel H as a one point distribution $p_H(h) = \delta(h - \mu_H)$. This is here extended by a covariance matrix Σ_H. Hence, we arrive at a Gaussian distribution for the channel (as originally proposed in Frey [175] and Faubel [15]):

$$p_H(h) = \mathcal{N}(h; \mu_H, \Sigma_{HH}). \tag{6.9}$$

This model allows for all the variables to be treated in a uniform fashion, so that the distributions of clean speech, channel and noise can be combined in one "large" joint Gaussian mixture distribution

$$p_{X,H,N}(x,h,n) \;=\; \sum_{k=1}^{\kappa} c_k p_{X,H,N|k}(x,h,n) \tag{6.10}$$

where $p_{X,H,N|k}(x,h,n)$ denotes the k-th Gaussian component. If we further assume that X, H and N are statistically independent, each component $p_{X,H,N|k}(x,h,n)$ of the joint Gaussian mixture can be written $p_{X|k}(x)p_H(h)p_N(n)$ or, equivalently:

$$p_{X,H,N|k}(x,h,n) = \mathcal{N}\left(\begin{bmatrix} x \\ h \\ n \end{bmatrix}; \begin{bmatrix} \mu_{X|k} \\ \mu_H \\ \mu_N \end{bmatrix}, \begin{bmatrix} \Sigma_{XX|k} & 0 & 0 \\ 0 & \Sigma_{HH} & 0 \\ 0 & 0 & \Sigma_{NN} \end{bmatrix}\right) \tag{6.11}$$

[4] Note that these noise distribution parameters can be estimated from a noisy utterance by using the expectation maximization algorithm [121, 45], a sequential variant thereof [123], or a bank of interacting Kalman filters [184].

And as $p_{X,H,N|k}$ is considered to be time-invariant, we in particular have $p_{X_t,H_t,N_t|k} = p_{X,H,N|k}$ for all t. Hence, the joint distribution $p_{X_t,Y_t} = p_{X,Y}$ of clean and noisy speech is obtained by transforming the variables X, H and N according to Section 3.1:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \tilde{f}\left(\begin{bmatrix} X \\ H \\ N \end{bmatrix}\right) \quad \text{with} \quad \tilde{f}\left(\begin{bmatrix} x \\ h \\ n \end{bmatrix}\right) = \begin{bmatrix} x \\ f(x,h,n) \end{bmatrix}. \tag{6.12}$$

In this equation, f is the nonlinear interaction function from Section (5.2). The transformation from 6.12 can in particular be performed[5] by transforming each individual Gaussian component $(X,N,H|k)$ while keeping the weights c_k fixed:

$$p_{X,Y}(x,y) = \sum_{k=1}^{K} c_k p_{X,Y|k}(x,y), \qquad \begin{bmatrix} X|k \\ Y|k \end{bmatrix} = \tilde{f}\left(\begin{bmatrix} X|k \\ H \\ N \end{bmatrix}\right). \tag{6.13}$$

Due to the nonlinearity of f, the resulting variables $(X,Y)|k$ have a non-Gaussian distribution in general. This is portrayed in Figure 6.2, which shows the noisy speech distribution that is obtained by transforming one-dimensional clean speech and noise distributions according to the interaction function from Section 5.2.4.

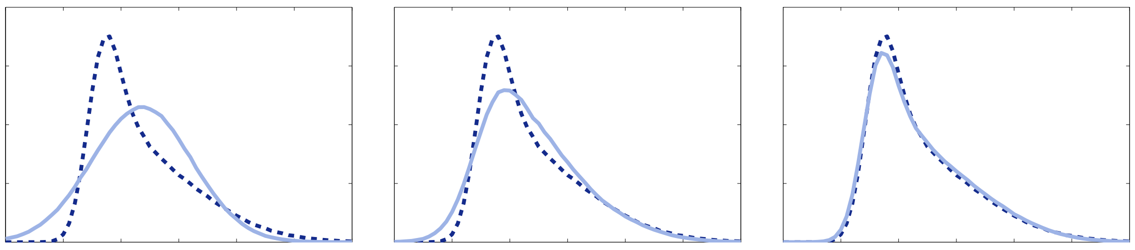

(a) ALoDT-1 (b) ALoDT-4 (c) ALoDT-16

Figure 6.2: *Transformed Distribution. The dashed curve shows the true distribution. The solid curves show approximations obtained with the adaptive level of detail transform (ALoDT) with 1, 4 and 16 Gaussians (see [12] for details). Also note that these result have been obtained under a Gaussian assumption of the original (i.e. non-transformed) distributions.*

Nevertheless, in most of the literature $p_{X,Y|k}$ is approximated as a single Gaussian. That is achieved through local linearization with a 1-st order vector Taylor series approximation [45], through a vector Taylor series expansion of higher order [176, 185, 186], through statistical linear approximation [122], or through use of the unscented transform [50, 15]. After this approximation, MMSE and MAP estimation can be performed as described at the start of Section 6.3. Algorithm 6.1 again subsumes this approach at the example of the unscented transform.

[5] see Section 2.2.8 for a proof

Bayesian Speech Feature Enhancement with the Unscented Transform

1. **Preparation**: For each clean speech mode $X|k$, $k=1,\dots,\kappa$, build the joint distribution of clean speech $X|k$, channel H and noise N according to:

$$p_{X,H,N|k}(x,h,n)=\mathcal{N}\left(\begin{bmatrix}x\\h\\n\end{bmatrix};\begin{bmatrix}\mu_{X|k}\\\mu_H\\\mu_N\end{bmatrix},\begin{bmatrix}\Sigma_{XX|k}&0&0\\0&\Sigma_{HH}&0\\0&0&\Sigma_{NN}\end{bmatrix}\right). \tag{6.14}$$

2. **Constructing the Joint**: Construct the joint distribution of clean and noisy speech by transforming each Gaussian mode $p_{X,H,N|k}$, $k=1,\dots,\kappa$, according to the augmented interaction function:

$$\begin{bmatrix}X|k\\Y|k\end{bmatrix}=\tilde{f}\left(\begin{bmatrix}X|k\\H\\N\end{bmatrix}\right)\quad\text{with}\quad\tilde{f}\left(\begin{bmatrix}x\\h\\n\end{bmatrix}\right)=\begin{bmatrix}x\\f(x,h,n)\end{bmatrix}. \tag{6.15}$$

This is achieved with the augmented unscented transform from Section 3.5.2; and it results in the following Gaussian mixture approximation (with $p_{X_t,Y_t}=p_{X,Y}$):

$$p_{X_t,Y_t}(x_t,y_t)=\sum_{k=1}^{\kappa}c_k\;\mathcal{N}\left(\begin{bmatrix}x_t\\y_t\end{bmatrix};\begin{bmatrix}\mu_{X|k}\\\mu_{Y|k}\end{bmatrix},\begin{bmatrix}\Sigma_{XX|k}&\Sigma_{XY|k}\\\Sigma_{YX|k}&\Sigma_{YY|k}\end{bmatrix}\right). \tag{6.16}$$

3. **Mode Probabilities**: Calculate the posterior probability $p_{K|y_t}(k)$ of each mode $p_{X_t,Y_t|k}$ according to:

$$c_{k|y_t}\triangleq p_{K|y_t}(k)=\frac{c_k\mathcal{N}\left(y_t;\mu_{Y|k},\Sigma_{YY|k}\right)}{\sum_{k'=1}^{\kappa}c_{k'}\mathcal{N}\left(y_t;\mu_{Y|k'},\Sigma_{YY|k'}\right)}. \tag{6.17}$$

4. **Clean Speech Estimation**: For mode-dependent bias correction, calculate the estimated clean speech spectrum $\hat{x}_t$ as

$$\hat{x}_t=y_t-\sum_{k=1}^{\kappa}c_{k|y_t}\delta_k\quad\text{with}\quad\delta_k=\left(\mu_{Y|k}-\mu_{X|k}\right). \tag{6.18}$$

For the full-conditional expectation, calculate

$$\hat{x}_t=\sum_{k=1}^{\kappa}c_{k|y_t}\mu_{X_t|y_t,k}\quad\text{with}\quad\mu_{X_t|y_t,k}=\mu_{X|k}+\Sigma_{XY|k}\Sigma_{YY|k}^{-1}\left(y_t-\mu_{Y|k}\right). \tag{6.19}$$

Algorithm 6.1: Bayesian Speech Feature Enhancement with the Unscented Transform

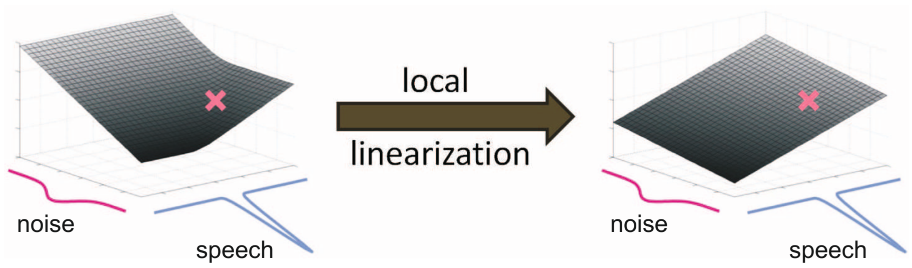


Figure 6.3: *Local linearization of the interaction function without consideration of the channel. The plot shows the one-dimensional case for the zero-phase factor model.*

6.3.4 Discussion

It is interesting to note that the above procedure is implicitly used in most MMSE approaches to speech feature enhancement. This includes the original vector Taylor series approach [45], Kim's sequential expectation maximization and interacting multiple model (IMM) approaches [123, 184], Segura's Model-based compensation [187], Frey's ALGONQUIN [175], Deng's Bayesian approach to speech feature enhancement [115] as well as more recent work [176, 186, 50, 119]. Indeed, even the Wiener filter can be interpreted as doing the same. It constructs the joint covariance matrix of clean and noisy speech spectra, X and Y, as

$$\Sigma = \begin{bmatrix} \sigma_{XX}(\omega) & \sigma_{XY}(\omega) \\ \sigma_{YX}(\omega) & \sigma_{YY}(\omega) \end{bmatrix} = \begin{bmatrix} \sigma_X^2(\omega) & \sigma_X^2(\omega) \\ \sigma_X^2(\omega) & \sigma_X^2(\omega)+\sigma_N^2(\omega) \end{bmatrix}.$$

This is done under the assumption that speech and noise are uncorrelated, i.e. $\sigma_{XY} = \sigma_{XX} = \sigma_X^2$. Furthermore, it is assumed that the speech and noise signals are zero-mean, from which it follows that their joint (circularly symmetric complex) Gaussian distribution can be written:

$$p_{X,Y}\left(\begin{bmatrix} X(\omega) \\ Y(\omega) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} X(\omega) \\ Y(\omega) \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_X^2(\omega) & \sigma_X^2(\omega) \\ \sigma_X^2(\omega) & \sigma_X^2(\omega)+\sigma_N^2(\omega) \end{bmatrix}\right) \tag{6.20}$$

In this case, the MMSE (i.e. Wiener) solution is obtained by conditioning (6.20) on a realized noisy speech spectrum $Y(\omega)$. This gives (according to Section 3.2.1):

$$\hat{X}_t(\omega) = 0 + \sigma_{XY}(\omega)\sigma_{YY}(\omega)^{-1}(Y_t(\omega) - 0) = \frac{\sigma_X^2(\omega)}{\sigma_X^2(\omega)+\sigma_N^2(\omega)} Y_t(\omega). \tag{6.21}$$

As it turns out that $\sigma_X^2(\omega)$ and $\sigma_N^2(\omega)$ are not easy to obtain in practice, Lim and Oppenheim [188] proposed an iterative approach which determines $\sigma_X^2(\omega)$ and $\sigma_N^2(\omega)$ by iterating between Wiener filtering and LPC parameter estimation. This procedure was later [189] shown to be an instance of the EM algorithm. More recent work [45, 124, 15] uses a strong prior model for the distribution of clean speech features and estimates the noise distribution with the EM algorithm. This idea is elaborated in more detail in the upcoming section.

6.4 Expectation Maximization Based Noise Estimation

Methods for compensating the statistical effects of noise either directly enhance speech features based on the approach from Section 6.3 or they adapt the acoustic models as described in [170, 124]. As both cases require an accurate estimate of the noise distribution, it is often assumed that (a) the noise distribution is known a priori or (b) that it can reliably be estimated from "suspected" noise only frames (e.g. at the start of an utterance). Acero [121] and Moreno [45] went one step further and proposed to estimate it from the entire utterance. This approach is explained in more detail in the following. In particular, it is here derived in a generalized fashion which allows for several extensions that have been proposed in the framework of this thesis [15, 16]. Following [183] and [124], noise spectra are modeled as a Gaussian random variable N with distribution

$$p_N(n) = \mathcal{N}(n; \mu_N, \Sigma_{NN}). \tag{6.22}$$

With this model, the noise estimation problem consists in estimating the noise mean μ_N and covariance matrix Σ_{NN} from a sequence $\bar{y} \triangleq (y_1, \ldots, y_\tau)$ of noisy speech features. This can be achieved in a maximum likelihood fashion by finding that parameter $\theta = \{\mu_N, \Sigma_{NN}\}$ which maximizes the likelihood $\mathscr{L}(\theta; \bar{y}) = p_{Y|\theta}(\bar{y})$:

$$\theta_{ML} \triangleq \underset{\theta}{\operatorname{argmax}} \prod_{t=1}^{\tau} p_{Y_t|\theta}(y_t) = \underset{\theta}{\operatorname{argmax}} \log \prod_{t=1}^{\tau} p_{Y_t|\theta}(y_t) = \underset{\theta}{\operatorname{argmax}} \sum_{t=1}^{\tau} \log p_{Y_t|\theta}(y_t).$$

As direct maximization of the likelihood turns out to be difficult, Rose [183] proposed to use the EM algorithm [32], with the rationale in mind that the estimation problem would be easier to solve if noise spectra were introduced as a hidden variable. This can be achieved by augmenting the observed noisy speech spectra $\bar{y} = (y_1, \ldots, y_\tau)$ with unobserved, i.e. unknown, noise spectra $\bar{n} \triangleq (n_1, \ldots, n_\tau)$. Subsequently declaring the augmented data $\{\bar{y}, \bar{n}\}$ complete and the observed data $\bar{y}$ incomplete, we have a maximum likelihood problem with incomplete data, i.e. a scenario in which the EM algorithm can be applied. The EM algorithm solves this problem by iterating between two steps:

1. an expectation step, in which the auxiliary function $\mathcal{Q}(\theta|\theta^{(l)})$ is constructed as the expectation of the log likelihood function $\mathscr{L}(\theta; \bar{y}, \bar{n})$ of the complete data $\{\bar{y}, \bar{n}\}$, given the incomplete data $\bar{y}$ as well as the current parameter estimate $\theta^{(l)}$:

$$\mathcal{Q}(\theta|\theta^{(l)}) \triangleq \int \log p_{Y,N|\theta}(\bar{y}, \bar{n}) p_{N|\bar{y},\theta^{(l)}}(\bar{n}) d\bar{n} \tag{6.23}$$

2. a maximization step in which the next parameter estimate $\theta^{(l+1)}$ is chosen to be a value θ which maximizes the auxiliary function:

$$\theta^{(l+1)} = \underset{\theta}{\operatorname{argmax}} \mathcal{Q}(\theta|\theta^{(l)}) \tag{6.24}$$

Iterating these two steps causes the parameter θ to converge to a local maximum of the likelihood function [32].

6.4.1 The General Solution

In order to derive a general EM algorithm for noise estimation, let $\theta^{(l)} = \{\mu_N^{(l)}, \Sigma_{NN}^{(l)}\}$ be the current parameter estimate of the noise distribution. Then, the expectation step requires constructing the auxiliary function $\mathscr{Q}(\theta|\theta^{(l)})$ for the current iteration, with $\theta = \{\mu_N, \Sigma_{NN}\}$. This is done under the assumption of statistical independence of the complete data samples (y_t, n_t), for $t = 1, \dots, \tau$, so that the auxiliary function (6.23) simplifies to:

$$\mathscr{Q}(\theta|\theta^{(l)}) = \sum_{t=1}^{\tau} \int \log p_{Y_t,N_t|\theta}(y_t, n_t) p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t. \tag{6.25}$$

Further using the fact that noisy speech spectra y_t are not dependent on the noise parameter θ once the noise spectrum n_t is known, $p_{Y_t,N_t|\theta}(y_t, n_t)$ can be written:

$$p_{Y_t,N_t|\theta}(y_t, n_t) = \underbrace{p_{Y_t|n_t,\theta}(y_t)}_{p_{Y_t|n_t}(y_t)} \; \underbrace{p_{N_t|\theta}(n_t)}_{\mathscr{N}(n_t;\mu_N,\Sigma_{NN})} ,$$

i.e. $\log\big(p_{Y_t,N_t|\theta}(y_t, n_t)\big) = \log\big(p_{Y_t|n_t}(y_t)\big) + \log\big(p_{N_t|\theta}(n_t)\big)$. Substituting this into (6.25) gives:

$$\begin{aligned} \mathscr{Q}(\theta|\theta^{(l)}) &= \sum_{t=1}^{\tau} \int \log p_{Y_t|n_t}(y_t) p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t \\ &\quad + \sum_{t=1}^{\tau} \int \log p_{N_t|\theta}(n_t) p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t. \end{aligned} \tag{6.26}$$

That is the expectation step. In the maximization step, we have to take the derivative of this equation with respect to $\theta = \{\mu_N, \Sigma_{NN}\}$ in order to find that parameter which maximizes the auxiliary function $\mathscr{Q}(\theta|\theta^{(l)})$. Having a closer look at (6.26), it becomes clear that the sum in the first line is independent of θ. Hence, the derivative with respect to μ_N is:

$$\begin{aligned} \frac{d\mathscr{Q}(\theta|\theta^{(l)})}{d\mu_N} &= \sum_{t=1}^{\tau} \int -\frac{1}{2}\Sigma_{NN}^{-1} \big(n_t - \mu_N\big) p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t \\ &= -\frac{1}{2}\Sigma_{NN}^{-1} \sum_{t=1}^{\tau} \left(\int n_t p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t - \mu_N \right) \end{aligned}$$

where it was used that (1) the term $(-\frac{1}{2}\Sigma_{NN}^{-1})$ is not dependent on the integration variable n_t and (2) that the integral $\int p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t$ over a pdf is 1. Equating the derivative $d\mathscr{Q}(\theta|\theta^{(l)})/d\mu_N$ to zero and further solving for μ_N gives:

$$\mu_N^{(l+1)} = \frac{1}{\tau} \sum_{t=1}^{\tau} \int n_t p_{N_t|y_t,\theta^{(l)}}(n_t) dn_t. \tag{6.27}$$

In order to also find the covariance matrix Σ_{NN} that maximizes $\mathscr{Q}(\theta|\theta^{(l)})$, let us take the derivative of (6.26) with respect to Σ_{NN}^{-1}. This gives:

$$\frac{d\mathscr{Q}(\theta|\theta^{(l)})}{d\Sigma_{NN}^{-1}} = \sum_{t=1}^{\tau}\int\left(\frac{1}{2}\Sigma_{NN}-\frac{1}{2}\left(n_t-\mu_N\right)\cdot\left(n_t-\mu_N\right)^T\right)p_{N_t|y_t,\theta^{(l)}}(n_t)dn_t$$

where it was used that, firstly, the derivative of $(n_t-\mu_N)^T\Sigma_{NN}^{-1}(n_t-\mu_N)$ with respect to Σ_{NN}^{-1} is $(n_t-\mu_N)(n_t-\mu_N)^T$; secondly, the log of the inverse of the normalizing constant can be written $\log((2\pi)^n\det(\Sigma_{NN}))^{-1/2} = 1/2\log\det(\Sigma_{NN}^{-1}) - n/2\log(2\pi)$; thirdly the derivative of the determinant $\det(\Sigma_{NN}^{-1})$ with respect to Σ_{NN}^{-1} is the adjunct, i.e. $\det(\Sigma_{NN}^{-1})\Sigma_{NN}$; and fourthly, that as a consequence of this: $d\log(\det(\Sigma_{NN}^{-1}))/d\Sigma_{NN}^{-1} = \Sigma_{NN}$. Finally equating the derivative $d\mathscr{Q}(\theta|\theta^{(l)})/d\Sigma_{NN}^{-1}$ to zero and solving for Σ_{NN} yields:

$$\Sigma_{NN}^{(l+1)} = \frac{1}{\tau}\sum_{t=1}^{\tau}\int n_t n_t^T p_{N_t|y_t,\theta^{(l)}}(n_t)dn_t - \mu_N^{(l)}\left(\mu_N^{(l)}\right)^T. \tag{6.28}$$

This shows that the general EM algorithm for noise estimation consist in iteratively updating the noise parameters $\theta^{(l)}$ according to (6.27) and (6.28), for $l = \{1,2,3,\ldots\}$ [16]. As particular implementations differ only in how they approximate the occurring integrals, the differences rely mainly on which approximation of the instantaneous noise distribution $p_{N_t|y_t,\theta^{(l)}}$ is used. Based on this criterion, existing approaches can be grouped into two categories: ones that are based on Gaussian mixture approximations [45, 124, 177, 15] and ones that are based on Monte Carlo approximations [16].

6.4.2 Gaussian Mixture Approximations

The majority of approaches [45, 124, 177, 15] is based on Gaussian mixture approximations of the instantaneous noise distribution $p_{N_t|y_t,\theta^{(l)}}$. This has its origin in Moreno et al's vector Taylor series (VTS) approach [45, 46] in which the distribution p_X of clean speech features is modeled as a mixture of Gaussians:

$$p_X(x) = \sum_{k=1}^{K} c_k\mathcal{N}\left(x;\mu_{X|k},\Sigma_{XX|k}\right). \tag{6.29}$$

Using this model, Moreno et al. introduced the index k of the clean speech component as a hidden variable and then wrote the instantaneous noise distribution $p_{N_t|y_t,\theta^{(l)}}$ as a marginal distribution of $p_{N_t,K|y_t,\theta^{(l)}}$:

$$p_{N_t|y_t,\theta^{(l)}}(n_t) = \sum_{k=1}^{K}\underbrace{p_{N_t|y_t,k,\theta^{(l)}}(n_t)p_{K|y_t,\theta^{(l)}}(k)}_{=p_{N_t,K|y_t,\theta^{(l)}}(n_t,k)}. \tag{6.30}$$

This has the advantage that the instantaneous noise distribution can be estimated individually for each clean speech mode $\mathcal{N}\left(x;\mu_{X|k},\Sigma_{XX|k}\right)$ if it is further weighted with the probability

$p_{K|y_t,\theta^{(l)}}(k)$ of being in that mode. This is possible as $p_{K|y_t,\theta^{(l)}}(k)$ can be evaluated with Bayes rule, with $c_k = p_K(k) = p_{K,\theta^{(l)}}(k)$:

$$p_{K|y_t,\theta^{(l)}}(k) = \frac{p_{Y_t|k,\theta^{(l)}}(y_t)c_k}{\sum_{k'=1}^{K} p_{Y_t|k',\theta^{(l)}}(y_t)c_{k'}}. \tag{6.31}$$

6.4.2.1 Constructing the Required Distributions

The evaluation of (6.31) requires predicting the noisy speech distribution $Y|k,\theta^{(l)}$ from (1) the clean speech mode $X|k$ and (2) the noise estimate $N|\theta^{(l)}$ from the last iteration. This is achieved by transforming the joint distribution of $N|\theta^{(l)}$ and $X|k$,

$$p_{N,X|k,\theta^{(l)}}(n,x) = \mathcal{N}\left(\begin{bmatrix} n \\ x \end{bmatrix}; \begin{bmatrix} \mu_N^{(l)} \\ \mu_{X|k} \end{bmatrix}, \begin{bmatrix} \Sigma_{NN}^{(l)} & 0 \\ 0 & \Sigma_{XX|k} \end{bmatrix}\right), \tag{6.32}$$

according to the interaction function f from Section 5.2. As noise parameter estimation also requires knowledge of the relationship between noisy speech Y and noise N, the transformation $N,X|k,\theta^{(l)} \longmapsto Y|k,\theta^{(l)}$ is extended to [15]:

$$\tilde{X} \triangleq \begin{bmatrix} N|\theta^{(l)} \\ X|k \end{bmatrix} \underset{\tilde{f}}{\longmapsto} \begin{bmatrix} N|\theta^{(l)} \\ Y|k,\theta^{(l)} \end{bmatrix} \triangleq \tilde{Y} \text{ with } \tilde{f}\left(\begin{bmatrix} n \\ x \end{bmatrix}\right) \triangleq \begin{bmatrix} n \\ f(x,0,n) \end{bmatrix}. \tag{6.33}$$

The result is a joint distribution $p_{N,Y|k,\theta^{(l)}}$ of $N|\theta^{(l)}$ and $Y|k,\theta^{(l)}$ from which both the instantaneous noise distributions $p_{N_t|y_t,k,\theta^{(l)}}$ and the observation likelihoods $p_{Y_t|k,\theta^{(l)}}(y_t)$ can be obtained as conditional and marginal distributions. Note again that X, Y and N are considered to be time-invariant, such that $p_{N,Y|k,\theta^{(l)}} = p_{N_t,Y_t|k,\theta^{(l)}}$ and so on.

6.4.2.2 Vector Taylor Series Expansion

Due to the nonlinearity of f, the transformed random variables $(N,Y)|k,\theta^{(l)}$ are no longer Gaussian. This greatly complicates the evaluation of the instantaneous noise distribution $p_{N_t|y_t,k,\theta^{(l)}}$ as well as the corresponding observation likelihood $p_{Y_t|k,\theta^{(l)}}(y_t)$. And it led Moreno et al. [45, 46] to approximate (6.33) through local linearization of $\tilde{f}$ around the means of the clean speech and noise distributions, $\mu_{X|k}$ and $\mu_N^{(l)}$ (see Section 3.4). The result is a Gaussian approximation of the joint distribution $p_{N,Y|k,\theta^{(l)}}$:

$$p_{N,Y|k,\theta^{(l)}}(n,y) \approx \mathcal{N}\left(\begin{bmatrix} n \\ y \end{bmatrix}; \begin{bmatrix} \mu_N^{(l)} \\ \mu_{Y|k,\theta^{(l)}} \end{bmatrix}, \begin{bmatrix} \Sigma_{NN}^{(l)} & \Sigma_{NY|k,\theta^{(l)}} \\ \Sigma_{YN|k,\theta^{(l)}} & \Sigma_{YY|k,\theta^{(l)}} \end{bmatrix}\right) \tag{6.34}$$

where the mean and covariance matrix are calculated according to Section 3.4:

$$\begin{aligned} \mu_{Y|k,\theta^{(l)}} &= f\left(\mu_{X|k}, 0, \mu_N^{(l)}\right), \quad \Sigma_{YN|k,\theta^{(l)}} = C_k \Sigma_{NN}^{(l)}, \\ \Sigma_{NY|k,\theta^{(l)}} &= \Sigma_{YN|k,\theta^{(l)}}^T, \quad \Sigma_{YY|k,\theta^{(l)}} = C_k \Sigma_{NN}^{(l)} C_k^T + B_k \Sigma_{XX|k} B_k^T. \end{aligned} \tag{6.35}$$

In these equations, $B_k = \nabla_x f(\mu_{X|k}, 0, \mu_N^{(l)})$ and $C_k = \nabla_n f(\mu_{X|k}, 0, \mu_N^{(l)})$ denote the Jacobians of the interaction function $f(x,0,n)$ with respect to x and n, evaluated at the mean $\mu_{X|k}$ of the k-th clean speech component as well as the mean $\mu_N^{(l)}$ of the current noise estimate. Marginalizing $p_{N,Y|k,\theta^{(l)}} = p_{N_t,Y_t|k,\theta^{(l)}}$ with respect to N_t leads to a Gaussian approximation of the observation likelihood:

$$p_{Y_t|k,\theta^{(l)}}(y_t) \approx \mathcal{N}\left(y_t; \mu_{Y|k,\theta^{(l)}}, \Sigma_{YY|k,\theta^{(l)}}\right). \tag{6.36}$$

The corresponding approximation of the instantaneous noisy distribution $p_{N_t|y_t,k,\theta^{(l)}}$ is obtained by conditioning $p_{N_t,Y_t|k,\theta^{(l)}}$ on the realized noisy speech spectrum y_t. This gives:

$$p_{N_t|y_t,k,\theta^{(l)}}(n_t) \approx \mathcal{N}\left(n_t; \mu_{N_t|k,y_t,\theta^{(l)}}, \Sigma_{N_t|k,y_t,\theta^{(l)}}\right), \tag{6.37}$$

where $\mu_{N_t|k,y_t,\theta^{(l)}}$ and $\Sigma_{N_t|k,y_t,\theta^{(l)}}$ are calculated as described in Section 2.2.5 [15]:

$$\mu_{N_t|k,y_t,\theta^{(l)}} = \mu_N^{(l)} + \Sigma_{NY|k,\theta^{(l)}} \Sigma_{YY|k,\theta^{(l)}}^{-1} (y_t - \mu_{Y|k,\theta^{(l)}}), \tag{6.38}$$

$$\Sigma_{N_t|k,y_t,\theta^{(l)}} = \Sigma_{NN}^{(l)} - \Sigma_{NY|k,\theta^{(l)}} \Sigma_{YY|k,\theta^{(l)}}^{-1} \Sigma_{YN|k,\theta^{(l)}}. \tag{6.39}$$

Substituting these Gaussian approximations of $p_{N_t|y_t,k,\theta^{(l)}}$ and $p_{Y_t|k,\theta^{(l)}}$ back into (6.30) and (6.31) gives a Gaussian mixture approximation of the instantaneous noise distribution.

6.4.2.3 Gaussian Mixture Implementation of the EM Algorithm

To subsume: the Gaussian mixture implementation of the EM algorithm uses the noise parameters $\theta^{(l)} = \left\{\mu_N^{(l)}, \Sigma_{NN}^{(l)}\right\}$ from the previous iteration in order to transform each clean speech Gaussian mode $X|k$ according to (6.32) and (6.33). Performing these transformation with the vector Taylor series (VTS) approach from Section 6.4.2.2 results in joint Gaussian approximations $p_{N,Y|k,\theta^{(l)}} = p_{N_t,Y_t|k,\theta^{(l)}}$ of noise and noisy speech. The marginal density $p_{Y_t|k,\theta^{(l)}}$ simulates how the noise estimate $\theta^{(l)}$ changes the k-th clean speech mode. Hence, the probability that the noisy speech spectrum y_t at time t originated from the k-th clean speech Gaussian can be evaluated as:

$$p_{K|y_t,\theta^{(l)}}(k) = \frac{p_{Y_t|k,\theta^{(l)}}(y_t) c_k}{\sum_{k'} p_{Y_t|k',\theta^{(l)}}(y_t) c_{k'}}. \tag{6.40}$$

The corresponding instantaneous noise distribution $p_{N_t|y_t,k,\theta^{(l)}}$ can be calculated according to (6.37). Plugging these $p_{K|y_t,\theta^{(l)}}$ and $p_{N_t|y_t,k,\theta^{(l)}}$ into (6.30) and substituting the result into (6.27) and (6.28) gives the noise parameter set $\theta^{(l+1)} = \left\{\mu_N^{(l+1)}, \Sigma_{NN}^{(l+1)}\right\}$ for the next iteration:

$$\mu_N^{(l+1)} = \frac{1}{\tau} \sum_{t=1}^{\tau} \sum_{k=1}^{K} p_{K|y_t,\theta^{(l)}}(k) \mu_{N_t|k,y_t,\theta^{(l)}} \tag{6.41}$$

$$\Sigma_{NN}^{(l+1)} = \frac{1}{\tau} \sum_{t=1}^{\tau} \sum_{k=1}^{K} p_{K|y_t,\theta^{(l)}}(k) \left(\Sigma_{N_t|k,y_t,\theta^{(l)}} + \mu_{N_t|k,y_t,\theta^{(l)}} \mu_{N_t|k,y_t,\theta^{(l)}}^T - \mu_N^{(l+1)} \mu_N^{(l+1)\,T} \right) \tag{6.42}$$

where $\mu_{N_t|k,y_t,\theta^{(l)}}$ and $\Sigma_{N_t|k,y_t,\theta^{(l)}}$ denote the mean and covariance of the instantaneous noise distribution from (6.38) and (6.39). These re-estimation equations can be interpreted as accumulating statistics of the instantaneous noise distributions $p_{N_t|y_t,k,\theta^{(l)}}$, weighted with the probability $p_{K|y_t,\theta^{(l)}}(k)$ that y_t was generated by the k-th clean speech mode.

6.4.3 Historical Perspective & Discussion

Regarding the results from the preceding sections, it should be mentioned that the noise estimation equations given here differ from those given in other work in the literature [46, 124, 177]. Before discussing this in more detail, with theoretical comparisons to Kim [124] and Li's [177] approaches in Sections 6.4.3.2 and 6.4.3.3, let us briefly sketch the historical development of EM based noise estimation. This approach had its beginnings in the early nineties, with Acero and Stern's seminal work on "Environmental Robustness in Automatic Speech Recognition" [121]. More precisely: with the codeword-dependent cepstral normalization (CDCN) technique published therein, which used a clean speech Gaussian mixture model in order to estimate noise based on the cepstral bias that it introduces at the individual modes[6]. Six years after the publication of [121], Moreno, Raj and Stern extended the CDCN approach by a vector Taylor series expansion [45, 46] which more accurately captures the statistical relationship between clean and noisy speech. This was in turn extended by Kim et al. [124] who gave the first proper derivation of an EM algorithm which also estimates the noise covariance matrix rather than just the mean [121, 46]. Kim's extension was based on modifying the auxiliary function from [46] by introducing noise samples as a hidden variable (similar as it had been done before by Rose et al. [183]). After the publication of [124], there was a period of relative silence until Li et al. [177] (some 10 years later) revived Moreno's approach with some slight modifications.

6.4.3.1 Discussion

In the following, it will be shown that all of the above mentioned approaches can be viewed as special cases of the results from Section 6.4.2.3. Starting from this point, the approximations made in [124, 177] are investigated regarding stability issues and bias in estimation. This analysis reveals in particular that

1. Kim et al's [124] parameter estimates of the instantaneous noise distribution can become unstable.
2. Li, Deng and Acero [177] implicitly use the noise variance in their mean estimate, although they claim not to use it.
3. Li, Deng and Acero's approach [177] tends to overestimate the variance of the instantaneous noise distribution.

[6] Note that both mode and codeword here refer to the very same thing: a Gaussian component of the clean speech Gaussian mixture distribution.

In order to prove these statements, let us first expand the parameters of the instantaneous noise distribution from Section 6.4.2.2. For this, the Jacobians of the interaction function $f(x,0,n)$ with respect to x and n are again denoted by $B_k = \nabla_x f(\mu_{X|k}, 0, \mu_N^{(l)})$ and $C_k = \nabla_n f(\mu_{X|k}, 0, \mu_N^{(l)})$. Then combining (6.38) and (6.35), the mean of the instantaneous noise distribution can be written

$$\mu_{N_t|k,y_t,\theta^{(l)}} \approx \mu_N^{(l)} + \underbrace{\Sigma_{NN}^{(l)} C_k^T}_{\Sigma_{NY|k,\theta^{(l)}}} \underbrace{\left(C_k \Sigma_{NN}^{(l)} C_k^T + B_k \Sigma_{XX|k} B_k^T\right)}_{\Sigma_{YY|k,\theta^{(l)}}}^{-1} \left(y_t - f(\mu_{X|k}, 0, \mu_N^{(l)})\right). \tag{6.43}$$

The covariance matrix of the instantaneous noise distribution can be expanded analogously, by combining (6.39) and (6.35):

$$\Sigma_{N_t|k,y_t,\theta^{(l)}} \approx \Sigma_{NN}^{(l)} - \underbrace{\Sigma_{NN}^{(l)} C_k^T}_{\Sigma_{NY|k,\theta^{(l)}}} \underbrace{\left(C_k \Sigma_{NN}^{(l)} C_k^T + B_k \Sigma_{XX|k} B_k^T\right)}_{\Sigma_{YY|k,\theta^{(l)}}}^{-1} \underbrace{C_k \Sigma_{NN}^{(l)}}_{\Sigma_{YN|k,\theta^{(l)}}}. \tag{6.44}$$

In order to conform to the notation used in [124, 177], let us further rewrite the update equations (6.41) and (6.42) from Section 6.4.2.3 by expressing τ by means of the posterior mode probabilities $p_{K|y_t,\theta^{(l)}}(k)$:

$$\tau = \sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k). \tag{6.45}$$

This equality holds as the sum over the $p_{K|y_t,\theta^{(l)}}(k)$, $k = 1, \ldots, \kappa$ is 1. Then, plugging (6.45) into (6.41) and (6.42), we get a formulation that is compatible with Kim [124] and Li's [177] noise update equations:

$$\mu_N^{(l+1)} = \frac{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k) \mu_{N_t|k,y_t,\theta^{(l)}}}{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k)}, \tag{6.46}$$

$$\Sigma_{NN}^{(l+1)} = \frac{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k) \left(\Sigma_{N_t|k,y_t,\theta^{(l)}} + \mu_{N_t|k,y_t,\theta^{(l)}} \mu_{N_t|k,y_t,\theta^{(l)}}^T\right)}{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k)} - \mu_N^{(l+1)} \mu_N^{(l+1)\,T} \tag{6.47}$$

which can now be directly compared to [124] and [177].

6.4.3.2 Comparison to Kim's Approach

Starting with Kim et al's approach, it should be mentioned that [124] uses different approximations for the mean and the covariance of the instantaneous noise distribution. Regarding the mean, [124] implicitly assumes that $\Sigma_{XX|k} = 0$. This claim is verified by substituting $\Sigma_{XX|k} = 0$ into (6.43),

$$\mu_{N_t|k,y_t,\theta^{(l)}} \approx \mu_N^{(l)} + \underbrace{\Sigma_{NN}^{(l)} C_k^T C_k^{-T} \left(\Sigma_{NN}^{(l)}\right)^{-1} C_k^{-1}}_{=C_k^{-1}} \left(y_t - f(\mu_{X|k}, 0, \mu_N^{(l)})\right),$$

and then pulling $\mu_N^{(l)}$ into $\left(y_t - f(\mu_{X|k}, 0, \mu_N^{(l)})\right)$, which directly gives the solution from [124]:

$$\mu_{N_t|k,y_t,\theta^{(l)}} \approx C_k^{-1}\left(y_t - f(\mu_{X|k}, 0, \mu_N^{(l)}) + C_k\mu_N^{(l)}\right). \tag{6.48}$$

A quick analysis reveals that the inverse of C_k (i.e. the Jacobian of f with respect to n) becomes numerically unstable if C_k does not have full rank. This can easily happen if the speech power is significantly stronger than that of noise, even if it just happens in one spectral bin. In order to also compare the covariance matrices of the instantaneous noise distributions, let us first rewrite the predicted noisy speech covariance from (6.35) by means of the variables A, B, C and D:

$$\Sigma_{YY|k,\theta^{(l)}} = \underbrace{C_k\Sigma_{NN}^{(l)}C_k^T}_{A} + \underbrace{B_k}_{B}\,\underbrace{\Sigma_{XX|k}}_{C}\,\underbrace{B_k^T}_{D}$$

Then, making use of the Sherman-Morrison-Woodbury (matrix inversion) formula, we find that $\Sigma_{YY|k,\theta^{(l)}}^{-1}$ can be written:

$$\underbrace{\left(C_k^{-T}\Sigma_{NN}^{-1}C_k^{-1}\right)}_{A^{-1}} + \underbrace{\left(C_k^{-T}\Sigma_{NN}^{-1}C_k^{-1}\right)}_{A^{-1}}\underbrace{B_k}_{B}\left(\underbrace{\Sigma_{XX|k}^{-1}}_{C^{-1}} + \underbrace{B_k^T}_{D}\underbrace{\left(C_k^{-T}\Sigma_{NN}^{-1}C_k^{-1}\right)}_{A^{-1}}\underbrace{B_k}_{B}\right)^{-1}\underbrace{B_k^T}_{D}\underbrace{\left(C_k^{-T}\Sigma_{NN}^{-1}C_k^{-1}\right)}_{A^{-1}}$$

where the superscript (l) has been dropped for better readability. Substituting this into (6.44) gives:

$$\Sigma_{N_t|k,y_t,\theta^{(l)}} = C_k^{-1}B_k\left(\Sigma_{XX|k}^{-1} + B_k^T C_k^{-T}\left(\Sigma_{NN}^{(l)}\right)^{-1}C_k^{-1}B_k\right)^{-1}B_k^T C_k^{-T}, \tag{6.49}$$

which when compared to [124] reveals that Kim et al's derivation of the instantaneous noise distribution assumes that $B_k^T C_k^{-T}\Sigma_{NN}^{-1}C_k^{-1}B_k = 0$. This gives the following estimate:

$$\Sigma_{N_t|k,y_t,\theta^{(l)}} = C_k^{-1}B_k\Sigma_{XX|k}B_k^T C_k^{-T}, \tag{6.50}$$

which again becomes unstable if C_k does not have full rank, i.e. if speech dominates, as explained above for (6.48).

6.4.3.3 Comparison to Li's Approach

In order to understand the approximations made in Li et al's approach [177, 190], let us first combine (6.43) and (6.46) in one equation and then pull out[7] the $\mu_N^{(l)}$ and $\Sigma_{NN}^{(l)}$ from the double sum in (6.46). This obviously gives:

$$\mu_N^{(l+1)} = \mu_N^{(l)} + \Sigma_{NN}^{(l)}\left(\frac{\sum_{t=1}^{\tau}\sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k)C_k^T\Sigma_{YY|k,\theta^{(l)}}^{-1}\left(y_t - f(\mu_{X|k}, 0, \mu_N^{(l)})\right)}{\sum_{t=1}^{\tau}\sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k)}\right). \tag{6.51}$$

[7] This is possible as $\mu_N^{(l)}$ and $\Sigma_{NN}^{(l)}$ are obviously independent of k and t.

If we now compare this result to equation (20) in [177], it becomes clear that Li et al. approximate $\Sigma_{NN}^{(l)}$ as:

$$\Sigma_{NN}^{(l)} \approx \left(\frac{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k) C_k^T \Sigma_{YY|k,\theta^{(l)}}^{-1} C_k}{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k)} \right)^{-1}. \tag{6.52}$$

This can be interpreted as accumulating the inverse covariance matrices $\Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1}$ of the instantaneous noise distributions and then obtaining $\Sigma_{NN}^{(l)}$ through inversion:

$$\Sigma_{NN}^{(l)} \approx \left(\frac{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k) \Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1}}{\sum_{t=1}^{\tau} \sum_{k=1}^{\kappa} p_{K|y_t,\theta^{(l)}}(k)} \right)^{-1} \quad \text{with} \quad \Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1} \approx C_k^T \Sigma_{YY|k,\theta^{(l)}}^{-1} C_k. \tag{6.53}$$

This obviously is more stable than directly accumulating the $\Sigma_{N_t|k,y_t,\theta^{(l)}}$, as it does not require inverting the individual C_k. But it is of interest to understand the assumptions which are made when $\Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1}$ is approximated by $C_k^T \Sigma_{YY|k,\theta^{(l)}}^{-1} C_k$. For that, let us solve the equation for $\Sigma_{YY|k,\theta^{(l)}}$ in (6.35) for the noisy speech covariance matrix $\Sigma_{NN}^{(l)}$. This yields:

$$\Sigma_{N_t|k,y_t,\theta^{(l)}} = C_k^{-1} \Sigma_{YY|k,\theta^{(l)}} C_k^{-T} - C_k^{-1} B_k \Sigma_{XX|k} B_k^T C_k^{-T}.$$

Comparing this equation to the above approximation of $\Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1}$, we find that Li et al. [177, 190] assume that $C_k^{-1} B_k \Sigma_{XX|k} B_k^T C_k^{-T}$ is 0 in order to get:

$$\Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1} \approx \left(C_k^{-1} \Sigma_{YY|k,\theta^{(l)}} C_k^{-T} \right)^{-1} = C_k^T \Sigma_{YY|k,\theta^{(l)}}^{-1} C_k.$$

This avoids the stability issues associated with the inversion of C_k. But it causes an overestimation of the noise covariance matrix, as the $\Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1} \approx C_k^T \Sigma_{YY|k,\theta^{(l)}}^{-1} C_k$ are underestimated in regions where speech dominates (due to $C_k \approx 0$ and, hence, $\Sigma_{N_t|k,y_t,\theta^{(l)}}^{-1} \approx 0$). The estimate in this work instead "backs off" to the prior noise distribution if speech is dominant – just verify that (6.43) and (6.44) reduce to $\mu_N^{(l)}$ and $\Sigma_{NN}^{(l)}$, respectively, if $C_k \approx 0$.

6.4.4 Estimation with the Unscented Transform

Instead of using the vector Taylor series approximation from Section 6.4.2.2, the joint distribution of noise and noisy speech $p_{N,Y|k,\theta^{(l)}}$ can also be obtained with the unscented transform [15]. This is achieved[8] by transforming $p_{N,X|k,\theta^{(l)}}$ according to the augmented interaction function $\tilde{f}$ from (6.33). Subsequently, the observation likelihood $p_{Y_t|k,\theta^{(l)}}$ and the instantaneous noise distribution $p_{N_t|y_t,k,\theta^{(l)}}$ are obtained as marginal and conditional distributions of $p_{N_t,Y_t|k,\theta^{(l)}} = p_{N,Y|k,\theta^{(l)}}$, in analogy to (6.36) and (6.37); and the total noise estimate $\theta^{(l)}$ is updated according to Section 6.4.2.3. All these steps are again summarized in Algorithm 6.2. The resulting noise estimates are accurate up to the second order term of the Taylor series expansion [15], just as is the unscented transform [52].

[8] see the generalized derivation of EM-based noise estimation, which has been developed in the framework of this thesis [15, 16] (and Sections 6.4.1 and 6.4.2.1).

EM-Based Noise Estimation with the Unscented Transform

1. **Preparation**: For each clean speech mode $X|k$, $k=1,\dots,\kappa$, build the joint distribution of noise N and clean speech $X|k$ according to:

$$p_{N,X|k,\theta^{(l)}}(n,x)=\mathcal{N}\left(\begin{bmatrix}n\\x\end{bmatrix};\begin{bmatrix}\mu_N^{(l)}\\\mu_{X|k}\end{bmatrix},\begin{bmatrix}\Sigma_{NN}^{(l)} & 0\\0 & \Sigma_{XX|k}\end{bmatrix}\right).$$

2. **Constructing the Joint**: Construct the joint distribution of noise and noisy speech by transforming each $p_{N,X|k,\theta^{(l)}}(n,x)$ according to the augmented interaction function:

$$\begin{bmatrix}N\\Y|k\end{bmatrix}=\tilde{f}\left(\begin{bmatrix}N\\X|k\end{bmatrix}\right)\quad\text{with}\quad\tilde{f}\left(\begin{bmatrix}n\\x\end{bmatrix}\right)=\begin{bmatrix}n\\f(x,0,n)\end{bmatrix}.$$

   This is achieved with the augmented unscented transform from Section 3.5.2; and it results in the following Gaussian mixture approximation:

$$p_{N,Y|\theta^{(l)}}(n,y)=\sum_{k=1}^{\kappa}c_k\ \mathcal{N}\left(\begin{bmatrix}n\\y\end{bmatrix};\begin{bmatrix}\mu_N^{(l)}\\\mu_{Y|k,\theta^{(l)}}\end{bmatrix},\begin{bmatrix}\Sigma_{NN}^{(l)} & \Sigma_{NY|k,\theta^{(l)}}\\\Sigma_{YN|k,\theta^{(l)}} & \Sigma_{YY|k,\theta^{(l)}}\end{bmatrix}\right).$$

3. **Mode Probabilities**: Calculate the posterior probability $p_{K|y_t}(k)$ of each mode $p_{X,Y|k,\theta^{(l)}}$ according to:

$$p_{K|y_t,\theta^{(l)}}(k)=\frac{c_k\mathcal{N}\left(y_t;\mu_{Y|k,\theta^{(l)}},\Sigma_{YY|k,\theta^{(l)}}\right)}{\sum_{k'=1}^{\kappa}c_{k'}\mathcal{N}\left(y_t;\mu_{Y|k',\theta^{(l)}},\Sigma_{YY|k',\theta^{(l)}}\right)}.$$

4. **Noise Re-estimation**: Calculate the means $\mu_{N_t|k,y_t,\theta^{(l)}}$ and covariance matrices $\Sigma_{N_t|k,y_t,\theta^{(l)}}$ of the instantaneous noise distribution:

$$\begin{aligned}\mu_{N_t|k,y_t,\theta^{(l)}} &= \mu_N^{(l)}+\Sigma_{NY|k,\theta^{(l)}}\Sigma_{YY|k,\theta^{(l)}}^{-1}(y_t-\mu_{Y|k,\theta^{(l)}}),\\ \Sigma_{N_t|k,y_t,\theta^{(l)}} &= \Sigma_{NN}^{(l)}-\Sigma_{NY|k,\theta^{(l)}}\Sigma_{YY|k,\theta^{(l)}}^{-1}\Sigma_{YN|k,\theta^{(l)}}\end{aligned}$$

   for $k=1,\dots,\kappa$. Then update the noise parameters $\theta^{(l+1)}=\left\{\mu_N^{(l+1)},\Sigma_{NN}^{(l+1)}\right\}$ for the next iteration according to

$$\begin{aligned}\mu_N^{(l+1)} &= \frac{1}{\tau}\sum_{t=1}^{\tau}\sum_{k=1}^{\kappa}p_{K|y_t,\theta^{(l)}}(k)\mu_{N_t|k,y_t,\theta^{(l)}}\\ \Sigma_{NN}^{(l+1)} &= \frac{1}{\tau}\sum_{t=1}^{\tau}\sum_{k=1}^{\kappa}p_{K|y_t,\theta^{(l)}}(k)\Big(\Sigma_{N_t|k,y_t,\theta^{(l)}}+\mu_{N_t|k,y_t,\theta^{(l)}}\mu_{N_t|k,y_t,\theta^{(l)}}^T-\mu_N^{(l+1)}\mu_N^{(l+1)^T}\Big).\end{aligned}$$

Algorithm 6.2: EM-Based Noise Estimation with the Unscented Transform

6.4.5 Monte Carlo Approximation

As an alternative to the Gaussian mixture approximations from Section 6.4.2, the instantaneous noise distribution $p_{N_t|y_t,\theta^{(l)}}$ can be approximated through samples [16]. This idea can be formulated more concisely as (1) drawing samples $n_t^{(j)}$, $j = 1,\ldots,L$, from the prior distribution $p_{N_t|\theta^{(l)}} = p_{N|\theta^{(l)}}$ of the noise, and then (2) converting these prior samples to posterior samples by weighting them with the appropriate importance[9] weights $\omega_t^{(j)} = p_{N_t|y_t,\theta^{(l)}}\left(n_t^{(j)}\right)/p_{N_t|\theta^{(l)}}\left(n_t^{(j)}\right)$. The results is a weighted empirical approximation of the instantaneous noise distribution:

$$\tilde{p}_{N_t|y_t,\theta^{(l)}}(n_t) = \sum_{j=1}^{L} \tilde{\omega}_t^{(j)} \delta\left(n_t - n_t^{(j)}\right) \quad \text{with} \quad \tilde{\omega}_t^{(j)} \triangleq \frac{\omega_t^{(j)}}{\sum_{i=1}^{L} \omega_t^{(i)}},$$

Plugging $\tilde{p}_{N_t|y_t,\theta^{(l)}}(n_t)$ into (6.27) and (6.28) gives a Monte Carlo approximation of the next noise parameter estimate $\theta^{(l+1)} = \left\{\mu_N^{(l+1)}, \Sigma_{NN}^{(l+1)}\right\}$ where $\mu_N^{(l+1)}$ and $\Sigma_{NN}^{(l+1)}$ are calculated according to:

$$\mu_N^{(l+1)} = \frac{1}{\tau} \sum_{t=1}^{\tau} \sum_{j=1}^{L} \tilde{\omega}_t^{(j)} n_t^{(j)}, \tag{6.54}$$

$$\Sigma_{NN}^{(l+1)} = \frac{1}{\tau} \left(\sum_{t=1}^{\tau} \sum_{j=1}^{L} \tilde{\omega}_t^{(j)} n_t^{(j)} \left(n_t^{(j)}\right)^T \right) - \mu_N^{(l+1)} \left(\mu_N^{(l+1)}\right)^T. \tag{6.55}$$

The advantage of this Monte Carlo approach is that is shifts the whole complexity of EM-based noise estimation to the calculation of importance weights. In particular, it avoids the local Gaussian fits of both the vector Taylor series expansion and the unscented transform. To proceed with the weight calculation, let us rewrite $p_{N_t|y_t,\theta^{(l)}}$ through use of Bayes rule:

$$p_{N_t|y_t,\theta^{(l)}}\left(n_t^{(j)}\right) = \frac{p_{Y_t,N_t|\theta^{(l)}}\left(y_t, n_t^{(j)}\right)}{p_{Y_t|\theta^{(l)}}(y_t)} = \frac{p_{Y_t|n_t^{(j)}}(y_t) p_{N_t|\theta^{(l)}}\left(n_t^{(j)}\right)}{\int p_{Y_t|n_t}(y_t) p_{N_t|\theta^{(l)}}(n_t)\, dn_t}.$$

Substituting this expansion back into $\omega_t^{(j)} = p_{N_t|y_t,\theta^{(l)}}\left(n_t^{(j)}\right)/p_{N_t|\theta^{(l)}}\left(n_t^{(j)}\right)$ and further using a remark in [191] that $\frac{1}{L}\sum_{i=1}^{L} p_{Y_t|n_t^{(i)}}(y_t)$ provides a consistent estimator for the denominator $p_{Y_t|\theta^{(l)}}(y_t)$ in Bayes equation, it is found that the $\omega_t^{(j)}$ can be approximated as:

$$\omega_t^{(j)} = \frac{p_{Y_t|n_t^{(j)}}(y_t)}{\int p_{Y_t|n_t}(y_t) p_{N_t|\theta^{(l)}}(n_t)\, dn_t} \approx \frac{p_{Y_t|n_t^{(j)}}(y_t)}{\frac{1}{L}\sum_{i=1}^{L} p_{Y_t|n_t^{(i)}}(y_t)}$$

Normalization of the weights finally yields [16]:

$$\tilde{\omega}_t^{(j)} \approx \frac{p_{Y_t|n_t^{(j)}}(y_t)}{\sum_{i=1}^{L} p_{Y_t|n_t^{(i)}}(y_t)}. \tag{6.56}$$

[9] see Section 2.5.4 on importance sampling

EM-Based Noise Estimation with the Monte Carlo Method

1. **Sample**: For each time $t \in \{1,\ldots,\tau\}$, draw L noise samples $n_t^{(j)}$ from the prior noise distribution $p_{N|\theta^{(l)}}$ and M clean speech samples $x_t^{(i)}$ from the Gaussian mixture clean speech distribution p_X from (6.29):

$$\begin{aligned} n_t^{(j)} &\sim \mathcal{N}\left(n;\mu_N^{(l)},\Sigma_{NN}^{(l)}\right), & j=1,\ldots,L,\\ x_t^{(i)} &\sim \sum_{k=1}^{K} c_k \mathcal{N}\left(x;\mu_{X|k},\Sigma_{XX|k}\right), & i=1,\ldots,M. \end{aligned}$$

2. **Simulate Noisy Speech**: Predict the empirical distribution of noisy speech by transforming all pairs $\left(x_t^{(i)}, n_t^{(j)}\right)$ of speech/noise samples according to the interaction function f. For the zero-phase factor model from Section 5.2.4 this gives:

$$y_t^{(i,j)} = \log\left(\exp\left(x_t^{(i)}\right) + \exp\left(n_t^{(j)}\right)\right).$$

   for $i=1,\ldots,M$, $j=1,\ldots,L$.

3. **Calculate Importance Weights**: Approximate the normalized importance weight $\omega_t^{(j)}$ of each noise sample $n_t^{(j)}$ according to:

$$\tilde{\omega}_t^{(j)} \approx \frac{p_{Y_t|n_t^{(j)}}(y_t)}{\sum_{l=1}^{L} p_{Y_t|n_t^{(l)}}(y_t)} \quad \text{with} \quad p_{Y_t|n_t^{(j)}}(y_t) \approx \frac{1}{M}\sum_{i=1}^{M}\mathcal{N}\left(y_t;y_t^{(i,j)},\Sigma\right).$$

4. **Update Noise Estimate**: Update the noise parameter estimate $\theta^{(l+1)} = \left\{\mu_N^{(l+1)},\Sigma_{NN}^{(l+1)}\right\}$ by calculating $\mu_N^{(l+1)}$ and $\Sigma_{NN}^{(l+1)}$ according to:

$$\begin{aligned} \mu_N^{(l+1)} &= \frac{1}{\tau}\sum_{t=1}^{\tau}\sum_{j=1}^{L}\tilde{\omega}_t^{(j)} n_t^{(j)},\\ \Sigma_{NN}^{(l+1)} &= \frac{1}{\tau}\left(\sum_{t=1}^{\tau}\sum_{j=1}^{L}\tilde{\omega}_t^{(j)} n_t^{(j)}\left(n_t^{(j)}\right)^T\right) - \mu_N^{(l+1)}\left(\mu_N^{(l+1)}\right)^T. \end{aligned}$$

Algorithm 6.3: EM-Based Noise Estimation with the Monte Carlo Method

In order to calculate these weights, it is still necessary to determine the noise sample likelihoods $p_{Y_t|n_t^{(j)}}(y_t)$ under the observation y_t. For that, let us first draw M clean speech samples $\left\{x_t^{(1)},\ldots,x_t^{(M)}\right\}$ from a previously learned clean speech distribution and then transform all pairs $\left(x_t^{(i)},n_t^{(j)}\right)$ of speech/noise samples according to the interaction function in the log-Mel domain, e.g. the zero-phase factor model from Section 5.2.4:

$$y_t^{(i,j)} = f\left(x_t^{(i)},0,n_t^{(j)}\right) = \log\left(\exp\left(x_t^{(i)}\right)+\exp\left(n_t^{(j)}\right)\right).$$

Then, the set $\left\{y_t^{(i,j)}, i=1,\ldots,M\right\}$ simulates how noisy speech would look like under the j-th noise sample. Consequently, Parzen-window density estimation [192] can be applied, by approximating $p_{Y_t|n_t^{(i)}}$ with Gaussian kernels around the $y_t^{(i,j)}$. This allows for the noise sample likelihoods to be evaluated according to:

$$p_{Y_t|n_t^{(j)}}(y_t) \approx \frac{1}{M}\sum_{i=1}^{M}\mathcal{N}(y_t; y_t^{(i,j)},\Sigma). \tag{6.57}$$

Following [16], the Kernel covariance matrix Σ is considered to be radial: $\Sigma = \alpha\cdot I$ where $\alpha \in \mathbb{R}$ and where I denotes the identity matrix. With this, the noise parameter estimate $\theta^{(l+1)} = \left\{\mu_N^{(l+1)},\Sigma_{NN}^{(l+1)}\right\}$ can now be updated according to (6.54) and (6.55).

6.5 Particle Filter Based Noise Tracking

In the expectation maximization approach from the previous Section, the distribution of noise is estimated from a longer sequence of noisy speech features (such as a complete utterance of 5-10 seconds). This guarantees a good accuracy of estimation. But it also leads to a large noise variance if the noise is changing during that time. This triggered research efforts towards non-stationary noise estimation, i.e. methods which "track" the noise, such as Kim's sequential expectation maximization [123] and interaction multiple model (IMM) [184] approaches, Yao [193] and Raj's [127] particle filter approach as well as several extensions thereof – see e.g. the work of Fujimoto [194, 195], Haeb-Umbach [133], Faubel and Wölfel [136, 134, 196, 197, 198]. As [137] gives a thorough introduction to Raj's work, this section just briefly restates that approach before it is further extended in Section 6.5.3. The main idea is to approximate the instantaneous noise distribution $p_{N_t|y_{1:t}}(n_t)$ by a weighted empirical distribution:

$$\tilde{p}_{N_t|y_{1:t}}(n_t) = \sum_{j=1}^{L}\tilde{\omega}_t^{(j)}\delta\left(n_t - n_t^{(j)}\right) \tag{6.58}$$

where the $n_t^{(j)}$ constitute possible noise hypotheses at time t and where the $\tilde{\omega}_t^{(j)}$ are their relative (i.e. normalized) likelihoods under all past observations $y_{1:t} = \left\{y_1,\ldots,y_t\right\}$. This distribution is propagated forward in time by (1) sampling noise hypotheses $n_{t+1}^{(j)}$ for time $t+1$, as described in Section 6.5.1; (2) evaluating their observation likelihoods $p_{Y_{t+1}|n_{t+1}^{(j)}}(y_{t+1})$ as described in Section 6.5.2 and then (3) updating the weights $\tilde{\omega}_{t+1}^{(j)}$ according to:

$$\tilde{\omega}_{t+1}^{(j)} = \frac{p_{Y_{t+1}|n_{t+1}^{(j)}}(y_{t+1})}{\sum_{j'=1}^{N} p_{Y_{t+1}|n_{t+1}^{(j')}}(y_{t+1})}. \tag{6.59}$$

The minimum mean square error (MMSE) estimate of clean speech is subsequently obtained by Monte Carlo integration [137, 136]:

$$\hat{x}_{t+1} = \sum_{j=1}^{L} \omega_{t+1}^{(j)} \underbrace{g\left(y_{t+1}, 0, n_{t+1}^{(j)}\right)}_{=\hat{x}_{t+1}|y_{t+1}, n_{t+1}^{(j)}} \tag{6.60}$$

where g denotes the inverse interaction function from Section 5.3.3. Section 6.5.3 shows how this approach can be combined with the phase-averaged model from Section 5.3.2.

6.5.1 A Process Model for the Evolution of Noise Spectra

Following Raj et al. [127], the evolution of log-Mel noise spectra is modeled as a first-order autoregressive process that is exited by correlated Gaussian [197] noise:

$$n_{t+1} = f(n_t, \epsilon_t) = A n_t + \epsilon_t. \tag{6.61}$$

In this equation, n_t is the d_m-dimensional noise spectrum at time t, A is a $d_m \times d_m$ dimensional linear prediction matrix and $\epsilon_t \sim \mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ denotes excitation noise. The ϵ_t are assumed to be statistically independent of each other. Hence, the noise samples $\left\{n_t^{(1)}, \ldots, n_t^{(L)}\right\}$ can be propagated to time $t+1$ by drawing L excitation noise samples $\epsilon_t^{(j)}$ from $\mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ and then updating the noise samples $n_t^{(j)}$ according to

$$n_{t+1}^{(j)} = f\left(n_t^{(j)}, \epsilon_t^{(j)}\right). \tag{6.62}$$

In order to learn the parameters of the above noise model, the linear prediction matrix A is typically estimated in a minimum mean square error fashion, as described in [127]:

$$\hat{A} = \left(\sum_{t=2}^{\tau} n_t n_{t-1}^T\right)\left(\sum_{t=2}^{\tau} n_{t-1} n_{t-1}^T\right)^{-1}, \tag{6.63}$$

whereas the excitation noise parameters $\theta = \left\{\mu_\epsilon, \Sigma_\epsilon\right\}$ are estimated with the maximum likelihood method:

$$\hat{\mu}_\epsilon = \frac{1}{\tau-1}\sum_{t=2}^{\tau}(n_t - A n_{t-1}), \quad \hat{\Sigma}_\epsilon = \frac{1}{\tau-1}\sum_{t=2}^{\tau}(n_t - A n_{t-1})(n_t - A n_{t-1})^T - \hat{\mu}_\epsilon \hat{\mu}_\epsilon^T. \tag{6.64}$$

These parameters can also be estimated in an online fashion, as described in [198].

6.5.2 Calculating the Observation Likelihoods

Next to the propagation of noise samples, particle filter based noise tracking approaches require the calculation of observation likelihoods $p_{Y_t|n_t^{(j)}}(y_t)$. In Raj et al.'s approach [127], that is achieved by using the interaction function from Section 5.2.4 as a measurement equation:

$$y_t = h(x_t, n_t) \triangleq \log\left(e^{x_t} + e^{n_t}\right)$$

where n_t denotes the noise spectrum to be tracked and where the clean speech spectrum x_t is "measurement noise". Then, $p_{Y_t|n_t^{(j)}}$ can be obtained with the fundamental transformation law of probability by transforming the clean speech variable X according to $Y_t|n_t^{(j)} = h\left(X, n_t^{(j)}\right)$:

$$p_{Y_t|n_t^{(j)}}(y_t) = p_X\left(h^{-1}\left(y_t, n_t^{(j)}\right)\right)\left|\det\left(\frac{dh^{-1}\left(y_t, n_t^{(j)}\right)}{dy_t}\right)\right|. \tag{6.65}$$

In this equation, $h^{-1}\left(y_t, n_t\right) = g\left(y_t, 0, n_t^{(j)}\right)$ denotes the inverse interaction function from Section 5.3.3, whose Jacobian $dg\left(y_t, 0, n_t^{(j)}\right)/dy_t$ with respect to y_t has been given in Section 5.3.5. Further substituting (5.20) and (5.28) into (6.65), it becomes clear that the observation likelihood can be evaluated according to:

$$p_{Y_t|n_t^{(j)}}(y_t) = \frac{p_X\left(y_t + \log\left(\mathbf{1} - e^{n_t^{(j)} - y_t}\right)\right)}{\prod_{k=1}^{d_m}\left(1 - e^{n_{t,k}^{(j)} - y_{t,k}}\right)} \tag{6.66}$$

where k denotes the log-Mel bin and where the log operation in (6.66) is performed component-wise, on the elements of the vector. As the inverse zero-phase factor model assumes that speech and noise are strictly additive (see Sections 5.3.3 and 5.3.5), $p_{Y_t|n_t^{(j)}}(y_t)$ is set to 0 if $n_{t,k}^{(j)}$ exceeds $y_{t,k}$ in any of the spectral bins. This is justified by the fact that $n_{t,k}^{(j)} > y_{t,k}$ is physically impossible[10] according to (6.5.2), which translates to zero likelihood [133, 134]. The clean speech distribution p_X, which is required for evaluating (6.66), is typically modeled as a Gaussian mixture distribution whose parameters have been trained on a clean speech training corpus:

$$p_X(x) = \sum_{k=1}^{K} c_k \mathcal{N}\left(x; \mu_{X|k}, \Sigma_{XX|k}\right).$$

Following [137, 136, 134, 196], this work uses diagonal covariance matrices.

6.5.3 Combination with the Phase-Averaged Model

The zero-phase factor model from the outset excludes attenuation and cancellation of the clean speech spectrum. Consequently, it assigns a weight of zero if a noise hypothesis exceeds the

[10] that is because $y_t = \log(e^{x_t} + e^{n_t}) \geq \log(e^{n_t}) = n_t$ due to the monotonicity of the logarithm.

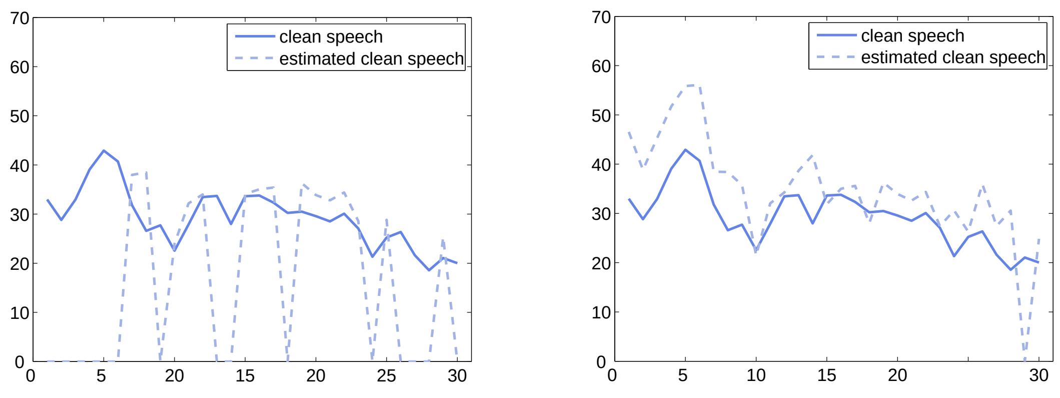
observed noisy speech spectrum (see Section 5.2). This can lead to a severe decimation of the particle (i.e. noise hypothesis) population, up to its complete annihilation if all weights are zero [133, 137]. In [134], it was found that the latter happens in about 4-6% of all cases. But even if it does not come to that, the decimation causes a considerable degradation of the tracking performance. Previous work tried to ameliorate this problem by introducing a reinitialization procedure [137] as well as a fast acceptance test (FAT) [134]. This thesis avoids the problem from the outset. That is achieved by using the phase-averaged model from Section 5.2.3, for which the observation likelihoods $p_{Y_t|n_t^{(j)}}(y_t)$ are calculated as follows [14]:

$$p_{Y_t|n_t^{(j)}}(y_t) = p_X\left(y_t + \mathscr{E}_{p_A(\alpha)}\left\{\tilde{g}^{\pm}(n_t^{(j)} - y_t, \alpha)\right\}\right) \cdot \det J\left(y_t, 0, n_t^{(j)}\right). \tag{6.67}$$

Here, $\left(y_t + \mathscr{E}_{p_A(\alpha)}\left\{\tilde{g}^{\pm}(n_t - y_t, \alpha)\right\}\right)$ denotes the component-wise evaluation of the MMSE clean speech estimate from (5.19). The Jacobian determinant

$$\det J\left(y_t, 0, n_t^{(j)}\right) = \prod_{k=1}^{d_m} \mathscr{E}_{p_{A_k}(\alpha_k)}\left\{\gamma_k^{\pm}(n_{t,k}^{(j)} - y_{t,k}, \alpha_k)\right\} \beta_k\left(n_{t,k}^{(j)} - y_{t,k}\right)$$

is calculated according to (5.24), (5.25), (5.26). The main advantage of the phase-averaged model is that the Jacobian determinant does not immediately drop to zero when a noise hypothesis exceeds the observed noisy speech spectrum. This effect can be seen in Figure 5.15. Another advantage consists in the fact that the phase-averaged model is the minimum mean square error solution. Hence, the above approach also gives more accurate clean speech estimates, as illustrated in Figure 6.4. A more detailed experimental analysis can be found in [14].



(a) zero-phase factor model *(b) phase averaged model*

Figure 6.4: *True and estimated clean speech spectra for the zero-phase factor as well as the phase-averaged model from Sections 5.3.3 and 5.3.2, respectively, at an SNR of 0 dB. Note that the figure shows an extreme example in which most of the speech spectrum is covered by noise.*

Particle Filter Based Noise Tracking

1. **Propagate Noise Hypotheses**: For each noise hypothesis $n_{t-1}^{(j)}$ from time $t-1$, draw a process noise sample $\epsilon_t^{(j)}$ from $\mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ and then predict the noise sample $n_t^{(j)}$ at time t according to:
$$n_t^{(j)} = A n_{t-1}^{(j)} + \epsilon_t^{(j)}$$
where A is the linear prediction matrix from Section 6.5.1.

2. **Calculate Weights**: For each noise hypothesis $n_t^{(j)}$, calculate the observation likelihood $p_{Y_t|n_t^{(j)}}(y_t)$ according to (6.66) or (6.67). In case (6.66) is used, $p_{Y_t|n_t^{(j)}}(y_t)$ is set to zero if $n_{t,k}^{(j)} > y_{t,k}$ for any bin $k \in \{1, \dots, d_m\}$. After calculating the observation likelihoods, the normalized weights $\tilde{\omega}_t^{(j)}$ are obtained as:
$$\tilde{\omega}_t^{(j)} = \frac{p_{Y_t|n_t^{(j)}}(y_t)}{\sum_{j'=1}^{N} p_{Y_t|n_t^{(j')}}(y_t)}.$$

3. **Estimate Clean Speech**: Calculate the minimum mean square error estimate of clean speech according to (6.60):
$$\hat{x}_t = \sum_{j=1}^{L} \omega_t^{(j)} g\left(y_t, 0, n_t^{(j)}\right),$$
where g is the inverse interaction function from (5.20) or (5.19), respectively.

4. **Resample**: Prune the noise hypotheses by importance resampling. This essentially multiplies hypotheses that have a high relative weight and it removes hypotheses that have a low relative weight. The result is an (equally-weighted) empirical distribution:
$$\hat{p}_{N_t|y_{1:t}}(n_t) = \frac{1}{L} \sum_{i=1}^{L} \delta\left(n_t - \tilde{n}_t^{(i)}\right),$$
where $\tilde{n}_t^{(i)}$ denotes a sample drawn from $n_t^{(j)}$ with probability $\tilde{\omega}_t^{(j)}$, $j = 1, \dots L$.

Algorithm 6.4: Particle Filter Based Noise Tracking

6.6 On the Use of Log-Mel Features

Regarding the use of log-Mel features in this thesis it should be emphasized that speech feature enhancement in the log-Mel domain [45, 175, 76, 119] is more exact than working with MFCC features [121, 176]. That is because it avoids further smoothing of the spectrum through the pseudo-inverse of the DCT [119]. In particular, it has no disadvantages compared to MFCC features – at least not if the enhancement is integrated into the feature extraction chain, as shown in Figure 6.5.

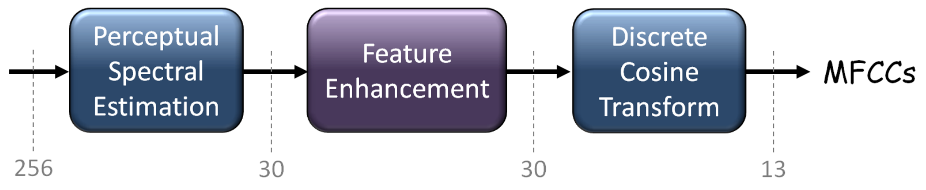


Figure 6.5: *Integration of log-Mel feature enhancement into the MFCC feature extraction chain. The resulting MFCC features may be further processed as indicated in Figure 5.7.*

6.7 Contributions of this Chapter

The following list again gives an overview of the individual contributions of this thesis to speech feature enhancement:

1. A generalized derivation of expectation maximization (EM) based noise estimation [15, 16], including theoretical comparisons to the approaches by Kim [124] as well as Li et al. [177, 190] (Section 6.4).
2. An algorithm for EM-based noise estimation with the unscented transform (Section 6.4.4) [15].
3. A Monte Carlo variant of the EM algorithm for estimating noise from noisy speech features (Section 6.4.5) [16].
4. A noise-tracking particle filter that uses the phase-averaged model in order to avoid stability issues due to the relative phase [14] (Section 6.5).

7 Missing Feature Reconstruction

The clean speech estimates of the particle filter approach from Section 6.5 can strong vary for adjacent frames. This becomes particularly pronounced in spectral valleys and it can be explained by the fact that inverse models fail to give accurate estimates if the noise power is comparable to that of speech (see Sections 5.3 and 6.5.3 and, in particular, Figure 6.4). In cases where the noise power is even stronger, the noise spectrum might actually mask parts of the clean speech spectrum, as shown in Figure 7.1.

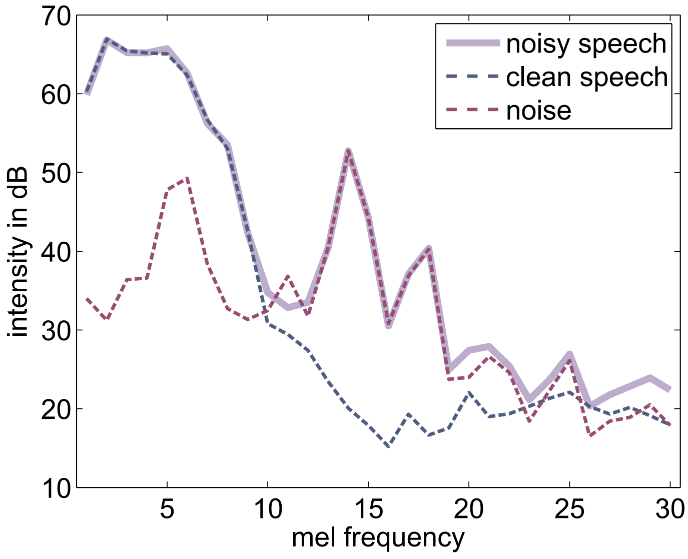


Figure 7.1: *Masking - in regions where the noise is over 10dB louder than speech (bins 13-18), the observed noisy speech spectrum is essentially independent of the clean speech spectrum.*

Conversely, noise has practically no effect on the observed spectrum if the speech power is 10 dB stronger than that of the noise. This means, in the limit, the effect of noise to log-Mel speech features can be approximated[1] by the log-max model:

$$y_i = \max(x_i, n_i), \tag{7.1}$$

where y denotes a noisy speech spectrum and where x and n denote corresponding clean speech and noise spectra in the log-Mel domain. To discuss (7.1) in more detail, let us have a look at Figure 7.2, which again illustrates the effect of masking for a single frequency bin.

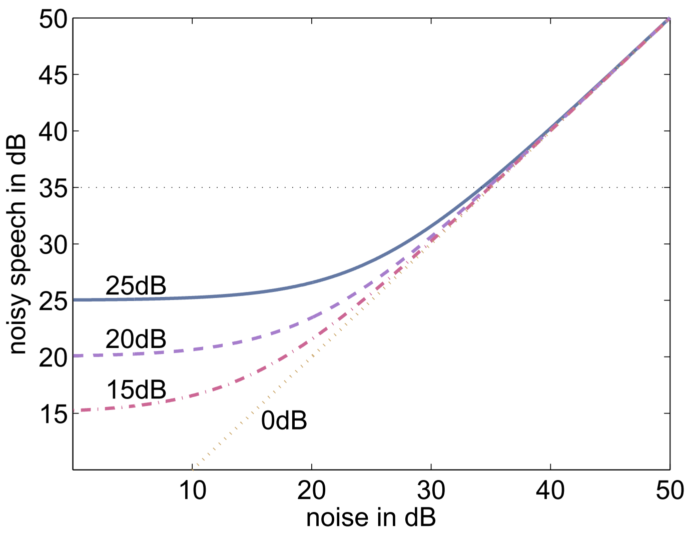


Figure 7.2: *Effect of additive noise to a clean speech spectral bin of 25 (solid curve), 20 (dashed curve), 15 (dotted dashed curve) and 0 dB (dotted curve), according to the zero-phase factor model from Section 5.2.4.*

The four curves show how clean speech at a power of 0, 15, 20 and 25 dB, respectively, is affected by additive noise of increasing intensity, starting with 0dB on the left and going up to 50dB on the right. At an observed power of 35 dB – marked by a dotted horizontal line – the curves are very close. Thus, a slight misestimation of the noise power, and be it as low as 1 dB, will cause the clean speech estimate to vary between 0 and 25 dB. This led Cooke, Morris and Green [199, 200] to consider the masked part missing and to then apply missing data theory in order to impute the masked spectral bins from non-masked ones. For this, the clean speech spectrum x (in the log-Mel domain) was reordered to form a partitioning

$$x = \left[x_m^T \;\; x_o^T \right]^T \tag{7.2}$$

with a masked part x_m and an observable part x_o. This reordering of x is essentially a permutation which is obtained as shown in Figure 7.3.

[1] Note that (7.1) is in particular justified in lower frequencies where the phase-averaged model is very close to the log-max approximation (see the discussion at the end of Section 5.2.4).

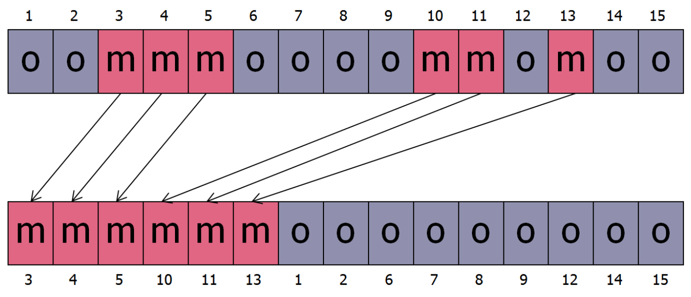


Figure 7.3: *Reordering the spectral bins partitions the clean speech spectrum into a missing and observable part. An "m" denotes a masked spectral bin while an "o" denotes an observable one.*

As the masking changes in time, the reordering needs to be done dynamically (once for each frame), based on the current "mask" $\theta = [\theta_1 \quad \cdots \quad \theta_d]$ whose components θ_i identify which bins are subject to masking:

$$\theta_i = \begin{cases} 1, & x_i \text{ is masked} \\ 0, & x_i \text{ is observable} \end{cases} \tag{7.3}$$

After reordering the bins, the masked part can be reconstructed as described in Section 7.1. The remaining part of the chapter consists of a theoretical comparison to Bayesian speech feature enhancement (Section 7.2) as well as the introduction of bounded estimates in Section 7.3. Section 7.4 finally presents a simple mask estimation technique, based on the noise tracking approach from Section 6.5.

7.1 Classical Missing Data Theory

Methods for the treatment of incomplete or missing data problems historically arose in the area of statistical data analysis [201, 202], out of the necessity to handle incompletely answered questionnaires or due to a general unavailability of data during the analysis of surveys and experimental studies. With the advancement of pattern recognition in the early 1990s, missing data techniques became of interest in computer vision [203] and automatic speech recognition [204], for handling visual occlusion as well as masking of speech by noise. This led to the emergence of two principally different approaches:

1. **Marginalization Techniques** that treat the missing data problem during classification by marginalizing over the missing portions of the feature vector [130, 131, 203, 205].

2. **Imputation Techniques** that estimate the missing portions of the feature vector prior to classification [199, 200, 206, 207].

This chapter focuses on the latter, with the aim of "reconstructing" the masked part x_m of a clean speech spectrum given the observable part x_o. Following [199, 200], this goal is here achieved with the method of conditional mean imputation.

7.1.1 Conditional Mean Imputation

Conditional mean imputation (CMI) originally refers to a work by Buck [208, 201, 202] in which the missing portion of the data was estimated based on linear regression. Motivated by the fact that this approach can be shown to minimize the mean square error if the data has a Gaussian distribution, the term "conditional mean imputation" is here used in a generalized sense for the minimum mean square error (MMSE) estimate. In order to derive this estimate, let us reformulate missing data imputation as a Bayesian estimation problem (see Chapter 3) where x_m denotes the hidden state to be estimated and where x_o denotes the corresponding observation. Then making use of (3.1), it turns out that the MMSE solution consists in calculating the conditional mean:

$$\widehat{x}_m = \delta_{MMSE}(x_o) = \mathcal{E}_{p_{X_m|x_o}(x_m)}\{x_m\}. \tag{7.4}$$

If we further assume that the joint distribution p_{X_m,X_o} of the observable and missing parts of the spectrum can be approximated as a Gaussian mixture,

$$p_{X_m,X_o}(x_m,x_o) = \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}\left(\begin{bmatrix} x_m \\ x_o \end{bmatrix}; \begin{bmatrix} \mu_{X_m|k} \\ \mu_{X_o|k} \end{bmatrix}, \begin{bmatrix} \Sigma_{X_mX_m|k} & \Sigma_{X_mX_o|k} \\ \Sigma_{X_oX_m|k} & \Sigma_{X_oX_o|k} \end{bmatrix}\right)}_{=p_{X_m,X_o|k}(x_m,x_o)}, \tag{7.5}$$

then (7.4) can be calculated according to Section 3.2.2 by simply conditioning p_{X_m,X_o} on $X_o = x_o$. This gives the solution presented by Cooke, Morris and Green [199, 200]:

$$\widehat{x}_m = \sum_{k=1}^{K} c_k^+ \mu_{X_m|x_o,k} \tag{7.6}$$

where $\mu_{X_m|x_o,k}$ denotes the conditional mean $\mu_{X_m|x_o,k} = \mu_{X_m|k} + \Sigma_{X_mX_o|k}\Sigma^{-1}_{X_oX_o|k}\left(x_o - \mu_{X_o|k}\right)$ of the k-th Gaussian component and where c_k^+ denotes the corresponding posterior probability:

$$c_k^+ = \frac{c_k \mathcal{N}\left(x_o; \mu_{X_o|k}, \Sigma_{X_oX_o|k}\right)}{\sum_{k'=1}^{K} \mathcal{N} c_{k'}\left(x_o; \mu_{X_o|k'}, \Sigma_{X_oX_o|k'}\right)}. \tag{7.7}$$

7.1.2 Mean Imputation

If the missing part of the data is assumed to be statistically independent of the observable part then the conditional distribution of $p_{X_m|x_o}$ reduces to p_{X_m}. In this case, the conditional mean $\mathcal{E}_{p_{X_m|x_o}}\{x_m\}$ consists only of the mean $\mathcal{E}_{p_{X_m}(x_m)}\{x_m\} = \mu_{X_m}$ of the missing part and the estimate from (7.6) reduces to:

$$\widehat{x}_m = \sum_{k=1}^{K} c_k^+ \mu_{X_m}. \tag{7.8}$$

This particularly simple estimate is generally referred to as mean imputation [200].

7.1.3 Partitioning the Mean and Covariance Matrices

Mean imputation and conditional mean imputation rely on the partitioned Gaussian mixture from (7.5). This partitioned distribution can be obtained from a general (i.e. non-partitioned) Gaussian mixture model of clean speech,

$$p_X(x) = \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}\left(x; \mu_{X|k}, \Sigma_{XX|k}\right)}_{=p_{X|k}(x)}, \tag{7.9}$$

by first reordering the means in analogy to (7.2) and then using the same reordering to permute the rows and columns of the covariance matrices [199]. This procedure is illustrated in Figures 7.4 and 7.5; and it needs to be performed for each frame.

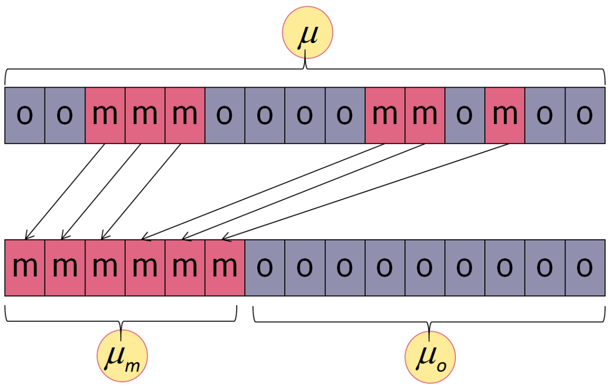
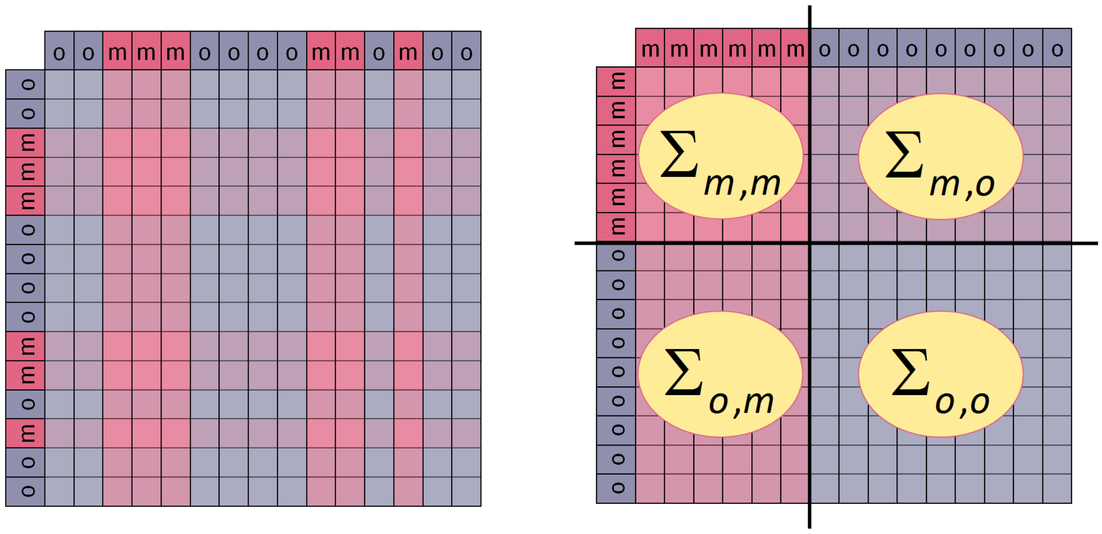


Figure 7.4: *Partitioning the Mean*



(a) original covariance matrix. *(b) partitioned covariance matrix.*

Figure 7.5: *Partitioning the Covariance Matrix*

7.2 The Relationship to Bayesian Speech Enhancement

The conditional mean imputation (CMI) estimate from (7.6) obviously looks quite similar to the MMSE solution (6.5) of Bayesian speech feature enhancement (see Section 6.3). This motivates a theoretical comparison between these two approaches. The surprising result is that in the limit, i.e. with the interaction function from Section 5.2 approaching the log-max approximation (7.1), both methods actually do give a very similar estimate. In order to show this, let us partition the clean and noisy speech spectra, x and y, in analogy to (7.2):

$$x = \begin{bmatrix} x_m \\ x_o \end{bmatrix}, \quad y = \begin{bmatrix} y_m \\ y_o \end{bmatrix} = \begin{bmatrix} n_m \\ x_o \end{bmatrix} \tag{7.10}$$

where x_o and x_m again denote the observed and missing part of the clean speech spectrum and where y_o and y_m denote the corresponding parts of the noisy speech spectrum. The term n_m denotes the part of the noise which masks the clean speech spectrum; and the equality on the right hand side of (7.10) is due to the use of the log-max model (7.1). Now, the joint distribution of X and Y can be constructed as described in Section 6.3, just with $f(x_i, 0, n_i) = \max(x_i, n_i)$, and the MMSE estimate of the k-th Gaussian component can be calculated according to (6.5):

$$\mu_{X|y,k} = \mu_{X|k} + \Sigma_{XY|k}\Sigma_{YY|k}^{-1}(y - \mu_{Y|k}). \tag{7.11}$$

This is the MMSE estimate of the Bayesian speech feature enhancement approach from Section 6.3. Now dropping the dependency on k for ease of notation, the mean μ_Y of noisy speech can be expressed as

$$\mu_Y \approx \mathscr{E}_{p_{N_m,X_o}}\left\{\begin{bmatrix} n_m \\ x_o \end{bmatrix}\right\} = \begin{bmatrix} \mu_{N_m} \\ \mu_{X_o} \end{bmatrix}. \tag{7.12}$$

Further making use of the fact that speech and noise are uncorrelated, i.e. $\Sigma_{N_m X_o} = 0$, the covariance matrix Σ_{YY} of noisy speech may be written:

$$\Sigma_{YY} \approx \mathscr{E}_{p_{N_m,X_o}}\left\{\begin{bmatrix} n_m \\ x_o \end{bmatrix}\begin{bmatrix} n_m & x_o \end{bmatrix}\right\} = \begin{bmatrix} \Sigma_{N_m N_m} & \Sigma_{N_m X_o} \\ \Sigma_{X_o N_m} & \Sigma_{X_o X_o} \end{bmatrix} = \begin{bmatrix} \Sigma_{N_m N_m} & 0 \\ 0 & \Sigma_{X_o X_o} \end{bmatrix} \tag{7.13}$$

and the cross-covariance matrix between speech and noisy speech may be calculated as:

$$\Sigma_{XY} \approx \mathscr{E}\left\{\begin{bmatrix} x_m \\ x_o \end{bmatrix}\begin{bmatrix} n_m & x_o \end{bmatrix}\right\} = \begin{bmatrix} \Sigma_{X_m N_m} & \Sigma_{X_m X_o} \\ \Sigma_{X_o N_m} & \Sigma_{X_o X_o} \end{bmatrix} = \begin{bmatrix} 0 & \Sigma_{X_m X_o} \\ 0 & \Sigma_{X_o X_o} \end{bmatrix}. \tag{7.14}$$

With these approximations, the regression term $\left(\Sigma_{XY}\Sigma_{YY}^{-1}\right)$ in (7.11) becomes:

$$\Sigma_{XY}\Sigma_{YY}^{-1} \approx \begin{bmatrix} 0 & \Sigma_{X_m X_o} \\ 0 & \Sigma_{X_o X_o} \end{bmatrix}\begin{bmatrix} \Sigma_{N_m N_m}^{-1} & 0 \\ 0 & \Sigma_{X_o X_o}^{-1} \end{bmatrix} = \begin{bmatrix} 0 & \Sigma_{X_m X_o}\Sigma_{X_o X_o}^{-1} \\ 0 & I \end{bmatrix}, \tag{7.15}$$

as a consequence of which $\mu_{X|y} = \mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(y - \mu_Y)$ evaluates to:

$$\begin{aligned}\mu_{X|y} &\approx \begin{bmatrix}\mu_{X_m}\\ \mu_{X_o}\end{bmatrix} + \begin{bmatrix}0 & \Sigma_{X_mX_o}\Sigma_{X_oX_o}^{-1}\\ 0 & I\end{bmatrix}\left(\begin{bmatrix}x_m\\ x_o\end{bmatrix} - \begin{bmatrix}\mu_{N_m}\\ \mu_{X_o}\end{bmatrix}\right)\\ &= \begin{bmatrix}\mu_{X_m} + \Sigma_{X_mX_o}\Sigma_{X_oX_o}^{-1}(x_o - \mu_{X_o})\\ x_o\end{bmatrix}. \end{aligned} \tag{7.16}$$

Comparing this result to (7.6), we find that the upper part $\mu_{X_m} + \Sigma_{X_mX_o}\Sigma_{X_oX_o}^{-1}(x_o - \mu_{X_o})$ of the vector corresponds to the conditional mean imputation estimate $\mu_{X_m|x_o}$ whereas the lower part x_o is simply the observable part of the clean speech spectrum. Hence, for a single Gaussian, the Bayesian speech feature enhancement approach from Section 6.3 approximates conditional mean imputation when the noise is significantly stronger than speech. For the Gaussian mixture case, it remains to be shown that the posterior weights c_k^+ from (6.4) are equivalent to the posterior weights $c_{k|y}$ from (7.7). In order to do this, let us factorize $p_{Y|k}(y)$ as:

$$\begin{aligned}p_{Y|k}(y) &\approx \mathcal{N}\left(\begin{bmatrix}n_m\\ x_o\end{bmatrix}; \begin{bmatrix}\mu_{N_m}\\ \mu_{X_o|k}\end{bmatrix}, \begin{bmatrix}\Sigma_{N_mN_m} & 0\\ 0 & \Sigma_{X_oX_o|k}\end{bmatrix}\right)\\ &= \mathcal{N}\left(n_m; \mu_{N_m}, \Sigma_{N_mN_m}\right) \cdot \mathcal{N}\left(x_o; \mu_{X_o|k}, \Sigma_{X_oX_o|k}\right).\end{aligned}$$

Then, making use of the fact that the noise likelihood $\mathcal{N}\left(n_m; \mu_{N_m}, \Sigma_{N_mN_m}\right)$ is independent of the index k of the clean speech component, the posterior weight calculation in (6.4) reduces to (7.7):

$$\begin{aligned}c_{k|y} &\approx \frac{c_k p_{Y|k}(y))}{\sum_{k'=1}^{K} c_{k'} p_{Y|k'}(y)} = \frac{c_k \mathcal{N}\left(n_m; \mu_{N_m}, \Sigma_{N_mN_m}\right)\mathcal{N}\left(x_o; \mu_{X_o|k}, \Sigma_{X_oX_o|k}\right)}{\sum_{k'=1}^{K} c_{k'} \mathcal{N}\left(n_m; \mu_{N_m}, \Sigma_{N_mN_m}\right)\mathcal{N}\left(x_o; \mu_{X_o|k'}, \Sigma_{X_oX_o|k'}\right)}\\ &= \underbrace{\frac{\mathcal{N}\left(n_m; \mu_{N_m}, \Sigma_{N_mN_m}\right)}{\mathcal{N}\left(n_m; \mu_{N_m}, \Sigma_{N_mN_m}\right)}}_{=1} \cdot \frac{c_k \mathcal{N}\left(x_o; \mu_{X_o|k}, \Sigma_{X_oX_o|k}\right)}{\sum_{k'=1}^{K} c_{k'} \mathcal{N}\left(x_o; \mu_{X_o|k'}, \Sigma_{X_oX_o|k'}\right)} = c_k^+.\end{aligned}$$

This shows the equivalence of Bayesian speech feature enhancement and conditional mean imputation under the approximations from (7.12) - (7.14). For diagonal covariance matrices, the cross-covariance term $\Sigma_{X_mX_o}$ in (7.16) becomes zero and the conditional mean imputation estimate from (7.16) reduces to the mean imputation estimate from (7.8):

$$\mu_{X|y} \approx \begin{bmatrix}\mu_{X_m}\\ x_o\end{bmatrix}.$$

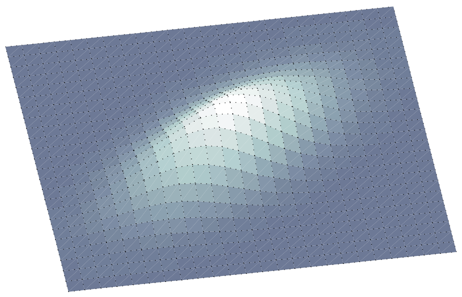
It is interesting to note that (7.12) - (7.14) implicitly assume that (a) for the missing part, the clean speech Gaussians are entirely buried below the noise Gaussian, and (b) for the observable part, the noise Gaussian is entirely buried below the clean speech Gaussians. Those are the assumptions that are made in missing feature reconstruction.

7.3 Bounded Conditional Mean Imputation

While classical missing data approaches assume that the masked parts of the clean speech spectrum are entirely missing, Holmes and Sedgwick [131] made use of the fact that the noisy speech spectrum also constitutes an upper bound, i.e. $x_m \le y_m = n_m$ (this is so because the coefficients of x_m would not be masked if they were larger than the corresponding coefficients of $y_m = n_m$). Consequently, the distribution of the masked part can be modeled by truncating the original clean speech distribution from (7.5) above at y_m and below at 0:

$$p_{X_m}(x_m) = \sum_{k=1}^{K} c_k \mathcal{N}\left(x_m; \mu_{X_m|k}, \Sigma_{X_m X_m|k}\right)\Big|_0^{y_m} .$$

This led to the bounded marginalization technique from [131], which would later be investigated in more detail by Josifovski et al. [166]. Almost 20 years after the publication of [131], Raj and Singh [209] translated Holmes and Sedgwick's approach to missing data imputation by deriving a bounded minimum mean square error (MMSE) estimate. The use of diagonal covariance matrices in that work allowed for each masked bin to be treated independently, as a doubly truncated Gaussian distribution [20, 210]. But it also restricted the approach to mean imputation, due to the fact that diagonal covariance matrices imply statistical independence of the observable and masked parts (see Section 7.1.2). This work shows how Raj and Singh's approach can be extended to full covariance matrices. This leads to the bounded conditional mean imputation estimate [17] which is described in Section 7.3.2. Its derivation requires the introduction of box-truncated Gaussian distributions.


(a) original distribution

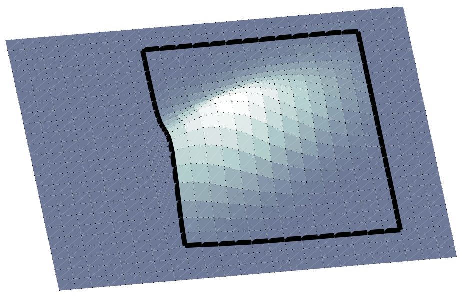

(b) truncated distribution

Figure 7.6: *Example of a 2-dimensional box-truncated multivariate Gaussian distribution*

7.3.1 The Box-Truncated Multivariate Gaussian Distribution

Following [17], the box-truncated Gaussian distribution is here defined as a multivariate Gaussian distribution that is truncated to a D-dimensional box $[L_1, U_1] \times \cdots \times [L_D, U_D]$ with upper and lower bounds U_i and L_i, $i = 1, \ldots, D$. This distribution can be formally described as

$$\mathcal{N}^{[L,U]}(x; \mu, \Sigma) \triangleq \frac{1}{c^{[L,U]}} \mathcal{N}(x; \mu, \Sigma)\Big|_L^U \tag{7.17}$$

where $c^{[L,U]}$ is the normalization constant derived in Section 7.3.1.1 and where

$$\mathcal{N}(x;\mu,\Sigma)\Big|_L^U \triangleq \begin{cases} \mathcal{N}(x;\mu,\Sigma), & x \in \bigotimes_{i=1}^D [L_i, U_i] \\ 0, & \text{otherwise} \end{cases}$$

For the imputation of masked spectral bins, the upper bound U will be set to the observed noise $n_m = y_m$ while the lower bound L will be set to 0 [17]. Next to calculating the normalizing constant in Section 7.3.1.1, the mean $\mu^{[L,U]}$ of the box-truncated Gaussian distribution is also derived in Section 7.3.1.2, as this is necessary for bounded conditional mean imputation.

7.3.1.1 Normalizing Constant

As the truncation of a Gaussian distribution removes all probability mass outside the D-dimensional box given by L and U, we need to recompute the normalizing constant

$$c^{[L,U]} \triangleq \int_L^U \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} dx. \tag{7.18}$$

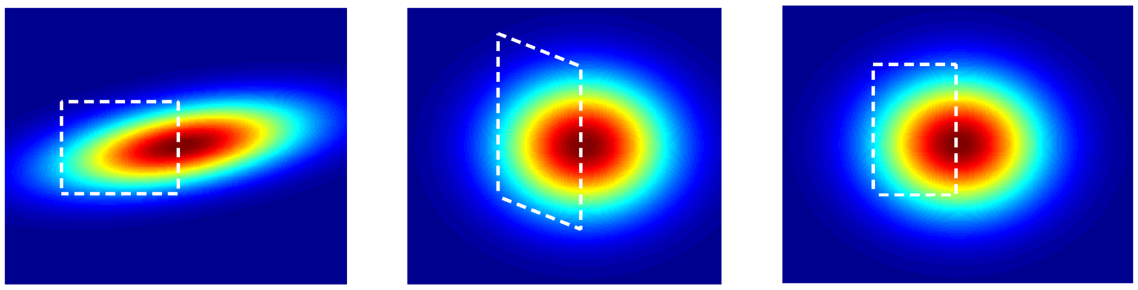
This, however, is complicated by the fact that the integral over a multivariate Gaussian distribution cannot be calculated analytically. One solution would be to use Genz's Monte Carlo approach [211]. But that proved to be too slow for speech feature imputation where a couple of thousand integrals have to be calculated for each second of input data, and that in an up to 30-dimensional space. Hence, this thesis resorts to computationally less demanding alternatives [17]: a diagonal covariance approximation that is obtained by zeroing off-diagonal elements; and an axis-parallel box approximation that is described in more detail in the following. In order to proceed, let us denote the Cholesky decomposition of the inverse covariance matrix by $\Sigma^{-1} = A^T A$, with an upper triangular matrix A. Then substituting $A(x-\mu)$ by z, the normalizing constant from (7.18) can be approximated as:

$$c^{[L,U]} = \int_L^U \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T A^T A(x-\mu)} dx \approx \int_{L'}^{U'} \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-\frac{1}{2}z^T z}\sqrt{|\Sigma|} dz, \tag{7.19}$$

where $L' = A(L-\mu)$, $U' = A(U-\mu)$, where $|\cdot|$ denotes the determinant and where the multiplication by the Jacobian determinant

$$\left|\frac{dx}{dz}\right| = |A^{-1}| = \sqrt{|A^{-1}||A^{-T}|} = \sqrt{|\Sigma|}$$

is due to the change of variables from x to z. This change of variables simplifies the integration as it causes a translation and rotation of the variable X such that the distribution is aligned with the axes, as shown in Figure 7.7-(b). The assumption made here is that the rotated integration region (i.e. the dashed quadrilateral in Figure 7.7-(b)) can reasonably be approximated by an axis-parallel box which is given by the transformed lower and upper bounds, L' and U'. This



(a) original distribution (b) change of variables (c) axis-parallel box approx.

Figure 7.7: *Effect of the change of variables, including the axis parallel box approximation.*

axis-parallel box approximation is again illustrated in Figure 7.7-(c). Now proceeding by first canceling the $\sqrt{|\Sigma|}$ in the nominator and denominator of (7.19) and then writing the sum in the exponent as a product of exponentials, (7.19) can be written:

$$c^{[L,U]} \approx \int_{L'}^{U'} \frac{1}{\sqrt{(2\pi)^D}} e^{-\frac{1}{2}\sum_{i=1}^{D} z_i^2} dz = \int_{L'}^{U'} \prod_{i=1}^{D} \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} z_i^2} \right) dz$$

Finally, pulling the product over i out of the integral, it is found that the normalizing constant from (7.18) can be approximated as

$$c^{[L,U]} \approx \prod_{i=1}^{D} \int_{L_i'}^{U_i'} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} z_i^2} dz_i = \prod_{i=1}^{D} \left(\mathscr{C}(U_i') - \mathscr{C}(L_i') \right), \tag{7.20}$$

where $\mathscr{C}$ denotes the cumulative density function of the standard (zero mean, unit variance) normal distribution. This result is exact (i.e. not an approximation) if the covariance matrix is diagonal. For general Σ, the accuracy of (7.20) is dependent on how diagonally dominant the corresponding coherence matrix

$$\Gamma = \begin{bmatrix} \frac{\sigma_{1,1}}{\sqrt{\sigma_{1,1}\sigma_{1,1}}} & \cdots & \frac{\sigma_{1,D}}{\sqrt{\sigma_{1,1}\sigma_{D,D}}} \\ \vdots & \ddots & \vdots \\ \frac{\sigma_{D,1}}{\sqrt{\sigma_{D,D}\sigma_{1,1}}} & \cdots & \frac{\sigma_{D,D}}{\sqrt{\sigma_{D,D}\sigma_{D,D}}} \end{bmatrix} \tag{7.21}$$

is, whose (i, j)-th element is the correlation coefficient $\rho_{i,j} \triangleq \sigma_{i,j}/\sqrt{\sigma_i^2\sigma_j^2}$ between the variables X_i and X_j.

7.3.1.2 Mean

This section approximates the mean of the box-truncated Gaussian distribution in analogy to the normalizing constant from Section 7.3.1.1. In order to get a start on this, let us first write down the formal definition of the mean [17]:

$$\mu^{[L,U]} \triangleq \frac{1}{c^{[L,U]}} \int_L^U x \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} dx. \tag{7.22}$$

Then substituting $A(x-\mu)$ by z, as in Section 7.3.1.1, and again using the axis-parallel box approximation, we arrive at:

$$\begin{aligned} \mu^{[L,U]} &\approx \frac{1}{c^{[L,U]}} \int_{L'}^{U'} (A^{-1}z+\mu) \frac{1}{\sqrt{2\pi}^D} e^{-\frac{1}{2}z^T z} dz \\ &= \mu \underbrace{\frac{1}{c^{[L,U]}} \int_{L'}^{U'} \frac{1}{\sqrt{2\pi}^D} e^{-\frac{1}{2}z^T z} dz}_{=c^{[L,U]}} - A^{-1} \underbrace{\frac{1}{c^{[L,U]}} \int_{L'}^{U'} \frac{1}{\sqrt{2\pi}^D} (-z) e^{-\frac{1}{2}z^T z} dz}_{\triangleq m}. \end{aligned} \tag{7.23}$$

So the "truncated mean" $\mu^{[L,U]}$ can be approximated as $\left(\mu - A^{-1}m\right)$. But for evaluating this term we still need to compute the integral in m. Let us start by breaking m into its components, by first writing z as a linear combination of standard basis vectors e_i and then pulling the sum over the basis vectors out of the integral:

$$\begin{aligned} m &= \frac{1}{c^{[L,U]}} \int_{L'}^{U'} \sum_{i=1}^{D} (-z_i) e_i \frac{1}{\sqrt{2\pi}^D} e^{-\frac{1}{2}z^T z} dz \\ &= \sum_{i=1}^{n} e_i \underbrace{\frac{1}{c^{[L,U]}} \int_{L'}^{U'} (-z_i) \prod_{j=1}^{D} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_j^2} dz}_{=m_i}. \end{aligned}$$

Now using the fact that the integral of the product of two independent factors, $f_1(x_1)$ and $f_2(x_2)$, can be calculated as the product of the integrals of the individual factors, i.e.

$$\iint f_1(x_1) f_2(x_2) dx_1 dx_2 = \int f_1(x_1) dx_1 \int f_2(x_2) dx_2,$$

the i-th component of m can be calculated as:

$$m_i = \frac{1}{c^{[L,U]}} \underbrace{\int_{L'_i}^{U'_i} (-z_i) \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2} dz_i}_{=\mathcal{N}(U'_i)-\mathcal{N}(L'_i)} \prod_{\substack{j=1 \\ j\neq i}}^{D} \underbrace{\int_{L'_j}^{U'_j} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_j^2} dz_j}_{=\mathscr{C}(U'_j)-\mathscr{C}(L'_j)}$$

where $\mathcal{N}$ denotes the standard normal density with zero mean and unit variance and where $\mathscr{C}$ denotes the corresponding cumulative density function. Further substituting $c^{[L,U]}$ by the approximation $\prod_{j=1}^{n}\left(\mathscr{C}(U'_j)-\mathscr{C}(L'_j)\right)$ from (7.20) and then canceling the $\left(\mathscr{C}(U'_j)-\mathscr{C}(L'_j)\right)$ in the nominator and denominator, it is found that m can be written:

$$m = \left[\frac{\mathcal{N}(U'_1)-\mathcal{N}(L'_1)}{\mathscr{C}(U'_1)-\mathscr{C}(L'_1)} \quad \cdots \quad \frac{\mathcal{N}(U'_D)-\mathcal{N}(L'_D)}{\mathscr{C}(U'_D)-\mathscr{C}(L'_D)} \right]^T. \tag{7.24}$$

So, in total we have the following approximation of the mean:

$$\mu^{[L,U]} \approx \mu - A^{-1}\left[\tfrac{\mathcal{N}(U_1')-\mathcal{N}(L_1')}{\mathcal{C}(U_1')-\mathcal{C}(L_1')} \quad \cdots \quad \tfrac{\mathcal{N}(U_D')-\mathcal{N}(L_D')}{\mathcal{C}(U_D')-\mathcal{C}(L_D')}\right]^T. \tag{7.25}$$

This is consistent with standard theory on one-dimensional, doubly truncated Gaussian distributions [14, 210], as in the one-dimensional case $\Sigma = \sigma^2$, $A^{-1} = \sigma$ and hence

$$\mu^{[L,U]} \approx \mu - \sigma \frac{\mathcal{N}(U') - \mathcal{N}(L')}{\mathcal{C}(U') - \mathcal{C}(L')}. \tag{7.26}$$

The equality of (7.26) with the equations in [14, 210] shows that the above approximation is exact in the one-dimensional case. The same obviously holds for diagonal covariance matrices [17]. In the general case, the accuracy of approximation is dependent on how diagonally dominant the coherence matrix Γ is (see the end of Section 7.3.1.1).

7.3.2 Bounding the Conditional Mean Imputation Estimate

Now that the box-truncated Gaussian distribution has been introduced, it can be used to bound the conditional mean imputation estimate from Section 7.1.1. This is achieved by calculating the expectation of $X_m|x_o$,

$$\widehat{x}_m = \mathcal{E}_{p_{X_m|x_o}}\{x_m\} = \int_0^{y_m} x_m p_{X_m|x_o}(x_m) dx_m \tag{7.27}$$

with $U = y_m$ and $L = 0$ as lower and upper bounds. These bounds are motivated by the facts that (1) the missing part x_m is bounded above by the noisy speech spectrum, i.e. $x_m \le y_m = n_m$, and (2) the log-Mel spectra used here cannot assume negative values (see [17]).

7.3.2.1 The Gaussian Case

Under the simplistic assumption that clean speech spectra follow a Gaussian distribution, the conditional distribution of $X_m|x_o$ is obtained by conditioning the partitioned joint distribution

$$p_{X_m,X_o}(x_m, x_o) = \mathcal{N}\left(\begin{bmatrix} x_m \\ x_o \end{bmatrix}; \begin{bmatrix} \mu_{X_m} \\ \mu_{X_o} \end{bmatrix}, \begin{bmatrix} \Sigma_{X_m X_m} & \Sigma_{X_m X_o} \\ \Sigma_{X_o X_m} & \Sigma_{X_o X_o} \end{bmatrix}\right) \tag{7.28}$$

on $X_o = x_o$. This gives $p_{X_m|x_o}(x_m) = \mathcal{N}\left(x_m; \mu_{X_m|x_o}, \Sigma_{X_m|x_o}\right)$ where $\mu_{X_m|x_o}$ and $\Sigma_{X_m|x_o}$ are calculated according to Section 2.2.5:

$$\mu_{X_m|x_o} = \mu_{x_m} + \Sigma_{X_m X_o} \Sigma_{X_o X_o}^{-1}\left(x_o - \mu_{X_o}\right), \quad \Sigma_{X_m|x_o} = \Sigma_{X_m X_m} - \Sigma_{X_m X_o} \Sigma_{X_o X_o}^{-1} \Sigma_{X_o X_m}.$$

Plugging this distribution back into (7.27), we find that the bounded conditional mean imputation estimate is just the mean $\mu^{[L,U]}$ of a box-truncated Gaussian distribution:

$$\widehat{x}_m = \int_0^{y_m} x_m p_{X_m|x_o}(x_m) dx_m = \int x_m \mathcal{N}^{[0,y_m]}\left(x_m; \mu_{X_m|x_o}, \Sigma_{X_m|x_o}\right) dx_m. \tag{7.29}$$

Approximating the mean according to Section 7.3.1.2 finally gives the following estimate for the missing part [17]:

$$\hat{x}_m \approx \mu_{X_m|x_o} - A^{-1}_{X_m|x_o} \left[\frac{\mathcal{N}(y'_{m,1})-\mathcal{N}(l'_{m,1})}{\mathcal{C}(y'_{m,1})-\mathcal{C}(l'_{m,1})} \quad \cdots \quad \frac{\mathcal{N}(y'_{m,D})-\mathcal{N}(l'_{m,D})}{\mathcal{C}(y'_{m,D})-\mathcal{C}(l'_{m,D})} \right]^T \tag{7.30}$$

where $l'_m = A_{X_m|x_o}(0-\mu_{X_m|x_o})$, $y'_m = A_{X_m|x_o}(y_m - \mu_{X_m|x_o})$ and where $A_{X_m|x_o}$ denotes the upper triangular matrix of the Cholesky decomposition of $\Sigma^{-1}_{X_m|x_o}$. For the diagonal covariance approximation (see the start of Section 7.3.1.1), $\Sigma_{X_m|x_o}$ is diagonalized before $A_{X_m|x_o}$ is calculated. This is achieved by setting all non-diagonal elements to zero.

7.3.2.2 The Gaussian Mixture Case

The Gaussian case from the previous section can easily be extended to Gaussian mixture distributions. In order to do so, let us partition the clean speech Gaussian mixture distribution as described in Section 7.1.1:

$$p_{X_m,X_o}(x_m,x_o) = \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}\left(\begin{bmatrix} x_m \\ x_o \end{bmatrix}; \begin{bmatrix} \mu_{X_m|k} \\ \mu_{X_o|k} \end{bmatrix}, \begin{bmatrix} \Sigma_{X_mX_m|k} & \Sigma_{X_mX_o|k} \\ \Sigma_{X_oX_m|k} & \Sigma_{X_oX_o|k} \end{bmatrix} \right)}_{=p_{X_m,X_o|k}(x_m,x_o)}. \tag{7.31}$$

Then, the MMSE estimate of the missing part x_m is obtained by first calculating the conditional distribution of $X_m|x_o$ according to Section 3.2.2, plugging the result back into (7.27) and then making use of (7.29). This gives [17]:

$$\begin{aligned} \hat{x}_m &= \int_0^{y_m} x_m \sum_{k=1}^{K} c_k^+ \mathcal{N}\left(x_m; \mu_{X_m|x_o,k}; \Sigma_{X_m|x_o,k}\right) dx_m \\ &= \sum_{k=1}^{K} c_k^+ \underbrace{\int x_m \mathcal{N}^{[0,y_m]}\left(x_m; \mu_{X_m|x_o,k}; \Sigma_{X_m|x_o,k}\right) dx_m}_{\hat{=}\hat{x}_{m|k}} \end{aligned} \tag{7.32}$$

where the conditional mean and covariance matrices $\mu_{X_m|x_o,k}$ and $\Sigma_{X_m|x_o,k}$ of the Gaussians are calculated according to (3.10):

$$\mu_{X_m|x_o,k} = \mu_{x_m|k} + \Sigma_{X_mX_o|k}\Sigma^{-1}_{X_oX_o|k}\left(x_o - \mu_{X_o|k}\right), \quad \Sigma_{X_m|x_o,k} = \Sigma_{X_mX_m|k} - \Sigma_{X_mX_o|k}\Sigma^{-1}_{X_oX_o|k}\Sigma_{X_oX_m|k}.$$

Further approximating the individual estimates $\hat{x}_{m|k}$ in (7.32) in analogy to (7.30), we arrive at:

$$\hat{x}_m \approx \sum_{k=1}^{K} c_k^+ \left(\mu_{X_m|x_o,k} - A^{-1}_{X_m|X_o,k} \left[\frac{\mathcal{N}(y'_{m,k,1})-\mathcal{N}(l'_{m,k,1})}{\mathcal{C}(y'_{m,k,1})-\mathcal{C}(l'_{m,k,1})} \quad \cdots \quad \frac{\mathcal{N}(y'_{m,k,D})-\mathcal{N}(l'_{m,k,D})}{\mathcal{C}(y'_{m,k,D})-\mathcal{C}(l'_{m,k,D})} \right]^T \right) \tag{7.33}$$

where $l'_{m,k} = A_{X_m|x_o,k}(0-\mu_{X_m|x_o,k})$, $y'_{m,k} = A_{X_m|x_o,k}(y_m - \mu_{X_m|x_o,k})$ and where $A_{X_m|x_o,k}$ denotes the upper triangular matrix of the Cholesky decomposition of $\Sigma^{-1}_{X_m|x_o,k}$. In order to calculate

Bounded Conditional Mean Imputation

1. For each $k \in \{1, \dots, \kappa\}$ do:

   (a) Calculate the mean $\mu_{X_m|x_o,k}$ and covariance matrix $\Sigma_{X_m|x_o,k}$ of the conditional Gaussian distribution $p_{X_m|x_o,k}$ according to

   $$\begin{aligned} \mu_{X_m|x_o,k} &= \mu_{x_m|k} - \Sigma_{X_m X_o|k} \Sigma^{-1}_{X_o|k} \left(x_o - \mu_{X_o|k} \right), \\ \Sigma_{X_m|x_o,k} &= \Sigma_{X_m X_m|k} - \Sigma_{X_m X_o|k} \Sigma^{-1}_{X_o X_o|k} \Sigma_{X_o X_m|k}. \end{aligned}$$

   (b) In case of the diagonal covariance approximation, set all non-diagonal elements of $\Sigma_{X_m|x_o,k}$ to 0. In case of the axis-parallel box approximation: do nothing.

   (c) Calculate the right Cholesky factor $A_{X_m|x_o,k}$ of the inverse covariance matrix $\Sigma^{-1}_{X_m|x_o,k}$. Then calculate the transformed upper and lower bounds

   $$l'_{m,k} = A_{X_m|x_o,k} \cdot (0 - \mu_{X_m|x_o,k}), \quad y'_{m,k} = A_{X_m|x_o,k} (y_m - \mu_{X_m|x_o,k}).$$

   (d) Calculate the observation likelihood $p_{X_o|y_m,k}(x_o)$ of the k-th Gaussian component as

   $$p_{X_o|y_m,k}(x_o) = \mathcal{N}\left(x_o; \mu_{X_o|k}, \Sigma_{X_o X_o|k}\right) \prod_{i=1}^{D} \left(\mathcal{C}(y'_{m,k,i}) - \mathcal{C}(l'_{m,k,i}) \right).$$

   (e) Calculate the bounded conditional mean imputation estimate $\hat{x}_{m|k}$ of the k-th Gaussian component according to:

   $$\hat{x}_{m|k} = \mu_{X_m|x_o,k} - A^{-1}_{X_m|X_o,k} \left[\frac{\mathcal{N}\left(y'_{m,k,1}\right) - \mathcal{N}\left(l'_{m,k,1}\right)}{\mathcal{C}\left(y'_{m,k,1}\right) - \mathcal{C}\left(l'_{m,k,1}\right)} \quad \cdots \quad \frac{\mathcal{N}\left(y'_{m,k,D}\right) - \mathcal{N}\left(l'_{m,k,D}\right)}{\mathcal{C}\left(y'_{m,k,D}\right) - \mathcal{C}\left(l'_{m,k,D}\right)} \right]^T.$$

   where $\mathcal{N}$ and $\mathcal{C}$ denote the probability density function and cumulative density function of the standard Gaussian distribution.

2. Calculate the total observation likelihood $L = \sum_{k=1}^{\kappa} c_k p(x_o | y_m, k)$.

3. Calculate the final bounded conditional mean imputation estimate $\hat{x}_m$ according to:

   $$\hat{x}_m = \sum_{k=1}^{\kappa} \underbrace{\left[\frac{c_k p(x_o | y_m, k)}{L} \right]}_{= c_k^+} \hat{x}_{m|k}.$$

Algorithm 7.1: Bounded Conditional Mean Imputation

the posterior probabilities $c_k^+ = p_{K|y}(k)$ in (7.33), let us first make use of Bayes rule with $y = [y_m^T \ x_o^T]^T$, $c_k = p_K(k) = p_{K|y_m}(k)$,

$$c_k^+ = p_{K|x_o,y_m}(k) = \frac{c_k p_{X_o|y_m,k}(x_o)}{\sum_{k'=1}^{K} c_{k'} p_{X_o|y_m,k'}(x_o)}, \tag{7.34}$$

and then calculate the $p_{X_o|y_m,k}(x_o)$ in this equation according to [173, 17]:

$$p_{X_o|y_m,k}(x_o) = \int_0^{y_m} \underbrace{p_{X_o|k}(x_o) \cdot p_{X_m|x_o,k}(x_m)}_{=p_{X_m,X_o|k}(x_m,x_o)} dx_m = p_{X_o|k}(x_o) \int_0^{y_m} p_{X_m|x_o,k}(x_m) dx_m.$$

This gives:

$$p_{X_o|y_m,k}(x_o) = \mathcal{N}\left(x_o; \mu_{X_o|k}, \Sigma_{X_oX_o|k}\right) \underbrace{\int_0^{y_m} \mathcal{N}(x_m; \mu_{X_m|x_o,k}, \Sigma_{X_m|x_o,k}) dx_m}_{=C_k^{[0,y_m]}} \tag{7.35}$$

where the term $C_k^{[0,y_m]}$ is the normalizing constant of a box-truncated Gaussian distribution, which can be approximated according to (7.20):

$$C_k^{[0,y_m]} \approx \prod_{i=1}^{D} \left(\mathscr{C}(y'_{m,k,i}) - \mathscr{C}(l'_{m,k,i})\right). \tag{7.36}$$

Algorithm 7.1, again summarizes this result with a compact description of the bounded conditional mean imputation technique.

7.4 Mask Estimation

The imputation methods from the previous sections are all based on partitioning the clean speech feature vectors x_t into an observable part $x_{t,o}$ and a missing part $x_{t,m}$. This is generally achieved by reordering the coefficients according to a mask θ_t, whose components $\theta_{t,i}$ identify which bins are "missing" due to masking by the noise (see the start of this chapter). These masks are sometimes considered to be "known" in order to compare different imputation methods under ideal conditions [207]. In practice, however, they need to be estimated with a mask estimation algorithm. Examples in the literature here include computational auditory scene analysis (CASA) [199], the neg-energy criterion [212], the difference between cube root signal and noise energy [213], Bayesian classifiers [214] or the soft Max-VQ algorithm from [209]. The approach taken in this chapter combines mask estimation with the particle filter based noise tracking framework from Section 4.7. More specifically, it uses the empirical noise distribution

$$p_{N_t|y_{1:t}}(n_t) \approx \sum_{j=1}^{L} \omega_t^{(j)} \delta_{n_t^{(j)}}(n_t), \tag{7.37}$$

i.e. the filtering density from (6.58), in order to approximate the implied clean speech distribution $p_{X_t|y_{1:t}}$. This is achieved by individually transforming the noise samples $n_t^{(j)}$ according to the inverse interaction function from Section 5.3, $x_t^{(j)} \triangleq g\left(y_t, 0, n_t^{(j)}\right)$, and then constructing an empirical approximation of $p_{X_t|y_{1:t}}$ according to:

$$p_{X_t|y_{1:t}}(x_t) \approx \sum_{j=1}^{L} \omega_t^{(j)} \delta_{x_t^{(j)}}(x_t). \tag{7.38}$$

With this approximation, the probability that the i-th clean speech bin $x_{t,i}$ is masked by noise, i.e. $\mathscr{P}(n_{t,i}$ masks $x_{t,i})$, can be evaluated as

$$\begin{aligned} \mathscr{P}(n_{t,i} \geq x_{t,i} + \tau) &= \int_{-\infty}^{\infty} \left(\int_{x_{t,i}+\tau}^{\infty} p(n_{t,i}) d n_{t,i} \right) p(x_{t,i}) d x_{t,i} \\ &\approx \sum_{j=1}^{L} \sum_{j'=1}^{L} \omega_t^{(j)} \omega_t^{(j')} u(n_{t,i}^{(j)} - x_{t,i}^{(j)} - \tau) \end{aligned} \tag{7.39}$$

where u denotes the unit step function which is 1 for $x_{t,i} \geq 0$ and 0 otherwise; τ is the masking threshold [212], i.e. the difference in power at which masking is assumed to occur. The masking probability (7.39) can now directly be used as a soft-mask estimate $\hat{\theta}_i = \mathscr{P}(n_{t,i} \geq x_{t,i} + \tau)$, in combination with a soft-decision approach [20], or it may be quantized to a binary mask [17]

$$\hat{\theta}_{t,i} = \begin{cases} 1, & \mathscr{P}(n_{t,i} > x_{t,i}) \geq 0.5 \\ 0, & \mathscr{P}(n_{t,i} > x_{t,i}) < 0.5 \end{cases} \tag{7.40}$$

for use with any the missing feature reconstruction approaches that have been described in this chapter. A detailed comparison of other mask estimation techniques is found in [215, 216].

7.5 Contributions of this Chapter

The following list again gives an overview of the individual contributions of this thesis to missing feature reconstruction:

1. Introduction of the box-truncated Gaussian distribution, including two approximations for its mean and normalization constant (Section 7.3.1) [17].

2. The bounded conditional mean imputation technique from Section 7.3.2 [17].

3. Particle filter based mask estimation (Section 7.4) [20].

4. An analysis of the relationship between conditional mean imputation and Bayesian speech feature enhancement (Section 7.2).

8
Speech Recognition Experiments

This chapter evaluates the usefulness of the proposed speech feature enhancement and reconstruction techniques. That is done by means of an experimental comparison in which a clean speech corpus is contaminated with prerecorded noise and then processed with an automatic speech recognition (ASR) system. The detailed results are presented in Sections 8.4.1 - 8.4.6, right after the ASR system has been described along with the used corpora. As it is well known that artificially added noise differs from a true noisy speech corpus (see e.g. the Lombard effect [217, 218]), the most important experiments are finally repeated on a real in-car corpus. This is done in Section 8.5.

8.1 Measuring ASR Performance

In order to compare the performance of different feature enhancement and reconstruction techniques at the hand of speech recognition results, we need to specify a quantitative measure for ASR performance. The most widely accepted measure for this purpose is the word error rate (WER). It is formally defined [101] as the minimum edit distance – by means of the number of word insertions, deletions and substitutions – between an ASR hypothesis and the corresponding reference transcription. This distance is further divided by the number of words:

$$\text{WER} \;=\; \frac{\#(\text{insertions}) + \#(\text{deletions}) + \#(\text{substitutions})}{\#(\text{words in reference transcription})} \times 100\%$$

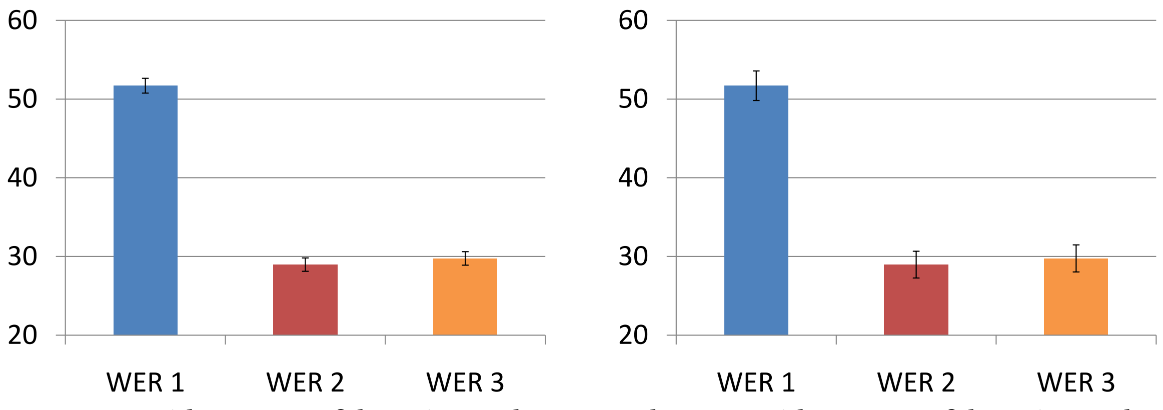
where $\#(\cdot)$ denotes the counting symbol. The reference transcription is the text which was really said in the utterances. In practice, the WER is calculated with standard tools such as the "HResults" program which is provided along with the Hidden Markov Model Toolkit (HTK) [219] or the "sclite" engine of the National Institute of Standards (NIST) Scoring Toolkit (SCTK).

8.1.1 Standard Error

When comparing word error rates, it is of interest to know whether a difference in WER is statistically significant. In order to do so, the standard error [220, 221] of each measured WER is calculated according to [222]:

$$SE = \sqrt{\frac{(WER/100)\cdot(100 - WER)/100}{\#(\text{words in reference transcription})}}.$$

Then, the probability that the true WER lies in the interval $[WER - SE, WER + SE]$ is 68.3% (under certain assumptions [222]). In particular, if two intervals $I_1 = [WER_1 - SE_1, WER_1 + SE_1]$ and $I_2 = [WER_2 - SE_2, WER_2 + SE_2]$ do not overlap then the difference between the word error rates WER_1 and WER_2 is considered to be statistically significant with 1-sigma (68.3%) confidence. In practice, the significance is checked by plotting standard error bars which correspond to the intervals. For a more thorough justification the reader is referred to [222].



(a) WERs with 68.3% confidence intervals *(b) WERs with 95.4% confidence intervals*

Figure 8.1: Bar plots of word error rates with corresponding error bars (black vertical lines). While WER 2 and WER 3 are significantly lower than WER 1 - even with 95.4% (2-sigma) confidence, i.e. with intervals $I = [WER - 2\cdot SE_1, WER + 2\cdot SE_1]$ that have been extended to 2 times the standard deviation - there is no significant difference between WER 2 and WER 3.

8.2 Description of the ASR System

The ASR experiments, which are reported in the following, have all been performed with the "Millennium" toolkit [223]. This toolkit is described in more detail in [102]. It supports several speech feature extraction schemes, monophone and triphone acoustic models, n-gram and grammar-based language models, lattice rescoring as well as various speaker adaptation techniques. The toolkit is implemented as a set of *C++* libraries that have been compiled for use in *python* [224]. Training, decoding and adaptation can all be performed with relatively short python scripts (that are about 500 to 1000 lines long).

8.2.1 WFST Decoder

The decoder is implemented as suggested by Saon et al. [225], including fast on-the-fly composition [226], [102, §7.4] of weighted finite state transducers (WFSTs). It generates word lattices which are then optimized with WFST operations, as described in [227]. The resulting word lattices can be used for lattice rescoring as well as for forced alignments during speaker adaptation. In order to get as meaningful results as possible, the dynamic beam width (see [102]) is rather wide. It is 160 on the first pass (without adaptation) and 180 on the second pass (with CMLLR adaptation – see Section 8.2.4). There is no maximum number of expandable states.

8.2.2 Lexicon and Phone-Set

As a pronunciation lexicon, the freely available CMU SPHINX [228] dictionary is used. This lexicon is based on the standard ARPA phone-set (ARPAbet) [229].

8.2.3 Acoustic Models

For digits recognition tasks, a simple monophone [101] acoustic model is used with a 3 state left-to-right hidden Markov model (HMM) for each phone. Such an HMM is shown in Figure 8.2 at the example of the phoneme *AH*. For mid-size vocabulary tasks, such as the 5000 (5k) word American wall street journal corpus [230] and its British counterpart [231], 1,743 context-clustered triphone states [101], [102, §7.3.4] are used with the same 3 state left-to-right model. Silence between words (SIL) and an optional breath (+BREATH+) symbol are separately modeled as simple one-state HMMs. The output densities are Gaussian mixture distributions with up to 46 Gaussians per state. The Gaussians have diagonal covariance matrices and they are trained with a variant of the split and merge EM algorithm [64]

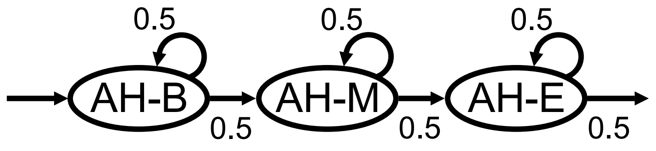


Figure 8.2: *Example of a 3 state left-to-right model for the monophone AH with begin, middle and end states AH-B/-M/-E. The transition probabilities are not trained but statically set to* 0.5.

8.2.4 Speaker Adaptation

State-of-the-art ASR systems can typically adapt their acoustic models to new speakers or environments (see Section 6.1.4). Millennium provides two adaptation methods for this purpose: constrained MLLR (CMLLR) adaptation [158, 232] (with a single regression class) and full MLLR adaptation [232] (with multiple regression classes). This work uses CMLLR adaptation only. This is done in an unsupervised fashion, by (1) performing a first (unadapted) speech recognition pass in order to obtain an initial word lattice; (2) using the decoder to obtain a forced alignment of the lattice; and (3) estimating the CMLLR parameters based on that alignment. The implementation used here re-estimates the parameters three times with realignment (i.e. steps 2 and 3 are repeated three times). Subsequently, the estimated CMLLR parameters are used for a final speech recognition pass.

8.2.5 Feature Extraction

In the ASR experiments which are reported below, the feature extraction is performed as described in Section 5.1. It starts with calculating 13 dimensional Mel frequency cepstral coefficients (MFCCs) with a 30 bin triangular Mel filterbank. After cepstral mean subtraction (CMS) [233, 234, 235] with variance normalization (CVN) [109, 110, 111], 15 consecutive frames of 13-coefficient MFCCs are concatenated and subsequently reduced by linear discriminant analysis (LDA). The resulting 42-dimensional features are used for recognition (unless explicitly stated otherwise). For CMS and CVN, the cepstral mean and variance are estimated on speech frames only (as determined by an energy based voice activity detector). This was found to give better results in practice than estimating them on the entire utterance [236]. Speech Feature enhancement is performed in the log-Mel domain and it is succeeded by the remaining feature extraction steps (DCT, CMS, etc.).

8.3 Description of the Corpora

The following describes the corpora, which are used in the experiments, including specifications of the test and training sets as well as an outline of the key features of the used acoustic and language models.

8.3.1 MC-WSJ-AV

The Multi-Channel Wall Street Journal Audio-Visual corpus [117] provides an intermediate task between simple digits recognition and large vocabulary conversational speech recognition. It consists of read sentences from the Wall Street Journal which have been recorded under three different conditions: (A) single stationary speaker; (B) single moving speaker; and (C) two speakers speaking at the same time. All the data was captured simultaneously using headsets, lapel microphones and two circular 8-channel microphone arrays. The reason for choosing this somewhat unusual corpus for speech enhancement experiments is simply that a readily trained ASR system was available [226, 237, 238] for this corpus. This ASR system has a performance comparable to that of other state-of-the-art systems [117, 239]. The triphone acoustic model was trained with 30 hours WSJ0 [230] and 12 hours WSJCAM0 [231] data, as described earlier in [237, 238]. It has 1,743 fully continuous codebooks with a total of 67,017 Gaussians. The much smaller (128 component) auxiliary clean speech model, which is used for speech feature enhancement and reconstruction, has been trained on the same data set. As a test set for speech recognition experiments, this work uses the headset recordings of the 5000-word sub-corpus of the MC-WSJ-AV data – more specifically: the single stationary speaker condition of the EVAL-1 and DEV-1 sets. Following Kumatani et al. [237, 238], the following utterances were excluded from the corpus:

- AMI_WSJ16-$*$_T6c020u (1 utterance)
- AMI_WSJ17-$*$_T7c020[o-z] (11 utterances)
- AMI_WSJ17-$*$_T7c021[0-3] (4 utterances)

This left a total of 352 clean speech utterances – 189 in the EVAL-1 set and 163 in the DEV-1 set – which amounts to approximately 40 minutes of speech. All the experiments, which are reported on this corpus, use a full trigram language model that has been trained on the transcriptions of the WSJ0 and WSJCAM0 corpora. The recognition network is composed at runtime, as described in [226].

8.3.2 NOISEX-92

In some of the experiments, noisy environments are simulated by adding noise to clean speech signals. This is done using the NOISEX-92 database [118] which provides recordings of various noise types that might be encountered in a military context, including cockpit noise from fighter jets (Buccaneer, F-16), destroyer engine and operations room noise, cabin noise from tanks (leopard, m109), babble, machine gun and factory noise as well as car noise from a Volvo 340. The original audio data was captured at a sampling rate of 19.98 kHz. Hence, all audio files were downsampled to 16kHz in order to match the sampling of the MC-WSJ-AV corpus.

8.3.3 AVICAR

In addition to artificially adding noise to the MC-WSJ-AV corpus, this thesis also considers experiments on a real noisy speech corpus. The corpus used for these experiments is the AVICAR corpus [240]. It was recorded in different cars driving with up to 55 miles per hour with a microphone array[1] mounted on the sun visor. The signal to noise ratio varies from 15dB to -10dB, where the latter value is assumed when the windows are open. Next to adverse noise conditions there are sporadic outages of microphones in about 15% of the utterances, due to defective contacts in the recording equipment [242]. Hence, the corpus was cleaned by running a script that automatically detects damaged utterances by their lower average power signature as well as by the signal bias, which occurs in some of the damaged recordings [16]. For ASR experiments,

[1] Note that this work uses the first channel of the array as a single distant microphone. Results for multiple channels are reported in [241].

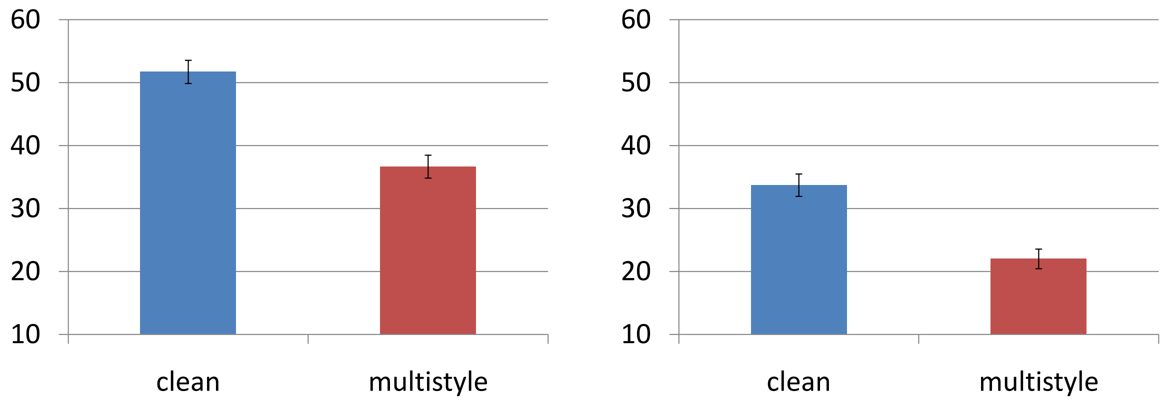
the AVICAR corpus was split into one training set and two test sets. The training set consisted of the “IDL” condition of the digits and phone number tasks, in which the car is standing still with the engine running (idle). This set was used to train a monophone acoustic model with 62 states (and a total of 2,742 Gaussians) as well as an auxiliary clean speech model for speech feature enhancement. Decoding was performed with a grammar-based language model that allows arbitrary sequences of numbers with optional pauses (silence) in between. The test set consists of the “35U” and “35D” conditions of the phone number task. In these conditions, the car is driving at 35 miles per hour with the windows being up (U) or down (D). This comprises 1324 and 1265 10-digit phone numbers, spoken by 79 different speakers. The total recording lengths are 114 and 108 minutes. There are several noise sources present, including noise from the engine, wind blowing into the microphones and cars passing by.

8.4 Experiments with Added Noise

This section compares all the feature enhancement and reconstruction techniques from Chapters 6 and 7 under the same conditions. That is achieved by (1) contaminating the MC-WSJ-AV corpus [117] (headset data) with noise from the NOISEX-92 database [118], and then (2) performing experiments on the resulting, contaminated recordings. The used noise types are destroyer engine (destroyereng) and factory (factory2) noise at signal-to-noise-ratios (SNRs) of 5, 10, 15 and 20 dB.

8.4.1 The Baseline

Figure 8.3 shows the baseline word error rates for clean speech training as well as for multi-condition (multi-style) training [155, 156], averaged over all 8 noise conditions. Note that these results were obtained with the standard Millennium front end, without speech feature enhancement and reconstruction.



(a) without speaker adaptation *(b) with CMLLR adaptation*

Figure 8.3: *Baseline ASR results for clean speech training and multi-style training. The word error rates are shown with and without CMLLR adaptation and with 2 sigma (95.4%) error bars.*

The plot to the left shows the first, unadapted pass. The plot to the right shows the more relevant second pass – after CMLLR adaptation (see Section 8.2.4). In both cases, multi-condition training gave significantly better speech recognition results than clean speech training, with relative improvements of 29.3% and 34.0%, respectively, on the first and second pass. Table 8.1 shows the exact WERs in dependence of the noise condition.

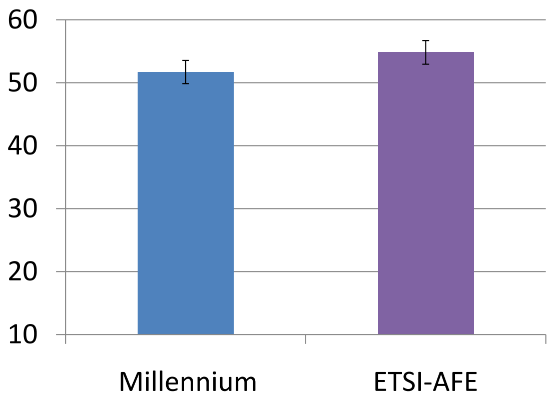
training method	speaker adaptation	destroyer engine 05dB	10dB	15dB	20dB	factory site 05dB	10dB	15dB	20dB	average WER
clean	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
multistyle	none	66.06	51.28	39.23	32.78	36.79	25.12	21.27	19.98	36.56
clean	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
multistyle	CMLLR	44.22	28.20	18.80	13.96	27.97	17.81	13.12	11.86	21.99

Table 8.1: *Word error rate under different noise conditions.*

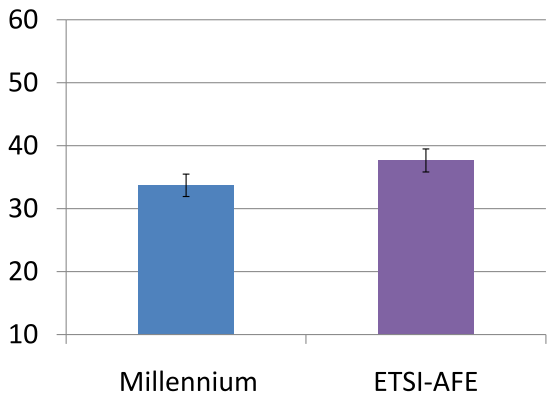
Note that for multi-condition training, the training data (consisting of the WSJ0 and WSJCAM0 corpora [230, 231]) was contaminated with 5, 10, 15 and 20 dB destroyer engine room (destroyereng), destroyer operations room (destroyerops), factory (factory2) and tank (leopard) noise. This led to a 4-fold increase of the training corpus; and it resulted in an acoustic model with 80178 Gaussians, compared to 67017 Gaussians for clean speech training. This increase can be explained by the fact that the split and merge EM algorithm [64], which has been used for Gaussian mixture training, automatically selects the most suitable number of Gaussians.

8.4.2 Comparison to ETSI-AFE features

Figure 8.4 gives a comparison between Millennium LDA features (see Section 8.2.5) and the 16 kHz version of the ETSI Advanced Front End (AFE) [243]. This comparison is of interest as the advanced front end includes noise reduction techniques such as two-stage Wiener filtering [145], SNR-dependent waveform processing [244] and blind equalization [243].



(a) without speaker adaptation



(b) with CMLLR adaptation

Figure 8.4: *Baseline ASR results for Millennium features and the ETSI advanced front end (AFE). The WERs are shown with 2 sigma (95.4%) error bars. Training was performed on clean speech.*

The plots again show the first, unadapted pass and a second pass with CMLLR adaptation. In both cases, Millennium features were slightly better than the ETSI-AFE, with 95.4% (2 sigma) significance on the second pass but just 68.3% (1 sigma) significance on the first pass. Table 8.2 shows the exact numbers in dependency of the noise condition. This table reveals that

(1) in non-stationary factory noise, Millennium features gave better results in general.

(2) with stationary destroyer engine room noise, the AFE did better on the first pass.

(3) in combination with CMLLR adaptation, Millennium features constantly outperformed the AFE except for 5dB destroyer engine noise.

feature	speaker	destroyer engine				factory site				average
extraction	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
Millennium	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
ETSI-AFE	none	87.12	72.37	54.78	41.17	68.06	50.10	36.91	28.04	54.82
Millennium	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
ETSI-AFE	CMLLR	73.60	51.04	33.05	22.15	51.62	31.63	21.19	16.88	37.65

Table 8.2: *WERs for Millennium features as well as for the ETSI-AFE. In both cases, training was performed on clean speech. The training and decoding parameters were identical.*

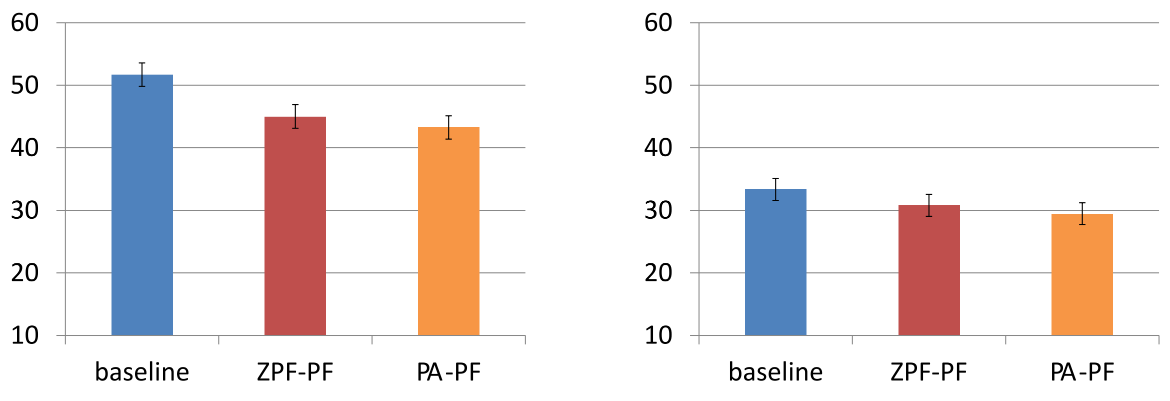
Also note that with AFE features the decoding times were almost 3 times higher on the first pass (622 hours compared to 230 hours) and over 4 times higher on the second pass (1438 hours compared to 336 hours). This might indicate a higher overlap of the triphone distributions, as the decoder automatically determines the search space (dynamic beam width) based on the current best hypothesis [102]. In these experiments, both systems (the one with Millennium features and the one with ETSI-AFE features) were trained with the same training scripts and parameters, including 63 iterations of split and merge EM training with a maximum of 46 Gaussians per state. The decoding parameters were identical. In case of the AFE, feature extraction was performed with the following commands (see [243]):

```
AdvFrontEnd <rawFile> <mfccFile> -F RAW -fs 16 -swap
derivCalc -COMB <mfcFile> <featFile>
```

This means: MFCC quantization (-q) and frame dropping (-FD) were disabled. The results were significantly worse when these switches were enabled.

8.4.3 Particle Filter Experiments

Proceeding in the chronological order of this thesis, the next experiments evaluate speech feature enhancement with the particle filter approach from Section 6.5. Figure 8.5 shows the resulting word error rates for the zero-phase factor model (ZPF) [127, 133] – including the fast acceptance test (FAT) [136, 134] – as well as for the phase-averaged model (PA), which has been introduced in Section 6.5.



(a) without speaker adaptation *(b) with CMLLR adaptation*

Figure 8.5: Average word error rates for particle filter (PF) based speech feature enhancement. The bar plots show results with the zero-phase factor (ZPF) model (see Section 6.5.2) as well as with the phase-averaged (PA) model from Section 6.5.3. The "baseline" is the clean speech (training) baseline from Section 8.4.1.

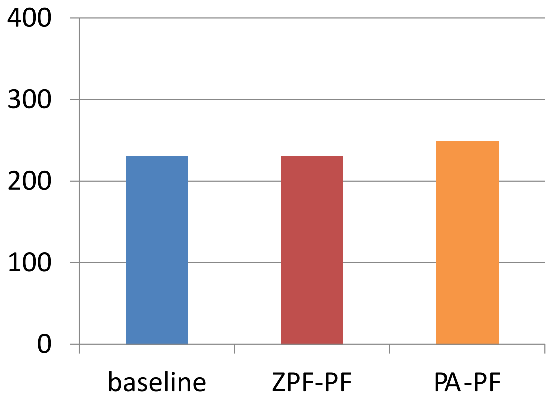
These experiments were again performed on the MC-WSJ-AV corpus, with added factory and destroyer room engine noise at SNRs of 5, 10, 15 and 20 dB. The plot to the left shows the first, unadapted pass. The plot to the right shows results after CMLLR adaptation. In both cases, both the PA-PF and the ZPF-PF performed significantly[2] better than the baseline, with relative WER reductions of 12.9% and 16.3% on the first pass, 7.5% and 11.6% on the second pass. The PA-PF performed slightly better. But this difference was just 68.3% (1-sigma) significant on the first pass; it was not significant on the second pass. Table 8.3 shows the exact word error rates in dependency of the noise condition.

enhancem.	speaker	destroyer engine				factory site				average
method	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
none	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
ZPF-PF	none	85.34	66.30	44.21	30.25	56.43	35.27	23.70	18.66	45.02
PA-PF	none	82.11	62.25	42.28	29.56	56.53	33.00	22.92	17.41	43.26
none	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
ZPF-PF	CMLLR	70.95	43.10	23.89	15.19	43.71	23.62	14.58	11.55	30.82
PA-PF	CMLLR	68.01	40.98	22.15	14.90	42.04	22.21	13.82	11.60	29.46

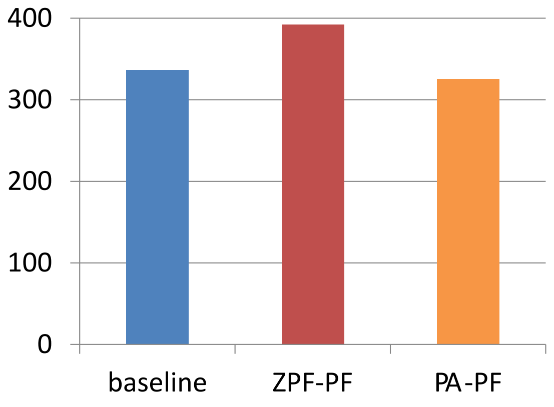
Table 8.3: *Word error rates for particle filter (PF) based speech feature enhancement.*

Having a closer look at this table, we find that the PA-PF performed better in almost every single experiment, with the exception of factory noise at 5dB (1st pass) and factory noise at 20dB (2nd pass). Figure 8.6 shows the computation times on a 3.0 GHz Intel Xeon CPU. The times are given in hours; they correspond to processing the whole corpus under all noise conditions; and they comprise the computation times of both speech feature enhancement and decoding.

[2] Note that the difference between PA-PF and baseline is 95.4% (2 sigma) significant. The difference between ZPF-PF and baseline is 94.5% significant on the first pass but just 68.3% on the second pass.



(a) without speaker adaptation



(b) with CMLLR adaptation

Figure 8.6: *Computation times of the particle filter based speech feature enhancement experiments. The times are given in hours and comprise both enhancement and decoding.*

These plots reveal that the PA-PF slightly increases the computation time of the first pass (by about 8% relative). Nevertheless, it lowers the computation time of the more relevant second pass, especially when compared to the ZPF-PF (relative reduction of 20.6%).

8.4.4 Missing Feature Reconstruction Experiments

This section evaluates the performance of the missing feature reconstruction techniques from Chapter 7. In order to have a balanced comparison, the methods are first compared under ideal conditions, by using so-called *oracle masks* [207, 214], i.e. masks that would be produced by an optimal mask estimation algorithm. This is achieved [207, 214] by calculating the oracle mask θ_t at time t from the true clean speech and noise spectra[3], x_t and n_t:

$$\theta_{t,i} = \begin{cases} 1, & n_{t,i} \geq x_{t,i} \\ 0, & n_{t,i} < x_{t,i} \end{cases} \tag{8.1}$$

Figure 8.7 shows the resulting word error rates for conditional mean imputation (CMI) [199, 200], bounded conditional mean imputation (BCMI) [17] as well as for the diagonal covariance matrix bounded mean imputation (DBMI) technique from [209, 20]. The first thing to notice is that all the feature reconstruction methods clearly outperform the baseline, with relative reductions of 15.3%, 45.6% and 33.3% in WER on the unadapted pass. These results almost completely translate to the second pass, with reductions of 19.7%, 40.1%, 33.0% after CMLLR adaptation. BCMI clearly performs the best. It is followed by DBMI, which performs 18.4% and 11.2% (relative) worse than BCMI but still 21.2% and 16.6% better than CMI (on the first and second pass, respectively).

[3] In our case, the true clean speech and noise spectra are known because we artificially add the noise. The resulting masks should be regarded for what they are: a hypothetical tool for comparing missing feature reconstruction methods under ideal conditions. They have no practical relevance.

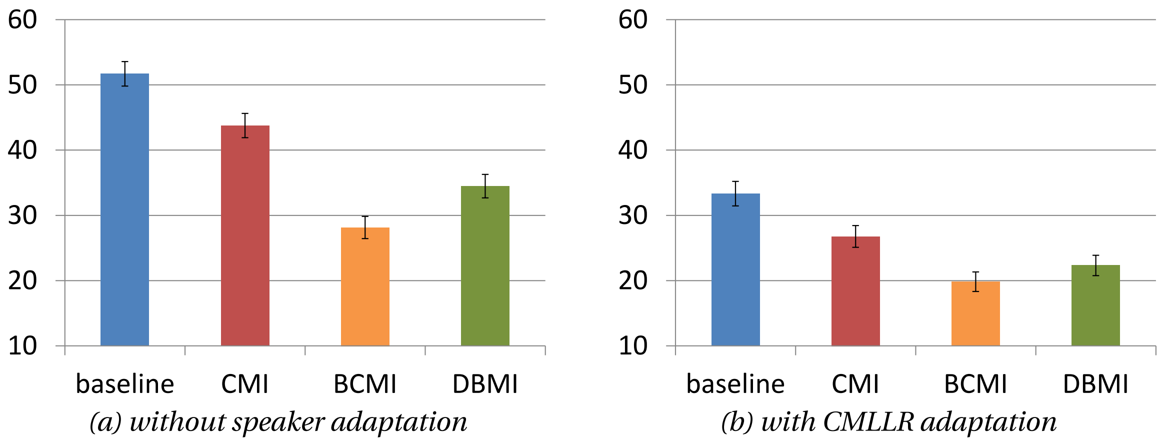


Figure 8.7: *Averaged word error rates for conditional mean imputation (CMI), bounded conditional mean imputation (BCMI) and diagonal (covariance) bounded mean imputation (DBMI), all using oracle masks. The results are shown for both the first, unadapted pass as well as for the second, adapted pass, with 95.4% (2 sigma) error bars, respectively.*

enhancem.	speaker	destroyer engine				factory site				average
method	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
none	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
CMI	none	79.97	61.84	41.08	28.93	57.89	35.68	24.91	19.89	43.77
BCMI	none	54.51	35.27	23.34	17.41	34.21	24.33	19.40	16.66	28.14
DBMI	none	68.01	47.95	32.33	25.10	35.97	26.38	20.08	19.99	34.48
none	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
CMI	CMLLR	57.72	36.14	21.29	14.22	38.41	21.22	13.60	11.59	26.77
BCMI	CMLLR	41.44	23.56	15.91	12.97	24.64	16.70	12.15	11.24	19.83
DBMI	CMLLR	51.79	30.49	18.25	13.43	24.93	16.54	12.41	10.70	22.32

Table 8.4: *Detailed word error rates of the missing feature construction experiments, in dependency of the noise condition.*

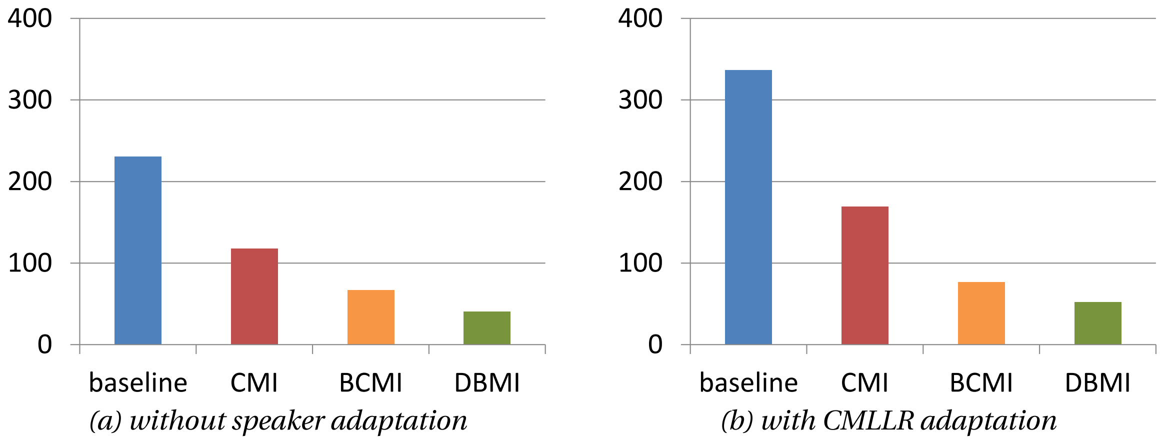
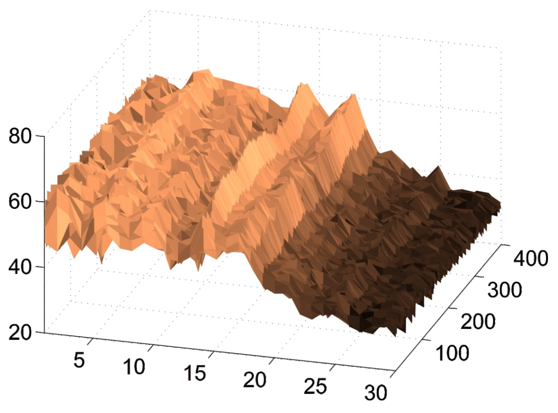
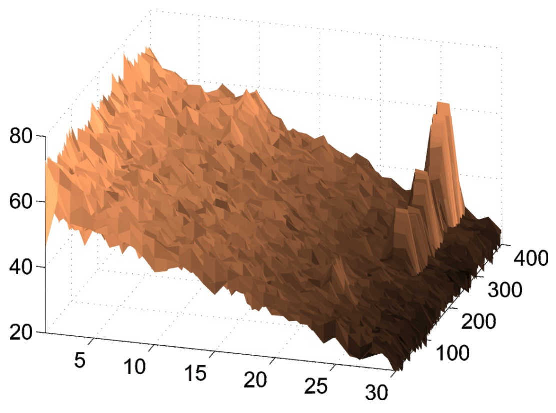


Figure 8.8: Computation times of the missing feature reconstruction experiments. The times are shown in hours, comprise both decoding and feature reconstruction and correspond to processing the whole corpus under all noise conditions.

All the differences in WER are 95.4% (2 sigma) significant, except for the difference between BCMI and DBMI, which is only 68.3% (1 sigma) significant on the second pass. Table 8.4 shows the exact word error rates in dependency of the noise condition. Having a closer look at these numbers, we find that BCMI significantly outperforms DBMI on destroyer engine room noise, while with factory noise both methods perform equally well. This discrepancy may be analyzed at the hand of Figure 8.9, which shows time-frequency representations of destroyer engine and factory noise. Turning to the surface plot on the left, we find that there are two main peaks in the 14th and 17th Mel frequency bins, with center frequencies of 1531 and 2125 Hz, respectively. These peaks explain the higher general WER in case of destroyer engine noise, as the range of 1200-2000 Hz is most important for speech recognition [245]. Now, having a look at Figure 8.10-(a), we see that it is exactly these critical frequencies, which are most often masked by noise. On average, 85.3% of the bins are missing, which leaves us with only 4.4 bins for re-

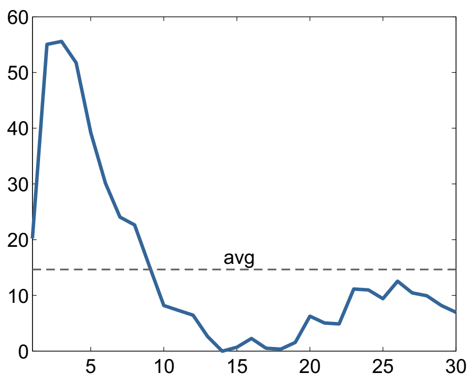


(a) destroyer engine room noise

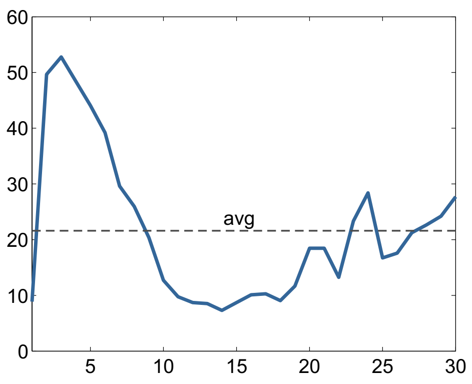


(b) factory site noise

Figure 8.9: *Time frequency representations of the used noise types. The x-axes represent Mel frequency bins (1-30). The y-axes represent the noise intensity in dB. The z-axes represent time in 100 millisecond ticks (1-400).*



(a) destroyer engine room noise



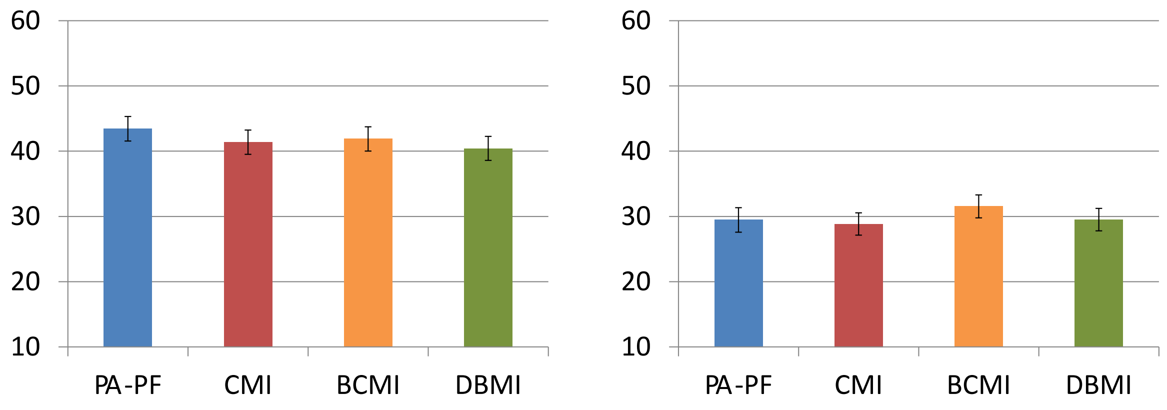
(b) factory site noise

Figure 8.10: *Percentage of time in which a Mel frequency bin was reliable, i.e. not masked by noise. The plots correspond to a signal-to-noise ratio of 5 dB.*

constructing 30-bin speech spectra. This compares to 6.5 bins with 5 dB factory noise (see Figure 8.10-(b)). Another difference between the two noise types is that for destroyer engine noise there are much fewer reliable (non-masked) values in the uppermost 20 frequency bins. This means, most of the reconstruction is based on a few low frequency bins; and it seems BCMI works better under such conditions. Figure 8.8 at the bottom of page 165 finally shows the computation times of the used feature reconstruction techniques in comparison to the baseline. Interestingly, CMI, BCMI and DBMI speed up the recognition by a factor 2.0, 3.4 and 5.6, respectively, on the first pass. On the second pass, the recognition is 2.0, 4.4 and 6.5 times faster. These speed-ups can be explained by the fact that better acoustic separability leads to better pruning during beam-search and, therewith, reductions in decoding time. This is a surprising result, especially regarding the fact that BCMI is computationally by far the most expensive technique, which has been considered in this thesis. It is about 6 times more expensive than CMI and over 50 times more expensive than DBMI.

8.4.5 Particle Filter + Missing Feature Reconstruction

This section considers a combination of the particle filter from Section 8.4.3 with a missing feature reconstruction (MFR) post-processing stage [20]. In this approach, the particle filter is used for both enhancing noisy speech spectra and for estimating masks for the subsequent MFR stage. This is done as described in Section 7.4. Figure 8.11 shows the resulting word error rates in comparison to the phase-averaged particle filter. The first pass results (bar plot to the left) indicate that the additional MFR stage gives relative improvements of 4.7%, 3.6% and 6.9% for CMI, BCMI and DBMI. But these results do not translate well to the second pass (bar plot to the right) where the improvements decrease to 2.1%, -7.1% and -1.0%, respectively.



(a) without speaker adaptation *(b) with CMLLR adaptation*

Figure 8.11: *Average WERs for the phase-averaged particle filter (PA-PF) with a missing feature reconstruction (MFR) post-processor. The masks for MFR were estimated with the approach from Section 7.4. The first bar shows results for the PA-PF only. The other bars show results for combinations with the indicated MFR techniques.*

The difference between CMI and the baseline (the PA-PF) is 1-sigma significant on the first pass and not significant on the second pass. The same holds for DBMI. In case of BCMI, the difference to the baseline is not significant on the first pass but 1-sigma significant on the second pass. These results should not be used to conclude that BCMI performs worse in general with estimated masks, especially not in the light of a more recent study [215, 216], which found that BCMI outperforms all the other methods – even sparse imputation [246] – when mask estimation is performed with the negative energy criterion [212, 166]. Table 8.5 again shows the detailed results.

MFR	speaker	destroyer engine				factory site				average
method	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
none	none	82.11	62.25	42.28	29.56	56.53	33.00	22.92	17.41	43.26
CMI	none	80.28	59.67	37.68	26.35	55.16	32.42	22.56	16.93	41.38
BCMI	none	83.70	59.89	37.80	24.06	57.16	33.71	22.23	16.52	41.88
DBMI	none	79.75	57.21	37.51	25.10	53.93	31.73	21.50	16.75	40.44
none	CMLLR	68.01	40.98	22.15	14.90	42.04	22.21	13.82	11.60	29.46
CMI	CMLLR	65.81	38.35	22.30	14.37	41.71	22.56	13.93	11.65	28.84
BCMI	CMLLR	71.92	43.90	24.28	15.16	46.60	24.08	14.35	12.06	31.54
DBMI	CMLLR	66.75	40.11	22.54	14.95	42.33	23.10	14.32	12.00	29.51

Table 8.5: *Word error rates for the phase-averaged particle filter (PA-PF) with a missing feature reconstruction post-processing stage. The table shows results for conditional mean imputation (CMI), bounded conditional mean imputation (BCMI) and diagonal bounded mean imputation (DBMI) in comparison to the plain PA-PF (none).*

8.4.6 Bayesian Speech Feature Enhancement - Oracle Experiments

The more recent part of this thesis [15, 16] is concerned with the Bayesian speech feature enhancement approach from Section 6.3. This approach necessitates prior knowledge of the clean speech and noise distributions and it uses the assumption that these distributions can be modeled as a Gaussian mixture and a single Gaussian, respectively. In order to evaluate the performance under ideal conditions, let us begin with a set of oracle experiments in which

(1) the channel difference of each speaker from the test corpus to the total average of the training corpus is perfectly known.

(2) the noise distribution of each individual utterance of the test corpus is perfectly known, in contrast to estimating it as in Section 8.4.7.

Throughout this section, the auxiliary clean speech model has been trained after application of an utterance-wise normalization procedure. This normalization procedure ensures that the average spectrum of each utterance in the training corpus coincides with the average spectrum of the total training corpus. This has a similar effect as cepstral mean subtraction (CMS) [233, 234] inasmuch as it decreases the variance due to inter-utterance differences. The differences to CMS are that (1) the method works in the log-Mel domain rather than on cepstra;

and (2) the spectra are normalized to the average clean speech spectrum instead of to zero. Figure 8.12 shows the resulting word error rates for the true MMSE estimate, i.e. the full conditional expectation (FCE) from (6.5), as well as for the mode-dependent bias correction (MDBC) approach from (6.6). In both cases, the joint distribution is constructed with the unscented transform [50], using the zero-phase factor model (Section 5.2.4) as an interaction function. The clean speech model as well as the oracle noise distributions have diagonal covariance matrices.

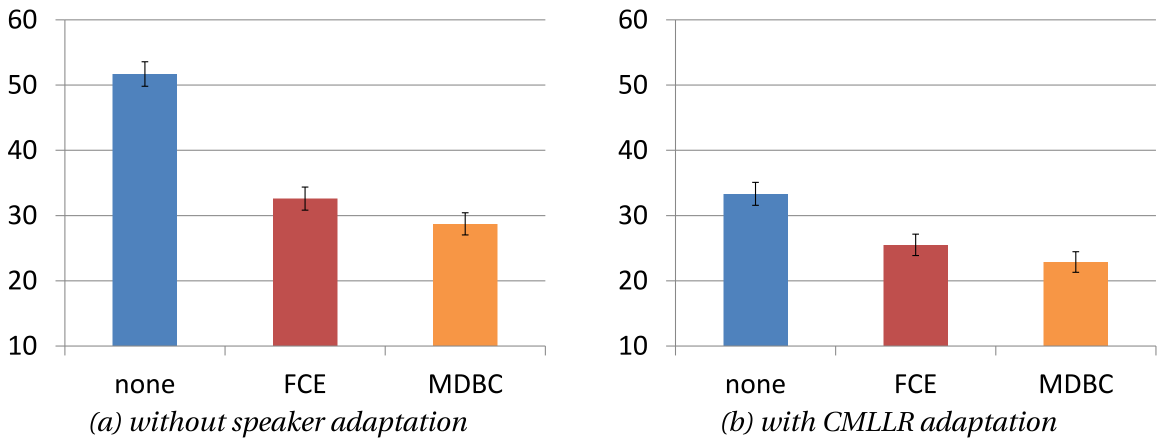


Figure 8.12: *Word error rates for the Bayesian speech feature enhancement approach from 6.3. Results are shown for the full conditional expectation (FCE) as well as for the mode-dependent bias correction (MDBC) approach from* (6.6).

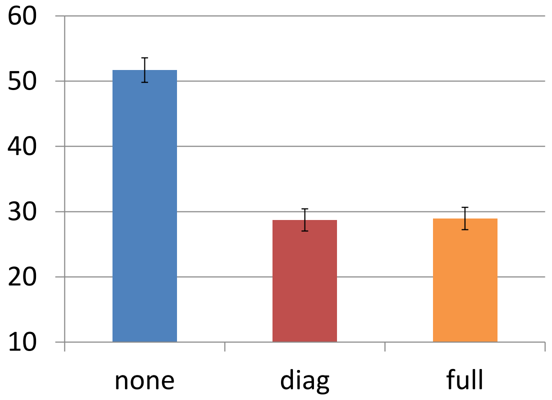
The first thing to notice is that FCE and MDBC perform greatly better than the baseline system, with relative reductions of 37.0% and 44.4% on the first pass and with reductions of 23.5% and 31.4% on the second pass. All the differences in WER are 2-sigma significant, except for the difference between MDBC and FCE, which is only 1-sigma significant on the second pass. The fact that MDBC outperforms FCE comes a little bit as a surprise but we assume that this is a result of stability issues[4], possibly due to a mismatched joint distribution. Table 8.6 again shows the detailed word error rates.

estimation	speaker	destroyer engine room				factory site				average
type	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
—	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
FCE	none	60.10	38.88	25.91	19.82	48.60	29.55	20.08	17.91	32.61
MDBC	none	55.28	34.88	23.48	18.66	39.15	24.50	18.22	15.65	28.73
—	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
FCE	CMLLR	50.31	30.90	20.08	13.79	37.85	23.10	14.87	13.17	25.51
MDBC	CMLLR	46.67	27.63	18.01	13.33	33.48	19.26	13.72	10.94	22.88

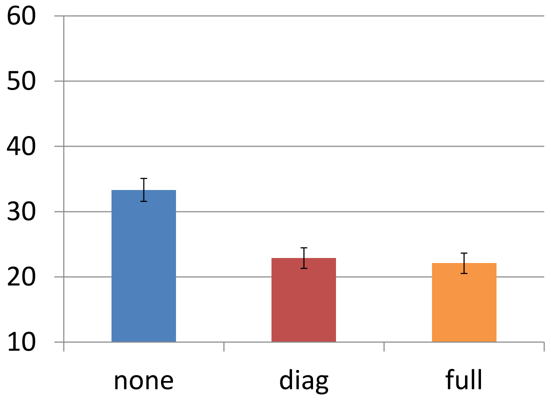
Table 8.6: *Mode-dependent bias correction (MDBC) versus full conditional expectation (FCE).*

[4] This claim is supported by the fact that FCE performed significantly worse than the baseline when full covariance matrices were used. The detailed results of these experiments are not reported here, however.

This table in particular reveals that MDBC outperforms FCE in every single experiment. Hence, only MDBC will be used from now on. In order to justify the use of full covariance matrices in [15, 16], a second set of experiments was performed. In these experiments, all the Gaussians of the clean speech and noise distributions had either diagonal or full covariance matrices. Figure 8.13 shows the results. Although diagonal covariance matrices are marginally better on the first pass, this trend reverses on the second pass. As neither of the differences is statistically significant, we can safely use full covariance matrices – possibly without any benefit but surely without harm either. Table 8.7 again shows the detailed results.



(a) without speaker adaptation



(b) with CMLLR adaptation

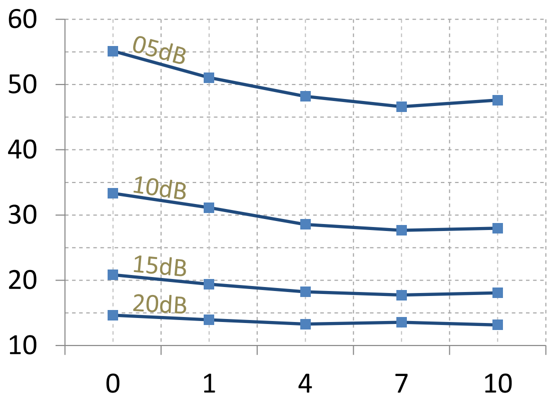
Figure 8.13: *Word error rates for mode-dependent bias correction (MDBC), with full and diagonal (diag) covariance matrices, respectively.*

covariance type	speaker adaptation	destroyer engine room 05dB	10dB	15dB	20dB	factory site 05dB	10dB	15dB	20dB	average WER
—	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
diag	none	55.28	34.88	23.48	18.66	39.15	24.50	18.22	15.65	28.73
full	none	54.70	34.95	24.49	19.33	37.44	24.85	18.90	17.00	28.96
—	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
diag	CMLLR	46.67	27.63	18.01	13.33	33.48	19.26	13.72	10.94	22.88
full	CMLLR	45.03	26.16	17.05	13.04	32.98	17.94	13.23	11.18	22.08

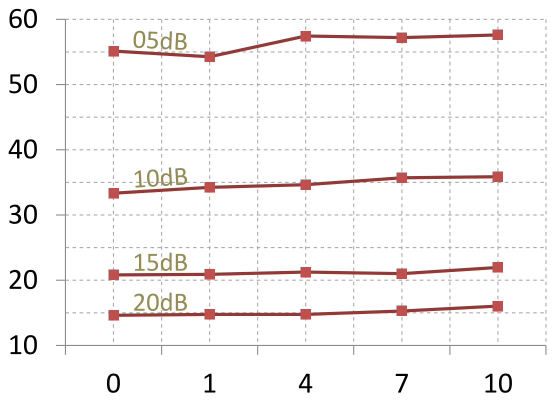
Table 8.7: *Detailed word error rates for the Bayesian speech feature enhancement experiments with full and diagonal (diag) covariance matrices.*

8.4.7 Bayesian Speech Feature Enhancement with Noise Estimation

This section performs experiments with estimated noise distributions while still maintaining the ideal channel. The noise is estimated on an utterance-by-utterance basis, through use of an energy-based voice activity detector (VAD), the Gaussian mixture expectation maximization (GM-EM) approach from Section 6.4.2 or the Monte Carlo EM (MC-EM) approach from Section 6.4.5. In contrast to [15, 16], the 0-th iteration of the EM algorithm is initialized with a VAD-based noise estimate. Figures 8.14 and 8.15 show the resulting word error rates in dependency of the iteration where the 0-th iteration indicates the VAD result. Having a closer look at Figure 8.14-(a) reveals that the Monte-Carlo EM algorithm significantly improves the results over voice activity detection. The WER drops continuously until it settles down in the 7-th iteration, with a relative improvement[5] of 15.5%, 17.0%, 14.8% and 7.4%, respectively, for 5, 10, 15 and 20 dB destroyer engine noise. Figure 8.15-(a) shows the corresponding results for factory noise. Here, we get an improvement of 12.1% at 5 dB but no significant improvements at other SNRs. Figures 8.14-(b) and 8.15-(b) indicate that the Gaussian mixture EM algorithm does not really improve the results over voice activity based noise estimation.

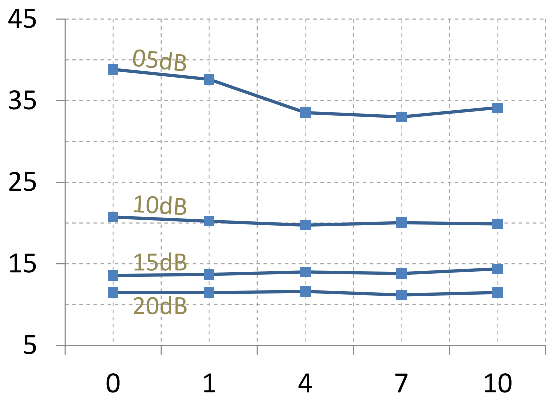


(a) MC-EM, destroyer engine room noise

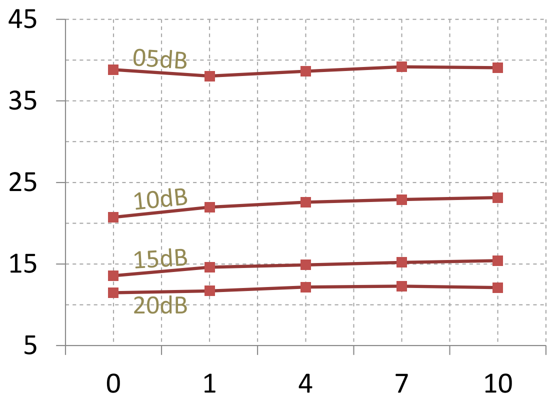


(b) GM-EM, destroyer engine room noise

Figure 8.14: *Word error rates (after CMLLR adaptation) for estimating destroyer engine room noise with the Monte Carlo (MC-EM) and Gaussian mixture EM (GM-EM) algorithms, respectively. Results are shown in dependency of the iteration (0, 1, 4, 7, 10), with the 0th iteration indicating the VAD based noise estimate. Feature enhancement has been performed with the MDBC approach. The joint distribution has been constructed with the unscented transform.*



(a) MC-EM, factory site



(b) GM-EM, factory site

Figure 8.15: *Word error rates (after CMLLR adaptation) for estimating factory site noise with the Monte Carlo (MC-EM) and Gaussian mixture EM (GM-EM) algorithms.*

[5] Note that we here measure the improvements in relation to VAD.

It is quite probable that the bad convergence behavior of the Gaussian mixture EM algorithm is related to the use of full covariance matrices, especially if we regard the fact that speech feature enhancement with FCE performed significantly worse than the baseline when full covariance matrices were used (see footnote on page 168). One explanation for this might be that full covariance matrices are more sensitive to model mismatches (such as neglecting the relative phase) as the full conditional expectation spreads the error across dimensions. On the other hand, there also were good reasons for using full covariance matrices in the first place:

(a) Full covariance matrices should be better in theory as they more accurately model the speech and noise distributions.

(b) Using full covariance matrices is closer to working in the cepstral domain[6], which most of the other authors are doing [123, 177].

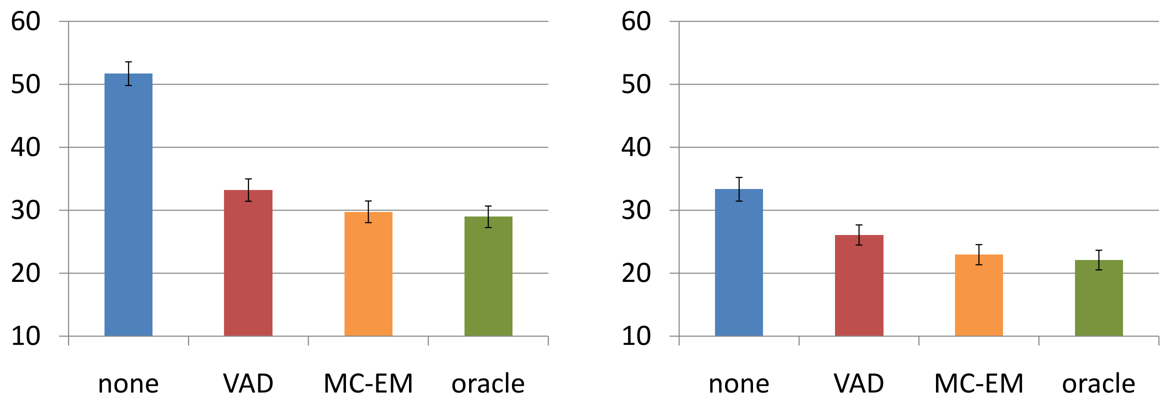
The latter point suggests that the problem with full covariance matrices may also apply to cepstral-based approaches such as [123, 177]. Table 8.8 again shows the detailed word error rates in dependency of the noise condition and iteration.

noise	iteration	destroyer engine room				factory site				average
estimation	number	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
none	—	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
VAD	—	55.13	33.34	20.83	14.63	38.84	20.73	13.55	11.47	26.07
MC-EM	1	51.06	31.13	19.40	13.93	37.61	20.47	13.70	11.45	24.84
MC-EM	4	48.19	28.55	18.22	13.29	33.54	19.74	14.00	11.60	23.39
MC-EM	7	46.60	27.65	17.74	13.55	33.00	20.03	13.79	11.18	22.94
MC-EM	10	47.62	27.97	18.06	13.16	34.13	19.89	14.37	11.48	23.34
GM-EM	1	54.25	34.24	20.90	14.76	38.04	21.98	14.61	11.69	26.31
GM-EM	4	57.43	34.64	21.24	14.75	38.64	22.57	14.88	12.17	27.04
GM-EM	7	57.18	35.71	21.00	15.29	39.17	22.90	15.19	12.27	27.34
GM-EM	10	57.59	35.87	21.96	16.03	39.08	23.15	15.41	12.10	27.65

Table 8.8: *Detailed word error rates for noise estimation with the Monte Carlo and Gaussian mixture EM algorithms. The results refer to the second pass, after CMLLR adaptation.*

As Gaussian mixture based noise estimation performs significantly worse than the Monte Carlo approach, it is no longer considered in the following. Hence, we proceed by comparing the remaining noise estimation techniques to the oracle noise from Section 8.4.6. This is done at the hand of Figure 8.16, which shows word error rates for MDBC based speech feature enhancement with (1) VAD-based noise estimation (2) MC-EM based noise estimation and (3) oracle noise. The first thing to notice is that VAD-based noise estimation gives a decent reduction of the WER, with relative improvements of 35.8% and 21.8% over the baseline on the first and second pass. The MC-EM approach further improves the results. In particular, it comes very close to oracle noise where the relative reductions were 44.0% and 33.8%, respectively.

[6] That is because the translation to the cepstral domain consists in a linear transformation. So, if we translate diagonal Gaussians in the cepstral domain back to the log-Mel domain, the Gaussians have full covariance matrices.

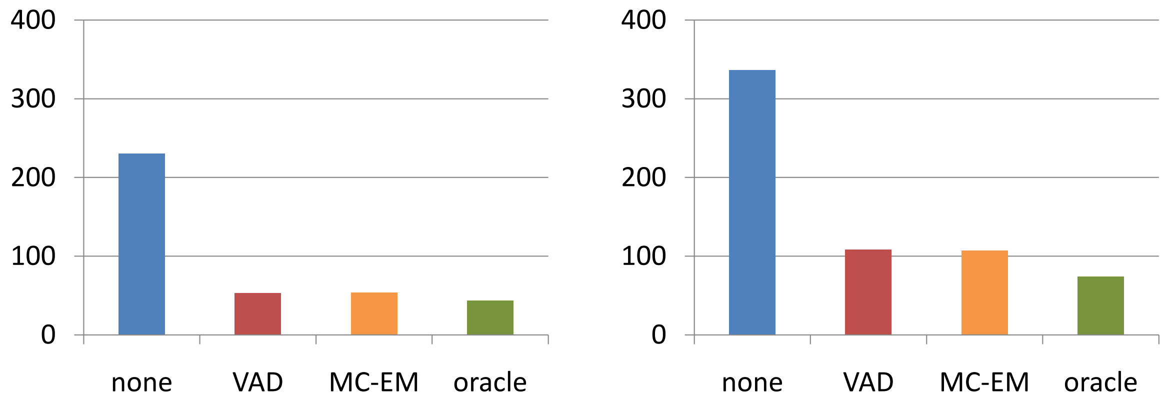


(a) without speaker adaptation *(b) with CMLLR adaptation*

Figure 8.16: *WERs for MDBC using different noise estimation schemes. The bar plots show results for energy based voice activity detection (VAD), the Monte Carlo EM algorithm (MC-EM) and oracle noise. The first bar shows the baseline (none), without speech feature enhancement.*

noise estimation	speaker adaptation	destroyer engine room				factory site				average WER
		05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	
none	none	89.56	74.54	55.21	37.17	63.06	42.86	28.62	22.56	51.70
VAD	none	64.22	42.17	28.21	21.09	44.75	28.40	20.03	16.71	33.26
MC-EM	none	56.41	36.57	24.47	19.05	40.74	25.46	18.66	16.66	29.75
oracle	none	54.70	34.95	24.49	19.33	37.44	24.85	18.90	17.00	28.96
none	CMLLR	76.69	49.01	26.37	16.70	45.75	24.50	15.40	12.18	33.33
VAD	CMLLR	55.13	33.34	20.83	14.63	38.84	20.73	13.55	11.47	26.07
MC-EM	CMLLR	46.60	27.65	17.74	13.55	33.00	20.03	13.79	11.18	22.94
oracle	CMLLR	45.03	26.16	17.05	13.04	32.98	17.94	13.23	11.18	22.08

Table 8.9: *Detailed word error rate for MDBC-based speech feature enhancement using different noise estimation techniques.*



(a) without speaker adaptation *(b) with CMLLR adaptation*

Figure 8.17: Computation times in hours. The times comprise both decoding and speech feature enhancement; and they correspond to processing the whole corpus, under all noise conditions.

While the difference between VAD and MC-EM is almost 2-sigma significant[7], the difference between MC-EM and oracle noise is not significant at all. A comparison of the computation times in Figure 8.9 shows that speech feature enhancement with either VAD or MC-EM based noise estimation speeds up the decoding time by a factor of 4.3 on the first pass compared to the baseline. On the second pass, it is a factor of 3.2. With oracle noise, the speed-up is 5.3 and 4.5, respectively. These results clearly demonstrate that speech feature enhancement has advantages with respect to both the word error rate and computational cost. Figure 8.18 finally gives a direct comparison to the multi-style system from Section 8.4.1. This system has been trained on noisy speech, using the noise types which are present in the test set plus two further noise types (see Section 8.4.1 for details).

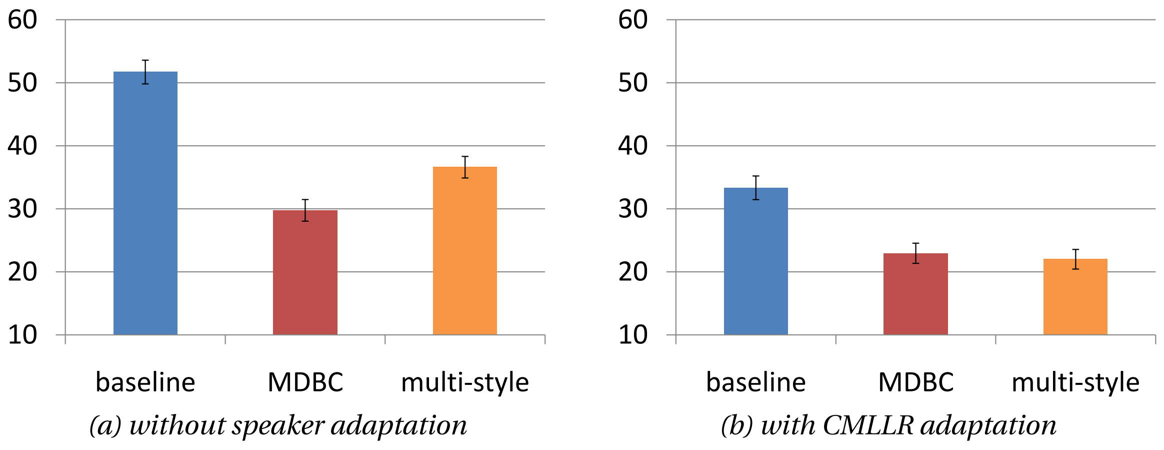


Figure 8.18: *MDBC in direct comparison to the multi-style trained system from Section 8.4.1. MDBC was using the MC-EM noise estimate. The multi-style system was trained on noisy speech only (without enhancement).*

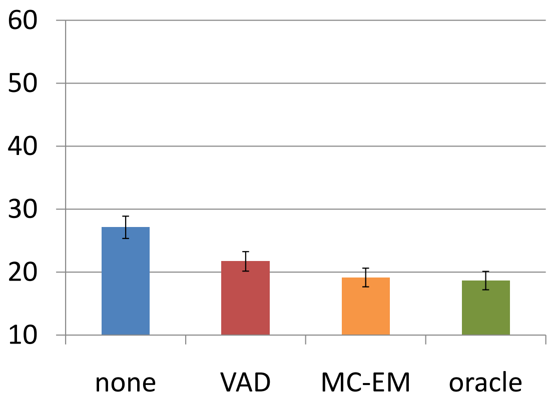
Obviously, the multi-style system performs well, with a relative reduction of more than 30% over the clean speech training baseline. Nevertheless, the multi-style system cannot compete with MDBC based speech feature enhancement – at least not on the first pass. This changes after CMLLR adaptation where the multi-style system is slightly (4.1%) but not significantly better than MDBC. Regarding these results, it is important to note that MDBC just used the baseline system, which has been trained on clean speech (i.e. the baseline system from Section 8.4.1). The noise was estimated with the MC-EM algorithm. The auxiliary clean speech model which was used in all speech feature enhancement and reconstruction experiments had only 128 Gaussian components. This compares to 67017 Gaussians for the clean speech acoustic model and to 80178 Gaussians for the multi-style acoustic model.

8.4.8 Validation on a Larger Set

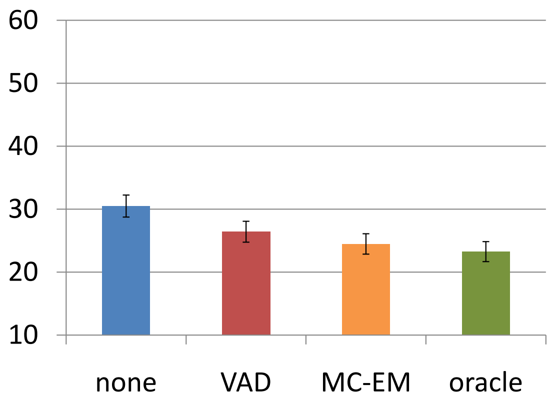
In order to validate the results from the previous section on a larger set, the most important experiments are finally reported under additional noise conditions. In particular, the noise

[7] It is definitely bigger than 1-sigma.

conditions are grouped into two categories: a "stationary" category, which consists of tank (leopard) and destroyer engine room (destroyereng) noise; and a "non-stationary" category, which consists of factory (factory2) and destroyer operations room (destroyerops) noise. Figure 8.19 shows the results on the second pass, after CMLLR adaptation. As expected, the baseline WER (none) is 11.1% higher with non-stationary noise than it is with stationary noise. At the same time, the relative improvement through speech feature enhancement is much lower: 13.4%, 19.7%, 23.7% with non-stationary noise compared to 20.0%, 29.4%, 31.2% with stationary noise, both respectively for MDBC with VAD, MC-EM and oracle noise estimation. In both cases, MC-EM based noise estimation performs significantly better than VAD with 1-sigma significance (with stationary noise it is almost 2-sigma). The difference between oracle noise and MC-EM noise estimation is not significant. The detailed results in tables 8.10 and 8.11 reveal that with tank noise MC-EM even outperforms oracle noise – at least on the first pass.



(a) *average WER with stationary noise*



(b) *average WER with non-stationary noise*

Figure 8.19: *Comparison of word error rates in stationary and non-stationary noise conditions. Results are shown for the second pass only and they are averaged over the sub-conditions of each category. The bars labeled with VAD, MC-EM and oracle again show MDBC results for the indicated noise estimation method. The first bar (none) shows the baseline.*

noise estimation	speaker adaptation	destroyer engine room 05dB	10dB	15dB	20dB	tank interior 05dB	10dB	15dB	20dB	average WER
none	none	89.56	74.54	55.21	37.17	30.98	19.94	15.72	14.08	42.15
VAD	none	64.22	42.17	28.21	21.09	24.73	18.13	16.11	14.05	28.59
MC-EM	none	56.41	36.57	24.47	19.05	22.78	17.91	14.97	14.66	25.85
oracle	none	54.70	34.95	24.49	19.33	24.08	18.46	16.29	14.88	25.90
none	CMLLR	76.69	49.01	26.37	16.70	17.46	11.76	9.76	9.13	27.11
VAD	CMLLR	55.13	33.34	20.83	14.63	17.63	11.55	10.27	10.20	21.70
MC-EM	CMLLR	46.60	27.65	17.74	13.55	15.53	11.26	10.77	10.03	19.14
oracle	CMLLR	45.03	26.16	17.05	13.04	15.93	11.23	10.73	10.03	18.65

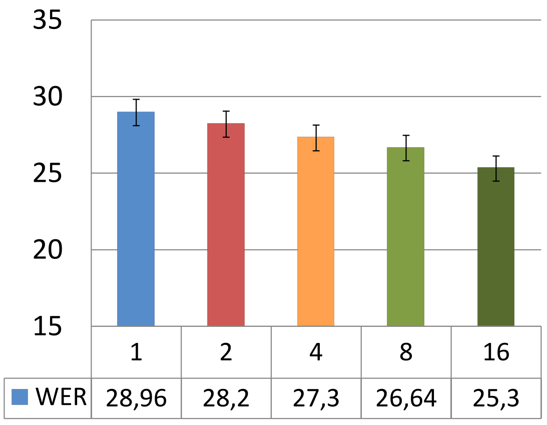
Table 8.10: *Detailed word error rates in stationary noise conditions. Results are shown without speech feature enhancement (none) as well as with MDBC. In the latter case, the noise estimate is obtained through VAD, MC-EM and oracle noise estimation.*

noise	speaker	destroyer operations room				factory site				average
estimation	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
none	none	81.27	56.22	33.71	20.71	63.06	42.86	28.62	22.56	43.63
VAD	none	66.64	43.49	26.86	19.19	44.75	28.40	20.03	16.71	33.26
MC-EM	none	68.63	43.73	26.11	17.91	40.74	25.46	18.66	16.66	32.24
oracle	none	60.65	37.10	23.58	17.62	37.44	24.85	18.90	17.00	29.64
none	CMLLR	70.86	40.74	21.29	13.16	45.75	24.50	15.40	12.18	30.49
VAD	CMLLR	59.67	34.31	19.28	13.40	38.84	20.73	13.55	11.47	26.41
MC-EM	CMLLR	55.35	31.94	17.84	12.66	33.00	20.03	13.79	11.18	24.47
oracle	CMLLR	52.48	29.73	16.80	11.64	32.98	17.94	13.23	11.18	23.25

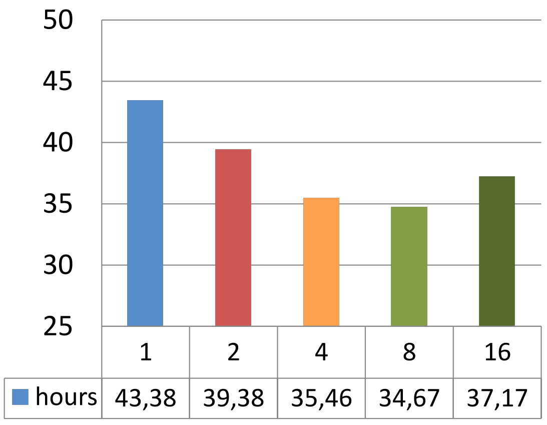
Table 8.11: *Detailed word error rates in non-stationary noise conditions. Results are shown without speech feature enhancement (none) as well as with MDBC. In the latter case, the noise estimate is obtained through VAD, MC-EM and oracle noise estimation.*

8.4.9 Using Gaussian Mixture Noise Distributions

So far, it has been presumed that the distribution of noise can be well approximated as a single Gaussian. But that is clearly violated in non-stationary environments where noise is time-varying. This section challenges the Gaussian approximation and demonstrates that better results can be achieved through use of a Gaussian mixture approximation. This is done at the hand of oracle experiments, in which a Gaussian mixture noise distribution is trained on the true noise signal (once for each utterance). Figure 8.20-(a) shows the resulting word error rates in dependency of the number of Gaussians (1, 2, 4, 8, 16). Roughly, whenever the number of Gaussians is doubled the WER undergoes a relative drop of 3.3% (on average), with the largest drop (of 5.0%) occurring when the number of Gaussians is increased from 8 to 16. In particular, with 16 Gaussians the total reduction is 12.6% compared to using a single Gaussian. This



(a) word error rate



(b) computation time in hours

Figure 8.20: *Word error rates and computation times for speech feature enhancement with oracle Gaussian mixture noise distributions. The results refer to the first pass. The errors bars indicate the 68.3% (1-sigma) confidence intervals.*

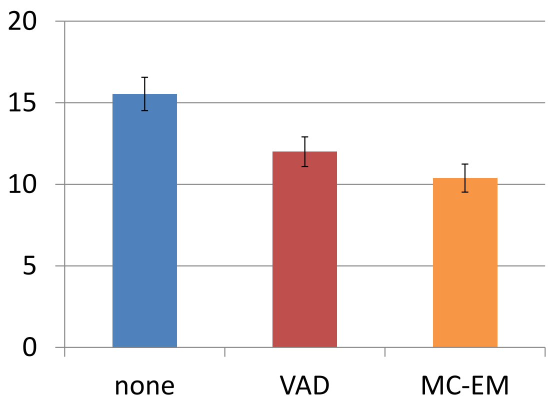
indicates that we have not yet tapped the full potential of the MDBC approach. Figure 8.20-(b) shows the corresponding computation times in hours. The bar plot reveals that the total computational cost – including both speech feature enhancement and decoding – is decreasing with the number of Gaussians; and that although the cost of MDBC linearly increases with the number of Gaussians. The minimum computation time seems to be reached at 8 Gaussians. Table 8.12 again shows the detailed word error rates of the individual experiments. There obviously are large improvements at lower SNRs, especially at 5 dB destroyer engine noise where the relative reduction is 23.1% with 16 Gaussians compared to a single Gaussian. And the fact that we get such large improvements with relatively stationary noise indicates that some of the gains might be due to other factors, such as a a better treatment of the nonlinear interaction function. This reasoning is supported by the fact that a Gaussian mixture approximation of the noise results in more Gaussians with smaller variances, which subsequently leads to lower approximation errors during construction of the joint distribution of speech and noise (see Section 3.8).

number of	speaker	destroyer operations room				factory site				average
Gaussians	adaptation	05dB	10dB	15dB	20dB	05dB	10dB	15dB	20dB	WER
1	none	60.65	37.10	23.58	17.62	37.44	24.85	18.90	17.00	29.64
2	none	54.07	33.82	23.80	19.63	35.90	23.34	18.61	16.44	28.20
4	none	52.02	33.30	23.19	18.01	34.23	23.36	18.56	15.74	27.30
8	none	50.85	31.77	22.61	17.72	32.43	23.55	17.93	16.25	26.63
16	none	46.65	29.99	21.29	17.36	29.90	22.23	18.46	16.54	25.30

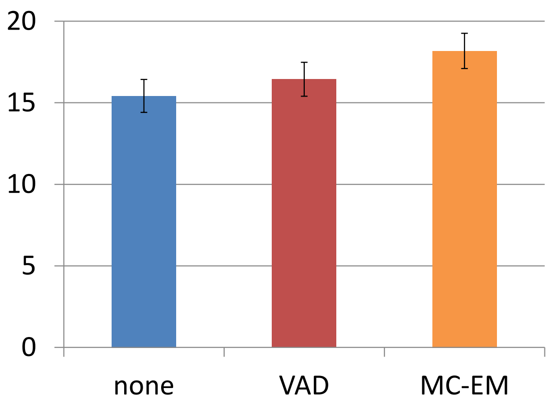
Table 8.12: *Detailed word error rates with Gaussian mixture noise distributions. The given results refer to the first pass; and they are shown in dependency of the number of Gaussians.*

8.5 Experiments with real Noise

This section validates the usefulness of Bayesian speech feature enhancement under real conditions. That is achieved by performing experiments on the AVICAR corpus [240], which has been recorded in several cars driving along a highway with up to 55 miles per hour (see Section 8.3.3 for the details). Based on the results from the previous sections, the noise is estimated with voice activity detection (VAD) or with the Monte-Carlo EM approach from Section 6.4.5. Feature enhancement is performed with mode-dependent bias correction (MDBC). Figure 8.21 shows the resulting word error rates on the connected digits (phone number) task of the "35D" condition. In this condition the cars are driving at 35 miles per hour with the windows down. The signal-to-noise ratio varies between 15 and -10 dB. The bar plots show results for two different feature sets: the LDA features which Millennium uses as a default (see Section 5.1) as well as 13-dimensional Mel frequency cepstral coefficients (MFCCs) with concatenated Delta and Delta-Delta features, which are essentially the first and second order derivatives of the MFCCs with respect to time [103, 104].

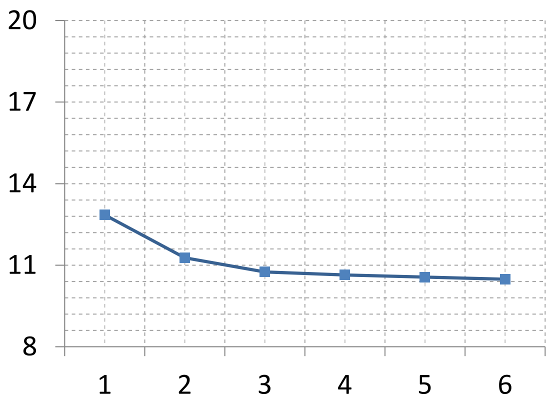


(a) LDA features

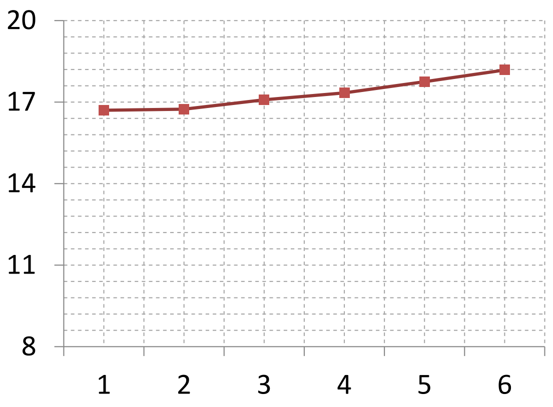


(b) MFCC + Delta features

Figure 8.21: *Word error rates for MDBC based speech feature enhancement with 68.3% (1-sigma) error bars. The noise has been estimate with voice activity detection (VAD) or with the Monte Carlo EM algorithm (MC-EM). The first bar shows the baseline (none), without speech feature enhancement.*



(a) LDA features



(b) MFCC + Delta features

Figure 8.22: *Word error rates for MDBC based speech feature enhancement with MC-EM based noise estimation. The plots show the results for the individual iterations.*

Without speech feature enhancement (none), LDA features perform equally well as MFCC + Delta and Delta-Delta features. This, however, changes after MDBC based speech feature enhancement. Here, LDA features[8] lead to a significant reduction of the word error rate, with relative improvements of 22.8% and 33.2% for VAD and MC-EM based noise estimation. The use of Delta features, on the other hand, increases the word error rate by 7.8% and 19.2%, respectively. This behavior might be explained by the fact that the LDA projects discriminatory irrelevant parts of the feature space to null. Plain MFCC features, on the other hand, can be expected to be more sensitive to inter-frame fluctuations – such as those introduced by log-Mel speech feature enhancement (see time ticks 10 to 65 in Figure 8.23-(b)). This sensitivity may be reduced by bandpass-filtering the MFCCs [187] or by re-synthesizing the speech signal with a spectral envelope Wiener filter [247] which smoothes the clean speech estimate in time.

[8] Note that this is consistent with the previous sections in which LDA features have been used exclusively.

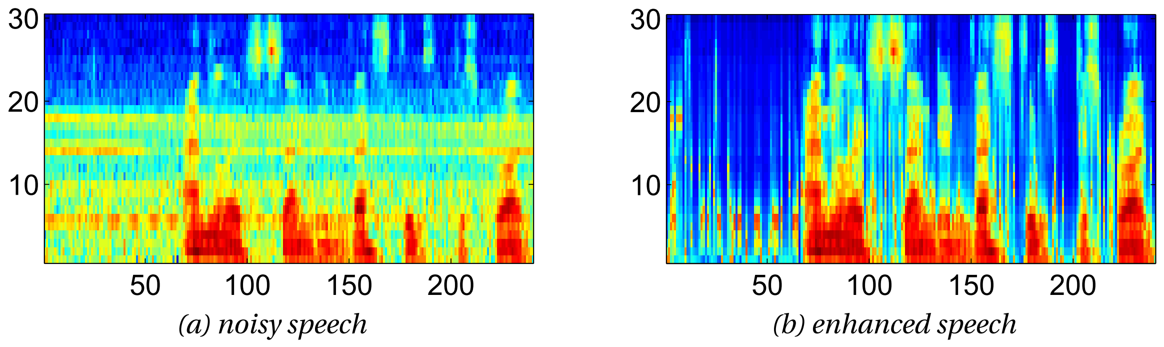


Figure 8.23: *Time-frequency representation of a noisy utterance. The x-axis represents time in 10 millisecond ticks. The y-axis represents Mel frequency bins. The color indicates the intensity in dB, with red marking high energy, yellow and green marking medium energy and blue marking low energy.*

Turning back to Figure 8.21-(a), we see that MC-EM based noise estimation outperforms VAD-based noise estimation. Although the difference between the two methods is not statistically significant, MC-EM gives a 2-sigma significant improvement over the baseline whereas VAD gives a 1-sigma significant improvement only. These results demonstrate that MDBC based speech feature enhancement works in practice. Figure 8.24 shows the computation times in minutes. In case of LDA features, speech feature enhancement with VAD and MC-EM based noise estimation gives a relative speed-up of 7.5% and 16.0%, respectively, compared to the baseline. With MFCC + Delta features, the speed-up is 57.6% and 54.6% (see Figure 8.24-(b)), although the WER does not improve in this case. Regarding the above experiments, it should be mentioned that the ideal channel is not known in this section (in contrast to the experiments with added noise from Section 8.4.6). Hence, speech feature enhancement is performed without channel compensation, using an auxiliary clean speech model that has been trained without the utterance-wise normalization procedure from Section 8.4.6.

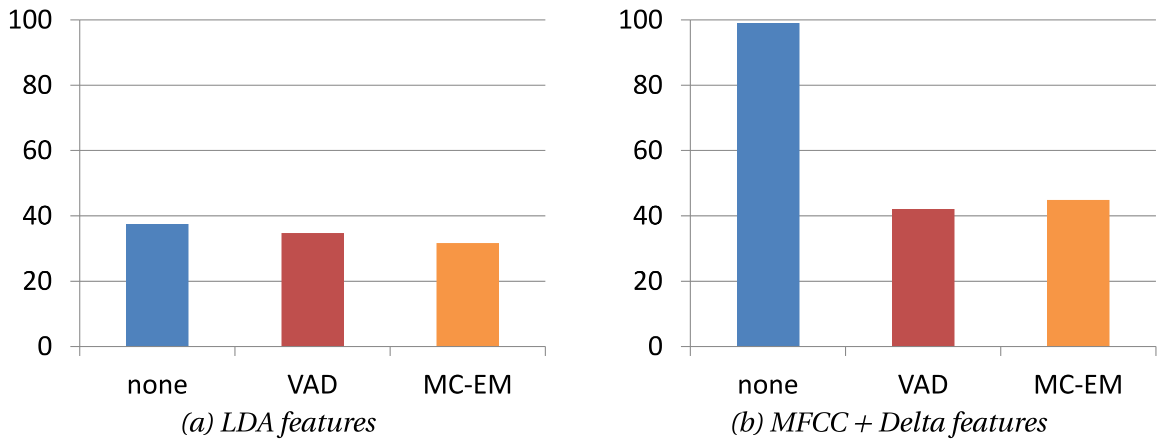


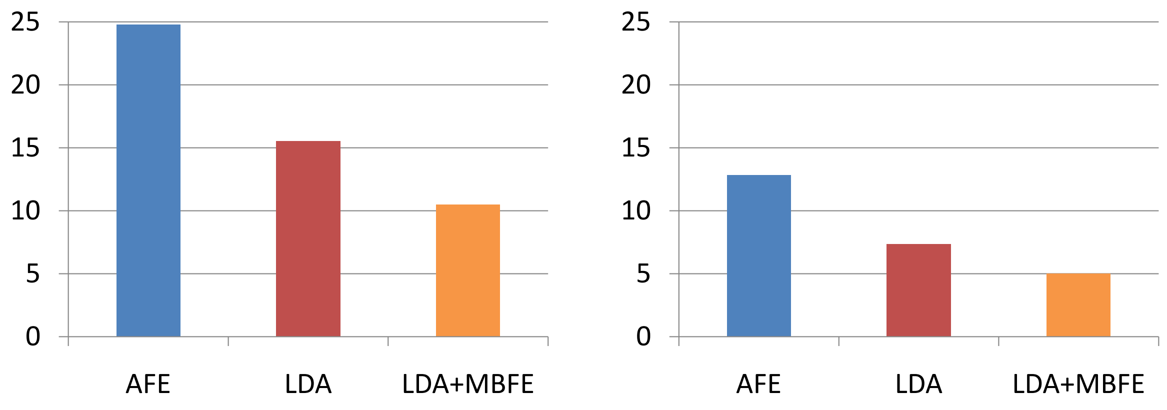
Figure 8.24: *Computation time in minutes for the baseline (none) as well as for MDBC based speech feature enhancement with VAD and MC-EM based noise estimation. Note that the times comprise both decoding and speech feature enhancement.*

8.5.1 Comparison to ETSI AFE Features

Table 8.13 again shows the detailed word error rates for LDA and Delta features in comparison to the ETSI AFE features that have been proposed by the European Telecommunications Standards Institute [243]. In particular note that the table shows results for both the 35D and 35U conditions in which the car is driving at 35 miles per hour with the windows down (D) or up (U). In both cases, the combination of LDA features with MDBC based speech feature enhancement gave significant improvements. ETSI AFE features [243] performed by far the worst – and that although the AFE includes speech enhancement steps such as two-stage Wiener filtering [145] and waveform processing [244]. The results are again subsumed in Figure 8.25; and they stand in contrast to the good performance that the AFE gave on the training corpus, i.e. the IDL condition of the AVICAR corpus for which the WER was 1.28% for the AFE compared to a WER of 2.57% for LDA features.

ASR features	condition	baseline	VAD	MC-EM, iteration				VAD + MC-EM, iteration			
				1	2	4	6	1	2	4	6
AFE	35D	24.78	—	—	—	—	—	—	—	—	—
Delta	35D	15.25	16.44	16.70	16.74	17.34	18.18	19.30	19.36	19.29	19.29
LDA	35D	15.54	12.00	12.85	11.27	10.64	10.38	11.91	11.16	10.70	10.72
AFE	35U	12.82	—	—	—	—	—	—	—	—	—
Delta	35U	7.94	8.97	9.83	9.71	9.65	10.01	9.18	9.29	9.32	9.56
LDA	35U	7.35	5.79	6.47	6.03	5.51	5.30	5.52	5.26	5.07	5.01

Table 8.13: *Detailed word error rates with the ETSI AFE [243], MFCC+Delta and LDA features. Results are shown for the 35D and 35U conditions of the AVICAR corpus, in which the car is driving at 35mph with the windows up (U) or down (D). The VAD column shows WERs for MDBC based speech feature enhancement with VAD based noise estimation. The MC-EM columns show the corresponding results with MC-EM based noise estimation with and without VAD based initialization.*



(a) 35D condition *(b) 35U condition*

Figure 8.25: *Word error rates on the 35D and 35U conditions of the AVICAR corpus, using ETSI AFE features, baseline LDA features and LDA features with MDBC based speech feature enhancement and MC-EM based noise estimation.*

8.6 Summary of the Speech Recognition Results

Regarding the mid-size (5000 word) vocabulary MC-WSJ-AV corpus [117] with artificially added noise from the NOISEX-92 database [118], is has been shown that:

1. Millennium LDA features (see Section 5.1) perform slightly better on noisy speech than the ETSI AFE (Section 8.4.2).
2. particle filter based speech feature enhancement improves the ASR results by $\geq 10\%$ (relative) on the first pass (Section 8.4.3).
3. the phase-averaged model from Section 5.2.3 performs a little better than the zero-phase factor model (Section 8.4.3).
4. with oracle masks, the proposed bounded conditional mean imputation (BCMI) technique from Section 7.3 significantly outperforms the other (considered) missing feature reconstruction techniques (Section 8.4.4).
5. in particular, BCMI reduces the baseline word error rate by 45% (relative) on the first pass and by 40% on the second pass (with CMLLR adaptation); this compares to 33% (on both passes) for bounded mean imputation (DBMI) [209, 20] (Section 8.4.4).
6. at the same time, BCMI lowers the total computation time of speech recognition (including feature reconstruction) by more than 70% (Section 8.4.4).
7. BCMI and DBMI significantly outperform particle filter based speech feature enhancement (with oracle masks).
8. the combination of particle filtering and missing feature reconstruction gives a slight improvement on the first pass only; hence, is not really worthwhile (Section 8.4.4).
9. mode-dependent bias correction (MDBC) from (6.6) performs better than the full conditional expectation (FCE) from (6.5); both perform significantly better than particle filter based speech feature enhancement (Section 8.4.6).
10. for MDBC there is no big difference between using full and diagonal covariance matrices; FCE, however, does not seem to work with full covariance matrices (Section 8.4.6).
11. noise estimation with the Monte Carlo expectation maximization (MC-EM) algorithm from Section 6.4.5 improves the ASR results by more than 10% compared to voice activity (VAD) based noise estimation (Section 8.4.7).
12. MC-EM noise estimation performs almost as well as oracle (perfectly known) noise; in particular, it performs better than the Gaussian mixture EM algorithm from Section 6.4.2 (with full covariance matrices).

13. MDBC performs significantly better than multi-style training on the first pass; it is only a little worse than multi-style training on the second pass.

14. MDBC reduces the total computation time of speech recognition (including feature enhancement) by up to 81% (Section 8.4.7).

15. MDBC with MC-EM noise estimation performs 42.5% and 31.2% (1st and 2nd pass) better than the baseline and therewith performs almost as well as BCMI with oracle masks (although it does not make use of prior knowledge about the noise such as BCMI).

16. BCMI (with oracle masks) and MDBC (with MC-EM based noise estimation) give better ASR results than (unsupervised) speaker-wise CMLLR adaptation, and that at a much lower computation time.

17. combining any of the feature enhancement and reconstruction techniques with CMLLR adaptation further improves the results; in particular, it always results in a lower WER than CMLLR adaptation only.

18. the ASR results and computation times, which are obtained with MDBC based speech feature enhancement, can be further improved by using Gaussian mixture noise distributions (Section 8.4.9).

enhancement method	feature type	training type	noise / mask estimation	word error rate		computation time	
				abs.	impr.	abs.	impr.
none	LDA	clean	none	51.70	0%	230h	0%
none	LDA	multi-style	none	36.56	29.3%	N/A	N/A
CMLLR	LDA	clean	CMLLR	33.33	35.5%	567h	-147%
AFE	AFE	clean	AFE	54.82	-6.0%	622h	-170%
PA-PF	LDA	clean	PF	43.26	16.3%	249h	-8.3%
PA-PF+DBMI	LDA	clean	PF	40.44	21.8%	282h	-22.6%
CMI	LDA	clean	oracle	43.77	15.3%	117h	49.1%
BCMI	LDA	clean	oracle	28.14	45.6%	67h	70.9%
DBMI	LDA	clean	oracle	34.48	33.3%	41h	82.2%
FCE (diag)	LDA	clean	oracle	32.61	36.9%	N/A	N/A
MDBC	LDA	clean	oracle	28.96	44.0%	43h	81.3%
16G-MDBC	LDA	clean	oracle	25.31	51.1%	37h	83.9%
MDBC	LDA	clean	VAD	33.26	35.7%	53h	77.0%
MDBC	LDA	clean	MC-EM	29.75	42.5%	54h	76.5%

Table 8.14: *Summary of the speech recognition results (first pass) with added noise, compared to speaker-wise noise adaptation (CMLLR), multi-style training and the ETSI AFE. The table shows the absolute (abs.) WER and the improvement (impr.) in relation to the baseline (first row). The last two columns show the corresponding computation times in hours. 16G-MDBC denotes MDBC with a 16-component Gaussian mixture noise distribution.*

Regarding the phone number task of the AVICAR corpus [240], which was recorded under real noise conditions, it has been shown that:

1. LDA features and MFCC + Delta features perform equally well without speech feature enhancement, although LDA features result in a lower computation time.
2. in combination with LDA features, MDBC based speech feature enhancement significantly improves the speech recognition results; with MFCC + Delta features, however, it increases the WER.
3. although VAD based noise estimation already gives good results in practice, the proposed MC-EM algorithm further improves the results.
4. the baseline LDA features significantly outperform AFE features on noisy speech (35D and 35U conditions) although, on the training corpus (IDAL condition), the AFE performs twice as well as baseline LDA features.
5. on noisy speech, the total computation time with MBFE (on top of LDA features) is comparable to the computation time with the ETSI-AFE.

enhancement method	feature type	training type	noise / mask estimation	word error rate		computation time	
				abs.	impr.	abs.	impr.
none	LDA	clean	none	15.54	0%	37.5m	0%
MDBC	LDA	clean	VAD	12.00	22.8%	34.7m	7.5%
MDBC	LDA	clean	MC-EM	10.72	31.0%	31.2m	16.8%
none	Delta	clean	none	15.25	1.9%	99.1m	-164%
MDBC	Delta	clean	VAD	16.44	-5.8%	42.0m	-12.0%
MDBC	Delta	clean	MC-EM	19.29	-24.1%	44.9m	-19.7%
AFE	AFE	clean	AFE	24.78	-59.5%	30.3m	19.2%

Table 8.15: *Word Error rates and computation times in minutes for the 35D condition. The improvement (impr.) is measured in comparison to baseline LDA features (first row).*

enhancement method	feature type	training type	noise / mask estimation	word error rate		computation time	
				abs.	impr.	abs.	impr.
none	LDA	clean	none	2.57	0%	9.2m	0%
none	Delta	clean	none	2.89	-12.5%	42.9m	-366%
AFE	AFE	clean	AFE	1.28	50.2%	8.8m	4.3%

Table 8.16: *Word Error rates and computation times in minutes for the IDL condition. The improvement (impr.) is measured in comparison to baseline LDA features (first row).*

9
Conclusions

This work has shown how the problem of speech recognition in noise can be treated within the framework of Bayesian state estimation. That started with **Chapter 3**, which systematically introduced the Bayesian state estimation framework. Next to developing a transformation-centric view, which shifts the whole complexity of estimation to the transformation of random variables, the chapter presented two new transformation methods, namely: the adaptive level of detail transform (Section 3.7) and the extensive unscented transform (Section 3.6). These transformations were shown be more accurate for nonlinear estimation problems, especially those which occur in the context of speech feature enhancement (Section 3.8). **Chapter 4** extended this approach to the sequential level. Apart from deriving all standard tracking algorithms from the transformation-centric point of view, the chapter also described a novel Gaussian mixture filter (Section 4.6.3). This filter was based on the adaptive level of detail transform and it was shown to have a superior performance compared to other nonlinear tracking algorithms (Section 4.8).

Chapter 5 took the first steps towards speech recognition by investigating the effect of noise to clean speech features (Section 5.2.1). This led to the development of a phase-averaged model (Section 5.2.3) which more accurately describes the interaction between speech and noise (see the experimental comparison in Section 5.2.5). Section 5.3.2 also derived the inverse of this model and therewith provided a noise-reduction rule in the feature domain. **Chapters 6 and 7** are the main content sections of this thesis. They describe two principally different approaches for removing noise from noisy speech features, namely:

1. Bayesian speech feature enhancement, which tries to stochastically map noisy speech to

clean speech features (Chapter 6)

2. missing feature reconstruction, which tries to infer the values of noise corrupted frequency bins from non-corrupted ones (Chapter 7).

These approaches are related in that they can both be derived within the framework of Bayesian state estimation, just under different models of how noise affects clean speech features (see Section 7.2). **Chapter 6** started by giving the minimum mean square error estimate of speech feature enhancement (Section 6.3), as well as a computationally efficient approximation thereof. The remaining part of the chapter mostly focused on noise estimation. This included the derivation of a general EM algorithm for noise estimation as well as two new implementations thereof: a Gaussian mixture implementation, which uses the unscented transform (Section 6.4.4), and a Monte Carlo implementation with Parzen window density estimation (Section 6.4.5). Section 6.4.3 showed that these implementation avoid some of the stability issues which are encountered in other work in the literature. In addition to stationary noise estimation with the EM algorithm, Section 6.5 presented a particle filtering approach that uses the phase-averaged interaction function. This approach avoids stability problems due to the relative phase. **Chapter 7** introduced classical missing feature reconstruction methods (Section 7.1) and then went on with deriving a bounded conditional mean imputation technique (Section 7.3). This required the introduction of box-truncated Gaussian distributions (Section 7.3.1).

The experimental evaluation in **Chapter 8** demonstrated that both speech feature enhancement and reconstruction techniques can significantly reduce the word error rate (WER) in noisy environments. The most important outcomes with added noise were:

1. under ideal conditions[1], the bounded conditional mean imputation (BCMI) technique from Section 7.3 outperforms all other feature domain noise reduction approaches at a relative improvement of 45% in WER compared to the baseline (see Section 8.4.4).

2. the speech feature enhancement approach from Section 6.3 achieves almost the same performance – i.e. a 42% reduction in WER – without making use of prior knowledge about the noise[2] (see Section 8.4.7).

3. both approaches reduce the total computing time of speech recognition, as they facilitate better pruning during decoding (see Sections 8.4.4. 8.4.6 and 8.4.7).

4. noise estimation with the Monte-Carlo (MC) EM implementation (Section 6.4.5) comes very close to perfectly knowing the noise (see Sections 8.4.6 and 8.4.7).

Experiments in a real noise environment showed improvements of 33% relative in WER (Section 8.5). Apart from the above, there are some lessons to be learned from the experimental

[1] i.e. when it is perfectly known, which parts of the speech spectrum are masked by noise
[2] meaning this result is much more realistic than the BCMI result from above

chapter. Firstly, the use of full covariance matrices appears to cause stability issues in combination with the minimum mean square error solution from speech feature enhancement (Section 6.3). This result may be explained by the fact that the correlation (which is introduced through full covariance matrices) allows errors to translate across frequency bands. That claim is supported by the fact that the problem is avoided by (a) using the mode-dependent bias correction technique from Section 6.3.2, which essentially throws away the covariance information, or (b) using diagonal covariance matrices instead of full covariance matrices (see the experiments in Section 8.4.6). It is further supported[3] by the good performance of Monte Carlo based noise estimation (Section 8.4.7) compared to the bad performance of Gaussian mixture based noise estimation (with full covariance matrices). Another lesson to be learned from Chapter 8 is that the use of a Gaussian mixture noise distribution can significantly improve the performance of speech feature enhancement (see Section 8.4.9). This approach is, however, hampered by the fact that the Gaussian mixture noise distribution is not so easy to estimate in practice. In principle, the Monte Carlo EM algorithm from Section 6.4.5 could be extended to (1) sample from a Gaussian mixture noise distribution and (2) re-estimating the Gaussian mixture noise distribution from the importance weighted samples by using a second EM algorithm (similar as in [54]). But an initial implementation in the framework of this thesis did not significantly improve the performance compared to a single Gaussian. Hence, Gaussian mixture noise estimation remains an open problem.

[3] Note that EM-based noise estimation essentially uses the same conditional expectation as the MMSE clean speech estimate.

Bibliography

[1] C. A. Halverson, D. B. Horn, C. Marie-Karat, and J. Karat, “Hearing what speech recognition can’t: A language to understand your users’ behavior,” in *Proceedings of the 1999 American Voice Input/Output Society Conference (AVIOS ’99)*, May 1999, pp. 267–276.

[2] S. Xu, S. Basapur, M. Ahlenius, and D. Matteo, “An empirical study on users’ acceptance of speech recognition errors in text-messaging,” in *Proceedings of the 12th International Conference on Human-Computer Interaction (HCI 2007)*, Jul. 2007, pp. 232–242.

[3] S. Choularton and R. Dale, “User responses to speech recognition errors: Consistency of behaviour across domains,” in *Proceedings of the 10th Australian International Conference on Speech Science & Technology (SST 2004)*, Dec. 2004, pp. 457–462.

[4] H. D. Green, “Adding user-friendliness and ease of implementation to continuous speech recognition technology with speech macros: Case studies,” *Journal of Healthcare Information Management*, vol. 18, no. 4, pp. 40–48, 2004.

[5] A. Alapetite, H. B. Andersen, and M. Hertzum, “Acceptance of speech recognition by physicians: A survey of expectations, experiences and social influence,” *International Journal of Human-Computer Studies*, vol. 67, no. 1, pp. 36–49, Jan. 2009.

[6] K. M. Lee and J. Lay, “Speech versus touch: A comparative study of the use of speech and DTMF keypad for navigation,” *International Journal of Human-Computer Interction*, vol. 19, no. 3, pp. 343–360, 2005.

[7] H.-G. Hirsch and D. Pearce, “The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions,” in *Proceedings of the ISCA Tutorial and Research Workshop ASR2000*, Sep. 2000.

[8] S. F. Boll, “Suppression of acoustic noise in speech using spectral subtraction,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 27, no. 2, pp. 113–120, Apr. 1979.

[9] M. W. Callahan, “Acoustic signal processing based on the short-time spectrum,” Ph.D. dissertation, Department of Computer Science, University of Utah, Salt Lake City (UT), USA, Mar. 1976.

[10] J. S. Lim and A. V. Oppenheim, "Enhancement and bandwidth compression of noisy speech," *Proceedings of the IEEE*, vol. 67, no. 12, pp. 1586–1604, Dec. 1979.

[11] A. Acero, "Acoustical and environmental robustness in automatic speech recognition," Ph.D. dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh (PA), USA, 1990.

[12] F. Faubel and D. Klakow, "An adaptive level of detail approach to nonlinear estimation," in *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2010)*, Mar. 2010, pp. 3958–3961.

[13] ——, "Further improvement of the adaptive level of detail transform: Splitting in direction of the nonlinearity," in *Proceedings of the 18th European Signal Processing Conference (EUSIPCO 2010)*, Aug. 2010, pp. 850–854.

[14] F. Faubel, J. McDonough, and D. Klakow, "A phase-averaged model for the relationship between noisy speech, clean speech and noise in the log-mel domain," in *Proceedings of Interspeech 2008*, Sep. 2008, pp. 553–556.

[15] ——, "On expectation maximization based channel and noise estimation beyond the Taylor series expansion," in *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2010)*, Mar. 2010, pp. 4294–4297.

[16] F. Faubel and D. Klakow, "Estimating noise from noisy speech features with a Monte Carlo variant of the expectation maximization algorithm," in *Proceedings of Interspeech 2010*, Sep. 2010, pp. 2046–2049.

[17] F. Faubel, J. McDonough, and D. Klakow, "Bounded conditional mean imputation with Gaussian mixture models : A reconstruction approach to partly occluded features," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, Apr. 2009, pp. 3869–3872.

[18] F. Faubel and D. Klakow, "A transformation-based derivation of the Kalman filter and an extensive unscented transform," in *Proceedings of the 15th Workshop on Statistical Signal Processing (SSP 2009)*. IEEE, Sep. 2009, pp. 161–164.

[19] F. Faubel, J. McDonough, and D. Klakow, "The split and merge unscented Gaussian mixture filter," *Signal Processing Letters*, vol. 16, no. 9, pp. 786–789, Sep. 2009.

[20] F. Faubel, H. Raja, J. McDonough, and D. Klakow, "Particle filter based soft-mask estimation for missing feature reconstruction," in *Proceedings of the International Workshop on Acoustic Echo and Noise Control (IWAENC 2008)*, Sep. 2008.

[21] P. S. Laplace, *A Philosophical Essay on Probabilities.* New York (NY), USA: John Wiley and Sons, 1902.

[22] S. B. McGrayne, *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy.* Yale University Press, 2011.

[23] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, 2nd ed., ser. Springer Texts in Statistics. Springer, 2004.

[24] J. A. Shohat and J. D. Tamarkin, *The Problem of Moments*, ser. Mathematical Surveys and Monographs. New York, USA: American Mathematical Society, 1943.

[25] J. S. Christiansen, "Indeterminate moment problems within the Askey-scheme," Ph.D. dissertation, Institute for Mathematical Sciences, University of Copenhagen, Denmark, 2004.

[26] A. D. Moivre, *The doctrine of chances: or, a method for calculating the probabilities of events in play*, 2nd ed. London, UK: Woodfall, 1738.

[27] C. F. Gauss, *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium.* Hamburg, Germany: Friedrich Perthes and I. H. Besser, 1809.

[28] C. H. Davis, *Theory of the Motion of the Heavenly Bodies Moving About the Sun in Conic Sections - A Translation of Gauss's "Theoria Motus".* Boston (MA), USA: Little, Brown and Company, 1857.

[29] F. Y. Edgeworth, "On the probable errors of frequency constants," *Journal of the Royal Statistical Society*, vol. 71, no. 3, pp. 381–397, 499–512, 651–678, Sep. 1908.

[30] ——, "Addendum on probable errors of frequency constants," *Journal of the Royal Statistical Society*, vol. 72, no. 1, pp. 81–90, Mar. 1909.

[31] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov Chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, Feb. 1970.

[32] A. P. Dempster, N. M. Laird., and D. B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, May 1977.

[33] N. Metropolis, "The beginning of the Monte Carlo method," *Los Alamos Science, Special Issue*, pp. 125–130, 1987.

[34] L. Devroye, *Nonuniform Random Variate Generation.* Springer, 1986.

[35] G. Marsaglia and A. Zaman, "The KISS generator," Department of Statistics, Florida State University, Tallahassee, FL, USA, Tech. Rep., 1993.

[36] G. Marsaglia and W. W. Tsang, “The Ziggurat method for generating random variables,” *Journal of Statistical Software*, vol. 5, no. 8, Oct. 2000.

[37] G. Marsaglia, “Generating a variable from the tail of the normal distribution,” *Technonometrics*, vol. 6, pp. 101–102, 1964.

[38] D. B. Rubin, “Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm,” *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 543–546, Jun. 1987.

[39] M. A. Tanner and W. H. Wong, “The calculation of posterior distributions by data augmentation,” *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 528–540, Jun. 1987.

[40] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME - Journal of Basic Engineering, Series D*, vol. 82, pp. 35–45, Mar. 1960.

[41] L. A. McGee and S. F. Schmidt, “Discovery of the Kalman filter as a practical tool for aerospace and industry,” National Aeronautics and Space Administration, Ames Research Center, Tech. Rep. NASA-TM-86847, Nov. 1985.

[42] E. Brookner, *Tracking and Kalman Filtering Made Easy.* John Wiley & Sons, 1998.

[43] S. O. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice Hall, 2001.

[44] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* John Wiley & Sons, 2006.

[45] P. J. Moreno, B. Raj, and R. M. Stern, “A vector Taylor series approach for environment-independent speech recognition,” in *Proceedings of the 1996 IEEE International Conference on Audio, Speech and Language Processing (ICASSP ’96)*, vol. 2, May 1996, pp. 733–736.

[46] P. J. Moreno, “Speech recognition in noisy environments,” Ph.D. dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh (PA), U.S.A., 1996.

[47] H. W. Sorenson and D. L. Alspach, “Recursive Bayesian estimation using Gaussian sums,” *Automatica*, vol. 7, no. 4, pp. 465–479, Jul. 1971.

[48] D. L. Alspach and H. W. Sorenson, “Nonlinear Bayesian estimation using Gaussian sum approximations,” *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, Aug. 1972.

[49] M. Afify, X. Cui, and Y. Gao, “Stereo-based stochastic mapping for robust speech recognition,” in *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, vol. 4, Apr. 2007, pp. 377–380.

[50] Y. Shinohara and M. Akamime, “Bayesian feature enhancement using a mixture of unscented transformations for uncertainty decoding,” in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, Apr. 2009, pp. 4569–4572.

[51] S. J. Julier and J. K. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in *Proceedings of the 1997 SPIE AeroSense Symposium*, Apr. 1997, pp. 182–193.

[52] ——, “Unscented filtering and nonlinear estimation,” *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[53] R. van der Merwe and E. A. Wan, “The square-root unscented Kalman filter for state and parameter-estimation,” in *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, vol. 6, May 2001, pp. 3461 – 3464.

[54] R. van der Merwe and E. Wan, “Gaussian mixture sigma-point particle filters for sequential probabilistic inference in dynamic state-space models,” in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2003)*, vol. 4, Apr. 2003, pp. 701–704.

[55] R. van der Merwe, “Sigma-point Kalman filters for probabilistic inference in dynamic state-space models,” Ph.D. dissertation, OGI School of Science and Engineering, Oregon Health & Science University, Hillsboro (OR), USA, 2004.

[56] K. Ito and K. Xiong, “Gaussian filters for nonlinear filtering problems,” *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 910–927, May 2000.

[57] S. Weisberg, *Applied Linear Regression*, 3rd ed., ser. Wiley Series in Probability and Statistics. John Wiley & Sons, 2005.

[58] D. J. Salmond, “Mixture reduction algorithms for uncertain tracking,” Royal Aerospace Establishment, Tech. Rep. TR 88004, Jan. 1988.

[59] ——, “Mixture reduction algorithms for target tracking in clutter,” in *Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets*, vol. 1305, Jan. 1990, pp. 434–445.

[60] J. L. Williams and P. S. Maybeck, “Cost-function-based Gaussian mixture reduction,” in *Proceedings of the Sixth International Conference on Information Fusion*, vol. 2, Mar. 2003, pp. 1047–1054.

[61] A. R. Runnalls, "Kullback-Leibler approach to Gaussian mixture reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989–999, Jul. 2007.

[62] U. D. Hanebeck, K. Briechle, and A. Rauh, "Progressive Bayes: A new framework for nonlinear state estimation," in *Proceedings of the 2003 SPIE AeroSense Symposium*, vol. 5099, May 2003, pp. 256–267.

[63] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for Gaussian mixture random vectors," in *Proceedings of the 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2008)*, Aug. 2008, pp. 181–188.

[64] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton, "Split and merge EM algorithm for improving Gaussian mixture density estimates," *Journal of VLSI Signal Processing Systems*, vol. 26, no. 1-2, pp. 133–140, 2000.

[65] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[66] J. R. Hershey and P. A. Olsen, "Approximating the Kullback-Leibler divergence between Gaussian mixture models," in *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, vol. 4, Apr. 2007, pp. 317–320.

[67] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. McGraw-Hill, 2002.

[68] G. Welch and G. Bishop, "An introduction to the Kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill (NC), USA, Tech. Rep. TR 95-041, Jul. 2006.

[69] J. L. Doob, *Stochastic Processes*, ser. Wiley Classics Library. John Wiley & Sons, 1953.

[70] H. W. Bode and C. E. Shannon, "A simplified derivation of linear least-squares smoothing and prediction theory," *Proceedings of the Institute of Radio Engineers*, vol. 38, pp. 417–425, Apr. 1950.

[71] R. E. Bellman, I. L. Glicksberg, and O. A. Gross, "Some aspects of the mathematical theory of control processes," RAND Corporation, USA, Tech. Rep. RAND Report R-313, 1958.

[72] R. E. Kalman and R. W. Koepcke, "Optimal synthesis of linear sampling control systems using generalized performance indexes," *Transactions of the American Society of Mechanical Engineers*, vol. 80, pp. 1820–1826, Nov. 1958.

[73] Y. Ho and R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, no. 4, pp. 333–339, Oct. 1964.

[74] R. J. Meinhold and N. D. Singpurwalla, “Understanding the Kalman filter,” *The American Statistician*, vol. 37, no. 2, pp. 123–127, May 1983.

[75] M. Athans, R. Wishner, and A. Bertolini, “Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements,” *IEEE Transactions on Automatic Control*, vol. 13, no. 5, pp. 504–514, Oct. 1968.

[76] L. Deng, J. Droppo, and A. Acero, “Enhancement of log mel power spectra of speech using a phase-sensitive model of the acoustic environment and sequential estimation of the corrupting noise,” *IEEE Transactions on Speech and Audio Processing*, vol. 12, pp. 133–143, Mar. 2004.

[77] P. Bruneau, M. Gelgon, and F. Picarougne, “Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach,” *Pattern Recognition*, vol. 43, no. 3, pp. 850–858, Mar. 2010.

[78] J. Goldberger, S. Gordon, and H. Greenspan, “An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures,” in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV 2003)*, vol. 2, Oct. 2003, pp. 487–493.

[79] J. Goldberger, H. K. Greenspan, and J. Dreyfuss, “Simplifying mixture models using the unscented transform,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1496–1502, Aug. 2008.

[80] Y. Kubo, T. Sato, and S. Sugimoto, “Modified Gaussian sum filtering methods for INS/GPS integration,” *Journal of Global Positioning Systems*, vol. 6, no. 1, pp. 65–73, 2007.

[81] I. Arasaratnam, S. Haykin, and R. J. Elliott, “Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature,” *Proceedings of the IEEE*, vol. 95, no. 5, pp. 953–977, May 2007.

[82] I. Arasaratnam and S. Haykin, “Cubature Kalman filters,” *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, Jun. 2009.

[83] J. H. Kotecha and P. M. Djuric, “Gaussian particle filtering,” *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, Oct. 2003.

[84] U. D. Hanebeck and O. Feierman, “Progressive Bayesian estimation for nonlinear discrete-time systems: The filter step for scalar measurements and multidimensional states,” in *Proceedings of the 42nd IEEE Conference on Decision and Control (CDC 2003)*, vol. 5, Dec. 2003, pp. 5366–5371.

[85] M. Huber, D. Brunn, and U. D. Hanebeck, "Closed-form prediction of nonlinear dynamic systems by means of Gaussian mixture approximation of the transition density," in *Proceedings of the 2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2006)*, Sep. 2006, pp. 98–103.

[86] V. Klumpp, F. Sawo, U. D. Hanebeck, and D. Fränken, "The sliced Gaussian mixture filter for efficient nonlinear estimation," in *Proceedings of the 11th International Conference on Information Fusion (FUSION 2008)*, Jul. 2008, pp. 24–31.

[87] W. I. Tam, K. N. Plataniotis, and D. Hatzinakos, "An adaptive Gaussian sum algorithm for radar tracking," *Signal Processing*, vol. 77, no. 1, pp. 85–104, Aug. 1999.

[88] N. M. Kwok, Q. P. Ha, S. Huang, G. Dissanayake, and G. Fang, "Mobile robot localization and mapping using a Gaussian sum filter," *International Journal of Control, Automation, and Systems*, vol. 5, no. 3, pp. 251–268, Jun. 2007.

[89] M. Flament, G. Fleury, and M.-E. Davoust, "Particle filter and Gaussian-mixture filter efficiency evaluation for terrain-aided navigation," in *Proceedings of the 12th European Signal Processing Conference (EUSIPCO 2004)*, Sep. 2004, pp. 605–608.

[90] R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan, "The unscented particle filter," Cambridge University Engineering Department, Cambridge, UK, Tech. Rep. CUED/F-INFENG/TR-380, Aug. 2000.

[91] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEEE Proceedings on Radar and Signal Processing*, vol. 140, pp. 107–113, Sep. 1993.

[92] J. F. de Freitas, M. Niranjan, and A. H. Gee, "Hybrid sequential Monte Carlo / Kalman methods to train neural networks in non-stationary environments," in *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '99)*, vol. 2, Mar. 1999, pp. 1057–1060.

[93] J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, Sep. 1998.

[94] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Cambridge University Engineering Department, Cambridge, UK, Tech. Rep. CUED/F-INFENG/TR 310, 1998.

[95] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[96] P. Müller, “Monte Carlo integration in general dynamic models,” *Contemporary Mathematics*, vol. 115, pp. 145–163, 1991.

[97] M. West, “Mixture models, Monte Carlo, Bayesian updating and dynamic models,” *Computing Science and Statistics*, vol. 24, pp. 325–333, 1992.

[98] A. Doucet, S. Godsill, and C. Andrieu, “On sequential Monte Carlo sampling methods for Bayesian filtering,” *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, Jul. 2000.

[99] J. F. G. de Freitas, M. A. Niranjan, A. H. Gee, and A. Doucet, “Sequential Monte Carlo methods to train neural network models,” *Neural Computation*, vol. 12, no. 5, pp. 955–993, 2000.

[100] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[101] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice Hall, 2001.

[102] M. Wölfel and J. McDonough, *Distant Speech Recognition.* John Wiley & Sons, 2009.

[103] S. Sagayama and F. Itakura, “On individuality in a dynamic measure of speech,” in *Proceedings of the 1979 Spring Conference of the Acoustical Society of Japan*, Jun. 1979, pp. 589–590.

[104] S. Furui, “Speaker-independent isolated word recognition using dynamic features of speech spectrum,” *IEEE Transactions on Acoustics, Speech and Signal Processsing*, vol. 34, no. 1, pp. 52–59, Feb. 1986.

[105] G. Yu, W. Russell, R. Schwartz, and J. Makhoul, “Discriminant analysis and supervised vector quantization for continuous speech recognition,” in *Proceedings of the 1990 International Conference of Acoustics, Speech and Signal Processing (ICASSP ’90)*, Apr. 1990, pp. 685–688.

[106] R. Haeb-Umbach and H. Ney, “Linear discriminant analysis for improved large vocabulary continuous speech recognition,” in *Proceedings of the 1992 International Conference of Acoustics, Speech and Signal Processing (ICASSP ’92)*, vol. 1, Mar. 1992, pp. 13–16.

[107] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, “Maximum likelihood discriminant feature spaces,” in *Proceedings of the 2000 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP 2000)*, vol. 2, Jun. 2000, pp. 1129–1132.

[108] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. Academic Press, 1990.

[109] J. P. Openshaw and J. S. Mason, "On the limitations of cepstral features in noise," in *Proceedings of the 1994 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP '94)*, vol. 2, Apr. 1994, pp. 49–52.

[110] C. Avendano, S. Tibrewala, and H. Hermansky, "Multiband and adaptation approaches to robust speech recognition," in *Proceedings of Eurospeech 1997*, Sep. 1997, pp. 1107–1110.

[111] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, pp. 133–147, Aug. 1998.

[112] A. Sehr, M. Zeller, and W. Kellermann, "Distant-talking continuous speech recognition based on a novel reverberation model in the feature domain," in *Proceedings of Interspeech 2006*, Sep. 2006, pp. 769–772.

[113] A. Sehr, "Reverberation modeling for robust distant-talking speech recognition," Ph.D. dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, Oct. 2009.

[114] A. Sehr, R. Maas, and W. Kellermann, "Reverberation model-based decoding in the logmelspec domain for robust distant-talking speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1676–1691, Sep. 2010.

[115] L. Deng, J. Droppo, and A. Acero., "A Bayesian approach to speech feature enhancement using the dynamic cepstral prior," in *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, vol. 1, Mar. 2002, pp. 829–832.

[116] T. T. Kristjansson, "Speech recognition in adverse environments: a probabilistic approach," Ph.D. dissertation, Department of Computer Science, University of Waterloo, Waterloo, Canada, 2002.

[117] M. Lincoln, I. McCowan, J. Vepa, and H. K. Maganti, "The multi-channel Wall Street Journal audio visual corpus (MC-WSJ-AV): specification and initial experiments," in *Proceedings of the 2005 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2005)*, Nov. 2005, pp. 357–362.

[118] A. Varga and H. J. M. Steeneken, "Assessment for automatic speech recognition: II. NOISEX-92: a database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Communication*, vol. 12, pp. 247–251, Jul. 1993.

[119] V. Leutnant and R. Haeb-Umbach, "An analytic derivation of a phase-sensitive observation model for noise robust speech recognition," in *Proceedings of Interspeech 2009*, Sep. 2009, pp. 2395–2398.

[120] D. Van Compernolle, "Increased noise immunity in large vocabulary speech recognition with the aid of spectral subtraction," in *Proceedings of the 1987 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP '87)*, vol. 12, Apr. 1987, pp. 1143–1146.

[121] A. Acero and R. Stern, "Environmental robustness in automatic speech recognition," in *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '90)*, vol. 2, Apr. 1990, pp. 849–852.

[122] N. S. Kim, "Statistical linear approximation for environment compensation," *IEEE Signal Processing Letters*, vol. 5, no. 1, pp. 8–10, Jan. 1998.

[123] ——, "Nonstationary environment compensation based on sequential estimation," *IEEE Signal Processing Letters*, vol. 5, no. 3, pp. 57–59, Mar. 1998.

[124] D. Y. Kim, C. K. Un, and N. S. Kim, "Speech recognition in noisy environments using first-order vector Taylor series," *Speech Communication*, vol. 24, pp. 39–49, May 1998.

[125] K. Yao, B. E. Shi, S. Nakamura, and Z. Cao, "Residual noise compensation by a sequential EM algorithm for robust speech recognition in nonstationary noise," in *Proceedings of Interspeech 2000*, vol. 1, Oct. 2000, pp. 770–773.

[126] K. Yao, K. K. Paliwal, and S. Nakamura, "Sequential noise compensation by a sequential Kullback proximal algorithm," in *Proceedings of Interspeech 2001*, Sep. 2001, pp. 1139–1142.

[127] B. Raj, R., Singh, and R. Stern, "On tracking noise with linear dynamical system models," in *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004)*, vol. 1, May 2004, pp. 965–968.

[128] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in *Proceedings of the 1979 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '79)*, vol. 4, Apr. 1979, pp. 208–211.

[129] D. Klatt, "A digital filter bank for spectral matching," in *Proceedings of the 1976 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '76)*, vol. 1, Apr. 1976, pp. 573–576.

[130] J. S. Bridle and et al., "A noise compensating spectrum distance measure applied to automatic speech recognition," in *Proceedings of the Institute of Acoustics Autumn Conference*, vol. 6, Nov. 1984, pp. 307–314.

[131] J. N. Holmes and N. C. Sedgwick, "Noise compensation for speech recognition using probabilistic models," in *Proceedings of the 1986 IEEE International Conference on Acoustics, Speech and Language Processing (ICASSP '86)*, vol. 11, Apr. 1986, pp. 741–744.

[132] A. Nadas, D. Nahamoo, and M. A. Picheny, "Speech recognition using noise-adaptive prototypes," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 10, pp. 1495–1503, Oct. 1989.

[133] R. Haeb-Umbach and J. Schmalenstroeer, "A comparison of particle filtering variants for speech feature enhancement," in *Proceedings of Interspeech 2005*, Sep. 2005, pp. 913–916.

[134] F. Faubel and M. Wölfel, "Overcoming the vector Taylor series approximation in speech feature enhancement – a particle filter approach," in *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, vol. 4, Apr. 2007, pp. 557–560.

[135] R. Singh and B. Raj, "Tracking noise via dynamical systems with a continuum of states," in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2003)*, vol. 1, Apr. 2003, pp. 396–399.

[136] F. Faubel and M. Wölfel, "Coupling particle filters with automatic speech recognition for speech feature enhancement," in *Proceedings of Interspeech 2006*, Sep. 2006, pp. 37–40.

[137] F. Faubel, "Speech feature enhancement for speech recognition by sequential Monte Carlo methods," Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, Germany, Aug. 2006.

[138] M. R. Schröder, "U. S. patent no. 3,180,936," filed Dec. 1960, issued Apr. 1965.

[139] M. R. Weiss, E. Aschkenasy, and T. W. Parsons, "Processing speech signals to attenuate interference," in *Proceedings of the IEEE Symposium on Speech Recognition*, Apr. 1974, pp. 292–295.

[140] ——, "Study and development of the INTEL technique for improving speech intelligibility," Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base (NY), USA, Tech. Rep. RADC-TR-75-108, Apr. 1975.

[141] N. Wiener, *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* John Wiley & Sons, 1949.

[142] G. Neben, R. J. McAulay, and C. J. Weinstein, "Experiments in isolated word recognition using noisy speech," in *Proceedings of the 1983 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP '83)*, vol. 8, Apr. 1983, pp. 1156–1159.

[143] S. F. Boll and R. E. Wohlford, "Event driven speech enhancement," in *Proceedings of the 1983 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP '83)*, vol. 8, Apr. 1983, pp. 1152–1155.

[144] J. E. Porter and S. F. Boll, “Optimal estimators for spectral restoration of noisy speech,” in *Proceedings of the 1984 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP ’84)*, vol. 9, Mar. 1984, pp. 53–56.

[145] A. Agarwal and Y. Cheng, “Two-stage mel-warped Wiener filter for robust speech recognition,” in *Proceedings of the 1999 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU ’99)*, Dec. 1999, pp. 67–70.

[146] Y. Ephraim and D. Mallah, “Speech enhancement using a minimum mean-square error log-spectral amplitude estimator,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 2, pp. 443–445, Apr. 1985.

[147] R. Gemello, F. Mana, and R. De Mori, “Automatic speech recognition with a modified Ephraim-Malah rule,” *IEEE Signal Processing Letters*, vol. 13, no. 1, pp. 56–59, Jan. 2006.

[148] R. Martin, “Noise power spectral density estimation based on optimal smoothing and minimum statistics,” *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, Jul. 2001.

[149] H.-G. Kim, M. Schwab, N. Moreau, and T. Sikora, “Speech enhancement of noisy speech using log-spectral amplitude estimator and harmonic tunneling,” in *Proceedings of the International Workshop on Acoustic Echo and Noise Control (IWAENC 2003)*, Sep. 2003, pp. 119–122.

[150] H. G. Hirsch and C. Ehrlicher, “Noise estimation techniques for robust speeech recognition,” in *Proceedings of the 1995 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP ’95)*, vol. 1, May 1995, pp. 153–156.

[151] N. W. D. Evans, J. S. D. Mason, W. M. Liu, and B. Fauve, “On the fundamental limitations of spectral subtraction: An assessment by automatic speech recognition,” in *Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2005)*, Sep. 2005.

[152] J. F. et. al, *Automatic Speech Recognition in Severe Environments.* Washington DC, USA: National Academic Press, 1984.

[153] C. A. Simpson, C. R. Coler, and E. M. Huff, “Human factors of voice I/O for aircraft cockpit controls and displays,” in *Proceedings of the Workshop on Standardization of Speech I/O Technology.* Gaithersburg (MD), USA: National Bureau of Standards, 1982, pp. 159–166.

[154] Z. A. Kersteen, “An evaluation of automatic speech recognition under three ambient noise levels,” in *Proceedings of the Workshop on Standardization of Speech I/O Technology.* Gaithersburg (MD), USA: National Bureau of Standards, 1982, pp. 63–68.

[155] R. P. Lippmann, M. M. Mack, and D. P. Paul, “Multi-style training for robust speech recognition under stress,” *Journal of the Acoustical Society of America*, vol. 79, no. 1, pp. 95–95, Apr. 1986.

[156] R. P. Lippmann, E. A. Martin, and D. P. Paul, “Multi-style training for robust isolated-word speech recognition,” in *Proceedings of the 1987 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’87)*, vol. 12, Apr. 1987, pp. 705–208.

[157] J. Jaschul, “Speaker adaptation by a linear transformation with optimised parameters,” in *Proceedings of the 1982 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP ’82)*, vol. 7, May 1982, pp. 1657–1660.

[158] V. V. Digalakis, D. Rtischev, and L. G. Neumeyer, “Speaker adaptation using constrained estimation of Gaussian mixtures,” *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 5, pp. 357–366, Sep. 1995.

[159] C. J. Leggetter and P. C. Woodland, “Speaker adaptation of continuous density HMMs using multivariate linear regression,” in *Proceedings of the 1994 International Conference on Spoken Language Processing (ICSLP ’94)*, Sep. 1994, pp. 451–454.

[160] ——, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models,” *Computer Speech & Language*, vol. 9, pp. 171–185, Apr. 1995.

[161] M. J. F. Gales and P. C. Woodland, “Mean and variance adaptation within the MLLR framework,” *Computer, Speech and Language*, vol. 10, no. 4, pp. 249–264, Oct. 1996.

[162] V. N. Parikh, B. Raj, and R. M. Stern, “Speaker adaptation and environmental compensation for the 1996 Broadcast News task,” in *Proceedings of the 1997 DARPA Speech Recognition Workshop*, Feb. 1997.

[163] M. Yamada, A. Baba, S. Yoshizawa, Y. Mera, A. Lee, H. Saruwatari, and K. Shikano, “Unsupervised noisy environment adaptation algorithm using MLLR and speaker selection,” in *Proceedings of Interspeech 2001*, Sep. 2001, pp. 869–872.

[164] M. Matassoni, M. Omologo, A. Santarelli, and P. Svaizer, “On the joint use of noise reduction and MLLR adaptation for in-car hands-free speech recognition,” in *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, vol. 1, May 2002, pp. 289–292.

[165] D. Van Compernolle, “Spectral estimation using a log-distance error criterion applied to speech recognition,” in *Proceedings of the 1989 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP ’89)*, vol. 1, May 1989, pp. 258–261.

[166] L. Josifovski, M. Cooke, P. Green, and A. Vizinho, “State based imputation of missing data for robust speech recognition and speech enhancement,” in *Proceedings of Eurospeech 1999*, Sep. 1999, pp. 2837–2840.

[167] A. P. Varga and R. K. Moore, “Hidden Markov model decomposition of speech and noise,” in *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’90)*, vol. 2, Apr. 1990, pp. 845–848.

[168] M. J. F. Gales and S. Young, "An improved approach to the hidden Markov model decomposition of speech and noise," in *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '92)*, vol. 1, Mar. 1992, pp. 233–236.

[169] M. J. F. Gales and S. J. Young, "HMM recognition in noise using parallel model combination," in *Proceedings of Eurospeech 1993*, Sep. 1993, pp. 837–840.

[170] ——, "Robust continuous speech recognition using parallel model combination," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 352–359, Sep. 1996.

[171] M. J. F. Gales, "Model-based techniques for noise robust speech recognition," Ph.D. dissertation, Gonville and Caius College, University of Cambridge, Cambridge, UK, 1996.

[172] R. C. Van Dalen, "Statistical models for noise robust speech recognition," Ph.D. dissertation, Queens' College, University of Cambridge, Cambridge, UK, 2011.

[173] B. Raj, M. L. Seltzer, and R. M. Stern, "Reconstruction of missing features for robust speech recognition," *Speech Communication*, vol. 43, no. 4, pp. 275–296, Sep. 2004.

[174] A. Erell and M. Weintraub, "Spectral estimation for noise-robust speech recognition," in *Proceedings of the 1989 Workshop On Speech And Natural Language (HLT '89)*, Feb. 1989, pp. 319–324.

[175] B. J. Frey, L. Deng, A. Acero, and T. Kristjansson, "ALGONQUIN: Iterating Laplace's method to remove multiple types of acoustic distortion for robust speech recognition," in *Proceedings of Interspeech 2001*, Sep. 2001, pp. 901–904.

[176] V. Stouten, H. Van Hamme, and P. Wambacq, "Effect of phase-sensitive environment model and higher order VTS on noisy speech feature enhancement," in *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005)*, vol. 1, Mar. 2005, pp. 433–436.

[177] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "High-performance HMM adaptation with joint compensation of additive and convolutive distortions via vector Taylor series," in *Proceedings of the 2007 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2007)*, Dec. 2007, pp. 65–70.

[178] H. Liao and M. J. F. Gales, "Joint uncertainty decoding for noise robust speech recognition," in *Proceedings of Interspeech 2005*, Sep. 2005, pp. 3129–3132.

[179] ——, "Issues with uncertainty decoding for noise robust automatic speech recognition," *Speech Communication*, vol. 50, no. 4, pp. 265–277, Apr. 2008.

[180] M. Afify, X. Cui, and Y. Gao, "Stereo-based stochastic mapping for robust speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 7, pp. 1325–1334, Sep. 2009.

[181] L. Deng, A. Acero, M. Plumpe, and X. Huang, "Large-vocabulary speech recognition under adverse acoustic environments," in *Proceedings of Interspeech 2000*, vol. 3, Oct. 2000, pp. 806–809.

[182] L. Deng, A. Acero, L. Jiang, J. Droppo, and X. Huang, "High-performance robust speech recognition using stereo training data," in *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, vol. 1, May 2001, pp. 301–304.

[183] R. C. Rose, E. M. Hofstetter, and D. A. Reynolds, "Integrated models of signal and background with application to speaker identification in noise," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 245–257, Apr. 1994.

[184] N. S. Kim, "IMM-based estimation for slowly evolving environments," *IEEE Signal Processing Letters*, vol. 5, no. 6, pp. 146–149, Jun. 1998.

[185] V. Stouten, "Robust automatic speech recognition in time-varying environemnts," Ph.D. dissertation, ESAT, Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 2006.

[186] J. Du and Q. Huo, "A feature compensation approach using high-order vector Taylor series approximation of an explicit distortion model for noisy speech recognition," in *Proceedings of Interspeech 2008*, Sep. 2008, pp. 1257–1260.

[187] J. C. Segura, A. de la Torre, M. C. Benitez, and A. M. Peinado, "Model-based compensation of the additive noise for continuous speech recognition. experiments using the AURORA II database and tasks," in *Proceedings of Interspeech 2001*, Sep. 2001, pp. 221–224.

[188] J. Lim and A. Oppenheim, "All-pole modeling of degraded speech," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 3, pp. 197–210, Jun. 1978.

[189] M. Feder, , A. V. Oppenheim, and E. Weinstein, "Maximum likelihood noise cancellation using the EM algorithm," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 2, pp. 204–216, Feb. 1989.

[190] J. Li, D. Yu, Y. Gong, and L. Deng, "Unscented transform with online distortion estimation for HMM adaptation," in *Proceedings of Interspeech 2010*, Sep. 2010, pp. 1660–1663.

[191] A. F. M. Smith and A. E. Gelfand, "Bayesian statistics without tears: A sampling-resampling perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, May 1992.

[192] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962.

[193] K. Yao and S. Nakamura, “Sequential noise compensation by sequential Monte Carlo method,” in *Proceedings of the 14th Conference on Neural Information Processing Systems (NIPS*2001)*, Dec. 2001.

[194] M. Fujimoto and S. Nakamura, “Particle filter based non-stationary noise tracking for robust speech recognition,” in *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005)*, vol. 1, Mar. 2005, pp. 257–260.

[195] ——, “Particle filtering and Polyak averaging-based non-stationary noise tracking for ASR in noise,” in *Proceedings of the 2005 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2005)*, Nov. 2005, pp. 337–342.

[196] M. Wölfel and F. Faubel, “Considering uncertainty by particle filter enhanced speech features in large vocabulary continuous speech recognition,” in *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, vol. 4, Apr. 2007, pp. 1049–1052.

[197] M. Wölfel, “Predicted walk with correlation in particle filter speech feature enhancement for robust automatic speech recognition,” in *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, Apr. 2008, pp. 4705–4708.

[198] ——, “Integration of the predicted walk model estimate into the particle filter framework,” in *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, Apr. 2008, pp. 4725–4728.

[199] M. Cooke, A. Morris, and P. Green, “Missing data techniques for robust speech recognition,” in *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’97)*, vol. 2, Apr. 1997, pp. 863–866.

[200] A. Morris, M. Cooke, and P. Green, “Some solutions to the missing feature problem in data classification, with application to noise robust ASR,” in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’98)*, vol. 2, May 1998, pp. 737–740.

[201] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data.* John Wiley & Sons, 1987.

[202] ——, *Statistical Analysis with Missing Data*, 2nd ed. John Wiley & Sons, 2002.

[203] S. Ahmad and V. Tresp, “Some solutions to the missing feature problem in vision,” in *Proceedings of the 5th Conference on Neural Information Processing Systems (NIPS*1992)*, Dec. 1992, pp. 393–400.

[204] M. P. Cooke, P. D. Green, and M. Crawford, “Handling missing data in speech recognition,” in *Proceedings of the 3rd International Conference on Speech and Language Processing (ICSLP ’94)*, Sep. 1994, pp. 1555–1558.

[205] P. D. Green, M. P. Cooke, and M. D. Crawford, “Auditory scene analysis and hidden Markov model recognition of speech in noise,” in *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’95)*, vol. 1, May 1995, pp. 401–404.

[206] B. Raj, R. Singh, and R. M. Stern, “Inference of missing spectrographic features for robust speech recognition,” in *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP ’98)*, Dec. 1998.

[207] B. Raj, *Reconstruction of Incomplete Spectrograms for Robust Speech Recognition.* Pittsburgh (PA), USA: Department of Electrical and Computer Engineering, Carnegie Mellon University, 2000.

[208] S. F. Buck, “A method of estimation of missing values in multivariate data suitable for use with an electronic computer,” *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 22, no. 2, pp. 302–306, 1960.

[209] B. Raj and R. Singh, “Reconstructing spectral vectors with uncertain spectrographic masks for robust speech recognition,” in *Proceedings of the 2005 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2005)*, Nov. 2005, pp. 65–70.

[210] J. K. Patel and C. B. Read, *Handbook of the Normal Distribution*, 2nd ed., ser. Statistics: Textbooks and Monographs. Marcel Dekker Inc., 1996.

[211] A. Genz, “Numerical computation of multivariate normal probabilities,” *Journal of Computational and Graphical Statistics*, vol. 1, no. 2, pp. 141–149, Jun. 1992.

[212] A. Drygajlo and M. El-Maliki, “Speaker verification in noisy environments with combined spectral subtraction and missing feature theory,” in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’98)*, vol. 1, May 1998, pp. 121–124.

[213] J. Barker, L. Josifovski, M. Cooke, and P. Green, “Soft decisions in missing data techniques for robust automatic speech recognition,” in *Proceedings of Interspeech 2000*, vol. 1, Oct. 2000, pp. 373–376.

[214] M. L. Seltzer, B. Raj, and R. M. Stern, “A Bayesian classifier for spectrographic mask estimation for missing feature speech recognition,” *Speech Communication*, vol. 43, no. 4, pp. 373–393, Sep. 2004.

[215] M. Braun, “A comparative study of missing feature imputation techniques for noise-robust speech recognition,” M.Eng., Department of Physics and Mechatronics, Saarland University, Saarbrücken, Germany, 2012.

[216] M. Braun, F. Faubel, and D. Klakow, “A comparative study of missing feature imputation techniques,” in *Proceedings of the 10th ITG Symposium of Speech Communication*, Sep. 2012.

[217] J.-C. Junqua, “The influence of acoustics on speech production: A noise-induced stress phenomenon known as the Lombard reflex,” *Speech Communication*, vol. 20, pp. 13–22, Nov. 1996.

[218] J. Hansen, “Analysis and compensation of stressed and noisy speech with application to robust automatic speech recognition,” Ph.D. dissertation, Georgia Institute of Technology, Atlanta (GA), U.S.A., 1988.

[219] S. Young and P. Woodland, “The hidden Markov model toolkit (HTK),” Cambridge, UK.

[220] J. L. Phillips, *How to Think about Statistics*, 6th ed. Henry Holt and Company, LLC, 1999.

[221] R. C. Brinker and R. Minnick, *The Surveying Handbook*, 2nd ed. Kluwer Academic Publishers, 1995.

[222] D. Klakow and J. Peters, “Testing the correlation of word error rate and perplexity,” *Speech Communication*, vol. 38, pp. 19–28, Sep. 2002.

[223] “Millennium ASR toolkit,” http://distantspeechrecognition.sourceforge.net, [Online; accessed 03-June-2015].

[224] “The python programming language,” http://www.python.org, [Online; accessed 27-January-2013].

[225] G. Saon, D. Povey, and G. Zweig, “Anatomy of an extremely fast LVCSR decoder,” in *Proceedings of Interspeech 2005*, Sep. 2005, pp. 549–552.

[226] J. McDonough, E. Stoimenov, and D. Klakow, “An algorithm for fast composition of weighted finite-state transducers,” in *Proceedings of the 2007 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2007)*, Dec. 2007, pp. 461–466.

[227] A. Ljolje, F. Pereira, and M. Riley, “Efficient general lattice generation and rescoring,” in *Proceedings of Eurospeech 1999*, vol. 3, Sep. 1999, pp. 1251–1254.

[228] “Cmu sphinx,” http://cmusphinx.sourceforge.net, [Online; accessed 26-January-2013].

[229] “ARPAbet,” https://en.wikipedia.org/wiki/Arpabet, [Online; accessed 26-January-2013].

[230] D. B. Paul and J. M. Baker, “The design for the Wall Street Journal-based CSR corpus,” in *Proceedings of the 2nd International Conference on Spoken Language Processing (ICSLP ’92)*, Oct. 1992, pp. 899–902.

[231] J. Fransen, D. Pye, T. Robinson, P. Woodland, and S. Young, “WSJCAM0 corpus and recording description,” Cambridge University Engineering Department (CUED) Speech Group, Trumpington Street, Cambridge CB2 1PZ, UK, Tech. Rep. CUED/F-INFENG/TR.192, Sep. 1994.

[232] M. J. F. Gales, “Maximum likelihood linear transformations for HMM-based speech recognition,” *Computer Speech & Language*, vol. 12, no. 2, pp. 75–98, Apr. 1998.

[233] B. S. Atal, “Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification,” *Journal of the Acoustical Society of America*, vol. 55, no. 6, pp. 1304–1312, Jun. 1974.

[234] ——, “Automatic recognition of speakers from their voices,” *Proceedings of the IEEE*, vol. 64, no. 4, pp. 460–475, Apr. 1976.

[235] A. E. Rosenberg, C.-H. Lee, and F. K. Soong, “Cepstral channel normalization techniques for HMM-based speaker verification,” in *Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP ’94)*, Sep. 1994, pp. 1835–1838.

[236] M. Westphal, “The use of cepstral means in conversational speech recognition,” in *Proceedings of Eurospeech 1997*, Sep. 1997, pp. 1143–1146.

[237] K. Kumatani, J. W. McDonough, B. Rauch, P. N. Garner, W. Li, and J. Dines, “Maximum kurtosis beamforming with the generalized sidelobe canceller,” in *Proceedings of Interspeech 2008*, Sep. 2008, pp. 423–426.

[238] K. Kumatani, J. McDonough, B. Rauch, D. Klakow, P. N. Garner, and W. Li, “Beamforming with a maximum negentropy criterion,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 5, pp. 994–1008, Jul. 2009.

[239] M. J. F. Gales and Y.-Q. Wang, “Model-based approaches to handling additive noise in reverberant environments,” in *Proceedings of the 2011 Workshop on Hands-Free Speech Communication and Microphone Arrays*, May 2011, pp. 121–126.

[240] B. Lee et al., “AVICAR: Audio-visual speech corpus in a car environment,” in *Proceedings of Interspeech 2004*, Oct. 2004, pp. 2489–2492.

[241] F. Faubel, M. Georges, K. Kumatani, A. Bruhn, and D. Klakow, “Improving hands-free speech recognition in a car through audio-visual voice activity detection,” in *Proceedings of the 2011 Joint Workshop on Hands-free Speech Communication and Microphone Arrays*, May 2011, pp. 70–75.

[242] T. Kleinschmidt et al., “A continuous speech recognition evaluation protocol for the AVICAR database,” in *Proceedings of the 1st International Conference on Signal Processing and Communication Systems*, Dec. 2007.

[243] *ETSI ES-202-050, version 1.1.5, “Standard for Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms”*, European Telecommunications Standards Institute, 2007.

[244] D. Macho and Y. M. Cheng, “SNR-dependent waveform processing for improving the robustness of ASR front-end,” in *Proceedings of the 2001 IEEE International Conference of Acoustics, Speech and Signal Processing (ICASSP 2001)*, vol. 1, May 2001, pp. 305–308.

[245] S. R. Sharma, “Multi-stream approach to robust speech recognition,” Ph.D. dissertation, Oregon Graduate Institute of Science and Technology, Portland (OR), USA, Oct. 1999.

[246] J. F. Gemmeke, B. Cranen, and U. Remes, “Sparse imputation for large vocabulary noise robust ASR,” *Computer Speech & Language*, vol. 25, no. 2, pp. 462–479, Apr. 2011.

[247] N. Hadir, F. Faubel, and D. Klakow, “A model-based spectral envelope Wiener filter for perceptually motivated speech enhancement,” in *Proceedings of Interspeech 2011*, Aug. 2011, pp. 213–216.