Speech Feature Enhancement for Speech Recognition by Sequential Monte Carlo Methods

Friedrich Faubel
Betreuer: Matthias Wölfel, Prof. Alex Waibel
Institut fuer Theorethische Informatik
Universität Karlsruhe (TH), Germany

9. 8. 2006

Declarations

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne unzulässige fremde Hilfe angefertigt habe. Die verwendeten Literaturquellen sind im Literaturverzeichnis vollständig aufgeführt.

Pittsburgh, den 9. 8. 2006

Acknowledgments

I would like to thank the following people for their help and support: Matthias Wölfel for being a good adviser, giving me a great degree of freedom, for being patient, encouraging and helping me to write the paper for interspeech, which we finished in a Sunday night session at 4 am in the morning (I don't know too many advisers who would do that). Prof. Richard Stern for the nice and profitable discussions and for practically "adopting" me into his robust speech recognition group at the Carnegie Mellon University. Prof. Alex Waibel, for making it possible for me to write this Diploma thesis at the Carnegie Mellon University (CMU) in Pittsburgh. My very international friends in Pittsburgh for showing me around, for accommodation, friendship and help.

Abstract

Particle filters (PF)s, a.k.a. sequential Monthe Carlo methods, originally developed for typical tracking applications like pursuing airplanes in radars [21], or persons in video images [23], are increasingly pervading other fields of engineering covering navigation, robotics, communications and (industrial) process control. Recently, they have found their way into speech recognition [39, 14] where they are used for the enhancement of speech features corrupted by noise. The advantage over classical methods like *spectral subtraction* [6] or *Wiener filtering* is that the PF allows the noise to be non-stationary.

We have followed and extended Raj et al's approach [39], which is based on the Bayesian bootstrap filter (the SIR particle filter). A complete statistical derivation of this approach is given in this thesis together with several refinements and imporovements in computational time as well as *automatic speech recognition* (ASR) accuracy. The general speech model — used for the particle likelihood evaluations — was replaced by a phoneme-specific model that works on a phoneme transcription hypothesis of a previous ASR pass, effectively coupling the particle filter to the ASR system. Furthermore we have devised a fast acceptance test and a reinitialization procedure for the particle filter, which overcome some stability problems of the original approach. The acceptance test also decreases the computational cost of the method by allowing us to use less particles. Replacement of *vector Taylor series* (VTS) noise compensation by a much simpler and faster method showed the greatest gain in computational time while outperforming the original method. Finally we incorporated the relative phase between speech and noise, which showed a significant gain in ASR accuracy in a special case. Further work in this direction should be performed, to see whether this is generalizable.

Contents

1 Introduction

An *automatic speech recognition* (ASR) system basically consists of two stages: a *feature extraction* stage (also called *frontend*) and a *decoding* stage. In the feature extraction stage the audio signal is

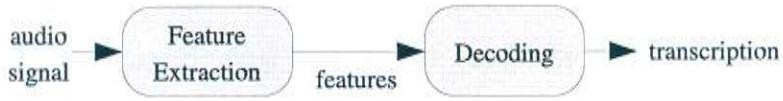


Figure 1.1: ASR overview

processed to obtain those aspects of the audio signal that are relevant for the recognition of speech. At the same time the dimension is reduced to cut the amount of data necessary for learning as well as the computational complexity of the decoder. Decoding means to obtain a transcription of the spoken utterances contained in the audio signal by aligning the features with states of acoustic, word and language models. The *acoustic model* describes the smallest acoustic units, phonemes or subphonemes, via the probability distribution of the corresponding features. Often phonemes are combined to larger units like triphones or — more general — n-phones. The *word model* specifies which sequences of states (phonemes or n-phones) are valid words. The *language model* gives the probability of a succession of words.

Employing ASR systems in a noise environment that differs from the training environment leads to a mismatch between features to be decoded and features used to train the acoustic model of the decoder. Since this discrepancy results in a severe degradation of the recognition performance [4], a variety of techniques have been devised to overcome this problem. Those techniques basically belong to one of the following categories: use of *noise robust features*, *model adaptation*, *hidden Markov model decomposition* (HMM) of speech and noise by the decoder and *speech feature enhancement*.

Model Adaptation
An extreme case of model adaptation consists in training different models for different environments. That, however, introduces new problems. First of all this only works for static noise environments, since the variance of the acoustic models greatly increases in the presence of non-stationary noises, which leads to increased overlaps of different phonetic units. Furthermore this is practically infeasible for large ASR systems which need a lot of training data. Therefore the acoustic model is typically trained with clean speech features and adapted by modifying the means and covariances of its Gaussian mixture distributions to compensate for the noise.

Hidden Markov Model - Decomposition
Varga and Moore's *Hidden Markov Model* (HMM) - decomposition approach [49] is entirely different. It uses separate models for speech and noise and searches the combined state space by an extended Viterbi algorithm in the decoding stage. Further work in this direction was performed by Gales and Young [16, 17] under the name *parallel model combination* (PMC). The drawback of this method is that it becomes computationally immensely expensive as the noise HMM become more complex.

Speech Feature Enhancement
Another powerfull approach is speech feature enhancement. It does not necessitate changing the decoder or adapting the models and it can be performed either in a pre-processing stage — completely independent of the ASR system — or during the feature extraction stage. The latter is more restrictive, but computationally less expensive, since signal processing is just performed

once. Furthermore, it is a good idea to conduct the enhancement in a domain as close as possible to the feature domain, since cleaning parts of the signal that are insignificant in the feature domain is probably useless. To discuss possible "entry points" for speech enhancement in our speech recognizer (Janus[1]) we give a short summary of its feature extraction stage. In "Windowing" the

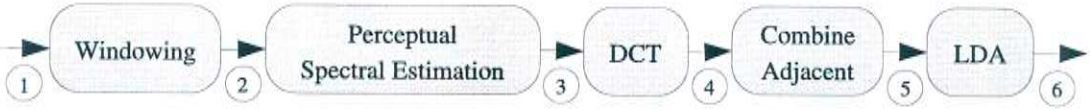


Figure 1.2: Janus Feature Extraction

digitized audio signal (1) is cut into overlapping frames of fixed length. Furthermore, a window function is multiplied (componentwise) with each frame to minimize leakage of spectral energy into neighboring frequency bands in the following spectral estimation step. Perceptual spectral estimation basically estimates the power of the different frequency bands using *Fourier transformation* FT, *linear prediction* (LP) or *minimum variance distortionless response* (MVDR), while mimicking the perception of frequency and loudness of the human auditory system. This is reasonable since it can well be assumed that production of speech and its perception are evolutionary tuned to each other. Experimental findings support this kind of reasoning. The perception of both frequency and loudness is logarithmic, which reflects the importance of the relative difference, not the absolute one. Note that the relative difference between 100 and 200 hertz (100%) is much higher than the relative difference between 1100 and 1200 hertz ($\approx 9\%$), while the absolute difference is the same. An example for a perceptual spectral estimation step implemented in Janus is given by figure 1.3. *Fast Fourier transformation* (FFT) is applied to each window of 256

Figure 1.3: Perceptual Spectral Estimation

samples. The power spectrum is obtained by calculating the componentwise magnitude square of the DFT's result, a complex vector of dimension 129. The *Mel filterbank* maps the linear into the logarithmic frequency domain and at the same time reduces the spectral dimension to 30, before componentwise application of the logarithm (log) transforms the power spectrum into the log Mel power domain.

After, *dicrete cosine transform* (DCT) is used to further reduce the spectral dimension while decorrelating the components. A DCT log mel spectrum is typically called cepstrum. The DCT log mel domain is also known as the cepstral domain. "Combine adjacent" merges successive cepstra, which are then processed with *linear discriminant analysis* (LDA) to extract the features that are most relevant for discrimination between acoustical units.

The ideal placement of speech feature enhancement is clearly at the end of the feature extraction stage — as close as possible to the features used by the speech recognizer. That, however, requires tracing the relationship of speech, noise and corrupted speech through the whole feature extraction stage, which will pose a big problem with our approach later. The problem is related to the application of the logarithm during the perceptual spectral estimation step, which makes it hard or even impossible to get a certain stochastic relationship through the DCT and LDA (see sections 4.4 and 3.2). That's why we will apply feature extraction before (see figure 1.4).

[1] *Janus Recognition Toolkit* (JRTk) is developed and maintained by the Interactive Systems Laboratories at two sites: Universität Karlsruhe (TH), Germany and Carnegie Mellon University, USA.

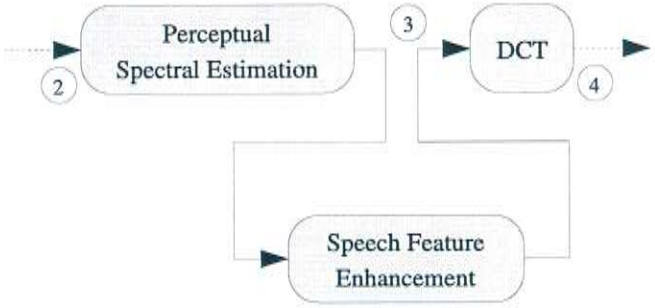


Figure 1.4: Placement of Speech Feature Enhancement

Early noise compensation techniques typically assumed the noise to be stationary, which might be true for thermal white or colored — not, however for environmental background noises. Members of this family are spectral subtraction [6], Wiener filtering and Ephraim and Malah's MMSE log spectral amplitude estimator [11, 12]. The argument for removing noise in the log spectral domain is that Minimization of the the *mean square error* (MSE) in the spectral domain does not necessary lead to a *minimum mean square error* (MMSE) estimate in the log spectral domain, where the speech recognizer gets its features from. The first serious attempts to non-stationary noise compensation emerged in the mid to late 1990s. Among those, probably the most prominent ones are Moreno's *vector Taylor series* (VTS) [35] and Kim's sequential EM approach [27] using statistical linear approximation (SLA) — Kim's extension of VTS. Some people cite Segura's "Model-based Compensation of the Additive Noise for Continuous Speech Recognition" [43], who combined the VTS approach with a band pass filter to remove the resiudal noise. Kim's sequential EM probably at least inspired Yao's sequential EM [52] and Kullback Proximal [51] algorithms, which — contrary to Kim's approach — use model adaptation to compensate for the noise. This kind of dynamic model adaptation, however, becomes computationally incredibly expensive for large ASR systems. That's why we perform speech feature enhancement.

On a parallel development track there had been ongoing research into non-stationary noise compensation for years. Paliwal and Basu [37] applied Kalman filtering as early as in 1987 to track the *linear prediction* (LP) coefficients of speech — which they assumed to be a time-varying autoregressive (TVAR) process — contaminated by white noise. Gibson [20] extended this to colored noise and developed an iterative signal and parameter estimation scheme, which was further examined by Gannot [18] and eventually enriched by usage of *unscented Kalman filters* (UKF)s [19] for joint and dual estimation of clean speech and the parameters. Despite all those improvements, there are some fundamental issues with the underlying approach to track LP coefficients. First of all, LP coefficients are known to be instable, meaning small changes in the coefficients might not translate to small changes of the corresponding signal. Moreover, linear prediction does not well model the spectral envelope for medium and high pitched voices [36]. Fong, Godsill, Doucet and West replaced the TVAR model by a *time-varying partial correlation* (TV-PARCOR) model to overcome the instability problem with LP coefficients. Furthermore, they proposed to use a *particle filter* instead of the Kalman filter. This, however, still doesn't overcome the major deficiency that all methods in this paragraph have in common: they neither work in the log spectral nor in the Mel domain, i.e. neglect perceptual relevance.

Kim [26] had the idea to track the noise spectrum — contaminating speech — in the (log Mel) spectral (power) domain by employing a bank of Kalman filters, a so-called interacting multiple model (IMM) widely used in the area of multiple target tracking [2]. Each Kalman filter represents a Gaussian noise hypothesis in the spectral domain that is compensated by SLA a.k.a. VTS. Kim's work inspired Raj et al. [39] to apply a sequential sampling importance resampling (SIR) particle filter — called Bootrap filter throughout this thesis like in the Gordon's original work [21] — to overcome the linear approximation of the relationship between clean speech, noise and corrupted speech in the (Mel) log spectral domain necessary for the application of the Kalman filter. We

followed this approach. Chapter 2 introduces the Bootstrap filter in the general framework of tracking. Chapter 3 gives a complete statistical derivation of Raj et al.'s approach.

2 Tracking Evolving Dynamical Systems

The objective of tracking is to estimate a sequence of system states on the basis of observations. To have a concrete scenario, let's say the system states describe the trajectory of an airplane and the observations are measurement values obtained by a radar. Then the tracking problem can be formulated as to reconstruct the target trajectory by evaluating of the relationships between possible trajectories and given measurements. For this we need a model that describes the system under consideration. Using a general model yields a general tracking algorithm.

2.1 A General Model for Tracking

The most general description of a system that facilitates tracking is the *observed stochastic dynamical system model*[1]. It consists of two stochastic processes that are interlocked with each other: a state process $(X_t)_{t\in\mathbb{N}}$ representing the evolution of a hidden, inner system and a corresponding observation process $(Y_t)_{t\in\mathbb{N}^*}$. The interlocking can be regarded as causal relationship where system states cause observations. As the observation might be subject to random disturbances or *measurement noise* the relationship is described by the conditional probability density[2] $p(y_{1:t}|x_{0:t})$, $t \in \mathbb{N}$. Choosing a state space determines which aspects of the real, physical system are

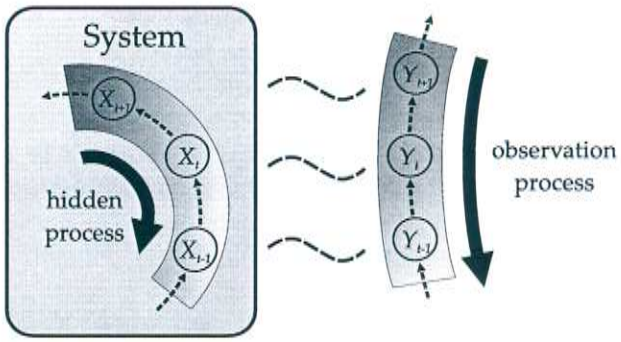


Figure 2.1: Observed Stochastic Dynamical System Model

incorporated into the model. For the airplane scenario one could choose the state vector (i.e. the state space) to describe the target's position $(p_x, p_y, p_z)^T$. A more sophisticated model would also include its velocity and acceleration $(p_x, p_y, p_z, v_x, v_y, v_z, a_x, a_y, a_z)^T$. The neglected quantities — speed of wind, variation in air presssure and gravity, maneuvering by a pilot etc — introduce indeterministic behavior into the model world. Therefore the evolution of the state process is described by a probability density, $p(x_{0:t})$, $t \in \mathbb{N}$. The observation process does not need to be specified since it is implicitly described by $p(y_{1:t}|x_{0:t})$ and $p(x_{0:t})$:

$$p(y_{1:t}) = \int p(y_{1:t}, x_{0:t}) dx_{0:t} = \int p(y_{1:t}|x_{0:t}) \cdot p(x_{0:t}) dx_{0:t} \tag{2.1}$$

[1] called *general state space model* throughout most of the particle filter literature

[2] For a purely deterministic relationship, the density is a Dirac delta , i.e. the whole probability mass is concentrated in one point.

where $p(x_0|y_{1:0}) := p(x_0)$. We still need to find a way to sequentially update $p(y_t|y_{1:t-1})$, the denominator of $q(x_t|x_{t-1}, y_t)$ in (2.7). Fortunately $p(y_t|y_{1:t-1})$ can be pulled out of the integral (2.7), since it does not depend on x_t. So

$$\begin{aligned} p(x_t, y_t|y_{1:t-1}) &= p(x_t|y_{1:t}) \cdot p(y_t|y_{1:t-1}) \\ &= \int p(y_t|x_t) \cdot p(x_t|x_{t-1}) \cdot p(x_{t-1}|y_{1:t-1}) dx_{t-1} \end{aligned}$$

and $p(y_t|y_{1:t-1})$ can be obtained as marginal density $p(y_t|y_{1:t-1}) = \int p(x_t, y_t|y_{1:t-1})dx_t$. This yields a tracking algorithm that sequentially calculates the E_{p_t} and requires only the computation of fixed dimensional integrals (having the dimension of state space) at each time step t, $t = 1, \ldots, \tau$.

Algorithm 2.1 Sequential Tracking

1. update $p(x_t|y_{1:t})$ by
   a) calculating $p(x_t, y_t|y_{1:t-1}) = p(y_t|x_t) \cdot \int p(x_t|x_{t-1}) \cdot p(x_{t-1}|y_{1:t-1})dx_{t-1}$
   b) calculating the marginal density $p(y_t|y_{1:t-1}) = \int p(x_t, y_t|y_{0:t-1})dx_t$
   c) combining those two densities to $p(x_t|y_{1:t}) = p(x_t, y_t|y_{1:t-1})/p(y_t|y_{1:t-1})$
2. evaluate $\mathrm{E}_{p_t}[h(x_t)|y_{1:t}] = \int h(x_t) \cdot p(x_t|y_{1:t})dx_t$

The idea behind this sequential approach is to develop a closed form representation of the filtering density, where $p(x_t|y_{1:t})$ has the same distribution for all t [5]. The *Kalman filter* (KF) [24] gets its closed form representation by assuming the filtering density to be Gaussian, which requires the deterministic parts of the state transition and the output probability to be linear. It is ideal — i.e. it can't be outperformed by any other method — if all the assumptions hold. Kalman filters can be applied to nonlinear problems by linearizing the nonlinearity through a first order Taylor series approximation. This procedure is generally called *extended Kalman filter* (EKF). Another way to obtain a closed form representation of the filtering density is to numerically approximate the integrals in algorithm 2.1. The advantage of this approach is that it does not impose the restrictions of Gaussianity and linearity on the model.

2.4 Approximating Expectations

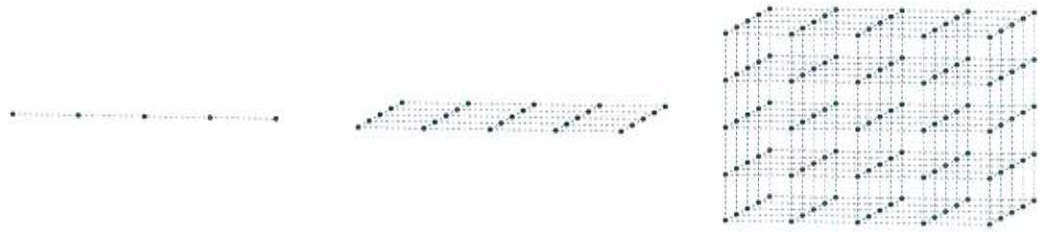
In this section we will establish a way to efficiently approximate expectations, i.e. integrals of the form $\int h(x) \cdot p(x)dx$ as occurring in our sequential tracking algorithm. The examination is restricted to numerical integration methods among which probably the most well-known one is grid-based integration.

Grid-based approximation
Let $\mathcal{G} = \{x^{(j)}, j = 1, ..., N\}$ be a set of support points which are aligned to form an equidistant grid (hence the name) and let V be the volume spanned by $\mathcal{G}$. Then $\mathrm{E}_{p(x)}[h(x)]$ can be approximated as

$$\int h(x) \cdot p(x)\mathrm{d}x \approx \frac{V}{N} \sum_{j=1}^{N} h(x^{(j)})p(x^{(j)}) \tag{2.8}$$

The number of support points where $h(x) \cdot p(x)$ has to be evaluated grows exponentially with the dimension. This is generally unavoidable. And though, grid-based methods tend to be particularly ineffective when it comes to higher dimensions. The reason for this is that they keep many points in regions of space that are relatively unimportant, i.e. points for which $h(x) \cdot p(x)$ is close to zero.

[5] An essential requirement for this is that the dimension stays the same.



Example 2.4.1. Consider a two-dimensional normal distribution with density $p(x_0, x_1) = 1/(2\pi) \cdot e^{-(x^2+y^2)/2}$. Let $h(x_0, x_1) = x_0^2$ and let's say we want to approximate $\mathrm{E}_p[h(x_0, x_1)]$ on the interval $[-5, 5] \times [-5, 5]$ with an equidistant grid of 5×5 points. The result of this approximation is 0.38 though we know the true result to be 1.0 since E_p is the variance of $p(x_0)$. Figure 2.3 depicts the addressed problem with grid-based integration.

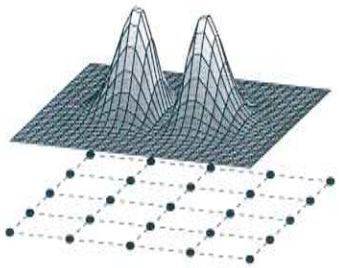


Figure 2.3: $x_0^2 \cdot p(x_0, x_1)$ with underlying grid (25 points)

||

Monte Carlo Approximation

Other deterministic numerical integration methods like the trapezoidal rule or Simpson's rule suffer from the same problems. In contrast, Monte Carlo integration, a stochastic numerical integration method and probably because of its use of randomness the method a gambler[6] would use, is known to work reasonably well in higher dimensions. The randomness comes in by drawing support points $x^{(1)}, ..., x^{(N)}$ at random from $p(x)$ — meaning they are selected with probability $p(x)$ among all x, which ensures that they are primarily located in regions where the probability mass of p is concentrated, i.e. not close to zero. Those points are now used to approximate p by the *empirical density function*

$$\hat{p}(x) := \frac{1}{N} \sum_{j=1}^{N} \delta_{x^{(j)}}(x)$$

where $\delta_{x^{(j)}}(x) := \delta(x - x^{(j)})$ is the translated Dirac delta function. Looking at the densities alone it might not be obvious that this is a reasonable approximation. As stated before, however, the relevant aspect is that the probability mass is concentrated in the same regions, which means that the cumulative density functions

$$P(X \leq x) = \int_{-\inf}^{x} p(x')dx' \qquad \hat{P}(X \leq x) = \int_{-\inf}^{x} \hat{p}(x')dx'$$

should be alike.

[6] Monte Carlo integration was given its name at the suggestion of Nicholas Metropolis in honor of Stanislaw Ulam's uncle, who reportedly was a gambler [48]. Monte Carlo is a district of Monaco which is known for its famous "Casino Monte Carlo". Metropolis and Ulam were both pioneers in the field of what today is known as Monte Carlo methods.

Example 2.4.2. Figure 2.4(a) is an illustration of a continuous, normal density function (solid line) with an exemplary empirical normal density function (dashed lines with dots marking Dirac deltas). Figure 2.4(b) shows the corresponding cumulative density functions.

(a) density (b) cumulative density

Figure 2.4: Continuous vs. Empirical Densities

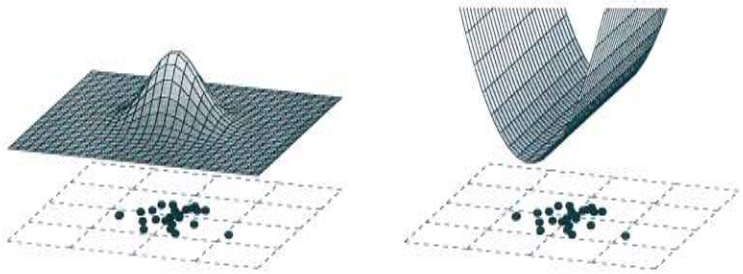
||

Similar to this approximation of the cumulative density function, arbitrary expectations $\mathrm{E}_p[h(x)]$ can be approximated by replacing the continuous density function with the empirical density:

$$\int h(x) \cdot p(x)dx \quad \approx \quad \int h(x)\hat{p}(x)dx \quad = \quad \frac{1}{N}\sum_{j=1}^{N} h(x^{(j)}) \tag{2.9}$$

This is what is known as Monte Carlo integration.

Example 2.4.3 (Continuation of example 2.4.1)**.** Let's again approximate the expectation $E_{p(x)}[h(x)]$ of example 2.4.1, but this time using Monte Carlo integration. Figure 2.5(a) shows a possible two-dimensional empirical normal density $\hat{p}$ (there is one for each drawn sample-set $\{x^{(1)}, \ldots, x^{(N)}\}$) with its continuous counterpart $p(x_0, x_1)$. Figure 2.5(b) shows the function



(a) empirical $\hat{p}$ (dots) and continuous normal density p (solid) (b) empirical density below x_0^2

Figure 2.5: Monte Carlo Approximation

$h(x_0, x_1) = x_0^2$ together with the samples $x^{(j)}$ (dots) where it it evaluated for the approximation by a sum (see equation 2.9). Depending on the drawn sample-set the result can be extremely precise (i.e. close to 1.0) as well as extremely poor (worse than grid-based integration). The following values show some possible outcomes:

0.91, 1.01, 1.00, 1.01, 1.70, 1.37, 1.09, 0.70, 1.26, 0.96

All but one of these values (1.70) are better than the grid-based result (0.38). The average of the values is 1.1, their variance is 0.07. Results like these are quite common for Monte Carlo integration. There is no guarantee for a good result, since the method is probabilistic. But the more samples $x^{(j)}$ are used for the approximation, the more unlikely is the chance of a bad result. ||

The comparison is manipulated in a way, since we used a disadvantageous grid in example 2.4.1 to portray the problem. In low dimensional spaces — typically up to dimension 3 — grid based integration outperforms Monte Carlo integration, if the grid is selected appropriately, but with increasing dimension the problem with disadvantages support points will sooner or later show up. A theoretical prove of *almost sure convergence* of the Monte Carlo approximation towards $E_{p(x)}[h(x)]$ with $N \to \infty$ can be found in [40].

Importance Sampling

In practice it is often hard to obtain samples from p, since the distribution might be hard to model or to "learn". Another problem is that $h(x) \cdot p(x)$ is not always dominated by $p(x)$, meaning $h(x)$ is close to zero in the region where the probability mass of p is concentrated, but conversely has significant values in regions where $p(x)$ is close to zero. Monte Carlo integration fails to provide accurate results in these cases. The idea behind *importance sampling* is to draw samples from another probability density with the aim of overcoming the mentioned problems. So, let's say we draw samples from $\pi(x)$ instead of $p(x)$. Requiring $\pi(x) \neq 0$ where $p(x) \neq 0$, $\omega(x) := p(x)/\pi(x)$ is well defined and

$$\begin{aligned} \mathrm{E}_p[h(x)] &= \int h(x) \cdot p(x) dx_{0:t} \\ &= \int h(x) \cdot \frac{p(x)}{\pi(x)} \cdot \pi(x) dx_{0:t} \\ &= \int h(x) \cdot \omega(x) \pi(x) dx_{0:t} \\ &= \mathrm{E}_\pi[h(x) \cdot \omega(x)] \end{aligned}$$

This is called the *importance sampling fundamental identity* (Robert & Casella [40]). Furthermore, $\mathrm{E}_\pi[h(x) \cdot \omega(x)]$ can be approximated by Monte Carlo integration, i.e. by using the empirical density $\hat{\pi}$ obtained by drawing samples from π. Thus $\mathrm{E}_p[h(x)]$ can be approximated as

$$\mathrm{E}_p[h(x)] \approx \frac{1}{N} \sum_{j=1}^{N} h(x_t^{(j)}) \cdot \omega(x_t^{(j)}) \tag{2.10}$$

The name importance sampling stems from the view that the samples are drawn from regions of "importance". π is called importance or proposal density, the $\omega(x)$ are called importance weights. It isn't hard to figure out that approximating $\mathrm{E}_p[h(x)]$ by replacing p with the *weighted empirical density*

$$\tilde{p}(x) := \frac{1}{N} \sum_{j=1}^{N} \omega(x^{(j)}) \delta_{x^{(j)}}(x)$$

with samples $x^{(j)}$ drawn from π yields the same result.

Example 2.4.4. Let $p = \mathcal{N}(0, 1)$ be the one-dimensional standard normal distribution, i.e. a zero mean Gaussian distribution with covariance 1 and let $\pi = \mathcal{U}[0, 3]$ be a uniform distribution on $[0, 3]$. Figure 2.6(a) shows a possible empirical density with samples $x^{(j)}$ drawn from π. Figure 2.6(b) shows the weighted empirical density $\tilde{p}(x)$ resulting from the samples $x^{(j)}$ with the corresponding importance weights $\omega(x^{(j)}) = 1/3 \cdot \mathcal{N}(x; 0, 1)$.

(a) empirical density (b) weighted empirical density

Figure 2.6: Importance Sampling

The missing left side of the Gaussian distribution clearly portrays the problem arising if the condition $[\pi(x) \neq 0 \text{ where } p(x) \neq 0]$ does not hold. Samples are never drawn from regions where the probability mass is zero, i.e. outside of $[0, 3]$ and therefore importance sampling can only represent the Gaussian on this interval. ||

Importance Resampling

Any normalized weighted empirical density $\bar{p}(x)$ can be transformed into an empirical density $\hat{p}$ simply by sampling from it. Note that the normalization of a weighted empirical density is equivalent to the normalization of its weights

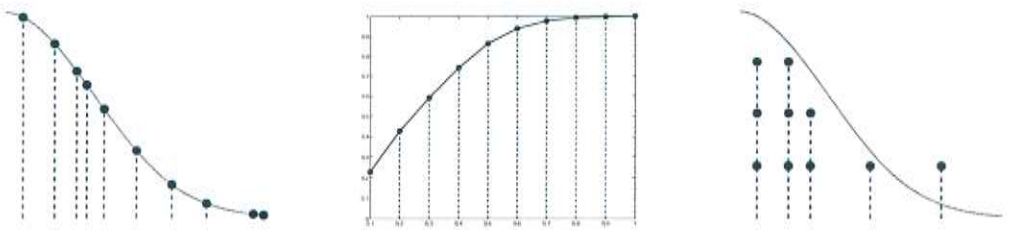
$$\bar{p}(x) = \frac{\frac{1}{N}\sum_{j=1}^{N}\omega(x^{(j)})\delta_{x^{(j)}}(x)}{\frac{1}{N}\sum_{j=1}^{N}\omega(x^{(j)})} = \sum_{j=1}^{N}\underbrace{\frac{\omega(x^{(j)})}{\sum_{j=1}^{N}\omega(x^{(j)})}}_{=:\tilde{\omega}(x^{(j)})}\delta_{x^{(j)}}(x)$$

Sampling from $\bar{p}(x)$ can be performed by drawing a sample u from the standard unitary distribution $\mathcal{U}[0, 1]$, looking up in which interval $[c_{j-1}, c_j]$, $j \in \{1, \ldots, N\}$, with

$$c_i = \sum_{l=1}^{i}\tilde{\omega}(x^{(j)})$$

and $c_0 = 0$ it is located and returning the corresponding $x^{(j)}$. This basically multiplies samples with high importance weights and eliminates samples with low importance weights.

Example 2.4.5 (Continuation of example 2.4.4). Figure 2.7(a) shows the weighted empirical density of example 2.4.4. Figure 2.7(b) shows the corresponding cumulative density function c_i, figure 2.7(c) a possible (there are many) resampled empirical density.



(a) weighted empirical density (b) cumulative density c_i (c) resampled empirical density

Figure 2.7: Importance Resampling

||

2.5 The Bayesian Bootstrap Filter

The technique of approximating expectations $\mathrm{E}_p[h(x)]$ by Monte Carlo integration and importance sampling/resampling can now be applied to the sequential tracking algorithm (Algorithm 2.1) by replacing the continuous filtering density $p(x_t|y_{1:t})$ with its empirical counterpart. This way the representation of the filtering density stays the same throughout time, which makes the problem of calculating the $\mathrm{E}_{p_t}[h(x_t)|y_{1:t}]$, $t = 1, \ldots, T$ tractable. So, let's say we have $\hat{p}(x_{t-1}|y_{1:t-1})$, the empirical filtering density at time $(t-1)$. Then the update equation (Algorithm 2.1 - Step 1a) gives us

$$\begin{aligned} p(x_t, y_t|y_{1:t-1}) &= p(y_t|x_t) \cdot \int p(x_t|x_{t-1}) \cdot \hat{p}(x_{t-1}|y_{1:t-1}) dx_{t-1} \\ &= p(y_t|x_t) \cdot \int p(x_t|x_{t-1}) \cdot \frac{1}{N} \sum_{j=1}^{N} \delta_{x_{t-1}^{(j)}}(x_{t-1}) dx_{t-1} \\ &= \frac{1}{N} \sum_{j=1}^{N} p(y_t|x_t) \cdot \int p(x_t|x_{t-1}) \cdot \delta_{x_{t-1}{}^{(j)}}(x_{t-1}) dx_{t-1} \\ &= \frac{1}{N} \sum_{j=1}^{N} p(y_t|x_t) \cdot p(x_t|x_{t-1}^{(j)}) \end{aligned} \tag{2.11}$$

Our aim is to construct the empirical density of $p(x_t|y_{1:t}) = p(x_t, y_t|y_{1:t-1})/p(y_t|y_{1:t-1})$ (Algorithm 2.1 - Step 1c) by sampling from it. This can be achieved by drawing samples from (2.11) instead, because the calculation of (Algorithm 2.1 - Step 1b) and division by $p(y_t|y_{1:t-1})$ can be postponed as this term is constant as soon as the current observation y_t is known. Since a sample from (2.11) can come from any of the terms $p(y_t|x_t) \cdot p(x_t|x_{t-1}^{(j)})$, we draw exactly [7] one sample $x_t^{(j)}$ from each $p(y_t|x_t) \cdot p(x_t|x_{t-1}^{(j)})$, which can be obtained by using $p(x_t|x_{t-1}^{(j)})$ as an importance density. The corresponding importance weight of such a sample $x_t^{(j)}$ is

$$\omega(x_t^{(j)}) = \frac{p(y_t|x_t^{(j)}) \cdot p(x_t^{(j)}|x_{t-1}^{(j)})}{p(x_t^{(j)}|x_{t-1}^{(j)})} = p(y_t|x_t^{(j)})$$

A good way to understand this is to regard the samples as supposable state hypotheses and the weights $\omega(x_t^{(j)})$ as their likelihoods $l(x_t^{(j)}; y_t)$ given the observation y_t. The resulting weighted empirical density — constructed by using all N samples and their importance weights — is

$$\tilde{p}(x_t, y_t|y_{1:t-1}) = \frac{1}{N} \sum_{j=1}^{N} p(y_t|x_t^{(j)}) \cdot \delta_{x_t^{(j)}}(x_t)$$

which is now used to approximate the normalizing factor $p(y_t|y_{1:t-1}) = \int p(x_t, y_t|y_{1:t-1}) dx_t$ by Monte Carlo integration:

$$\begin{aligned} p(y_t|y_{1:t-1}) &\approx \int \frac{1}{N} \sum_{j=1}^{N} p(y_t|x_t^{(j)}) \cdot \delta_{x_t^{(j)}}(x_t) dx_t \\ &= \frac{1}{N} \sum_{j=1}^{N} p(y_t|x_t^{(j)}) =: \tilde{p}(y_t|y_{1:t-1}) \end{aligned}$$

Dividing $\tilde{p}(x_t, y_t|y_{1:t-1})$ by $\tilde{p}(y_t|y_{1:t-1})$ gives a weighted empirical density of $p(x_t|y_{1:t})$:

$$\tilde{p}(x_t|y_{1:t}) := \frac{\tilde{p}(x_t, y_t|y_{1:t-1})}{\tilde{p}(y_t|y_{1:t-1})} = \frac{\sum_{j=1}^{N} p(y_t|x_t^{(j)}) \cdot \delta_{x_t^{(j)}}(x_t)}{\sum_{j=1}^{N} p(y_t|x_t^{(j)})}$$

It might seem intriguing, but it is by design that this division is equivalent to normalizing the importance weights:

$$\bar{\omega}(\tilde{x}_t^{(j)}) := \frac{\omega(\tilde{x}_t^{(j)})}{\sum_{j=1}^{N} \omega(\tilde{x}_t^{(j)})} = \frac{p(y_t|x_t^{(j)})}{\sum_{j=1}^{N} p(y_t|x_t^{(j)})}$$

[7] We want the number of samples, N, to stay constant. It is also possible to draw $N' > N$ samples and to reduce the number of samples to N during the following resampling. This was originally proposed by Rubin [42].

Hence, the empirical filtering density $\hat{p}(x_t|y_{1:t})$ for $p(x_t|y_{1:t})$ can be obtained by the method of importance resampling, i.e. by sampling from the normalized weighted empirical density $\tilde{p}(x_t|y_{1:t})$. The evaluation step (Algorithm 2.1 - Step 2) can be performed by using either this empirical density $\hat{p}(x_t|y_{1:t})$ or the weighted empirical density $\tilde{p}(x_t|y_{1:t})$ to approximate $\mathrm{E}_{p_t}[h(x_t)|y_{1:t}]$ by Monte Carlo integration:

$$\begin{aligned}\mathrm{E}_{p_t}[h(x_t)|y_{1:t}] &= \int h(x_t) \cdot p(x_t|y_{1:t}) dx_t \\ &\approx \sum_{j=1}^{N} h(\tilde{x}_t^{(j)}) \tilde{\omega}(\tilde{x}_t^{(j)}) \\ &\approx \sum_{j=1}^{N} h(x_t^{(j)})\end{aligned}$$

This procedure can be formulated as an algorithm that is iterated over t starting with $t = 1$, where the initial samples $x_0^{(j)}$ are drawn from the prior distribution $p(x_0)$.

Algorithm 2.2 The Boostrap Filter

1. update $\hat{p}(x_t|y_{1:t})$ by
   a) drawing one sample $\tilde{x}_t^{(j)}$ from each $p(x_t|x_{t-1}^{(j)})$ for $j = 1, \ldots, N$.
   b) calculating the normalized importance weights $\tilde{\omega}_t^{(j)}(\tilde{x}_t^{(j)}) = p(y_t|\tilde{x}_t^{(j)}) / \sum_{j=1}^{N} p(y_t|\tilde{x}_t^{(j)})$ and constructing $\tilde{p}(x_t|y_{1:t}) = \sum_{k=1}^{K} \tilde{\omega}_t^{(j)} \delta_{\tilde{x}_t^{(j)}}(x_t)$.
   c) obtaining the empirical density $\hat{p}(x_t|y_{1:t})$ by importance resampling, i.e. by sampling $x_t^{(j)}$ from $\tilde{p}(x_t|y_{1:t})$ for $j = 1, \ldots, N$.
2. approximate $\mathrm{E}_{p_t}[h_t(x_t)|y_{1:t}]$ as $\sum_{j=1}^{N} h_t(\tilde{x}_t^{(j)}) \tilde{\omega}(\tilde{x}_t^{(j)})$ or as $\sum_{j=1}^{N} h_t(x_t^{(j)})$.

This algorithm was first proposed by Gordon [21], though it seems to have been independently developed by Kong, Liu and Wong in the slightly different context of Bayesian missing data problems [30]. The method of Importance resampling was developed by Rubin — alongside a comment [42] to Tanner and Wong's data augmentation algorithm [45]. It is clearly Rubin's brainchild though, motivated by his earlier work: the Bayesian bootstrap [41]. Gelfand and Smith [44] discovered that normalization of the importance weights as performed in Rubin's importance resampling implicitly approximates the *normalizing constant* $p(y)$ in Bayesian Statistics, i.e the denominator of

$$p(x|y) = \frac{p(y|x) \cdot p(x)}{p(y)} = \frac{p(y|x) \cdot p(x)}{\int p(y|x) \cdot p(x) dx}$$

The Bayesian Bootstrap filter, in turn, can be understood as an application of Smith and Gelfand's method to the tracking problem or more general to sequential Bayesian estimation. It comes as no surprise that Smith is a coauthor of Gordon's paper. Kitagawa's [29] idea to name samples "particles" eventually coined the term *particle filter* for a generalized version of the Bootstrap filter, which is exhaustively treated in the monographs by Liu and Chen [33] and Doucet [10]. A very good tutorial on particle filters can be found in [1].

The Importance of Resampling

Doucet [10], Liu and Chen [33] also consider *sequential importance sampling* (SIS), a Bootstrap filter without the resampling stage — hence not a bootstrap. This factually approximates the continuous filtering density $p(x_t|y_{1:t})$ by a weighted empirical one, causing that hypotheses with a low relative likelihood are kept. They are further propagated, evolve according to the state transition probability and sonly span a big portion of state space — not however the regions of importance. This phenomenon is called "weight degeneracy". Doucet [10] gives a theoretical

proof that the variance of the importance weights can only increase over time, meaning weight degeneracy has to occur. The resampling stage, in contrast, can be regarded as a pruning step where likely hypotheses are multiplied, unlikely ones are removed from the particle population, causing the state space to be thoroughly explored in regions of high relative likelihood, less explored in unlikely ones.

2.6 Semi-Deterministic Resampling

Real random resampling has a high variability and therefore necessitates a high number of samples to give a reliable approximation of the weighted empirical density. Furthermore it is computationally inefficient: the simple algorithm has a complexity of $\mathcal{O}(N^2)$. Searching the cumulative density with *binary search* yields a complexity of $\mathcal{O}(N \log(N))$. Kitagawa argued that it is not essential to do random resampling, since the purpose of resampling is solely to obtain an empirical density that mimics the weighted empirical density function. With this motivation, he proposed two new semi-deterministic resampling algorithms [29], which would later become known as *systematic resampling* (SR) and *residual resampling* (RR)[8].

Resampling can be divided into two stages: a stage that computes the *multiplication factor* $m[j]$ of each sample — giving the number of children the sample will have — and the actual resampling stage in which the multiplication factors are realized. This provides a maximum of flexibility when it comes to exchanging resampling algorithms. Algorithm 2.3 gives an implementation of systematic resampling. Algorithm 2.4 is a very efficient implementations of systematic resampling, proposed by Bolic, Djuric and Hong [5] under the name *systematic residual resampling* (RSR). It is also the sampling method of our choice, since its computational complexity is $\mathcal{O}(N)$ and since usage of systematic resampling is generally encouraged in [1].

Algorithm 2.3 computeSRMF(N, M, $\omega^{(j)}, j = 1, \ldots, N$)

calculate the stepwidth $s = 1/M$
draw initial $u \in \mathcal{U}[0, s]$
initialize the cumulative weight: $c = 0$

for $j = 1, \ldots, N$
    update the cumulative weight: $c = c + \omega^{(j)}$
    $k = 0$
    while $(u \leq c)$
        count: $k = k + 1$
        $u = u + s$
    end while
    store the jth multiplication factor $m[j] = k$
end for

Algorithm 2.4 computeRSRMF(N, M, $\omega^{(j)}, j = 1, \ldots, N$)

calculate the stepwidth $s = 1/M$
draw initial $u \in \mathcal{U}[0, s]$
initialize the cumulative weight: $c = 0$

for $j = 1, \ldots, N$
    $k = \lfloor (\omega^{(j)} - u) \cdot M \rfloor + 1$
    $u = u + k \cdot s - w^{(j)}$
    store the jth multiplication factor $m[j] = k$
end for

[8] Residual resampling is an optimized implementation of Kitagawa's *stratified resampling* proposed in [3].

3 The Bayesian Bootstrap Filter for Speech Feature Enhancement

This chapter formally derives the particle filter approach by Raj et al. [39], which applies a Bayesian Bootstrap filter to track the noise spectra corrupting speech and infers the clean speech spectra in step 2 of the Bootstrap filter (algorithm 2.2) by using the relationship between clean speech, noise and corrupted speech spectra to construct an appropriate inference function h_t. In the following spectra will always be log Mel power spectra if not stated otherwise, x_t, n_t and y_t will always denote clean speech, noise and observed corrupted speech spectra, respectively, in the log Mel spectral power domain. Note that — differing from the previous section — the state vector is n_t (not x_t), since we want to track noise spectra. The decision to use a Bootstrap filter stems from the fact that the relationship between observations y_t and state vectors n_t is nonlinear. So a *Kalman filter* can't be used. It is, however, possible to use an *extended Kalman filter* (EKF) that linearizes the nonlinearity by a first order Taylor series approximation. Kim employed a bank of EKFs in his IMM approach [26], which is strongly connected to this method. It, however, necessitates reducing the Kalman gain in a way that is not mathematically justified (see [39, 26]) to get any improvements in recognition performance. That's why Raj et al. decided to apply a Bootrap filter to the problem. For the Bootrap filter to be applicable we need to specify an observed dynamical state space system for the evolution of noise spectra corrupted by speech spectra, i.e. we have to specify the noise transition probability $p(n_t|n_{t-1})$ and the output probability $p(y_t|n_t)$ (see section 2.3).

3.1 A Dynamical System Model for the Noise

Singh, Raj and Stern [39] proposed to model the evolution of noise spectra as an lth order autoregressive process

$$n_t = \underbrace{\left(\boxed{A_1} \boxed{A_2} \ldots \boxed{A_l} \right)}_{=:A} \cdot \underbrace{\begin{pmatrix} n_{t-1} \\ n_{t-2} \\ \vdots \\ n_{t-l} \end{pmatrix}}_{=:\bar{n}_t} + \epsilon_t \tag{3.1}$$

where n_t is a d dimensional column vector representing the noise spectrum at time t and A is a $(l \cdot d) \times d$ Matrix learned for a specific noise type. The deterministic part, $\hat{n}_t = A \cdot \bar{n}_{t-1} = \sum_{i=1}^{l}(A_i) \cdot n_{t-1}$ can be compared to *linear prediction* (LP), which is very well known in automatic speech recognition. It is typically performed in the time (signal) domain to obtain an all-pole modell of speech. In this approach, however, the technique of linear prediction is used to predict the current noise spectrum given the last l noise spectra. Hence, the scalar LP-coefficients are replaced by $d \times d$ LP-matrices A_i, $i = 1, \ldots, l$. The indeterministic part, i.e. what can't be learned by linear prediction, is represented by the ϵ_t terms, which are considered to be (stochastically) independent identically distributed (i.i.d.) zero mean Gaussian with diagonal covariance matrix Σ_{noise}.

Learning the AR Noise Model

The autoregressive noise model consists of two components that have to be learned for each noise type: the LP matrix A and the covariance matrix Σ_{noise}. As stated without proof in [39],

minimization of the prediction error norm results in the following estimate of the LP matrix:

$$A = \mathrm{E}[n_t \bar{n}_{t-1}^T] \cdot \mathrm{E}[\bar{n}_t \bar{n}_{t-1}^T]^{-1} \tag{3.2}$$

where $\mathrm{E}[n_t \bar{n}_{t-1}^T]$ is a $d \times (d \cdot l)$ matrix and $\mathrm{E}[\bar{n}_t \bar{n}_{t-1}^T]$ is a $(d \cdot l) \times (d \cdot l)$ matrix. With noise data $n_1, \ldots, n_\tau$ available, these matrices can be learned as

$$\mathrm{E}[n_t \bar{n}_{t-1}^T] = \frac{1}{\tau} \sum_{t=l}^{\tau} n_t \bar{n}_{t-1}^T \qquad \text{and} \qquad \mathrm{E}[\bar{n}_t \bar{n}_{t-1}^T] = \frac{1}{\tau} \sum_{t=l}^{\tau} \bar{n}_t \bar{n}_{t-1}^T$$

The diagonal covariance for the ϵ_t terms can be learned as

$$\Sigma_{noise} = \begin{pmatrix} v_1 & 0 & \ldots & 0 \\ 0 & v_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & v_d \end{pmatrix}$$

with $v_i := \mathrm{E}[(n_{t,i} - (A\bar{n}_{t-1})_i)^2])$, where $n_{t,i}$ denotes the ith component of the vector n_t. In practice, this variance should be increased to enlarge the search space. Figure 3.1 shows the prediction error — the average of the v_i — for different noise types.

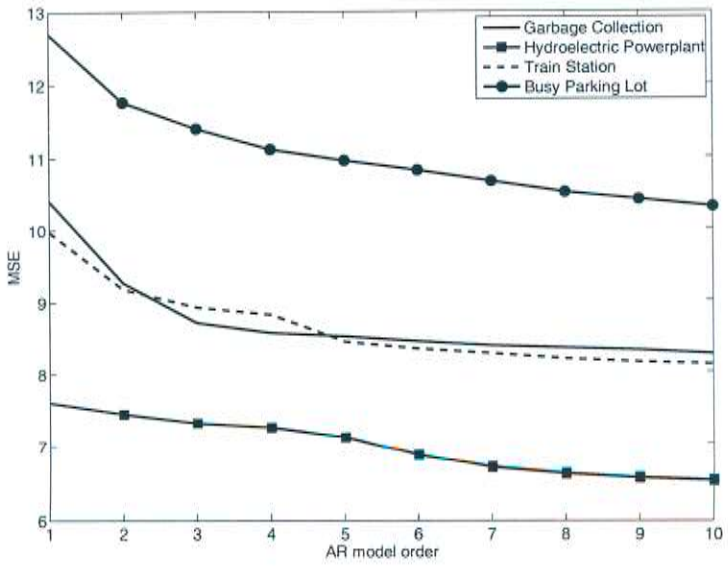


Figure 3.1: AR-Model Training Error

The State Transition Probability

The noise model implicitly defines the noise transition probability $p(n_t|\bar{n}_{t-1})$ as a Gaussian distribution with covariance Σ_{noise} around the predicted noise $A\bar{n}_{t-1}$:

$$p(n_t|\bar{n}_{t-1}) = A\bar{n}_{t-1} + p(\epsilon_t) = \mathcal{N}(n_t; A\bar{n}_{t-1}, \Sigma_{noise})$$

The state transition probability $p(\bar{n}_t|\bar{n}_{t-1})$ is just dependent on the probability of n_t, since the $n_{t-1}, \ldots, n_{t-(l-1)}$ are already determined. They are given by $\bar{n}_{t-1}$. Hence $p(\bar{n}_t|\bar{n}_{t-1}) = p(n_t|\bar{n}_{t-1})$, i.e.

$$p(\bar{n}_t|\bar{n}_{t-1}) = \mathcal{N}(n_t; A\bar{n}_{t-1}, \Sigma_{noise}) \tag{3.3}$$

Sampling form $p(\bar{n}_t|\bar{n}_{t-1})$ — as required by the Boostrap filter — can be performed by sampling n_t from the noise transition probability $p(n_t|\bar{n}_{t-1})$ and adding the $n_{t-1}, \ldots, n_{t-(l-1)}$ to form $\bar{n}_t = (n_t^T, \ldots, n_{t-(l-1)}^T)^T$. Using a model order l bigger than one factually circumvents the restrictive assumption of the noise being a Markov chain, which presumes that the current noise spectrum is just dependent on the last noise spectrum. This method of merging states to overcome the limitations of the Markov assumption is well known in the statistical literature and for example proposed in [34].

Problems with Higher Model Orders
We decided to use only first order models because of the problems introduced by higher model orders. Reliably learning an lth order LP matrix with $d \times (d \cdot l)$ coefficients necessitates a huge amount of training data. Furthermore, errors introduced by using noisy hypotheses of the $n_{t-1}, \ldots, n_{t-l}$ might sum up and actually worsen the particle prediction in the Bootstrap filter. Moreover, Raj et al. concluded before that higher order models don't result in better performance of the particle filter since noise prediction does not significantly improve with higher model orders in the first place [39].

The Prior Distribution
Initialization of the Bootstrap filter is performed by sampling from the (state) prior distribution. Raj et al. did not say, how they initialized the samples at time $t = 0$. We decided to learn $p(n_0)$ — the distribution of noise spectra — as a Gaussian mixture distribution, since we only used first order models in our experiments. Learning $p(\bar{n}_0)$ for higher model orders is problematic, if few noise data is available. It is, however, possible to use $p(n_0)$ to produce samples from $p(\bar{n}_0)$ by sampling $n_{-(l-1)}$ from $p(n_0)$ and completing the vector by predicting the $n_{-(l-1-i)}$, $i = 1, ..., l-1$ using an ith order model or by sampling from the noise transition probability of the respective AR model.

3.2 Relating States and Observations

Observations y_t are corrupted speech spectra in our case. It is necessary to find a way to relate them to the state vector $\bar{n}_t$ so that the likelihood (weight) can be calculated for a certain noise hypothesis (sample) $\bar{n}_t^{(j)}$. The current observation is caused by the current noise spectrum n_t, the current speech spectrum x_t and — as we will see later — the current relative phase between the both. Therefore it is reasonable to assume that preceding noise spectra $n_{t-1}, n_{t-(l-1)}$ do not induce new knowledge into the probability density $p(y_t|\bar{n}_t)$ as long as the noise is additive. Hence, we assume that $p(y_t|\bar{n}_t) = p(y_t|n_t)$, i.e.

$$p(y_t|\bar{n}_t) = p(y_t|n_t)$$

This might not be entirely true, if — what is considered to be "noise" — includes reverberations of speech like for example in far distance recording. Taking this into consideration, it is sufficient to investigate the relationship between n_t and y_t. The principle of superposition holds in the signal domain, i.e. for frames or windows we have

$$y_t^{(s)} = x_t^{(s)} + n_t^{(s)}$$

where superscript (s) denotes the signal domain for discrimination from x_t, n_t and y_t, which are always in the Mel log spectral power domain. Since the Fourier transformation is linear, this gives

$$y_t^{(f)} = x_t^{(f)} + n_t^{(f)}$$

where superscript (f) denotes the Fourier domain. The components of those vectors are complex numbers, which can be represented by magnitude and phase

$$c = \alpha_c \cdot e^{i\varphi_c}$$

with magnitude α_c and phase φ_c. Using this notation, the power spectrum of $y_t^{(f)}$ can be represented as $(\|y_{t,1}^{(f)}\|^2, \ldots, \|y_{t,d}^{(f)}\|^2)$, where

$$\|y_{t,i}^{(f)}(\omega)\|^2 = \alpha_{y_{t,i}} e^{i\varphi_{y_{t,i}}} \cdot \left(\alpha_{y_{t,i}} e^{y_{t,i}}\right)^* = \alpha_{y_{t,i}} e^{i\varphi_{y_{t,i}}} \cdot y_{t,i} e^{-y_{t,i}} = \alpha_{y_{t,i}}^2$$

$$\begin{aligned} \|x_{t,i}^{(f)} + n_{t,i}^{(f)}\|^2 &= \left(\alpha_{x_{t,i}} e^{i\varphi_{x_{t,i}}} + \alpha_{n_{t,i}} e^{i\varphi_{n_{t,i}}}\right) \cdot \left(\alpha_{x_{t,i}} e^{i\varphi_{x_{t,i}}} + \alpha_{n_{t,i}} e^{i\varphi_{n_{t,i}}}\right)^* \\ &= \alpha_{x_{t,i}}^2 + \alpha_{n_{t,i}}^2 + \underbrace{\alpha_{x_{t,i}} e^{i(\varphi_{x_{t,i}} - \varphi_{n_{t,i}})} + \alpha_{x_{t,i}} \alpha_{n_{t,i}} e^{i(\varphi_{x_{t,i}} - \varphi_{n_{t,i}})}}_{= \alpha_{x_{t,i}} \alpha_{n_{t,i}} \cdot 2\cos(\varphi_{x_{t,i}} - \varphi_{n_{t,i}})} \end{aligned}$$

Defining $\theta_{t,i} := (\varphi_{x_{t,i}} - \varphi_{n_{t,i}})$ to be the relative phase between $x_{t,i}^{(f)}$ and $x_{t,i}^{(f)}$, we obtain

$$\|y_{t,i}^{(f)}\|^2 = \|x_{t,i}^{(f)}\|^2 + \|n_{t,i}^{(f)}\|^2 + 2\cos(\theta_{t,i})\sqrt{\|x_{t,i}^{(f)}\|^2 \cdot \|n_{t,i}^{(f)}\|^2} \tag{3.4}$$

The relative phase is typically [35, 26, 39] considered to be zero with the argument that it is zero in average. That, however, leads to problems with the likelihood (weight) evaluations of the Bootstrap filter as we will see later. In chapter 6 we will abandon this simplification, but here — as in the original approach — the relative phase term is omitted, i.e. we have

$$\|y_{t,i}^{(f)}\|^2 = \|x_{t,i}^{(f)}\|^2 + \|n_{t,i}^{(f)}\|^2 \tag{3.5}$$

Denoting $(e^{y_{t,1}}, \ldots, e^{y_{t,d}})^T$ by e^{y_t}, this translates to $e^{y_t} = e^{x_t} + e^{n_t}$ in the log spectral (power) domain with

$$x_{t,i} = log\left(\|x_{t,i}^{(f)}\|^2\right), \quad n_{t,i} = log\left(\|n_{t,i}^{(f)}\|^2\right), \quad y_{t,i} = log\left(\|y_{t,i}^{(f)}\|^2\right)$$

Solving the equation (3.5) for the clean speech spectrum x_t yields

$$x_t = \log(e^{y_t} + e^{n_t}) \tag{3.6}$$

where $\log(x_1, \ldots, x_d)$ again denotes the componentwise application of the log function.

A General Speech Model

Knowing how typical speech spectra look like, we can learn $p(x)$, the probability density for x being clean speech, modelled as a *Gaussian mixture*

$$p(x) = \sum_{k=1}^{K} c_k \mathcal{N}(x; \mu_k, \Sigma_k) \tag{3.7}$$

with diagonal covariance matrices Σ_k as common in speech recognition. Such a mixture can easily be learned by using the *expectation maximization* (EM) algorithm [7] or one of its variants such as *Split and Merge EM* (SMEM) [47] and Monte Carlo EM (MCEM) [31]. This probability distribution can be used to calculate $p(y_t|n_t)$ by exploiting (3.6), the relation between x_t and y_t, n_t because of the following result.

The Fundamental Transformation Law of Probabilities

It is possible to evaluate an "unknown" probability density $p(y)$, if some other probability density $p(x)$ can be found such that x is functionally dependent on y: $x = f(y)$. Then, provided the derivative of $f(y)$ with respect to y is well defined, $p(y)$ can be calculated as

$$p_y(y) = p_x(f(y)) \cdot \left|\frac{df(y)}{dy}\right| \tag{3.8}$$

where $|\cdot|$ is the absolute value if x and y are scalars, the absolute value of the Jacobian determinant if x and y are vectors of the same dimension. This is called the *fundamental transformation law of probabilities* and it holds since the probability density $p(y)$ is just the derivative of its cumulative distribution function

$$P(y') = \int_{-\inf}^{y'} p_y(y)dy = \int_{f^{-1}(-\inf)}^{f^{-1}(y')} p_x(f(y))\frac{df(y)}{dy}dy$$

The substitution rule of integration was applied to obtain the rightmost term from the previous one. We have to take the absolute value of the derivative, since probability densities are supposed to be ≥ 0.

Applying the Fundamental Transformation Law
In our case we have $x_t = f_{n_t}(y_t) = \log(e^{y_t} + e^{n_t})$ (see equation (3.6)) or — split into the d components — $f_{n_t,i}(y_{t,i}) = \log(e^{y_{t,i}} + e^{n_{t,i}})$, $i = 1, \ldots, d$ which can be written

$$f_{n_t,i}(y_{t,i}) = \log\left(e^{y_{t,i}} \cdot (1 + e^{n_{t,i}-y_{t,i}})\right) = y_{t,i} + \log(1 + e^{n_{t,i}-y_{t,i}})$$

The $f_{n_t,j}(y_{t,i})$ are 0 for $i \neq j$. So, the corresponding Jacobian matrix is

$$J := \begin{pmatrix} \frac{df_{n_t,1}(y_t)}{dy_{t,1}} & \cdots & \frac{df_{n_t,1}(y_t)}{dy_{t,d}} \\ \vdots & \ddots & \vdots \\ \frac{df_{n_t,d}(y_t)}{dy_{t,1}} & \cdots & \frac{df_{n_t,d}(y_t)}{dy_{t,d}} \end{pmatrix} = \begin{pmatrix} J_1 & 0 & \ldots & 0 \\ 0 & J_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & J_n \end{pmatrix}$$

with $J_i := 1/(1 - e^{n_{t,i}-y_{t,i}})$ since

$$\frac{df_{n_t,i}(y_t)}{dy_{t,j}} = 1 + \frac{1}{1 + e^{n_{t,i}-y_{t,i}}} \cdot e^{n_{t,i}-y_{t,i}} = \frac{1}{1 + e^{n_{t,i}-y_{t,i}}}$$

for $i = j$ and 0 for $i \neq j$. This gives us the Jacobian determinant needed for the application of the fundamental transformation law of probabilities: $det(J) = \prod_{i=1}^{d} 1/(1 - e^{n_{t,i}-y_{t,i}})$ and $p(y_t|n_t)$ can be evaluated as

$$p(y_t|n_t) = \frac{p_x(f_{n_t}(y_t))}{\prod_{i=1}^{d} |1 - e^{n_{t,i}-y_{t,i}}|} \tag{3.9}$$

There is one remaining problem, however.

Logarithm of a Negative Number
If a noise hypothesis (log spectrum) n_t exceeds the observed (log) spectrum y_t in just one spectral bin, say the ith, then

$$n_{t,i} \geq y_{t,i} \Rightarrow e^{n_{t,i}} \geq e^{y_{t,i}} \Rightarrow e^{n_{t,i}-y_{t,i}} \geq 1$$

So $\log(1 - e^{n_t-y_t})$ and therefore the weight $\omega_t(n_t) = p(y_t|n_t)$ can't be evaluated. This is not a math problem but a modelling problem: we considered noise and speech power spectra (not in the log domain) to be strictly additive (see equation 3.5) by omitting the phase term. Hence it is impossible[1] that $\|n_{t,\omega}^{(f)}\|^2 > \|y_{t,\omega}{}^{(f)}\|^2$ — at least in the model world. That can be translated to probability by setting $p(y_t|n_t) := 0$. Haeb-Umbach and Schmalenstroeer [22] explicitly set the weight to zero for this case. Raj and Singh didn't address the problem in their papers [38, 39].

3.3 Applying the Bootstrap Filter

Now that the noise transition probability $p(\bar{n}_t|\bar{n}_{t-1})$ and the output probability $p(y_t|n_t)$ are known, the Bootstrap filter 2.2 can be applied to the problem. This is straightforward. The resulting

[1]The speech power spectrum $\|x_t^{(f)}\|^2$ is always positive. Superscript (f) again denotes the Fourier domain.

filter is outlined in Algorithm 3.1. Clean speech can be estimated in the inference step (step 4) by selecting an appropriate function h_t.

Algorithm 3.1 Bootstrap Filter for Noise Tracking

1. **Generate noise hypotheses**

   At time zero ($t = 0$) noise hypotheses or particles $\bar{n}_0^{(j)}$ ($j = 1, ..., N$) are drawn from the prior noise density $p(\bar{n}_0)$. If t is bigger than zero, $\bar{n}_t^{(j)}$ is sampled from the noise transition probability $p(\bar{n}_t|\bar{n}_{t-1}^{(j)})$ for $j = 1, ..., N$.

2. **Calculate the normalized weights**

   The likelihood of each noise hypothesis $\bar{n}_t^{(j)}$ is evaluated according to equation (3.9)

   $$p(y_t|\bar{n}_t^{(j)}) = \sum_{k=1}^{K} \frac{c_k \mathcal{N}(y_t + \log(\underline{1} - e^{n_t^{(j)} - y_t}); \mu_k, \Sigma_k)}{\prod_{i=1}^{d} \left|1 - e^{\hat{n}_{t,i}^{(j)} - y_{t,i}}\right|}$$

   if $n_{t,i}^{(j)} < y_{t,i}$ for $i = 1, \ldots, d$. Otherwise $p(y_t|\bar{n}_t^{(j)})$ is set to zero. The normalized weights are calculated as

   $$\tilde{\omega}_t^{(j)} = \frac{p(y_t|\bar{n}_t^{(j)})}{\sum_{m=1}^{N} p(y_t|\bar{n}_t^{(m)})}$$

3. **Resample among the noise hypotheses**

   The normalized weights are used to resample among the noise hypotheses $\bar{n}_t^{(j)}$ ($j = 1, ..., N$) by importance resampling.

4. **Infer clean speech**

   Clean speech is inferred as $\sum_{j=1}^{N} h_t(n_t^{(j)})\tilde{\omega}(n_t^{(j)})$

These steps are repeated with $t \rightarrow (t + 1)$ until all time-frames are processed. It is still necessary to construct an appropriate inference function h_t for inferring clean speech.

3.4 Inferring Clean Speech

The basic idea is to infer clean speech spectra x_t as the mean of x_t given the observations $y_{1:t}$:

$$\mathrm{E}[x_t|y_{1:t}] = \int x_t \cdot p(x_t|y_{1:t}) dx \tag{3.10}$$

Analogous to section 2.2, this can be shown to be the MMSE estimation for clean speech. The noise n_t can be introduced as a hidden variable since $p(x_t|y_{1:t})$ can be calculated as marginal density of $p(x_t, n_t|y_{1:t})$:

$$p(x_t|y_{1:t}) = \int p(x_t, n_t|y_{1:t}) dn_t$$

Further, using

$$p(x_t, n_t|y_{1:t}) = \frac{p(x_t, n_t, y_{1:t})}{p(n_t, y_{1:t})} \cdot \frac{p(n_t, y_{1:t})}{p(y_{1:t})} = p(x_t|y_{1:t}, n_t) \cdot p(n_t|y_{1:t})$$

and changing the order of integration we obtain

$$\mathrm{E}[x_t|y_{1:t}] = \int x \cdot \int p(x_t|y_{1:t}, n_t) \cdot p(n_t|y_{1:t}) dn \, dx$$

$$= \int \underbrace{\int x \cdot p(x_t|y_{1:t}, n_t)dx}_{=:h_t(n_t)} \; p(n_t|y_{1:t})dn \tag{3.11}$$

Since this is equivalent to calculating $E_{p(n_t|y_{1:t})}[h_t(n_t)|y_{1:t}]$, we can use the weighted empirical density $\bar{p}(n_t|y_{1:t})$ provided by the Bootstrap Filter to approximate (3.10) by Monte Carlo integration:

$$\mathrm{E}[x_t|y_{1:t}] \approx \sum_{j=1}^{N} h_t(n_t^{(j)})\tilde{\omega}_t(n_t^{(j)}) \tag{3.12}$$

Furthermore, x_t can be represented as

$$x_t = y_t + log(\underline{1} - e^{n_t - y_t}) \tag{3.13}$$

where $\underline{1} = (1, \ldots, 1)$ denotes a d-dimensional vector of ones. We still need to determine $h_t(n_t^{(j)}) = \int x_t \cdot p(x_t|y_{1:t}, n_t^{(j)})dx$.

Approach 1
The straight forward approach would be to use the established deterministic relationship from above: $x_t = y_t + log(\underline{1} - e^{n_t - y_t})$. Then $p(x_t|y_t, n_t) = \delta_{y_t + log(\underline{1} - e^{n_t - y_t})}(x_t)$ which yields

$$\begin{aligned} h_t^{(1)}(n_t) &= \int x_t \cdot \delta_{y_t + log(\underline{1} - e^{n_t - y_t})}(x_t)dx_t \\ &= y_t + log(\underline{1} - e^{n_t - y_t}) \end{aligned} \tag{3.14}$$

This can be regarded as spectral subtraction in the logarithmic domain (for one noise hypothesis).

Approach 2
The approach proposed by Raj et al. [39] is to use Moreno's *vector Taylor series* (VTS) method [35] which approximates $log(\underline{1} + e^{n_t - x_t})$ by its 0th[2] order Taylor series expansion around the kth Gaussian's mean μ_k. For the case of the Bootstrap filter, where the noise variance is implicitly contained in the different noise hypotheses, this can be derived directly. There is no need of an approximation as we will show in the following. The number of Gaussian in the Gaussian mixture $p(x)$ (see 3.7) can be introduced as a hidden variable k, since $p(x_t|y_t)$ can be represented as the marginal density $p(x_t|y_t) = \sum_{k=1}^{K} p(x_t, k|y_t)$. So, using the equality $p(x_t, k|y_t) = p(x_t|k, y_t) \cdot p(k|y_t, n_t)$, the integral $h_t(n_t)$ in (3.11) can be written

$$\begin{aligned} h_t(n_t) &= \int x_t \cdot \sum_{k=1}^{K} p(x_t|k, y_t, n_t) \cdot p(k|y_t, n_t)\, dx_t \\ &= \sum_{k=1}^{K} p(k|y_t, n_t) \int x_t \cdot p(x_t|k, y_t, n_t)\, dx_t \end{aligned}$$

The question is still how to calculate the $p(k|y_t, n_t)$. First of all

$$p(k|y_t, n_t) = \frac{p(k, y_t|n_t)}{p(y_t|n_t)} = \frac{p(y_t|n_t, k) \cdot p(k|n_t)}{p(y_t|n_t)}$$

Since k — the number of the Gaussian in the mixture that "produced" the clean speech spectrum — can be considered to be stochastically independent of the noise spectrum n_t, we have $p(k|n_t) = p(k) = c_k$ (compare to 3.7) and therefore

$$p(k|y_t, n_t) = \frac{c_k \cdot p(y_t|n_t, k)}{p(y_t|n_t)}$$

[2] Higher order approximations were also examined in [35]

Note that $p(y_t|n_t) = \sum_{k=1}^{K} c_k \cdot p(y_t|n_t, k)$. Now, the noise can be considered to shifts the means μ_k of clean speech to μ'_k[3]. The effect of n_t to the kth Gaussian in the log domain is

$$e^{\mu'_k} = e^{\mu_k} + e^{n_t}$$

(compare to equation (3.6)). Solving for μ_k yields

$$\mu'_k = \mu_k + \underbrace{log(1 + e^{n_t - \mu_k})}_{=:\Delta_{\mu_k, n_t}} \tag{3.15}$$

Instead of shifting the mean, we could conversely shift the corrupted spectrum y_t in the opposite direction to obtain the clean spectrum

$$x_t = y_t - \Delta_{\mu_k, n_t}$$

such that $\mathcal{N}(y_t; \mu'_k, \Sigma_k) = \mathcal{N}(x_t; \mu_k, \Sigma_k)$. Now, using $p(x_t|k, y_t, n_t) = \delta_{y_t - \Delta_{\mu_k, n_t}}(x_t)$ yields

$$\begin{aligned} h_t^{(2)}(n_t) &= \sum_{k=1}^{K} p(k|y_t, n_t) \int x \cdot \delta_{y_t - \Delta_{\mu_k, n_t}}(x_t)\, dx_t \\ &= \sum_{k=1}^{K} p(k|y_t, n_t)\, (y_t - \Delta_{\mu_k, n_t}) \\ &= y_t - \sum_{k=1}^{K} p(k|y_t, n_t) \Delta_{\mu_k, n_t} \end{aligned} \tag{3.16}$$

The algorithms 3.2 and 3.3 summarize how clean speech inference can be performed in step 4 of the noise tracking bootstrap filter (Algorithm 3.1) using the two presented approaches.

Algorithm 3.2 Inferring Clean Speech - Approach 1

1. for $j = 1$ to N do calculate $\hat{x}_t^{(j)} = y_t + log(1 - e^{n_t^{(j)} - y_t})$
2. $\hat{x}_t = \sum_{j=1}^{N} \bar{\omega}_t(n_t^{(j)}) \hat{x}_t^{(j)}$

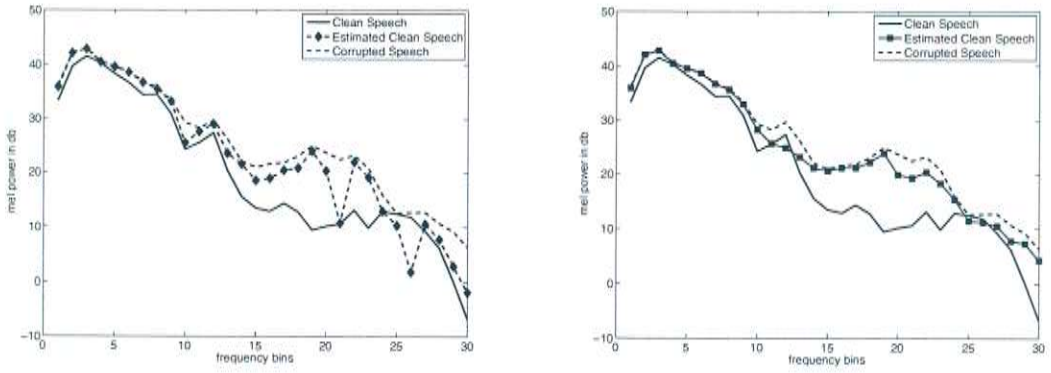
Algorithm 3.3 Inferring Clean Speech - Approach 2

1. for $j = 1$ to N do
   - for $k = 1$ to K do calculate $p(y_t|\bar{n}_t^{(j)}, k) = \frac{c_k \mathcal{N}(y_t + \log(1 - e^{n_t^{(j)} - y_t}); \mu_k, \Sigma_k)}{\prod_{i=1}^{d} \left| 1 - e^{\hat{n}_{t,i}^{(j)} - y_{t,i}} \right|}$
   - calculate $p(y_t|\bar{n}_t^{(j)}) = \sum_{k=1}^{K} p(y_t|\bar{n}_t^{(j)}, k)$
   - for $k = 1$ to K do
     calculate $p(k|y_t, n_t^{(j)}) = \frac{p(y_t|\bar{n}_t^{(j)}, k)}{p(y_t|\bar{n}_t^{(j)})}$
     calculate $\Delta_{\mu_k, n_t^{(j)}} = y_t + log(1 - e^{n_t^{(j)} - y_t})$
     end for
   - calculate $\hat{x}_t^{(j)} = y_t - \sum_{k=1}^{K} p(k|y_t, n_t^{(j)}) \Delta_{\mu_k, n_t^{(j)}}$

   end for
2. infer estimated clean speech as $\hat{x}_t = \sum_{j=1}^{N} \bar{\omega}_t(n_t^{(j)}) \hat{x}_t^{(j)}$

[3] This is the model adaptation approach

When looking at estimated clean speech, the spectra obtain with $h^{(1)}$ are typically more jagged, while those obtained with $h^{(2)}$ are smoother. Figure 3.2(a) shows a typical example of an estimated clean speech spectrum obtained with $h^{(1)}$. Figure 3.2(b) shows the corresponding estimated clean speech spectrum obtained with $h^{(2)}$.



(a) obtained with $h1$

(b) obtained with $h2$

Figure 3.2: Estimated Clean Speech

Infer the noise first

Haeb-Umbach and Schmalenstroeer [22] proposed another approach for estimating the clean speech spectrum. It consists in inferring the noise spectrum n_t in step 4 of algorithm 3.1 instead of the clean speech spectrum x_t. The inferred noise spectrum is simply the mean of n_t given $y_{0:t}$:

$$\hat{n}_t := \int n_t \cdot p(n_t|y_{0:t})dn_t \approx \sum_{j=1}^{N} n_t^{(j)} \cdot \bar{\omega}_t(n_t^{(j)})$$

Haeb-Umbach and Schmalenstroeer only considered $h_t^{(2)}$ following Raj et al.'s approach. Usage of $h_t^{(1)}$ has — to our knowledge — never been considered in the context, neither for the inference using equation (3.11) nor for this case.

4 Refinements

4.1 Getting the Filter to Work

In practice there are some issues that prevent the designed filter from working well. The major issue is the problem explained in section 3.2 — that noise hypotheses are not allowed to exceed the observed, contaminated spectrum. We assigned a zero weight in this case. Unfortunately, this procedure comes with a side-effect: overestimations of the actual noise as well as cancellation due to relative phase differences between noise and speech can cause severe decimation (attrition) among the particles up to a complete annihilation of the "population", if all weights are zero. We call the latter case a *dropout*. If a dropout occurs, the weights can't be normalized by dividing through their sum (the sum is zero). A reasonable workaround is to set the normalized weights to $1/N$, since all weights being zero means all noise hypotheses are equally bad. Furthermore, the estimated clean speech spectrum $\tilde{x}_t$ should be replaced by its corrupted counterpart y_t in this case.

Reinitialization
Setting all weights to $1/N$, however, introduces a new problem: the equal weighting causes the particles to evolve only according to the noise transition probability, completely independent of the observed corrupted spectra. Sometimes the noise trajectory is recovered, i.e. one of the weights gets a non-zero weight after some time. As the trajectory might also be lost forever, the particle filter should be reinitialized if the sum of weights

$$\sum_{j=1}^{N} \omega(n_t^{(j)}) = \sum_{j=1}^{N} p(y_t|n_t^{(j)})$$

is zero or unusually small[1] for a continuous period of time (in our case 100 ms)[2]. Reinitialization of the particle filter means repetition of step 1 of algorithm 3.1, i.e. initializing the particles with samples from the prior distribution $p(\bar{n})$ of noise spectra. If the current noise spectrum is "unusual" and therefore unlikely to be drawn from $p(\bar{n})$, it is sometimes necessary to perform several reinitializations in succession before the noise trajectory is recovered.

A More Sophisticated Initialization
Another point of criticism is that there is more knowledge available than the static prior distribution of noise. We actually have the corrupted spectrum y_t. Further, *voice activity detection* (VAD) or a previous ASR run could be used to identify the last silence or noise frame before the start of the current utterance. Denoting the corresponding log power spectrum by $\tilde{n}_{t-1}$, we could initialize the samples at the start of each utterance with $p(n_t|\tilde{n}_{t-1})$. Unfortunately, this "perfect" initialization is problematic in praxis since VAD and ASR don't work reliable. So, $\bar{n}$ is not really guaranteed to be a noise frame. Moreover, this procedure can't be used for reinitialization when the current frame is a speech frame. Another approach consists in generating clean speech samples $x^{(j)}$ from $p(x)$. Then, using the current observation y_t, the corresponding noise sample can be inferred as

$$n_t^{(j)} = y_t + log(1 - e^{x^{(j)} - y_t})$$

provided $x_{\omega_i}^{(j)} < y_{t,\omega_i}$ for all i. If this condition is not satisfied, another speech sample $x^{(j)}$ is drawn and the whole procedure is repeated until $x^{(j)}$ is finally accepted or a certain number of iterations

[1] this is another indicator for divergence from the actual noise trajectory
[2] We have already described this method in [13] which will appear in Proceedings of Interspeech 2006.

has passed. In the second case $n_t(j)$ is sampled from $p(\bar{n}_t)$ just to have a jth noise sample, though it might get a zero weight. The advantage of this approach is that the obtained noise samples have a very high acceptance rate, i.e. they get a weight $\neq 0$. The trajectory is often immediately found again. Successive reinitializations occur only seldom. Moreover there is typically far more data available for learning the distribution of clean speech than there is for the noise.

Underestimation
Occasionally noise hypotheses drop to unusually low levels (below -30 db) — especially when the current noise is atypical, i.e. mismatched to the noise model. The problem is that the hypotheses might stabilize there or just continue to drop without ever getting too low of a likelihood to be reinitialized. To prevent this it might be a good idea not to allow them to "go" there in the first place. That's why we set a noise hypothesis's likelihood to zero, if its average (over the dimensions)

$$\frac{1}{d}\sum_{i=1}^{d} n_{t,i}^{(j)}$$

drops below a certain lower bound like -30 db. Alternatively its median, maximum or minimum could be used.

A Fast Acceptance Test
The best solution to handling sample attrition and dropouts would of course be not to let them happen. In fact, the dropouts can be reduced by increasing the number of Samples (N), which however greatly increases the computational time. We propose to use a *fast acceptance test* that virtually increases the number of samples when necessary. It works as follows: when generating noise hypotheses for time t (Algorithm 3.1 - step 1) by sampling from the state transition probability, the drawn sample $n_t^{(j)}$ is rejected if not $(n_{t,i}^{(j)} < y_{t,i}\ \forall i)$. In case of rejection we randomly select $s \in \{1,\ldots,N\}$ and sample $n_t^{(j)}$ from $p(n_t|\bar{n}_{t-1}^{(s)})$ until it is accepted or a certain number B of iterations has passed.

Algorithm 4.1 Efficiently Sampling from the Transition Probability

```
for j = 1 to N do
    l = 0
    s = j
    accept = false
    while (l < B) and (accept == false)
        sample n_t^(H) from p(n_t | n̄_{t-1}^(s))
        if (n_{t,i}^(H) < y_{t,i} ∀i)
            accept = true
        else
            randomly select s ∈ {1,...,N}
            l = l + 1
    end while
    n_t^(j) = n_t^(H)
end for
```

The advantage of this approach is that the number of samples stays constant. The worst case computational time is limited by B.

4.2 Efficient Noise Compensation

Inferring clean speech using h_2 (see section 3.4) is computationally very inefficient, since the logarithm and the exponential function have to be evaluated K — this is the number of Gaussians in the mixture of the clean speech distribution — times more often than when using h_1. This matters since execution of those functions by the MPU typically takes several hundred clock cycles

while simpler operations like multiplications and additions take one. So, cutting down the number of times one of those "expensive" functions is computed, can greatly enhance the speed of the total algorithm, which is dominated by noise inference in case of h_2. This can be achieved by reducing the sum in equation 3.12

$$\sum_{j=1}^{N} h_t(n_t^{(j)})\tilde{\omega}_t(n_t^{(j)})$$

to the relevant summands, i.e. the ones with a $\tilde{\omega}_t(n_t^{(j)})$ that is bigger than a certain weight threshold

$$\left(\max_j \tilde{\omega}_t(n_t^{(j)})\right) \cdot \frac{1/a - 1}{b \cdot (N-1)}$$

where $a \in [0,1]$ is the accuracy and where b is an assumed upper bound for the dimension-wise distance between all $h_t(n_t^{(j)})$, $j = 1, \ldots, N$. A value of $b = 100$ db worked good. If a is 1 the accuracy is 100%, meaning every term of the sum has to be computed. A value of $a = 0.9$ means that the resulting estimated clean speech spectrum might be imprecise by up to 10% in a worst-case scenario, that will never occur in practice. We kept this value since it resulted in negligible errors accompanied by a considerable gain in speed - up to 10 times faster. Further reduction did not seem to greatly increase the execution time, which probably means that it is now dominated by the computational time of the particle filter.

4.3 AR-Model Adaptation

In the log spectral domain an AR model learned for a noise type is dependent on the intensity of the noise, since a mismatch in intensity is an additive term $a = (\alpha, \ldots, \alpha)$. So

$$A \cdot (\bar{n}_{t-1} + a) \;=\; A \cdot \bar{n}_{t-1} + A \cdot a \;\neq\; A \cdot \bar{n}_{t-1}$$

This can be compensated by first subtracting an estimate $\hat{a}$ of the noise intensity difference from $\bar{n}_t$ before multiplying with A and then adding it again, afterwards:

$$n_t = A \cdot (\bar{n}_{t-1} - \hat{a}) \;+\; \hat{a}$$

Using this equation for noise prediction instead of (3.1) enabled us to perform experiments with different SNRs without retraining the noise models. In those cases, however, the noise was synthetically added. In reality there will presumably be problems with this approach, since a noise that is less loud is typically farer away, which means there is more distortion due to reverberation or room acoustics. So the noise itself is probably different and has to be relearned.

4.4 Using Linearly Transformed Spectra

As mentioned earlier, speech feature enhancement should be performed as close as possible to the end of the feature extraction stage. This, however, is complicated by the fact that tracing the relationship between speech, noise, corrupted speech troughout the feature extraction stage, becomes increasingly complex. In section 3.2 we tacitly[3] assumed that applying a Mel filterbank doesn't change the established relationship in the log (spectral) power domain. To examine what really happens, we will now investigate the more general case of a linear transformation, which is applicable to Mel filterbanks as well as to discrete cosine transform (DCT) and linear discriminant analysis (LDA). As we will learn, there is a fundamental difference between transformations that were applied before and after going into the log domain.

Linear Transformation before Going into the Log Domain

[3] As it will turn out now, this is not quite true — at least not in general.

Let B be the linear transformation that is performed before going into the log power domain and let $y_t^{(B)} = \log(B \cdot \|y^{(f)}\|^2)$ where $y_t^{(f)}$ again denotes the corrupted frame in the Fourier domain. Using $\|y_t^{(f)}\|^2 = \|x_t^{(f)}\|^2 + \|n_t^{(f)}\|^2$, we have

$$\begin{aligned} y_t^{(B)} &= \log\left(B \cdot \|y_t^f\|^2)\right) \\ &= \log\left(B \cdot (\|x_t^f\|^2 + \|n_t^f\|^2)\right) \\ &= \log\left(B \cdot \|x_t^f\|^2 + B \cdot \|n_t^f\|^2\right) \\ &= \log\left(e^{\log(B \cdot \|x_t^f\|^2)} + e^{\log(B \cdot \|n_t^f\|^2)}\right) \\ &= \log\left(e^{x_t^{(B)}} + e^{n_t^{(B)}}\right) \end{aligned}$$

Solving for $x_t^{(B)}$ yields $x_t^{(B)} = \log\left(e^{y_t^{(B)}} - e^{n_t^{(B)}}\right)$.

Comparison with equation (3.6) shows that the noise compensation filter does not need to be modified. The clean speech distribution $p(x)$ as well as the noise model, consisting of the noise transition probability $p(n_t|n_{t-1})$, the prior distribution $p(n)$ and Σ_{noise} have to be relearned, however.

Linear Transformation after Going into Log Domain

Denoting the linear transformation that is performed after going into the log domain by C and writing $C \cdot \log(\|y^{(f)}\|^2)$ as $y_t^{(C)}$, we get

$$\begin{aligned} y_t^{(C)} &= C \cdot \log\left(\|y_t^f\|^2\right) \\ &= C \cdot \log\left(\|x_t^f\|^2 + \|n_t^f\|^2\right) \\ &= C \cdot \log\left(e^{\log(\|x_t^f\|^2)} + e^{\log(\|n_t^f\|^2)}\right) \\ &= C \cdot \log\left(e^{C^{-1}C\log(\|x_t^f\|^2)} + e^{C^{-1}C\log(\|n_t^f\|^2)}\right) \\ &= C \cdot \log\left(e^{C^{-1}x_t^{(C)}} + e^{C^{-1}n_t^{(C)}}\right) \end{aligned}$$

Solving for $x_t^{(C)}$ yields

$$f_{n_t,i}(y_t) := x_t^{(C)} = C_{i,:} \cdot log(e^{C^{-1}y_t^{(C)}} - e^{C^{-1}n_t^{(C)}})$$

where $C_{i,:}$ denotes the ith row vector of C. We have to calculate the corresponding Jacobian matrix to apply the fundamental transformation law of probabilities (see section (3.8). Defining $C^{-1} =: D = (d_{i,j})_{i,j}$ its elements $J_{i,j} = df_{n_t,i}(y_t)/dy_{t,j}$ are

$$\begin{aligned} J_{i,j} &= \frac{\sum_{k=1}^d c_{i,k} \log\left(\exp(\sum_{l=1}^d d_{k,l}y_{t,l}^{(C)}) - \exp(\sum_{l=1}^d d_{k,l}n_{t,l}^{(C)})\right)}{dy_{t,j}^{(C)}} \\ &= \sum_{k=1}^d c_{i,k} \cdot \frac{d_{k,j} \cdot \exp(\sum_{l=1}^d d_{k,l}y_{t,l}^{(C)})}{\exp(\sum_{l=1}^d d_{k,l}y_{t,l}^{(C)}) - \exp(\sum_{l=1}^d d_{k,l}n_{t,l}^{(C)})} \\ &= \sum_{k=1}^d c_{i,k} \cdot \frac{d_{k,j}}{1 - \exp(\sum_{l=1}^d d_{k,l}(n_{t,l}^{(C)} - y_{t,l}^{(C)})} \end{aligned}$$

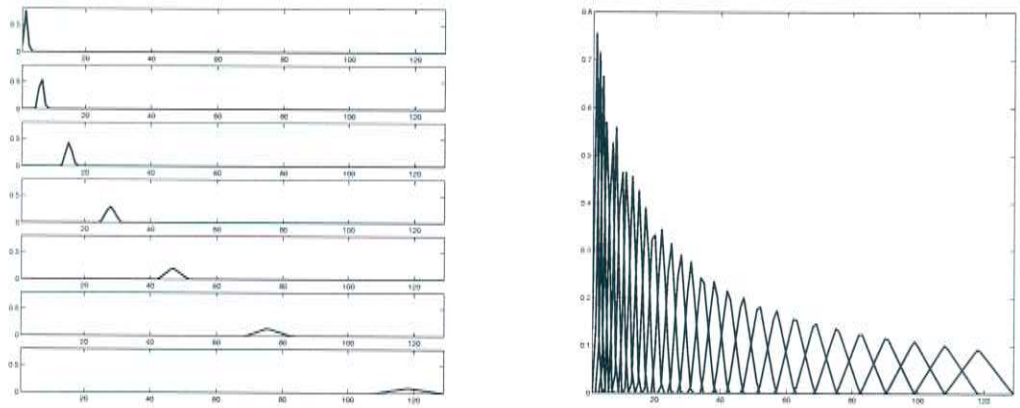
Thus, $p(y_t^{(C)})$ can be calculated as

$$p(y_t^{(C)}) = p(f_{n_t}(y_t^{(C)})) \cdot abs(|J|)$$

Unfortunately, this procedure only works if the linear transformation has rank d, i.e. if the transformation is really mapping the d-dimensional space onto itself and not just on a subspace. In speech recognition actually all transformations (be it MEL-FB, DCT or LDA) are used to reduce the dimension of the input space. So they don't have rank d and this is very problematic!

Mel Filterbanks

Mel filterbanks are typically applied during the feature extraction stage to reduce the spectral dimension while mimicking the logarithmic frequency perception of the human auditory system. It is a set of filters of ascending frequency and increasing bandwith, whose power is individually integrated over the frequency domain. In practice this is typically implemented by multiplying the d dimensional power spectra with a $d' \times d$ ($d' < d$) MEL-FB matrix, whose rows represent the different filters:



(a) some single rows

(b) all rows in one figure

Figure 4.1: MEL-FB matrix

This is a linear transformation with rank $< d$. Therefore it should be applied before going into the log domain, like it is specified for Sphinx III[4] — the speech recognizer used by Raj et al. [39].

Applying it after going into the log power domain causes unresolvable mathematical problems with the likelihood computations.

DCT & LDA

Discrete cosine transformation (DCT) and *linear discriminant analysis* (LDA) are always applied after going into the log domain. Furthermore, the dimension of the respective output vector is typically reduced by omitting the higher coefficients. However, the DCT can still be split into a rank d linear transformation into the Mel Cepstral domain and a following dimensional reduction step. Hence, the Bootstrap filter could be applied in the unreduced Cepstral domain and speech feature enhancement would be closer to the end of the feature extraction stage. Moreover, the assumption of stochastic independence of the frequency bins is closer to "satisfied" in the Cepstral domain because of the DCT's decorrelating property. For the LDA this gets more complicated.

[4]Sphinx is maintained and developed by both the computer science and electrical and computer engineering departments at the Carnegie Mellon University in Pittsburgh.

4.5 Warped MVDR Spectra

Focusing on robust features, the disadvantage of using log Mel power spectra is the equal weighting of spectral peaks and valleys as it is well known that noise is mainly present in low energy regions. To overcome this drawback we estimate the spectrum by the warped and scaled *minimum variance distortionless response* (MVDR) [50] spectral envelope as it provides an accurate description only for spectral peaks. For the representation of valleys no information about the fine spectral structure is preserved, limiting the description more or less to the energy levels. Therefore, spectralenvelopes are more robust to noise than their power spectrum counterparts. The MVDR is used instead of the widely known *linear prediction* (LP) as it has been shown that MVDR spectral estimation overcomes the problems in modeling voiced speech associated with LP spectral estimation techniques [36]. To provide a better approximation of the relevant aspects of the human auditory system, we have applied the well-known technique of pre-warping — a time-domain technique to estimate an all-pole model based on a warped frequency axis such as the Mel scale — to the MVDR spectral estimate. A linear filterbank consisting of equispaced isosceles triangular filters is used to reduce the spectral dimension to 30 like in the case of the Mel filterbank.

Figure 4.2: MVDR Perceptual Spectral Estimation

It is mathematically unclear, how estimating the log Mel power spectra with the warped MVDR affects the relationship between speech, noise and corrupted speech. We assumed the relationship to be the same as the one derived in section 3.2, though this is probably not entirely true.

5 A Phoneme Specific Filter

In section 3.2 the likelihood evaluation of noise hypotheses $n_t^{(j)}$ for the Bootstrap Filter was derived by using the fundamental transformation law of probability and a Gaussian mixture model for clean speech (equation (3.7)). This general speech model, proposed by Raj et al. [38, 39] and earlier by Moreno [35], is very general and in particular static: it just tells us the probability of a spectrum x being a clean speech spectrum — completely independent of time. Speech however isn't static. It consists of different phonemes or subphonemes whose spectra are approximately stationary. Therefore we propose to model speech as a time-dependent phoneme-specific Gaussian mixture distribution

$$p_{phon(t)}(x) = \sum_{k=1}^{K} c_{k,phon(t)} \mathcal{N}(x; \mu_{k,phon(t)}, \Sigma_{k,phon(t)})$$

where $phon(t)$ denotes the phoneme spoken at time t. This means that each phoneme is modelled as a Gaussian mixture distribution and the clean speech model at time t is the Gaussian mixture distribution of phoneme $phon(t)$. Figure 5.2 visualizes the differences of the models. While the probability distribution of the general model (figure 5.2(a)) is comparably broad, the phoneme-specific models (figures 5.2(b) - 5.2(d)) are more confined and obviously show more details. The problem is that the phoneme sequence is not known in advance, which would render the proposed approach purely hypothetical, if it wouldn't be possible to obtain a phoneme sequence hypothesis by using the following 'two-pass' approach:

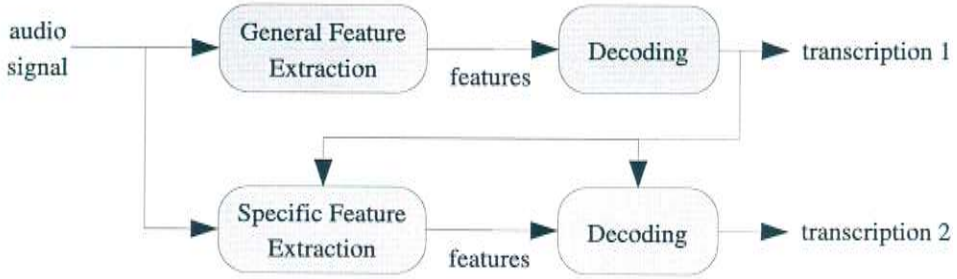


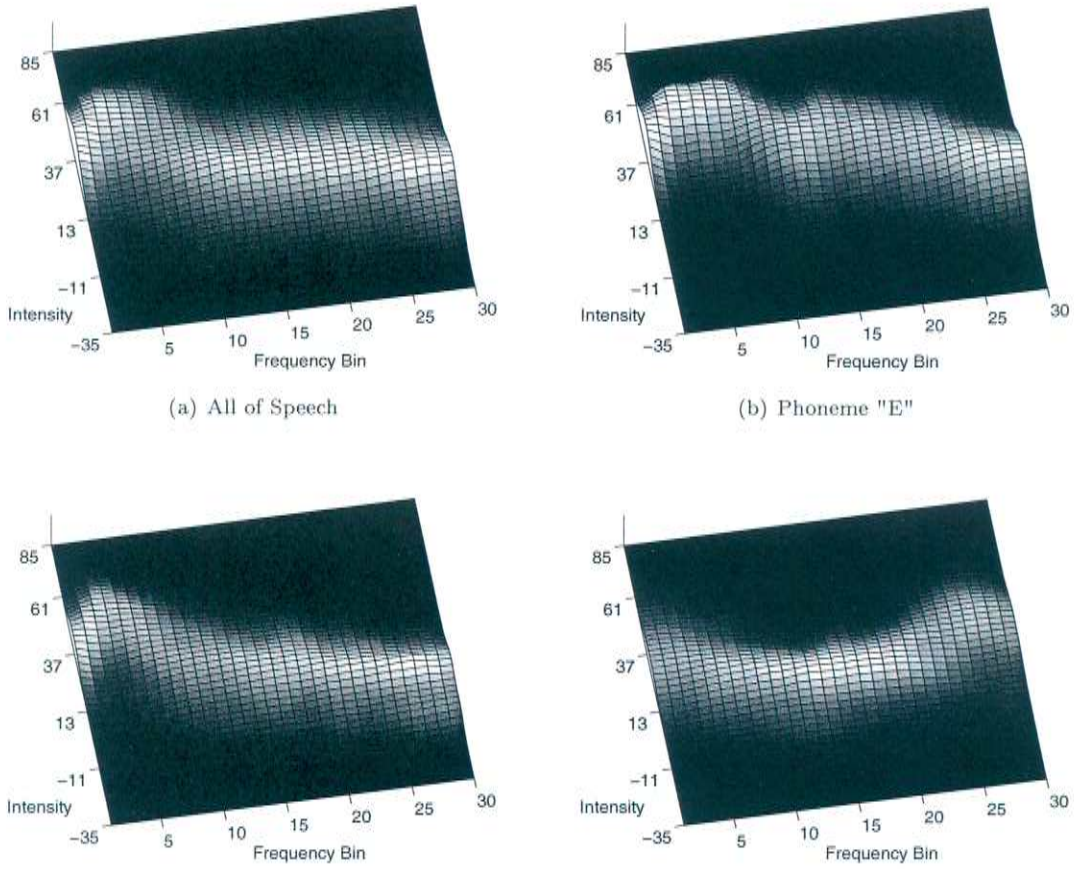
Figure 5.1: Two Pass Approach

In the first pass feature extraction includes speech feature enhancement with the Bootstrap filter using the general speech model (general feature extraction). The resulting features are fed into the decoder to obtain a first phoneme sequence hypothesis (transcription). In the second (and following) pass(es), the hypothesis of the previous pass enables us to use the Bootstrap Filter with the phoneme-specific speech model (specific feature extraction). This way the more sophisticated acoustic, word and language models of the speech recognizer are incorporated into speech feature enhancement.

Unfortunately, this procedure introduces a new problem: by correcting all corrupted speech spectra towards the hypothesis from the previous pass we might tie ourselves too strongly to that hypothesis. That hypothesis can, however, be wrong. Furthermore, the switching between phonemes can cause a very sudden change of the noise hypotheses' likelihoods which might destabilize the Bootstrap Filter. Therefore we decided to interpolate the phoneme-specific with the general speech model to form the *mixture model*

$$p_{mix(t)}(x) = \alpha \cdot p_{phon(t)}(x) + (1 - \alpha) \cdot p(x)$$

with mixture weight α. We used an equal weighting of the two models. Better results might be achieved by learning these weights for the data, or by dynamical adaptation based on the the speech recognizer's confidence.



(a) All of Speech

(b) Phoneme "E"

(c) Phoneme "M"

(d) Phoneme "S"

Figure 5.2: Probability Distributions of Speech Spectra

It is worth mentioning that this phoneme-specific approach is not restricted to the Bootstrap Filter. It can be regarded as a general extension to all methods that use this kind of general speech model — starting with the vector Taylor series approach [35] — and it is easy to implement, since replacement of the general speech model with a phoneme-specific or mixture model does not necessitate fundamental changes to the algorithms. Note, that the three models $p(x)$, $p_{phon(t)}(x)$, $p_{mix(t)}(x)$ are all Gaussian mixture distributions. So, the only thing that has to be done is to dynamically switch the speech model according to the current phoneme hypothesis $phon(t)$.

6 Incorporating the Relative Phase

In section 3.2 the relative phase was considered to be zero, which led to a simplified relation (equation (3.5)) between speech, noise and corrupted speech. As stated in section 4.1, the effect of this simplification is that noise hypotheses are not allowed to exceed the observed, contaminated spectrum. The result is that overestimations of the actual noise as well as cancellation due to relative phase differences between noise and speech can cause severe sample attrition, which prevents the Bootstrap filter from working well[1]. Section 4.1 was primarily concerned with lessening the repercussions. Here, however, we will approach the problem by not making the previous assumption that the phase term $2cos(\theta_t)\|x^{(f)}\|\|n^{(f)}\|$ is zero.

6.1 A New Relation between States and Observations

Generally solving equation (3.4) for $\|x^{(f)}\|^2$ yields

$$
\begin{aligned}
& \|y^{(f)}\|^2 = \|x^{(f)}\|^2 + \|n^{(f)}\|^2 + 2cos(\theta_t)\|x^{(f)}\|\|n^{(f)}\| \\
\Leftrightarrow\ & \|x^{(f)}\|^2 + 2cos(\theta_t)\|x^{(f)}\|\|n^{(f)}\| = \|y^{(f)}\|^2 - \|n^{(f)}\|^2 \\
\Leftrightarrow\ & \left(\|x^{(f)}\| + cos(\theta_t)\|n^{(f)}\|\right)^2 - cos(\theta_t)^2\|n^{(f)}\|^2 = \|y^{(f)}\|^2 - \|n^{(f)}\|^2 \\
\Leftrightarrow\ & \left(\|x^{(f)}\| + cos(\theta_t)\|n^{(f)}\|\right)^2 = \|y^{(f)}\|^2 - \|n^{(f)}\|^2 + cos(\theta_t)^2\|n^{(f)}\|^2 \\
\Leftrightarrow\ & \|x^{(f)}\| + cos(\theta_t)\|n^{(f)}\| = \sqrt{\|y^{(f)}\|^2 + \left(cos(\theta_t)^2 - 1\right)\|n^{(f)}\|^2} \\
\Leftrightarrow\ & \|x^{(f)}\| = \sqrt{\|y^{(f)}\|^2 + \left(cos(\theta_t)^2 - 1\right)\|n^{(f)}\|^2} - cos(\theta_t)\|n^{(f)}\| \\
\Leftrightarrow\ & \|x^{(f)}\|^2 = \left(\sqrt{\|y^{(f)}\|^2 + \left(cos(\theta_t)^2 - 1\right)\|n^{(f)}\|^2} - cos(\theta_t)\|n^{(f)}\|\right)^2
\end{aligned}
$$

Substituting $\|x_t^{(f)}\|^2$, $\|n_t^{(f)}\|^2$ and $\|y_t^{(f)}\|^2$ by

$$\|x_t^{(f)}\|^2 = e^{x_t}, \quad \|n_t^{(f)}\|^2 = e^{n_t}, \quad \|y_t^{(f)}\|^2 = e^{y_t}$$

to move into the log spectral domain gives

$$
\begin{aligned}
& e^{x_t} = \left(\sqrt{e^{y_t} + (cos(\theta_t)^2 - 1)e^{n_t}} - cos(\theta_t)\sqrt{e^{n_t}}\right)^2 \\
\Leftrightarrow\ & e^{x_t} = \left(\sqrt{e^{n_t}} \cdot \sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\sqrt{e^{n_t}}\right)^2 \\
\Leftrightarrow\ & e^{x_t} = \left(\sqrt{e^{n_t}} \cdot \left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\right)\right)^2 \\
\Leftrightarrow\ & e^{x_t} = e^{n_t} \cdot \left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\right)^2 \\
\Leftrightarrow\ & x_t = n_t + \log\left(\left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\right)^2\right)
\end{aligned}
$$

[1] This has before been identified as the source of the problem by Haeb-Umbach and Schmalenstroeer [22].

$$\Leftrightarrow \quad x_t = n_t + 2 \cdot \log\left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\right) \tag{6.1}$$

Analogous to section 3.2

$$f_{n_t,\theta_t}(y_t) := n_t + 2 \cdot \log\left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\right)$$

is defined in order to apply the fundamental transformation law of probabilities (equation (3.8)), only this time dependent on θ_t. The elements of the required Jacobian matrix $(f_{n_{t,i},\theta_{t,i}}(y_t)/dy_{t,j})_{i,j}$ are

$$\begin{aligned}
\frac{df_{n_{t,i},\theta_{t,i}}(y_t)}{dy_{t,j}} &= \frac{d\left(n_{t,i} + 2 \cdot \log\left(\sqrt{e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1} - cos(\theta_{t,i})\right)\right)}{dy_{t,j}} \\
&= 2 \cdot \frac{\left(e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1\right)^{-1/2} \cdot e^{y_{t,i} - n_{t,i}}}{\sqrt{e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1} - cos(\theta_{t,i})} \\
&= 2 \cdot \frac{e^{y_{t,i} - n_{t,i}}}{\left(\sqrt{e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1} - cos(\theta_{t,i})\right)\sqrt{e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1}}
\end{aligned}$$

if $i = j$ and 0 otherwise. Hence, the Jacobian determinant is the product of the diagonal elements and $p(y_t|n_t, \theta_t)$ can be determined as

$$p(y_t|n_t, \theta_t) = p_x(f_{n_t,\theta_t}(y_t)) \cdot \prod_{i=1}^{d} \left|\frac{df_{n_{t,i},\theta_{t,i}}(y_t)}{dy_{t,i}}\right| \tag{6.2}$$

if $(e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1) > 0$ and $(\sqrt{e^{y_{t,i} - n_{t,i}} + cos(\theta_{t,i})^2 - 1} - cos(\theta_{t,i}) > 0)$. If these condition are not met $f_{n_t,\theta_t}(y_t)$ can't be computed. The reason is an impossible constellation of y_t, n_t and θ_t (compare to section 3.2) and thus $p(y_t|n_t, \theta_t)$ is set to zero for this case. Now, assuming that the noise spectrum n_t and the relative phase θ_t between speech and noise are stochastically independent, i.e. $p(\theta_t|n_t) = p(\theta_t)$, we have

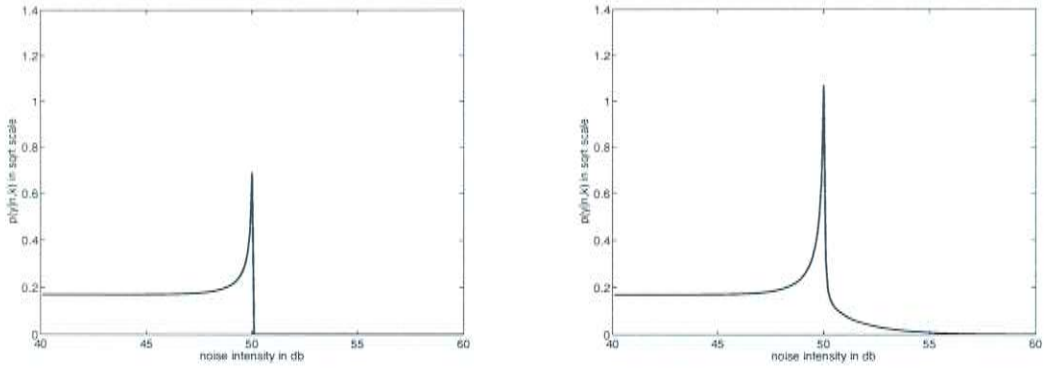
$$p(y_t, \theta_t|n_t) \;=\; p(y_t|n_t, \theta_t) \cdot p(\theta_t|n_t) \;=\; p(y_t|n_t, \theta_t) \cdot p(\theta_t)$$

and $p(y_t|n_t)$ can be calculated as the marginal density of $p(y_t, \theta_t|n_t)$:

$$p(y_t|n_t) \;=\; \int_{-\pi}^{\pi} p(y_t, \theta_t|n_t)\, d\theta_t \;=\; \int_{-\pi}^{\pi} p(y_t|n_t, \theta_t) \cdot p(\theta_t)\, d\theta_t \tag{6.3}$$

Figure 6.1 depicts the difference between this approach and the original calculation (equation 3.9) of $p(y_t|n_t)$. The two figures show the likelihood $p(y_{t,i}|n_{t,i}, k)$ for one spectral bin and for one Gaussian under the assumption that the phase is uniformly distributed. The Gaussian has a mean of 40 and a variance of 10. The corrupted speech spectrum is 50. It can clearly be seen that the original likelihood (figure 6.1(a)) is cut off if the noise hypothesis exceeds the corrupted speech spectrum, while the newly proposed method (figure 6.1(b)) still has some probability after that point and is generally broader. This might look good, but there is a problem with the new approach: the integral (6.3) does not seem to be analytical — at least we did not find an analytical solution nor a good Taylor series approximation — even with the simplifying assumption that θ_t is uniformly distributed on $[-\pi, \pi]$. Therefore we decided to approximate the likelihood numerically, which can be achieved by the Monte Carlo method with samples $\theta_t^{(m)}$, $m = 1, \ldots, M$ drawn from $p(\theta_t)$:

$$p(y_t|n_t) \;\approx\; \frac{1}{M} \sum_{m=1}^{M} p(y_t|n_t, \theta_t^{(m)}) \tag{6.4}$$



(a) original likelihood

(b) likelihood with incorporation of the phase

Figure 6.1: Comparison of the two approaches

This approximation has to be performed for each noise hypothesis $n_t^{(j)}$ — meaning $p(y_t|n_t^{(j)}), \theta_t^{(m)}$ must be evaluated $(N \cdot M)$ times. Furthermore, the approximation requires a lot of samples $\theta_t^{(m)}$ if a decent amount of precision is desired. So M should be large (close to $\approx N$), which renders this approach computationally inefficient. Section 6.3 will point out how this can be handled more efficiently. Before, however, we will investigate how the relative phase is distributed in the Mel spectral power domain.

6.2 A Distribution for the Phase

In the Fourier domain the relative phase can be considered to be uniformly distributed in all spectral bins. Further, it can be assumed that it is stochastically independent[2] for different spectral bins. Thus, we consider the θ_{t,ω_i} $i = 1, \ldots, M$, terms to be i.i.d. uniformly distributed. Dimensional reduction by application of a Mel (FFT) or linear filterbank (MVDR) results in a weighted sum of those random variables [9]. The distribution of such a sum was examined by Kamgar-Parsi [25][3]. Deng, Droppo and Acero [9] argue that such a weighted sum is approximately Gaussian distributed because of the *Central Limit Theorem*, which states that a sum of i.i.d random variables is Gaussian if the number of random variables approaches infinity. This, however, is not perfectly true for the lower frequency bins of Mel spectra, which are barely the sum of one spectral bin. We determined the relative phase distribution by simulation to verify those theoretical results. This is possible since the relative phase can be reconstructed by solving (3.4) for θ_t

$$
\begin{aligned}
& \|y^{(f)}\|^2 = \|x^{(f)}\|^2 + \|n^{(f)}\|^2 + 2cos(\theta_t)\|x^{(f)}\|\|n^{(f)}\| \\
\Leftrightarrow \quad & 2\cos(\theta_t)\|x^{(f)}\|\|n^{(f)}\| = \|y^{(f)}\|^2 - \|x^{(f)}\|^2 - \|n^{(f)}\|^2 \\
\Leftrightarrow \quad & \cos(\theta_t) = \frac{\|y^{(f)}\|^2 - \|x^{(f)}\|^2 - \|n^{(f)}\|^2}{2\|x^{(f)}\|\|n^{(f)}\|} \\
\Leftrightarrow \quad & \theta_t = \arccos\left(\frac{\|y^{(f)}\|^2 - \|x^{(f)}\|^2 - \|n^{(f)}\|^2}{2\|x^{(f)}\|\|n^{(f)}\|}\right)
\end{aligned}
$$

where $y_t^{(s)}$ is obtained by synthetically adding known speech $x_t^{(s)}$ and noise $n_t^{(s)}$ signals. Note that arccos doesn't tell us the sign of θ_t. Hence, the resulting distribution of θ_t on the interval $[0, \pi]$ is valid only if the distribution of θ_t is symmetric — which is not known — or if θ_t used as an

[2] In practice neighboring bins influence each other since multiplication with a window function in the time domain causes smoothing (convolution) in the Fourier domain.

[3] unfortunately I couldn't obtain a copy of that paper

argument of a symmetric function, like $\cos(\theta_t)$. Since we are only concerned with the latter case, we don't have to care. Figure 6.2 shows the empirical phase distribution for Mel spectra resulting from a simulation with 45 minutes of speech. Frequency bin #1 is clearly distributed uniformly,

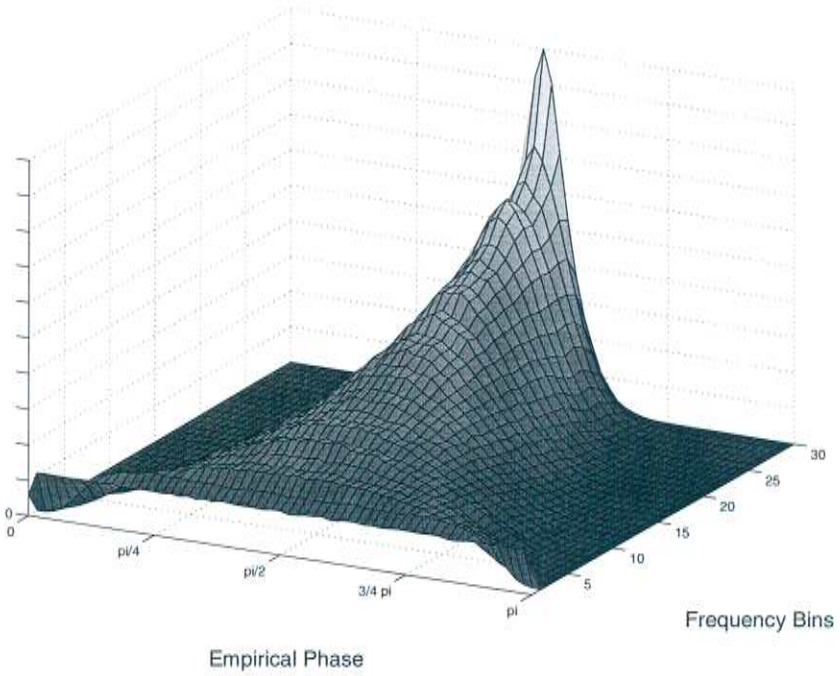
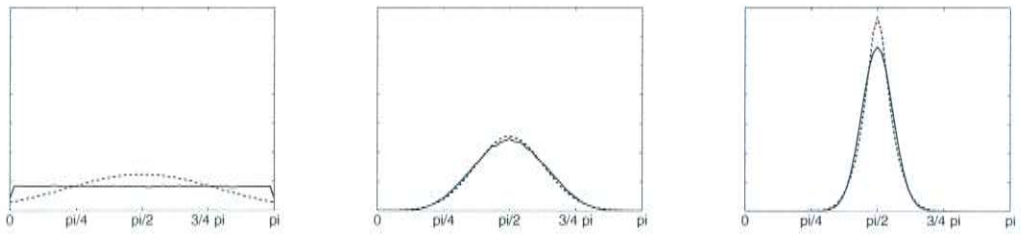


Figure 6.2: The Empirical Relative Phase Distribution

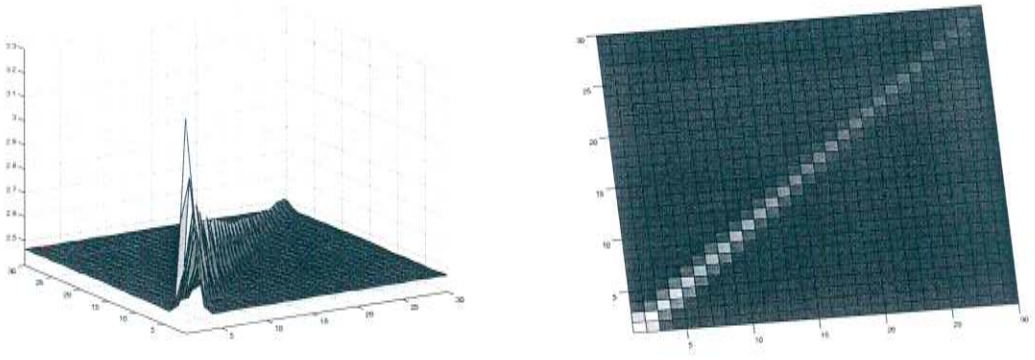
while higher frequency bins seem to follow a Gaussian distribution (on the interval $[0, \pi]$). This becomes clearer by looking at the figures 6.3(a) - 6.3(c) portraying the empirical phase distribution for different frequency bins (solid lines) and their corresponding Gaussian approximations (dashed lines). While a Gaussian assumption with mean $\pi/2$ is not exactly true for the lowest



(a) frequency bin #1 (b) frequency bin #15 (c) frequency bin #30

Figure 6.3: Comparison of the two approaches

and highest frequency bins it seems to be quite a reasonable fit nevertheless — provided that the Gaussian distribution is limited to $[0, \pi]$. The stochastic dependency of the relative phase for the different frequency bins can be determined experimentally by again using the simulation approach. The result is presented in figure 6.4, showing that the relative phase of Mel power spectra is approximately stochastically independent for different frequency bins.



(a) side view

(b) top down view

Figure 6.4: Correlation of the Phase among the Frequency Bins

6.3 An Efficient Approximation

Using the established assumption of stochastic independence among the $\theta_{t,i}$, $i = 1, \ldots, d$, it is possible to calculate (6.4) much more efficiently by splitting the d-dimensional integral (6.3) into d one-dimensional ones. First of all, the Gaussians $\mathcal{N}(x_t; \mu_k, \Sigma_k)$ with diagonal covariance matrices[4] in the Gaussian mixture $p(x)$ (see equation(3.7)) can be factorized

$$p_x(x_t) = \sum_{k=1}^{K} c_k \, \mathcal{N}(x_t; \mu_k, \Sigma_k) = \sum_{k=1}^{K} c_k \prod_{i=1}^{d} \underbrace{\mathcal{N}(x_{t,i}; \mu_{k,i}, \sigma_{k,i})}_{=: p_{x_i}(x_{t,i}|k)}$$

Using this decomposition of p_x , $p(y_t|n_t, \theta_t)$ (see equation 6.2) can be written

$$\begin{aligned} p(y_t|n_t, \theta_t) &= p_x(f_{n_t, \theta_t}(y_t)) \cdot \prod_{i=1}^{d} \left| \frac{df_{n_{t,i}, \theta_{t,i}}(y_t)}{dy_{t,i}} \right| \\ &= \sum_{k=1}^{K} c_k \prod_{i=1}^{d} p_{x_i}(f_{n_{t,i}, \theta_{t,i}}(y_t)|k) \left| \frac{df_{n_{t,i}, \theta_{t,i}}(y_t)}{dy_{t,i}} \right| \end{aligned} \tag{6.5}$$

Replacing $p(y_t|n_t, \theta_t)$ in the Monte Carlo approximation (6.4) and changing the order of the sums, we obtain

$$\begin{aligned} p(y_t|n_t) &\approx \frac{1}{M} \sum_{m=1}^{M} p(y_t|n_t, \theta_t^{(m)}) \\ &= \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} c_k \prod_{i=1}^{d} p_{x_i}(f_{n_{t,i}, \theta_{t,i}^{(m)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i}, \theta_{t,i}^{(m)}}(y_t)}{dy_{t,i}} \right| \\ &= \frac{1}{M} \sum_{k=1}^{K} c_k \sum_{m=1}^{M} \prod_{i=1}^{d} p_{x_i}(f_{n_{t,i}, \theta_{t,i}^{(m)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i}, \theta_{t,i}^{(m)}}(y_t)}{dy_{t,i}} \right| \end{aligned} \tag{6.6}$$

If the $\theta_{t,i}$, $i = 1, \ldots, d$ are stochastically independent as assumed, it is possible to generate — let's say M' — samples $\theta_{t,i}^{(1)}, \ldots, \theta_{t,i}^{(M')}$ from $p(\theta_i)$, $i = 1, \ldots, M'$. Then each $\theta_t^{(m)} := \left(\theta_{t,1}^{(m_1)}, \ldots, \theta_{t,d}^{(m_d)}\right)$ is a valid sample for $m_1, \ldots, m_d \in \{1, \ldots, M'\}$ with

[4] This means different dimensions $x_{t,i}$, $x_{t,j}$ of clean speech spectra x_t are considered to be stochastically independent.

$$\begin{array}{ccccc} \theta_{t,1}^{(m_1)} & \in & \theta_{t,1}^{(1)}, \dots \theta_{t,1}^{(M')} & \sim & p(\theta_1) \\ \vdots & & \vdots & & \vdots \\ \theta_{t,d}^{(m_d)} & \in & \theta_{t,d}^{(1)}, \dots \theta_{t,d}^{(M')} & \sim & p(\theta_d) \end{array}$$

The computational complexity for the evaluation of

$$\frac{1}{M} \sum_{m=1}^{M} \prod_{i=1}^{d} p_{x_i}(f_{n_{t,i},\theta_{t,i}^{(m)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i},\theta_{t,i}^{(m)}}(y_t)}{dy_{t,i}} \right|$$

in (6.6) using all those samples is $\mathcal{O}((M')^d \cdot d)$, since there are $M = (M')^d$ possible combinations for $\theta_t^{(m)}$. The law of distributivity, however, allows us to calculate the same expression in $\mathcal{O}(M' \cdot d)$ by evaluating

$$\prod_{i=1}^{d} \frac{1}{M'} \sum_{m_i=1}^{M'} p_{x_i}(f_{n_{t,i},\theta_{t,i}^{(m_i)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i},\theta_{t,i}^{(m_i)}}(y_t)}{dy_{t,i}} \right|$$

instead, because this is equivalent to calculating

$$\left(\frac{1}{M'}\right)^d \sum_{m_1=1}^{M'} \cdots \sum_{m_d=1}^{M'} \prod_{i=1}^{d} p_{x_i}(f_{n_{t,i},\theta_{t,i}^{(m_i)}}(y_t)|k) \left| \frac{df_{n_{t,i},\theta_{t,i}^{(m_i)}}(y_t)}{dy_{t,i}} \right|$$

So, the weight $p(y_t|n_t^{(j)})$ of a noise hypothesis $n_t^{(j)}$ can be approximated by Monte Carlo integration with samples $\theta_{t,i}^{(1)}, \dots \theta_{t,i}^{(M')}$ from $p(\theta_i)$, $i = 1, \dots, d$:

$$p(y_t|n_t^{(j)}) \approx \sum_{k=1}^{K} c_k \underbrace{\prod_{i=1}^{d} \frac{1}{M'} \sum_{m_i=1}^{M'} p_{x_i}(f_{n_{t,i}^{(j)},\theta_{t,i}^{(m_i)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i}^{(j)},\theta_{t,i}^{(m_i)}}(y_t)}{dy_{t,i}} \right|}_{=p(y_t|n_t^{(j)},k)} \tag{6.7}$$

This can be regarded as independent approximation of the different dimensions of each Gaussian mode of integral (6.3) by Monte Carlo integration

$$\int_{-\pi}^{\pi} p(y_{t,i}|n_{t,i},\theta_{t,i},k) \cdot p(\theta_{t,i})\, d\theta_{t,i} \approx \sum_{m_i=1}^{M'} p_{x_i}(f_{n_{t,i}^{(j)},\theta_{t,i}^{(m_i)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i}^{(j)},\theta_{t,i}^{(m_i)}}(y_t)}{dy_{t,i}} \right| \tag{6.8}$$

While Haeb-Umbach and Schmalenstroeer [22] proposed to infer the noise independently for each dimension (for all noise hypotheses), we propose to do that only for the phase of each noise hypothesis. Furthermore we propose to use a deterministic version of the Monte Carlo method for the approximation (6.7) — motivated by the fact that the integrals (6.8) are one-dimensional and that Monte Carlo integration does not perform as well as deterministic here[5], especially if we try to keep the number M' of samples small. This can be performed by choosing the samples $\theta_t^{(m_i)}$ such that

$$\int_0^{\theta_{t,i}^{(m_i)}} p(\theta_{t,i}) d\theta_{t,i} = \frac{j + 1/2}{M + 1}$$

which makes sure that the samples all represent the same probability mass.

[5] as mentioned in section 6.3

6.4 Inferring the Phase

Additionally to incorporating the relative phase into the likelihood calculation, it is further possible to estimate the relative phase θ_t with the aim of introducing it to clean speech estimation. Given only y_t, n_t and $p(\theta_t|y_t, n_t)$, the optimal estimator for θ_t with respect to the MMSE criterion is the conditional mean

$$\mathrm{E}_{p(\theta_t|y_t,n_t)}[\theta_t] = \int_{-\pi}^{\pi} \theta_t \cdot p(\theta_t|y_t, n_t)\, d\theta_t \tag{6.9}$$

Assuming again that the noise spectrum n_t and the relative phase θ_t are stochastically independent, i.e. $p(\theta_t|n_t) = p(\theta_t)$, $p(\theta_t|y_t, n_t)$ can be expressed as

$$p(\theta_t|y_t, n_t) = \frac{p(y_t, \theta_t|n_t)}{p(y_t|n_t)} = \frac{p(y_t|\theta_t, n_t) \cdot p(\theta_t|n_t)}{p(y_t|n_t)} = \frac{p(y_t|\theta_t, n_t) \cdot p(\theta_t)}{p(y_t|n_t)}$$

Applying this equality to (6.9) yields

$$\begin{aligned} \mathrm{E}_{p(\theta_t|y_t,n_t)}[\theta_t] &= \int_{-\pi}^{\pi} \theta_t \cdot \frac{p(y_t|\theta_t, n_t) \cdot p(\theta_t)}{p(y_t|n_t)}\, d\theta_t \\ &= \frac{1}{p(y_t|n_t)} \cdot \int_{-\pi}^{\pi} \theta_t \cdot p(y_t|\theta_t, n_t) \cdot p(\theta_t)\, d\theta_t \end{aligned} \tag{6.10}$$

As mentioned in 6.2, the empirical relative phase distribution should not be used in this case, since θ_t is antisymmetric, $p(y_t|\theta_t, n_t)$ is symmetric (see its definition in equation (6.2)) and their product is hence antisymmetric. If $p(\theta_t)$ should be symmetric, equation 6.10 is zero, which doesn't help us either. Fortunately θ_t never occurs directly throughout the derivation of the relation between speech, noise, corrupted speech and the relative phase in section 6.1. It only occurs as an argument in $\cos(\theta_t)$. Therefore the idea is to estimate $\cos(\theta_t)$ instead:

$$\mathrm{E}_{p(\theta_t|y_t,n_t)}[\cos(\theta_t)] = \int_{-\pi}^{\pi} \cos(\theta_t) \cdot p(\theta_t|y_t, n_t)\, d\theta_t \tag{6.11}$$

Analogous to equation (6.10), this can be written

$$\mathrm{E}_{p(\theta_t|y_t,n_t)}[\theta_t] = \frac{1}{p(y_t|n_t)} \cdot \int_{-\pi}^{\pi} \cos(\theta_t) \cdot p(y_t|\theta_t, n_t) \cdot p(\theta_t)\, d\theta_t$$

Using the representation $\theta_t = \sum_{l=1}^{d} \theta_{t,l} \cdot e_l$, where e_l is the lth unit vector, this becomes

$$\mathrm{E}_{p(\theta_t|y_t,n_t)}[\theta_t] = \frac{1}{p(y_t|n_t)} \cdot \sum_{l=1}^{d} e_l \cdot \int_{-\pi}^{\pi} \cos(\theta_{t,l}) \cdot p(y_t|\theta_t, n_t) \cdot p(\theta_t)\, d\theta_t \tag{6.12}$$

Note that $p(y_t|\theta_t, n_t)$ can't simply be split into the components $p(y_{t,l}|\theta_{t,l}, n_{t,l})$. Replacing $p(y_t|\theta_t, n_t)$ by equation (6.5) and using the stochastic independence of the $\theta_{t,i}$, $i = 1, \ldots, d$, $p(\theta_t) = \prod_{i=1}^{d} p(\theta_{t,i})$, we obtain

$$\begin{aligned} &\sum_{l=1}^{d} e_l \cdot \int \cos(\theta_{t,l}) \cdot p(y_t|\theta_t, n_t) \cdot p(\theta_t)\, d\theta_t \\ &= \sum_{l=1}^{d} e_l \int \ldots \int \cos(\theta_{t,l}) \sum_{k=1}^{K} c_k \prod_{i=1}^{d} \underbrace{p_{x_i}(f_{n_{t,i},\theta_{t,i}}(y_{t,i})|k) \left| \frac{df_{n_{t,i},\theta_{t,i}}(y_{t,i})}{dy_t} \right|}_{=p(y_{t,i}|n_{t,i},\theta_{t,1},k)} p(\theta_{t,i})\, d\theta_{t,1} \ldots d\theta_{t,d} \end{aligned}$$

Changing the order of integration, exploiting the linearity of integration and using $\int \prod = \prod \int$[6] yields

$$\sum_{l=1}^{d} e_l \sum_{k=1}^{K} c_k \int \cos(\theta_{t,l}) \cdot p(y_{t,l}|n_{t,l}, \theta_{t,l}, k) \cdot p(\theta_{t,l}) \left(\prod_{\substack{i=1 \\ i \neq l}}^{d} \int p(y_{t,i}|n_{t,i}, \theta_{t,i}, k) \cdot p(\theta_{t,i})\, d\theta_{t,i} \right) d\theta_{t,l}$$

Since θ_t is stochastically independent of n_t and k, i.e. $p(\theta_{t,i}) = p(\theta_{t,i}|n_{t,i}, k)$

$$\begin{aligned} \int p(y_{t,i}|n_{t,i}, \theta_{t,1}, k) \cdot p(\theta_{t,i})\, d\theta_{t,i} &= \int p(y_{t,i}|n_{t,i}, \theta_{t,i}, k) \cdot p(\theta_{t,i}|n_{t,i}, k)\, d\theta_{t,i} \\ &= \underbrace{\int p(y_{t,i}, \theta_{t,i}|n_{t,i}, k)\, d\theta_{t,i}}_{=p(y_{t,i}|n_{t,i},k)} \end{aligned}$$

Applying this equality to the previous equation and pulling the product of the $p(y_{t,i}|n_{t,i}, k)$ out of the integral gives

$$\sum_{l=1}^{d} e_l \sum_{k=1}^{K} c_k \left(\prod_{\substack{i=1, \\ i \neq l}}^{d} p(y_{t,i}|n_{t,i}, k) \right) \underbrace{\int \theta_{t,l} \cdot p(y_{t,l}|n_{t,l}, \theta_{t,l}, k) \cdot p(\theta_{t,l})\, d\theta_{t,l}}_{=:\theta_{y_{t,i}, n_{t,i}, k}} \tag{6.13}$$

The idea now is to approximate the $\theta_{y_{t,i},n_{t,i},k}$ as well as the $p(y_{t,i}|n_{t,i}, k)$ with the Monte Carlo method using the samples $\theta_{t,l}^{(m_l)}$ from (6.7), so that all those integrals are approximated with the same samples:

$$\int \theta_{t,l} \cdot p(y_{t,l}|n_{t,l}, \theta_{t,l}, k) \cdot p(\theta_{t,l})\, d\theta_{t,l} \approx \sum_{m_l=1}^{M'} \theta_{t,l}^{(m_l)} \cdot p(y_{t,l}|n_{t,l}, \theta_{t,l}^{(m_l)}, k) \tag{6.14}$$

$$\int p(y_{t,i}|n_{t,i}, \theta_{t,1}, k) \cdot p(\theta_{t,i})\, d\theta_{t,i} \approx \sum_{m_i=1}^{M'} p(y_{t,i}|n_{t,i}, \theta_{t,i}^{(m_i)}, k) \tag{6.15}$$

The normalizing constant $p(y_t|n_t)$ in (6.10) is approximated according to equation (6.7). Since

$$p(y_t|n_t, k) = \prod_{i=1}^{d} p(y_{t,i}|n_{t,i}, k)$$

the product of the $p(y_{t,i}|n_{t,i}, k)$ in (6.13) can be expressed as

$$\prod_{\substack{i=1, \\ i \neq l}}^{d} p(y_{t,i}|n_{t,i}, k) = \frac{p(y_t|n_t, k)}{p(y_{t,l}|n_{t,l}, k)}$$

This procedure can be summarized by the following algorithm that infers $\hat{\alpha} := \cos(\hat{\theta}_t)$ and at the same time estimates the likelihood $\hat{p}(y_t|n_t)$. It has to be performed for each noise hypothesis $n_t^{(j)}$.

[6] $\int \int f(x)g(y)dydx = \int f(x) \int g(y)dydx = \int f(x)dx \cdot \int g(y)dy$

Algorithm 6.1 Phase Inference

1. for $i = 1, \ldots, d$ do

   for $k = 1, \ldots, K$ do

   - calculate $\hat{p}(y_{t,i}|n_{t,i}^{(j)}, k) = \sum_{m_i=1}^{M'} p_{x_i}(f_{n_{t,i}^{(j)}, \theta_{t,i}^{(m_i)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i}^{(j)}, \theta_{t,i}^{(m_i)}}(y_{t,i})}{dy_{t,i}} \right|$
   - calculate $\hat{\alpha}_{y_{t,i}, n_{t,i}^{(j)}, k} = \sum_{m_i=1}^{M'} \cos(\theta_{t,i}^{(m_i)}) \cdot p_{x_i}(f_{n_{t,i}^{(j)}, \theta_{t,i}^{(m_i)}}(y_{t,i})|k) \left| \frac{df_{n_{t,i}^{(j)}, \theta_{t,i}^{(m_i)}}(y_{t,i})}{dy_{t,i}} \right|$

2. for $k = 1, \ldots, K$ do calculate $\hat{p}(y_t|n_t^{(j)}, k) = \prod_{i=1}^{d} \hat{p}(y_{t,i}|n_{t,i}^{(j)}, k)$

3. calculate the normalizing constant $\hat{p}(y_t|n_t^{(j)}) = \sum_{k=1}^{K} c_k \hat{p}(y_t|n_t^{(j)}, k)$

4. for $i = 1, \ldots, d$ do calculate $\hat{\alpha}_{y_{t,i}, n_{t,i}^{(j)}} = \frac{1}{\hat{p}(y_t|n_t^{(j)})} \sum_{k=1}^{K} c_k \frac{\hat{p}(y_t|n_t^{(j)}, k)}{\hat{p}(y_{t,i}|n_{t,i}^{(j)}, k)} \hat{\alpha}_{y_{t,i}, n_{t,i}^{(j)}, k}$

Note, that this approach is not restricted to the bootstrap filter. It presents a general way to calculate the likelihood $p(y_t|n_t)$ dependent on the distribution of the relative phase. The same thing applies to phase inferrence.

6.5 Inferring Clean Speech - Take 2

The clean speech spectrum can principally be inferred like in section 3.4, i.e. by using the general approximation given in equation 3.12:

$$\mathrm{E}[x_t|y_t] \approx \sum_{j=1}^{N} h_t(n_t^{(j)}) \tilde{\omega}_t(n_t^{(j)})$$

The probability $p(x_t|y_{1:t}, n_t)$ in the term $h_t = \int x \cdot p(x_t) \cdot p(x_t|y_{1:t}, n_t)$, however, has to be replaced by its phase enriched version $p(x_t|y_{1:t}, n_t, \theta_t)$.

Approach 1p

Using the relationship $x_t = n_t + 2 \cdot \log\left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - 1} - cos(\theta_t)\right)$ established in (6.1) we obtain

$$h_t^{(1p)}(n_t) = n_t + 2 \cdot \log\left(\sqrt{e^{y_t - n_t} + cos(\theta_t)^2 - \underline{1}} - cos(\theta_t)\right) \tag{6.16}$$

Approach 2p

Using equation (6.1) and writing $\alpha_t = \cos(\theta_t)$, the effect of n_t to the kth Gaussian mean can be determined analogous to equation (3.15):

$$\begin{aligned} & e^{\mu'_k} = e^{\mu_k} + e^{n_t} + \alpha_t \sqrt{e^{\mu_k} \cdot e^{n_t}} \\ \Leftrightarrow \quad & e^{\mu'_k} = e^{\mu_k} \cdot \left(1 + e^{n_t - \mu_k} + 2\alpha_t \sqrt{e^{\mu_k} \cdot e^{n_t}} e^{-\mu_k}\right) \\ \Leftrightarrow \quad & \mu'_k = \mu_k + \underbrace{\log\left(1 + e^{n_t - \mu_k} + 2\alpha_t \sqrt{e^{n_t - \mu_k}}\right)}_{\Delta_{\mu_k, n_t, \alpha_t}} \end{aligned}$$

Using $\hat{\alpha}_{y_t,n_t}$ as an approximation to α_t we get

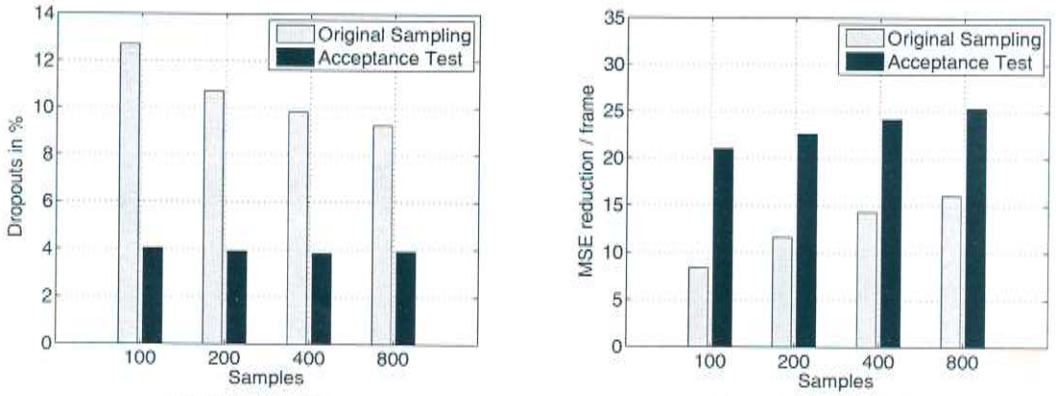
$$h_t^{(2)}(n_t) = y_t - \sum_{k=1}^{K} p(k|y_t, n_t)\Delta_{\mu_k,n_t,\hat{\alpha}_{y_t,n_t}} \tag{6.17}$$

analogous to equation (3.16).

7 Experiments

We chose to evaluate the performance of the speech feature enhancing Bootstrap filter and the proposed extensions under controlled conditions, i.e by artificially adding noise to approximately 45 minutes of seminar speech. This allows test with different *signal to noise ratios* (SNR)s and facilitates analysis of how good clean speech spectra estimation works. The latter is achieved by calculating the reduction of the *mean square error* (MSE) introduced by the noise. All the following experiments were performed using dynamic noise with a broad variety of sounds coming from a truck, slamming rubbish containers, distant voices, and shouts [46]. Warped MVDR spectral estimation was used instead of Mel FFT spectra, since we found it was found to perform better in earlier experiments [13].

First, we used a basic system to evaluate the performance of the the fast acceptance test proposed in section 4.1 - algorithm 4.1. This system consists of the bootstrap filter for speech feature enhancement as described in chapter 3 using a general speech model with 128 Gaussians plus the reinitialization procedure proposed in section 4.1. All of the following experiments were performed for a SNR of 0db. Figure 7.1(a) compares the dropout rates of the original sampling method and algorithm 4.1 — with a maximum number of 100 iterations — in dependency of the number N of samples used by the Bootstrap filter. Figure 7.1(b) shows the *mean square error reduction* (MSER) of the error introduced by the noise for the two approaches.



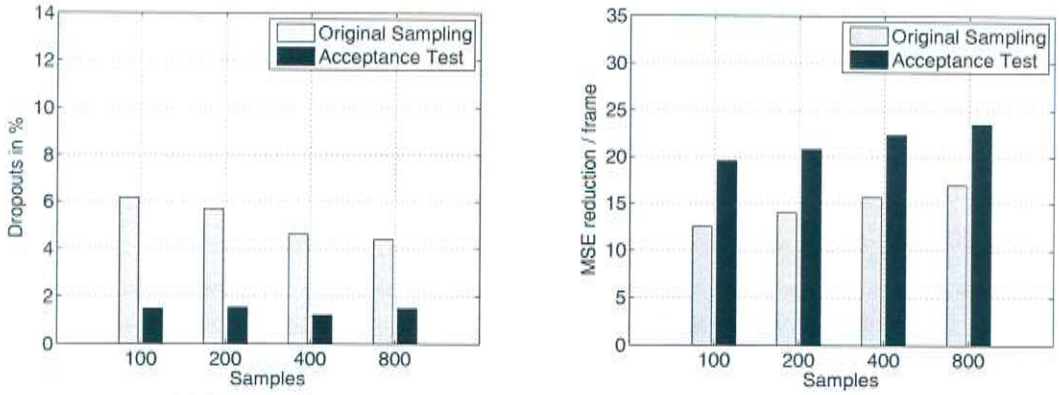
(a) Dropout Rate

(b) Reduction in MSER

Figure 7.1: Dropouts and MSE Reduction (GM128,LV)

The first thing that we see is that the dropout rate is significantly lower for the acceptance test. While it decreases with the number of particles for the original approach it seems to vary around 3.9% for the acceptance test. The MSE reduction increases with the number of samples and is significantly better with the fast acceptance test than with the original approach: 12% better (and more than twice as good) at 100 samples and still 8% better at 800 samples. Statically setting the noise variance (see section 3.1) to 20 instead of using the variance learned from the noise (VL) drastically reduced the dropout rate as can be seen in figure 7.2(a). Comparing figures 7.1(b) and 7.2(b) yields the interesting result that the MSE reduction with the acceptance test is stronger for the real noise variance than for the static one (figure??), though its dropout rate is almost twice as high. Without the acceptance test the MSER is conversely much lower. This however changes with the number of samples, reflecting the ability of the fast acceptance test to virtually increase

the number of samples (if necessary) without calculating the computationally intensive likelihoods or increasing the number of used samples.

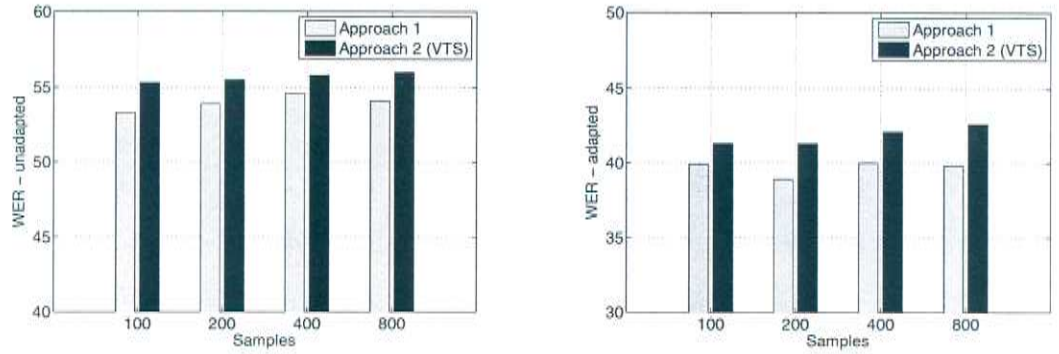


(a) Dropout Rate

(b) Reduction in MSER

Figure 7.2: Dropouts and MSE Reduction (GM128,SV)

Next, we will have a look at the differences between the two clean speech inference methods described in section 3.4: the simple, straight forward approach using $h^{(1)}$ and the VTS approach $h^{(2)}$. We compared the two approaches by using again the basic system with static noise variance and fast acceptance test (FAT100). The differences in MSE reduction were negligible (below 0.5%), but the *word error rates* (WER)s surprisingly differed noticeably. Figures 7.3(a) and 7.3(b) show the WERs for the unadapted and adapted[1] passes for static noise variance (SV). As it turns out, the simple approach $h^{(1)}$ is not only considerably faster than the VTS[2] approach but also improves the word error rate by almost 2% absolute for both passes, which means a gain of around 5% relative for the adapted pass.



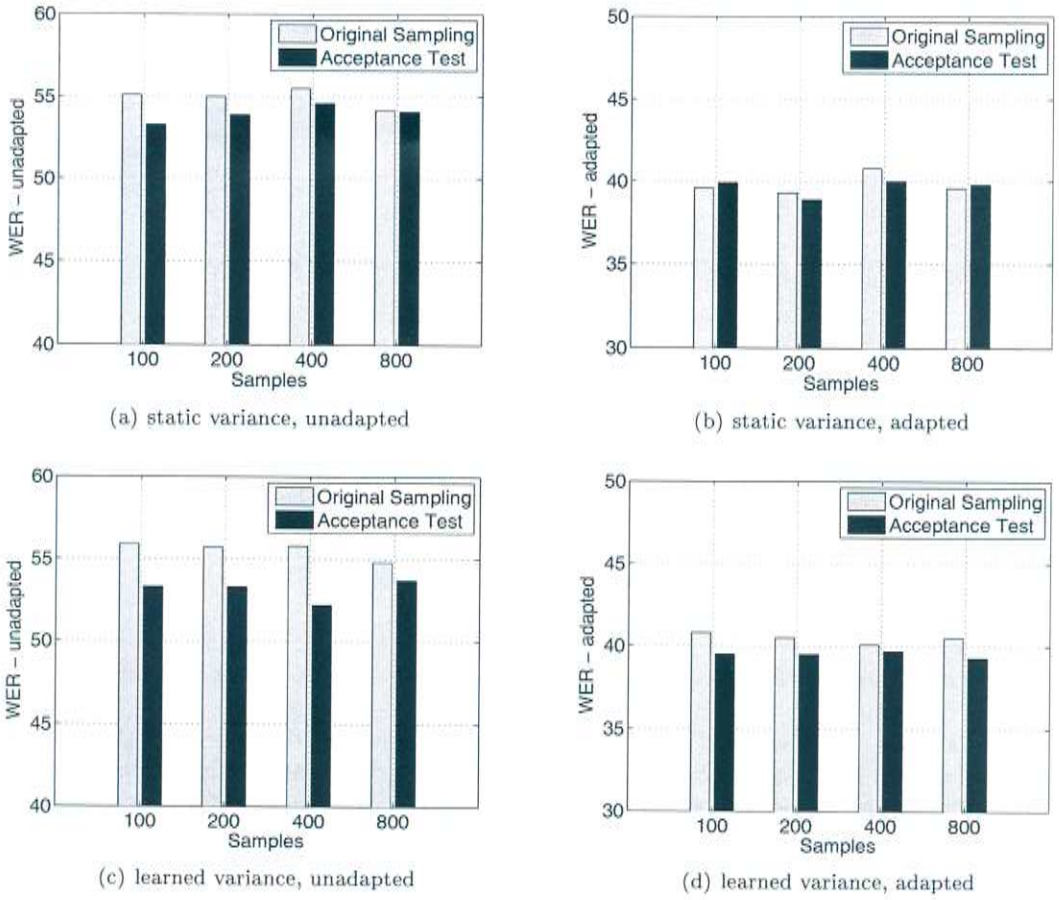
(a) Unadapted (SV)

(b) Adapted (SV)

Figure 7.3: Clean Speech Inference - $h^{(1)}$ vs $h^{(2)}$ (GM128, SV, FAT100)

Figure 7.4 shows the Word error rates of the different approaches using clean speech inference method $h^{(1)}$. While the WERs of the unadapted pass (figures 7.4(a) and 7.4(c)) seem to be better, if the acceptance test is used, this is much more unclear for the adapted pass (figures 7.4(b) and7.4(d)). Those results compare to a WER of 60.2% for the unadapted pass and 42.4 for the adapted pass without speech feature enhancement.

[1]The adapted pass uses *maximum likelihood linear regression* (MLLR) [32] and constrained MLLR (feature space



(a) static variance, unadapted

(b) static variance, adapted

(c) learned variance, unadapted

(d) learned variance, adapted

Figure 7.4: WERs for the Different Approaches (GM128, $h_{(1)}$)

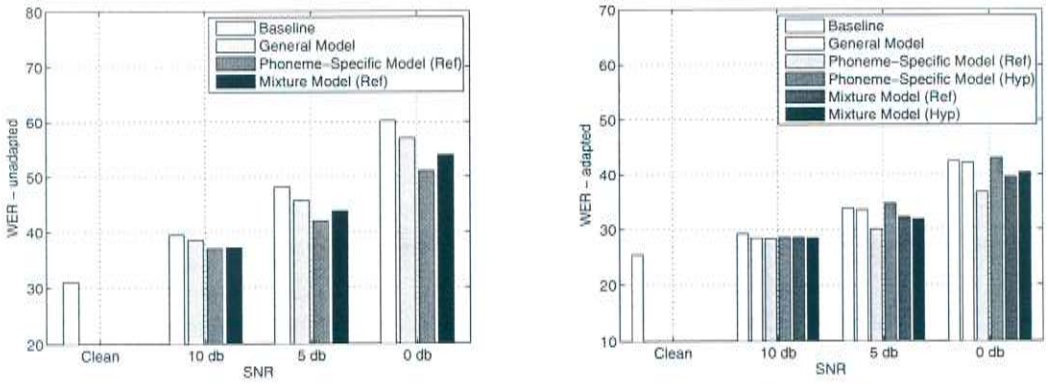
A phase inferring bootstrap filter with 100 samples, learned noise variance, without the acceptance test and using clean speech inference method $h^{(2)}$ could not improve upon those word error rates: the unadapted pass WER was 52.6% (compared to 52.4% for a system with 400 samples, learned noise variance, with acceptance test and using $h^{(1)}$), the adapted pass WER was 42.6% (compared to 39.4% for a system with 800 samples, learned noise variance, with acceptance test and using $h^{(1)}$). This comparison might be unfair since $h^{(2)}$ is known to perform worse than $h^{(1)}$, but this is not implemented yet for the phase inferring bootstrap filter as isn't the acceptance test. The MSE reduction was 35.14%, the dropout rate 0.32%, which is by far the best result in this respect. The best previous results where a dropout rate of 1.21 for a system with static noise variance, acceptance test, and 400 samples and a MSE reduction of 25.35 percent for a system with 800 samples, acceptance test and learned noise variance.

The last experiments involve the phoneme-specific model. They were all performed by using 1000 samples, no acceptance test, static noise variance and clean speech inference method $h^{(2)}$. Figure 7.5(a) shows the unadapted pass WERs of the baseline system (without filtering), a bootstrap filter with the general speech model (128 Gaussians), a bootstrap filter with the phoneme-specific speech model (16 Gaussians) and one with a mixture model (16 + 128 Gaussians) — both working on the the phoneme sequence reference (ref) obtained from alignment with clean speech. Figure 7.5(a) shows the adapted pass WERs for the same filters plus a phoneme-specific and a

adaptation) [15].

[2]even with its efficient implementation proposed in 4.2

mixture model filter working on the phoneme sequence hypothesis obtained from a first pass (see chapter 5). While the filter with the general speech model shows good improvements for the un-



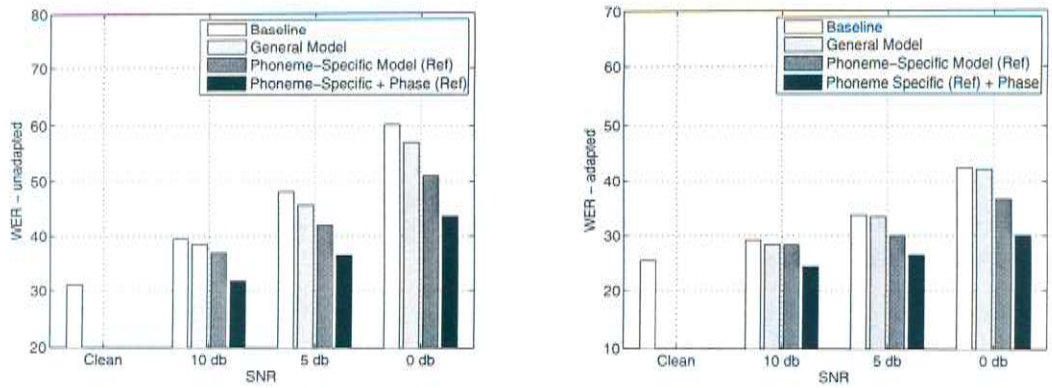
(a) unadapted

(b) adapted

Figure 7.5: Phoneme-Specific Models (128/16, SV, $h^{(2)}$)

adapted recognition pass, most of that gain levels off on the adapted pass. The phoneme-specific filter on the hypothesis of the first pass resulted in worse performance than the baseline for higher SNRs, which demonstrates the problem of 'model tying'. The mixture model filter, however, produced good results — not as good as the phoneme-specific filter on the reference (the theoretical optimum), but still better than the general filter. To make sure that the improvement in WER of the mixture model is not caused by the higher number of Gaussians (144 instead of 128), we compared it to a general model with 256 Gaussians. Its WER was 41.6% for the adapted pass at 0 db, while the mixture model had an adapted WER of 39.5. So it can be concluded that the number of Gaussians is not the crucial factor here.

Figure 7 shows the WER results of the phase-inferring filter in combination with the phoneme-specfic model on the reference. It performs much better than the phoneme-specific filter without phase-inference. Especially interesting is the result for 10dB on the adapted pass: the phase inferring filter on the phoneme-specific model is the only method showing a significant gain.



(a) unadapted

(b) adapted

Figure 7.6: Inferred Phase (128/16, SV, $h^{(2)}$)

8 Conclusions

The Bootstrap filter for noise Tracking as well as clean speech inference — both derived as *minimum mean square error* MMSE methods — do what they are supposed to do: they reduce the mean square error introduced by the noise. All proposed methods, the refinements as well as the phase approach improve upon the basic method in this respect. This does not completely translate to the *word error rate* (WER) of the speech recognizer. At times MSE reduction and WER even seemed to behave antithetic.

The problem with the MMSE approach is that it minimizes the average (global) error, but does not say too much about the local quality of the noise compensation. Hence, the MMSE approach is not guranteed to enhance the parts of speech that are relevant for speech recognition. This might — at least partially — be caused by working in the log Mel spectral power domain, too far away from the final features. Another problem might be that MMSE speech feature enhancement on a frame by frame basis can cause variations among successive frames, which might distort the speech recognizer. This could probably be improved by using a dynamical or switching linear dynamical model for clean speech as proposed in [28, 8].

The methods that significantly improve the word error rate are clean speech inference approach $h^{(1)}$ and the phoneme-specific (mixture) model. It should be examined whether those two methods can be combined to good effect. Furthermore it should be investigated how $h^{(1)}$ performs for lower SNRs and for different noises.

The fast acceptance test allows a reduction in the number of samples without increasing the number of dropouts. Furthermore it effectively facilitates using the learned noise variance, which in combination with the acceptance test outperformed the static noise variance (with FAT) in terms of MSE and WER while having a higher dropout rate. This motivates the development of a method that dynamically increases the variance if the number of accepted samples gets low.

The phase approach yields by far the best MSE reduction and the fewest dropouts. For the phoneme-specific model on the reference — though only hypothetical — this even translated to the WERs. It should be noted, however, that the implementaion of the phase incorporating boostrap filter is — deviating from its description in chapter 6 — still using an assumption of a uniformly distributed phase[1] and does not perform the weighting $\hat{p}(y_t|n_t^{(j)},k)/\hat{p}(y_{t,i}|n_{t,i}^{(j)},k)$ in algorithm 6.1 - step 4

$$\hat{\alpha}_{y_{t,i},n_{t,i}^{(j)}} = \frac{1}{\hat{p}(y_t|n_t^{(j)})}\sum_{k=1}^{K} c_k \frac{\hat{p}(y_t|n_t^{(j)},k)}{\hat{p}(y_{t,i}|n_{t,i}^{(j)},k)}\hat{\alpha}_{y_{t,i},n_{t,i}^{(j)},k}$$

The missing weighting causes that the phase for one dimension is inferred independent of the other dimensions. The results might get better, when those things are fixed.

[1] which is not true, but for the lowest Mel frequency bin

Bibliography

[1] 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, Issue 2:174–188, Feb. 2002.

[2] A. Averbuch, S. Itzikowitz, and T. Kapon. Radar target tracking - viterbi versus imm. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, Issue 3:550–563, May 1991.

[3] E.R. Beadle and P.M. Djuric. A fast-weighted bayesian bootstrap filter for nonlinear model state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 33, Issue 1:338–343, Jan. 1997.

[4] J. R. Bellegarda. Statistical techniques for robust asr: Review and perspectives. *Proceedings of EuroSpeech 1997*, Vol. 1:KN33–36, Sep 1997.

[5] M. Bolic, P. M. Djuric, and S. Hong. New resampling algorithms for particle filters. *Proc. of ICASP '03*, Vol. 2:589–592, Apr. 2003.

[6] S.F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *ASSP*, 27:113–120, Apr. 1979.

[7] 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. 1:1–38, 1977.

[8] J. Deng, M. Bouchard, and T. H. Yeap. Speech feature estimation under the presence of noise with a switching linear dynamic model. *Proceedings of ICASSP 2006*, pages 497–500, May. 2006.

[9] 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, No. 2:133–143, March 2004.

[10] A. Doucet. *On Sequential Simulation-Based Methods for Bayesian Filtering, Technical report CUED/F-INFENG/TR 310.* Cambridge University Department of Engineering, 1998.

[11] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. *IEEE Transactions on Accoustics, Speech and Signal Processing*, 32(6):1109–1121, Dec. 1984.

[12] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Accoustics, Speech and Signal Processing*, Vol. 33, No. 2:443–445, Apr. 1985.

[13] F. Faubel and M. Wölfel. Coupling particle filters with automatic speech recognition for speech feature enhancement. *to appear in Proc. of Interspeech*, Sep. 2006.

[14] M. Fujimoto and S. Nakamura. Particle filter based non-stationary noise tracking for robust speech feature enhancement. *Proc. of ICASSP*, 2005.

[15] M.J.F. Gales. Semi-tied covariance matrices. *Proc. of ICASSP*, 1998.

[16] M.J.F. Gales and S. Young. An improved approach to the hidden markov model decomposition of speech and noise. *Proceedings of ICASSP 1992*, pages 233–236, 1992.

[17] M.J.F. Gales and S.J. Young. Hmm recognition in noise using parallel model combination. *Proceedings of Eurospeech 1993*, Vol. 2:837–840, Apr 1993.

[18] S. Gannot, D. Burshtein, and E. Weinstein. Iterative and sequential kalman filter-based speech enhancement algorithms. *IEEE Transactions on Speech and Audio Processing*, Vol. 6, Issue 4:373–385, Aug. 1998.

[19] S. Gannot and M. Moonen. On the application of the unscented kalman filter to speech processing. *International Workshop on Acoustic Echo and Noise Control*, Sept. 2003.

[20] J.D. Gibson, B. Koo, and S.D. Gray. Filtering of colored noise for speech enhancement and coding. *IEEE Transactions on Signal Processing*, Vol. 39, Issue 8:1732–1742, Aug. 1991.

[21] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing*, 140:107–113, Sep. 1993.

[22] R. Haeb-Umbach and J Schmalenstroeer. A comparison of particle filtering variants for speech feature enhancement. *Proc. of Interspeech*, 2005.

[23] M. Isard and B. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29, 1:5–28, 1998.

[24] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[25] B. Kamgar-Parsi and M. Brosh. Distribution and moments of the weighted sum of uniform random variables, with applications in reducing monte carlo simulations. *The Journal of Statistical Computation and Simulation*, Vol. 52, No. 4:399–414, 1995.

[26] N. S. Kim. Imm-based estimation for slowly evolving environments. *IEEE Signal Processing Letters*, Vol. 5, No. 6:146–149, Jun. 1998.

[27] N. S. Kim. Nonstationary environment compensation based on sequential estimation. *IEEE Signal Processing Letters*, Vol. 5, No. 3:57–59, Mar. 1998.

[28] N.S. Kim, W. Lim, and R.M. Stern. Feature compensation based on switching linear dynamic model. *Signal Processing Letters, IEEE*, Vol. 12, Issue 6:473–476, Jun. 2005.

[29] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, Vol. 5, 1:1–25, Sep. 1996.

[30] A. Kong, J.S. Liu, and W.H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, Vol. 89, No. 425.:278–288, Mar. 1994.

[31] M. Lauer. *Entwicklung eines Monte-Carlo-Verfahrens zum selbstständigen Lernen von Gauss-Mischverteilungen.* Dissertation. Universität Osnabrück, Nov. 2004.

[32] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, pages 171–185, 1995.

[33] J.S. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, Sep. 1998.

[34] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Communication and Control Engineering. Springer, second edition, 1993.

[35] P.J. Moreno, B. Raj, and R.M. Stern. A vector taylor series approach for environment-independent speech recognition. *Proc. of ICASSP*, 1996.

[36] M.N. Murthi and B.D. Rao. All-pole modeling of speech based on the minimum variance distortionless response spectrum. *IEEE Transactions on Speech and Audio Processing*, 8(3):221–239, May 2000.

[37] K. K. Paliwal and A. Basu. A speech enhancement method based on kalman filtering. *IEEE Transactions on Accoustics, Speech and Signal Processing*, Vol. 12:177–180, Apr. 1987.

[38] B. Raj and R. Singh. Tracking noise via dynamical systems with a continuum of states. *Proc. of ICASSP*, 2003.

[39] B. Raj, R. Singh, and R. Stern. On tracking noise with linear dynamical system models. *Proc. of ICASSP*, 2004.

[40] C.P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, second edition, 2004.

[41] D.B. Rubin. The bayesian bootstrap. *The Annals of Statistics*, Vol. 9, No. 1:130–134, Jan. 1981.

[42] D.B. Rubin. A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when the fraction of missing information is modest: the sir algorithm. *Journal of the American Statistical Association*, Vol.82, No.398, Theory and Methods:543–546, June 1987.

[43] 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. *Proc. of Eurospeech '01*, Vol. 1:221–224, Sept. 2001.

[44] A.F.M. Smith and A.E. Gelfand. Bayesian statistics without tears: a sampling-resampling perspective. *The American Statistician*, Vol. 46, No. 2:84–88, May 1992.

[45] 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, Theory and Methods:528–540, June 1987.

[46] The Freesound Project. garbage.coll.serv.ds70p.mp3. *freesound.iua.upf.edu/samplesViewSingle.php?id=6986*.

[47] 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, Issue 1-2:133–140, Aug. 2000.

[48] S. Ulam. *Adventures of a Mathematician*. Charles Scribner's Sons, 1983.

[49] A.P. Varga and R.K. Moore. Hidden markov model decomposition of speech and noise. *Proceedings of ICASSP 1990*, Vol. 2:845–848, Apr 1990.

[50] M. Wölfel and J.W. McDonough. Minimum variance distortionless response spectral estimation, review and refinements. *IEEE Signal Processing Magazine*, 22(5):117–126, Sept. 2005.

[51] K. Yao, K.K. Paliwal, and S. Nakamura. Sequential noise compensation by a sequential kullback proximal algorithm. *Proceedings of Eurospeech*, pages 1139–1142, Sep. 2001.

[52] 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. *Proceedings of ICSLP*, Vol.1:770–773, Oct. 2000.