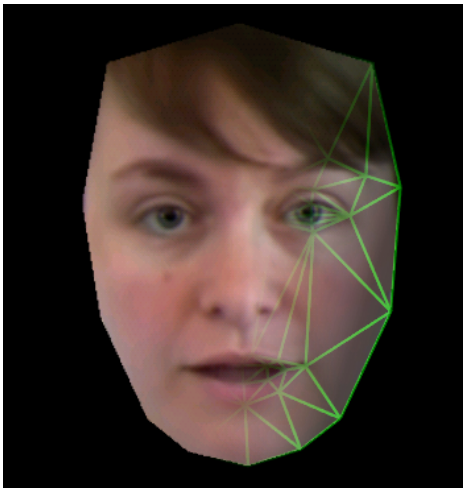
Making Faces – State-Space Models Applied to Multi-Modal Signal Processing

Tue Lehn-Schiøler

21st October 2005

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192, ISBN 87-643-0011-0



Abstract

The two main focus areas of this thesis are State-Space Models and multi modal signal processing. The general State-Space Model is investigated and an addition to the class of sequential sampling methods is proposed. This new algorithm is denoted as the Parzen Particle Filter. Furthermore, the Markov Chain Monte Carlo (MCMC) approach to filtering is examined and a scheme for MCMC to be used in on-line applications is proposed. In estimating parameters, it is shown that the EM-algorithm exhibits slow convergence especially in the low noise limit. It is demonstrated how a general gradient optimizer can be applied to speed up convergence.

The linear version of the State-Space Model, the Kalman Filter, is applied to multi modal signal processing. It is demonstrated how a State-Space Model can be used to map from speech to lip movements.

Besides the State-Space Model and the multi modal application an information theoretic vector quantizer is also proposed. Based on interactions between particles, it is shown how a quantizing scheme based on an analytic cost function can be derived.

Dansk Resumé

Hovedfokus for denne afhandling er klassen af tilstandsmodeller (State-Space Models). Den generelle version af modellen analyseres og inden for den specifikke gruppe af modeller, der estimerer tilstanden ved hjælp af sekvensiel sampling, foreslås en ny algoritme baseret på Parzens tæthedsestimator. Envidere vises det, hvordan metoder baseret på Markovkæde Monte-Carlo (MCMC) metoder kan anvendes til online estimation. Den informationsteoretiske tilgang til signalbehandling berøres igennem en algoritme til vektor kvantisering. Indenfor parameter estimation vises hvordan EM-algoritmen har sine begrænsninger og der foreslås en metode, der gør det muligt at anvende en standard optimeringstilgang istedet.

Endelig anvendes en stor del af afhandlingen på at gøre rede for hvordan tilstandsmodeller kan anvendes inden for multimodal signalbehandling. Det vises, hvordan den lineære version (Kalman Filteret) kan trænes til at oversætte mellem tale og mundbevægelser.

Preface

This thesis was prepared in the Intelligent Signal Processing (ISP) group at Informatics and Mathematical Modelling, the Technical University of Denmark in partial fulfillment of the official requirements for acquiring the PhD degree in engineering.

Applying for the PhD scholarship, I had to make a choice of what the topic of the studies was going to be. The choice fell on the binding problem, the task of combining the impressions received through different senses. The motivation for this was the fascinating ability of humans to do this. There are many approaches to this task, and at the beginning of the study I hoped that it would be possible to utilize knowledge from psychological studies to improve the artificial treatment of multi modal data. However, it quickly became clear that this was far from the general activities of the group, and focus was moved to applying more classical machine learning to mapping from speech to lip-movements.

Even this was in the outskirts of the group activity and finally – in keeping with the tradition in the ISP group – focus moved towards algorithmic, machine learning issues. On the algorithmic aspects, I have been able to cooperate with other PhDs, and it turned out that the problems found here are just as interesting. The change of focus has given me the insights that the essential thing is not the topic of the study but rather the process of studying. In the process, cooperation and interacting with other PhDs are extremely important.

Although the scientific work became centered on algorithms, it has been nice to have a real world problem to focus on. It is a great help when explaining to 'outsiders' why it is worth spending three years on research. For me, having an application – a goal – is of great importance, it makes it fun to do research and after all I am an engineer. That is why I am happy that I have managed to push the ISP group towards more fancy demos and hopefully also towards putting more emphasis on applications and cooperation.

Acknowledgements

On top of the official requirements, the ISP group has a number of unofficial requirements that must be fulfilled to get the degree. The past three years have been very eventful and, as almost all PhD students I am now married, have children, and own a house.

It has been a pleasure to work at the department, and I am very thankful to all

the nice and – in union – omniscient people here.

Besides the ISP group as a whole there are a number of people – both in the group and outside – that deserves mentioning individually.

The 'Junta' (Anders, Paul, Kristian and Alex) for comments on my demos and presentation skills. Lars for proof reading, ideas, and discussions. RE: (or what ever the name is these days) Martin, Lars, Anders and Paul for funny experiments. Mikkel for help with the AAM toolbox. Anders and Peter for help with sound features. Anders and Rasmus for help with state space calculations. Kåre, for help with matrix calculation, proof reading, and general discussions about everything. Niels for tex, php, and matlab help and for keeping me of the 'frikadelle' team almost to the end. Kåre, Niels, Lars, Jan, and Ole for the endless lunch speculation about house prices, bonds and real estate agents. Mogens for handyman tips, funny stories and excellent computer service. Ulla for keeping everything together, without her there would be no ISP group. The group at CNEL (Dr. Principe, Deniz, Ken, Anant and Robert) for taking me in and opening my eyes for information theory.

Finally, my family Christine, William, and Anna for support, patience, and inspiration.

_____ 21st October 2005

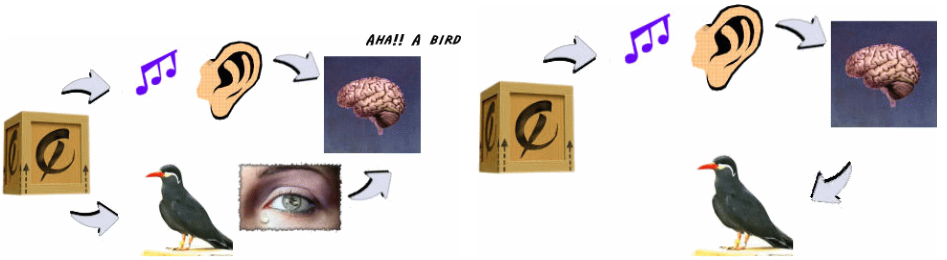
Tue Lehn-Schiøler

Contents

Chapter 1

Introduction

Animals and humans have the ability to combine impressions from different senses. This ability enables natural signal processing systems to extract information from and understand noisy and complex environments. If we see a man putting his hands together and shortly there after hear a sharp noise we are able to combine these two impressions to infer that what we witnessed was in fact a clap. In the same way, we are able to imagine what sound a dog would make, and what it would feel like to touch it just by looking at it. Figure 1.1 sketches two examples of audio and visual combination. In figure 1.1(a), the sound and the images are both observed and the joint impression leads to the conclusion that a bird is hidden in the box. In figure 1.1(b), the sound of the bird is heard and recognized, and from that, an image of the bird can be formed.
In fact, in most human sensing more than one modality like speech, facial expression, smell, gestures and haptics plays a role. In computer science, these modalities are traditionally modeled individually at a high levels of sophistication. Especially audio and visual signals have received much individual attention. The field of image processing has developed a range of techniques to describe pictures and the objects on them and lots of energy has been put into speech recognition. However, only in the last few years the combination of these fields have become the object of attention of a larger number of researchers.
With new MPEG standards for describing coding and labeling of both audio and visual signals it has suddenly become easier to combine the two modalities. Likewise, the fast propagation of web-cameras has made it possible to collect both audio and video signals in a simple manner. The MPEG standards for multimedia storage and transmission include not only the compressed signals



(a) There are two observations of what is in the box, an audio signal and a visual signal. Both signals may be unclear due to noise and distortions. However, by combining the two impressions, it is possible to figure out that the box contains a bird.

(b) From the bird's song an impression of the bird is made, and it is possible to imagine what the bird looks like.

Figure 1.1: Two examples of multi modal signal processing. The visual and the audio impressions can be combined or one can be derived from the other.

but also meta information, like what is happening in this scene, who is in it, what clothes are the actors wearing, and where to buy it. This will allow quick access to huge bodies of information while watching tv, but perhaps more importantly, it will make it possible to search in multimedia files. In the case of Hollywood movies with huge budgets it is no problem to hire a guy to manually type in all this information, but for live news casts, older content, and small budget productions, it is vital to have automatic extraction of the information. In this thesis, focus is put on combing audio and visual inputs, but the combination schemas can be used on any type of dual modality signals including brain scans like fMRI and EEG, map making from airplane and satellite images, military applications like tracking a tank using radar and infrared sensors and condition monitoring using acoustic emission, temperature, magnetic measurements, etc.
On an finer scale, research in audio and visual combination is driven by a range of different goals:

Lip-reading Improving speech recognition in noisy environments.

Lip synchronization Cloning and animation of faces, creating avatars, agents or virtual actors. Even changing the appearance of real actors to synchronize with another person speaking (perhaps in another language).

Human computer interface The computer should understand speech and gestures, and also communicate via speech and gestures.

Tracking of persons In a video conference, finding the person who is speaking and fixing the camera on him is important.

Classification Classification, of video sequences into e.g. news or sport, enables search.

Recognition Identification of the person speaking can be useful in security applications. In searches for a specific person or object in a database of video clips, recognition is necessary.

Even though the goals of the research are very different, the techniques that are used are similar. In this work, focus is on lip-synchronization, but the framework could just as easily have been applied to lip-reading, tracking or any other point of the list.

The motivations for generating facial expressions using a speech signal are at least threefold: Firstly, the language synchronization of movies often leaves the actor's mouth moving while there is silence or the other way around, this looks rather unnatural. If it was possible to manipulate the face of the actor to match the actual speech it would be much more pleasant to view synchronized movies (and cartoons).

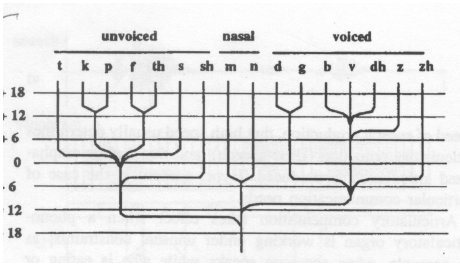
Secondly, even with increasing bandwidth, sending images via the cell phone is quite expensive, therefore a system that allows a single image to be sent in the beginning of the conversation and then models the face corresponding to the speech would be useful. Such a device would also help hearing impaired people lip-read over the phone without the person in the other end investing in expensive equipment[1].

Thirdly, when producing agents on the computer (like Mr. Clips) it would make communication more plausible if the agent could interact with lip movements corresponding to the (automatically generated) speech.

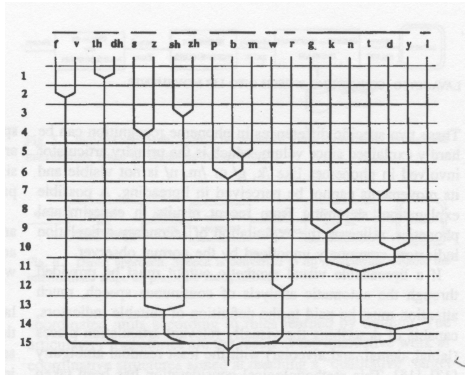
From a physiological view the combination of sensor information is also an interesting task. A wide range of experiments has been performed to reveal how humans perceive multi modal signals. In the audio visual community, the McGurk effect (McGurk and MacDonald, 1976) is perhaps the most important. It is an effect of mixing audio and visual signals. When a listener is presented with the sound of /ba/ and the visual impression (lip movements) of /ga/ most people will perceive /da/. That is /ba/+/ga/=/da/. The reason for this is that the sound /ba/ could not have been made with the lip movements for /ga/ and the other way around. The most likely sound given both audio and visual stimuli is /da/. Arnt Maasø has a video sequence illustrating the effect on his home page http://www.media.uio.no/personer/arntm/McGurk_english.html. The effect indicates that the visual system is important in speech recognition. An extended version of the McGurk effect is reported by Massaro et al. (1999) combining the audio "My gag kok me koo grive" with the visual "my bab pop me poo brive" the perceived sentence is "my dad taught me to drive".

In the above experiments, it is utilized that the confusion between phonemes is different in the audio and the visual representation. The mouth position

[1] The European project 'synface' http://www.speech.kth.se/synface/ is at the moment developing a commercial product for this.



(a) Auditory confusion in white noise with decreasing signal to noise ratio in DB.



(b) Visual confusion in white noise with decreasing signal to noise ratio.

Figure 1.2: Confusion between sounds in the audio and visual representation. From Dodd and Campbell (1987) and reproduced in Lavagetto (1995). The confusion is not the same in the two modalities, for example /f/ and /v/ are close in the visual domain but distinct in the audio domain.

of sounds like /b/,/p/ and /m/ or /k/,/n/ and /g/ cannot be distinguished even though the sounds can. Similarly, the sounds of /m/ and /n/ or /b/ and /v/ are very similar even though the mouth position is completely different. In figure 1.2, the audio and visual confusions are described. The difference in confusion can be utilized to improve speech recognition when using both audio and visual streams.

1.1 Speech to image mapping – Talking faces

The first attempts to control facial animation by speech was developed as early as the 1970s (Parke, 1975), however, not much attention was paid to field until the mid 1980s where development really began with the work of (Bergeron and Lachapelle, 1985; Hill et al., 1988; Lewis and Parke, 1987; Massaro and Cohen, 1990; Morishima et al., 1989).
In an early overview about state of the art lip-sync, Lewis (1991) concludes that using loudness to control the jaw is not a useful approach since sounds made with closed mouth can be just as loud as open mouth sounds. He also notes that the spectrum matching method has severe problems due to the formants independence of pitch. In this method, the shape of the mouth is determined from the frequency content of the speech. The problem is illustrated by the fact that the mouth shape is the same when a sound e.g. an 'a' is spoken with a high or a deep voice. Finally, he mentions that it is possible to automatically

generate speech from text and in this way gain control of which phoneme to visualize. In his view, the speech synthesis in 1991 was not of sufficient quality to sound natural, and although progress has been made in the field, automatically generated speech is still far from perfect[2]. The suggestion in Lewis (1991) is to extract phonemes using a Linear Prediction speech model and then map the phonemes to keyframes given by a lip reading chart.
The idea of extracting phonemes or similar high-level features from the speech signal before performing the mapping to the mouth position has been widely used in the lip-sync community. Goldenthal et al. (1997) suggested a system called "Face Me!". He extracts phonemes using Statistical Trajectory Modeling. Each phoneme is then associated with a mouth position (keyframe). In Mike Talk (Ezzat and Poggio, 1998), phonemes are generated from text and then mapped onto keyframes, however, in this system trajectories linking all possible keyframes are calculated in advance thus making the video more seamless. In "Video rewrite" (Bregler et al., 1997), phonemes are again extracted from the speech, in this case using Hidden Markov Models. Each triphone (three consecutive phonemes) has a mouth sequence associated with it. The sequences are selected from training data, if the triphone does not have a matching mouth sequence in the training data, the closest available sequence is selected. Once the sequence of mouth movements has been determined, the mouth is mapped back to a background face of the speaker. Other authors have proposed methods based on modeling of phonemes by correlational HMM's (Williams and Katsaggelos, 2002) or neural networks (Hong et al., 2002).
Methods where speech is mapped directly to facial movement are not quite as popular as phoneme based methods. However, in 'Picture my voice' (Massaro et al., 1999), a time dependent neural network, maps directly from 11×13 Mel Frequency Cepstral Coefficients (MFCC) as input to 37 facial control parameters. The training output is provided by a phoneme to animation mapping, but the trained network does not make use of the phoneme representation. Brand (1999) proposed a method based on (entropic) HMM's where speech is mapped directly to images. Methods that do not rely on phoneme extraction have the advantage that they can be trained to work in all languages, and that they are able to map non-speech sounds like yawning or laughing.
There are certain inherent difficulties in mapping from speech to mouth positions. The most profound is the confusion between visual and auditive information as described in the previous section. When mapping from speech to facial movements, one cannot hope to get a perfect result simply because it is very difficult to distinguish whether a "ba" or a "da" was spoken. Another difficulty is time-scales, sounds are typically recorded at 10-100 kHZ and images at 10-100 Hz, furthermore synchrony between the streams is not guarantied.
This short introduction captures a representative sample of the contributions to the speech to face problem, the list of authors is far longer. In appendix A a

[2] Just try to get Acrobat reader or Microsoft Sam to read a text out loud.

time line with an overview of the progress in the field is presented, this includes references to a broader range of authors. A historical view (dating back to Don Quixote) is provided by the excellent web page http://www.haskins.yale.edu/haskins/heads.html by Philip Rubin and Eric Vatikiotis-Bateson.

1.2 Why State-Space Models?

Studying the attempts that were made in the past, it became clear that most approaches use some kind of function approximation that does not take the temporal aspect into account. The approaches that do utilize the temporal structure of speech and images uses Hidden Markov Models (HMMs).
Popularized by Rabiner (1989), the HMM has been studied extensively by the speech recognition community and all kinds of extensions like coupled HMMs (Brand et al., 1997), asynchronous HMMs (Bengio and Bengio, 1996), and Factorial Hidden Markov Models (Ghahramani and Jordan, 1995) have been proposed. The nature of the HMM where the state vector contains a probability distribution over classes makes it suitable for a task like speech recognition. At each time step, the most probable phoneme can be found. However, the reason that this model has been studied only briefly in this work is that the discrete nature of the hidden space also makes the transitions in observation space jump. Given the temporal structure of the data and given that the movements of faces are smooth and continuous the idea arose to use continuous State-Space Models to create talking faces. The continuous State-Space Model shares graphical model with the HMM, and can in many regards be considered to fall into the same family (see e.g. Roweis and Ghahramani (1999)). For example the Viterbi algorithm (Viterbi, 1967) used to find the optimal state sequence in HMMs is similar to the Kalman smoothing algorithm.
Unlike the hidden Markov Model, the State Space Models work on continuous input and output spaces. It has many instances and even though the most straight forward one, the Kalman Filter (Kalman, 1960), has been studied in much detail in the past there are still many interesting aspects of State-Space Models to investigate.
The examination of State-Space Models for speech to image mapping led to many ideas and questions about the nature of State-Space Models and possible extensions. In this way, the studies was broadened to cover more theoretic algorithmic aspects rather than just application of the model to the specific problem.

1.2.1 Scientific contribution

The contributions of this work fall mainly into three categories: Information theory, State-Space Models and Talking faces. Information theory is used to derived a vector quantization scheme and it is also used in the derivation of a

novel approach for Particle Filtering. The general State-Space Model is examined in detail, besides Particle Filtering and the linear Gaussian case, Markov Chain Monte Carlo (MCMC) has been treated. The linear version of the State-Space Model is investigated particularly carefully and a new method for inferring parameter-values using gradients rather than expectation maximization has been derived.
The continuous State-Space Model is then used for multi-modal signal processing. Especially the problem of mapping from speech to lip-movements is considered.
Most of the work presented in this thesis has previously been published in the following papers:

Journal papers

Lehn-Schiøler, T., Hegde, A., Erdogmus, D., Principe, J. C. , *Vector-Quantization using Information Theoretic Concepts*, Natural Computing, vol. 4, pp. 39–51, 2005 (Lehn-Schiøler et al., 2005b)

Conference papers

Lehn-Schiøler, T., Hansen, L. K., Larsen, J. , *Mapping from Speech to Images Using Continuous State-Space Models*, Lecture Notes in Computer Science, vol. 3361, pp. 136 - 145, 2005 (Lehn-Schiøler et al., 2005a)

Lehn-Schiøler, T. , *Talking Faces a state-space approach*, Proceedings of DSAGM, pp. 103-111, 2004 (Lehn-Schiøler, 2004b)

Lehn-Schiøler, T. , *Multimedia Mapping using Continuous State-Space Models*, IEEE 6'th Workshop on Multimedia Signal Processing Proceedings, pp. 51–54, 2004 (Lehn-Schiøler, 2004a)

Lehn-Schiøler, T., Erdogmus, D., Principe, J. C. , *Parzen Particle Filters*, ICASSP, vol. 5, pp. 781-784, 2004 (Lehn-Schiøler et al., 2004)

Hegde, A., Erdogmus, D., Lehn-Schiøler, T., Rao, Y., Principe, J. , *Vector-Quantization by density matching in the minimum Kullback-Leibler divergence sense*, IEEE International Conference on Neural Networks - Conference Proceedings, vol. 1, pp. 105–109, 2004 (Hegde et al., 2004)

Publications in progress

Olsson, R. K., Petersen, K. B., Lehn-Schiøler, T. , *State-Space Models - from the EM algorithm to a gradient approach*, Submitted to Neural Computation, 2005 (Olsson et al., 2005)

Ferkinhoff-Borg, J., Lehn-Schiøler, T. , *The Failure of Particle Filters*, In progress, 2005 (Ferkinhoff-Borg et al., 2005)

1.3 The structure of the thesis

In chapter 2 the multi modal data sets are presented and relevant features are extracted. The chapter is concluded with the findings on Information Theoretic Vector Quantization originally presented in Hegde et al. (2004); Lehn-Schiøler et al. (2005b). Chapter 3 presents the State-Space Model, both the linear and the non-linear non-Gaussian cases are treated. The chapter is based on Lehn-Schiøler et al. (2004) and on sofar unpublished work (Ferkinhoff-Borg et al., 2005). Continuing the treatment of State-Space Models, chapter 4 describes parameter estimation with particular focus on the linear State-Space Model. Parts of the chapter build on work presented in Olsson et al. (2005). Finally, chapter 5 presents the results of using State-Space Models for generation of talking faces, the chapter relies on the results presented in Lehn-Schiøler (2004a,b); Lehn-Schiøler et al. (2005a).

Chapter 2

Data

In this chapter, the specific application mapping from speech to video is in focus. This is in contrast to the following two chapters that will deal with more general algorithmic questions.
The main focus of this chapter is on the data; it is obvious that a mapping from speech to images requires data in the form of video. It is also obvious that some kind of preprocessing is necessary. The preprocessing – or feature extraction – both in the image and the sound domain is treated in the following.
In the data fusion literature, three different methods are used; early fusion where data is treated directly without any preprocessing. Intermediate fusion where data is preprocessed, but the features are handled simultaneously. Finally, in late fusion, processing is done in each modality separately and the combination is done after the analysis. Even though the task at hand is mapping rather than fusion, the same categorizations can be used. As shall be clear from this chapter and chapter 5, the strategy chosen in this work mainly fits in the intermediate fusion framework.
To finish of this chapter, a vector quantizer based on information theory is presented. This algorithm was originally derived in Hegde et al. (2004); Lehn-Schiøler et al. (2005b) to aid in the feature extraction process.

2.1 Data acquisition

The data for automatic lip-sync are of course video clips containing images and sounds of speaking people. Unfortunately, unlike the speech recognition

community no standard data-sets are available for this task. The closest data-sets are the Vid-Timit database[1] (Sanderson, 2002a,b), the AV-Timit corpus [2], the AV-ViaVoice database (Potamianos et al., 2003) and the Johns Hopkins Lipreading Corpus (Bernstein and Eberhardt, 1986).
The VidTimit database was mainly created for multi-modal person authentication. It is comprised of video and corresponding audio recordings of 43 people (19 female and 24 male), reciting short sentences. The sentences – 10 per person – are from the test section of the TIMIT corpus. The first two sentences for all persons are the same, with the remaining eight generally different for each person. The sentences are all of length 3-5 seconds.
The AV-TIMIT corpus consist of continuous, phonetically balanced speech, spoken by multiple speakers, It uses 450 TIMIT-SX sentences. The corpus also contain time aligned phonetic transcriptions of the speech. The database contains 223 speakers, 117 of which are male and 106 are female.
The AV-ViaVoice database by IBM has 290-subject, uttering sentences from a large-vocabulary. Designed for speaker independent audio-visual speech recognition, the corpus consists of video and audio from a 290 subjects uttering a large set collection of sentences.
Unfortunately the AV-ViaVoice database is proprietary, the AV-TIMIT database was not public at the time when it was needed (and is still not, even though it might be possible to get access to it by contacting the authors), and also the John Hopkins Lipreading Corpus was un-available.
That leaves only the VidTimit database which has been used in most of this work, a single picture of one of the subjects is shown in figure 2.1.
As described above the VidTimit database contains many different speakers but only ten sentences from each speaker. In many cases ten samples are not enough to train and validate a model. To overcome this problem a new data set was gathered, the set contain two speakers uttering 20 sentences of length 5-15 seconds each. The sentences were chosen to contain all English phonemes chosen from the English Language Speech Database for Speaker Recognition (ELSDSR) database (Feng, 2004). In table 2.1 examples of sentences from both the VidTimit and the ELSDR databases are found. During the recordings the lighting was not set up properly causing shadow effects to appear in the movies and hence also in the final mappings. On the positive side the noise level was very low. Figure 2.2 show a single frame with each of the speakers[3]

2.2 Feature extraction sound

Processing of speech signals is perhaps one of the most active fields in signal processing and has been since the early 1970's. It began as early as 1936 in

[1] http://users.rsise.anu.edu.au/~conrad/vidtimit/
[2] http://www.csail.mit.edu/research/abstracts/abstracts04/html/191/191.html
[3] Which by the way are my wife and myself.

Table 2.1: Example sentences from the VidTimit and ELDSR database.

Database	Sentence
VidTimit	"Don't ask me to carry an oily rag like this." "She had your dark suit in greasy wash water all year."
ELDSR	"Chicken little was in the woods one day when an acorn fell on her head. It scared her so much she trembled all over. The poor girl shook so hard all her feathers fell out." "My friend Trisha suggests me to go to the woods to watch the poor bear being hunted for pleasure, and I said yes."

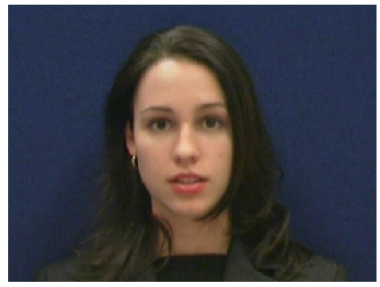


Figure 2.1: Sample image from the VidTimit database. The database contains ten sentences (5-8 seconds each) from a wide range of speakers.

(a) Tue

(b) Christine

Figure 2.2: Sample images from own recordings. Twenty sequences of each speaker each 8-15 seconds where recorded.

AT&T's Bell Labs.
The community has had plenty of standard data sets like the Timit database and many different approaches has been taken. Still, speaker independent large vocabulary speech recognition is difficult, and the best systems to date achieves word recognition rates of 60-90 % depending on the data-set. Since the mapping from speech to phonemes is at best 90% correct and since the goal is to animate the face rather than to understand speech, phoneme recognition was not employed in this project. However, the speech recognition society has also developed a host of preprocessing features to describe audio at shorter timescales. These include short term energy (STE), linear prediction coefficients (LPCs), Perceptual Linear Prediction (PLP), discrete Fourier transform (DFT), zero crossing rate (ZCR) and Mel Frequency Cepstral Coefficients (MFCCs).
In the talking face litterateurs many of these features has been used Brand (1999); Dupont and Luettin (2000) used PLP and a modification J-Rasta-PLP, McAllister et al. (1998) used DFT, Lewis (1991) used PLPs and Hong et al. (2002); Massaro et al. (1999) used MFCC's. In Morishima et al. (1989) a Vector Quantization (VQ) approach is used.
Regardless of the problems with recognition rate a number of researchers has chosen to base their talking head on phonemes. In this case phonemes must be extracted from speech (Ezzat and Poggio, 1998; Goldenthal et al., 1997) or speech must be artificially generated along with the lip movements (Pandzic et al., 1999).
Due to their close link to human perception and the wide use in speech recognition MFCC's were chosen as the sound feature. The choice of MFC coefficients as the sound feature is mainly based on the popularity of this feature in the literature. The sound is split into 40 ms blocks (to match the frame rate of the movie) and 13 MFCC features are extracted from each block. In figure 2.3 a speech signal and the first two MFCC features are shown for illustration.
In a perfect world a better set of sound features would be available. They should be able to capture the important aspects of sound, especially speech, and they should be generative. With such features the opposite mapping, i.e. from images to sound would also be possible.

2.3 Feature extraction images

In multi-modal signal processing a range of methods for processing the images has been used. When using the images to improve speech recognition or to infer speaker identity the goal is simply to extract information about the face. The facial model needed for this can be very simple and need not have the ability to reconstruct the face.
In most applications low level features are used for analysis of the face, this includes like Gabor Filters, edge detectors and color histograms. When more task specific knowledge is added, geometric properties like the distance between

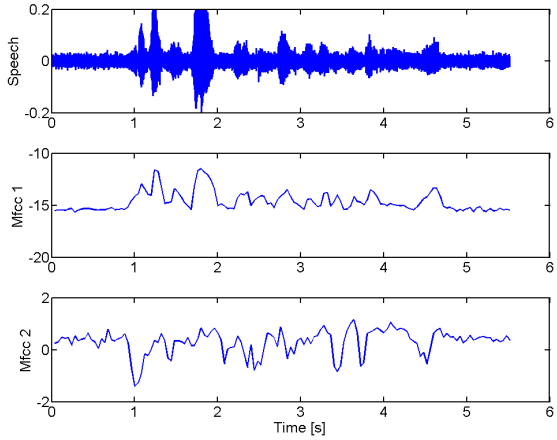


Figure 2.3: From the speech signal 13 MFCCs are extracted. The figure show an example of a speech signal and the first two MFCCs.

the eyes or the opening of the mouth are of interest. Examples of this can be found in multi-modal speech recognition (e.g. Bengio (2004)), in multi-modal person authentication (e.g. Sanderson and Paliwal (2002)) and in many other applications. However, when trying to animate a face it is necessary to make a mapping back into the image domain; hence, a generative model is needed.
The generative models needed for facial animation can be of two basic types: Computer graphics models as used in computer games and animated movies, and photo-realistic models.
In computer graphics the face is typically animated by manipulating control points in the model. Since the emergence of the MPEG-4 standard a set of control points called Facial Animation Points (FAPs) has been used in many animation schemas. In figure 2.4 the location of these points are shown. The long list of authors animating talking faces using the computer graphics approach include Aleksic and Katsaggelos (2003); Goto et al. (2001); Hong et al. (2002); Karlsson et al. (2003); Lewis (1991); McAllister et al. (1998); Pandzic et al. (1999).
The computer graphics models can be deformed to match the shape of a real person, and also texture can be applied to make the realism greater. However, at this stage in development the visual quality of the models does not match that of real video, at least not with standard hardware and reasonable computation times.
The photo realistic approach is to generate faces from video of a real person speaking. This can be done by selecting images from training data and rear-

ranging them to match the new speech as it is done in Brand (1999); Bregler et al. (1997). A video sequence is segmented into short clips each of which describes a specific triphone (sequence of three phonemes). Once new speech arrives the sequences are reordered by selecting the most similar sequence to the new phonemes. Similarity is measured by using a visual similarity lookup table. A similar reordering approach is proposed by Arslan and Talkin (1998) even though they use computer graphics models to do the animation.
Statistical models of faces can be created from a set of keyframes from video of real people. The keyframes are selected to describe the most important positions of the mouth, then a keyframe is assigned to each phoneme and some kind of morphing or optical flow is calculated to interpolate between the keyframes (Goldenthal et al., 1997; Tiddeman and Perrett, 2002). An extension of this is the Multidimensional Morphable Model (MMM). It was first introduced by Jones and Poggio (1998) and is based on morphing between keyframes. A set of important images are selected form the data set and one of them is selected to be the reference image. Then the optical flow vectors are calculated to morph each image to the reference. Shape parameters describing the warp from the reference to the current frame can then be extracted from real video. In Ezzat and Poggio (1998) it is used to create talking faces; each phoneme is described by a Mixture of Gaussians in the parameter space and the best possible path through parameter space can be calculated.
The Active Appearance Model (AAM) (Cootes et al., 1998) has been used to model the face in this work. It has previously been used to describe faces, both inter subject variations (Bettinger and Cootes, 2004; Stegmann, 2002) and deformations and expressions in single subjects (Gross et al., 2004; Lepsoy and Curinga, 1998; Matthews et al., 2002; Theobald et al., 2003). The AAM is closely related to the MMM in the way that it builds a statistical model of the face based on sample images. Like the computer vision modeling the AAM offer a parametric approach to generating faces. There is a bijective mapping between any point in a continuous parameter space and an image of a face. A closer description of the AAM is given in section 2.3.1. The AAM has recently been used to create talking faces by Theobald et al. (2004) and Cosker et al. (2004).

2.3.1 Active Appearance Models

As mentioned previously Active Appearance Models (AAMs) are used in a wide range of image processing tasks, among these the modeling of faces. For a detailed description of the AAM the original paper Cootes et al. (1998) gives a good insight, and a comprehensive description can be found in an unpublished report by the same author `http://www.isbe.man.ac.uk/~bim/Models/app_models.pdf`. Extensions and improvements to the model has been proposed amongst others by Matthews and Baker (2004).
The basic idea behind Active Appearance Models is to build a statistical model

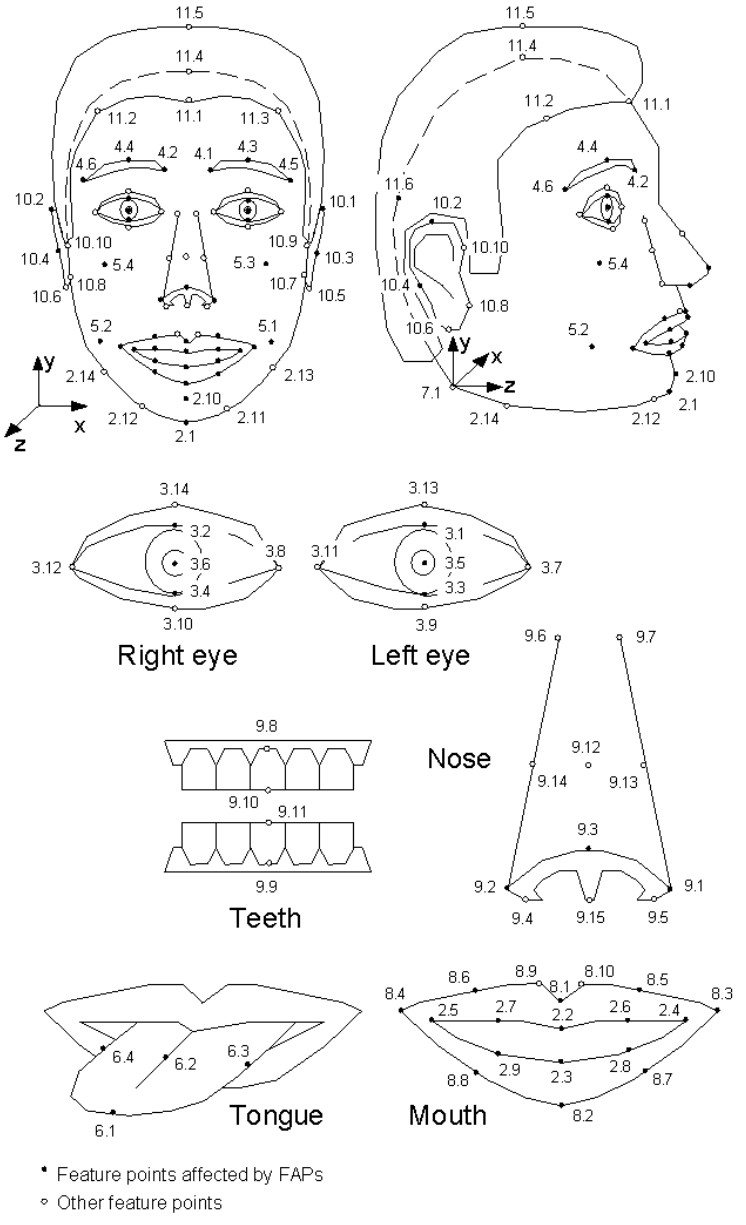


Figure 2.4: Illustration of the facial feature points (from www.research.att.com/projects/AnimatedHead). The points are chosen to represent the facial movements. In the modeling groups 2,3,4,8,10 and 11 representing the outline of the face, the eyes, and the mouth where used.

by supplying a training set of images. Each of the images must be annotated using the same set of landmarks, then the location (x-y coordinates) of landmarks in an image can be collected in a vector to describe the shape of the face, in figure 2.5 an annotated face can be seen. After gathering vectors from all training images the mean face can be found. Using triangulation on the original coordinates a linear mapping from each training image to the mean shape can be found, and for every image the texture can be mapped onto the mean shape. The data from each image now consist of two parts, the first is the shape containing the coordinates of the landmarks and the second a vector with the pixel values after mapping to the mean shape. Using a mean shape for the texture ensures that the texture vectors are of equal length. Assuming the features are Gaussian distributed a principal component analysis can be carried out on each of the two part of the data, it turns out that usually most of the variance in the data can be described by a small number of components. It is important to note that it is possible to map back from the coefficients to the image domain. In e.g. Theobald et al. (2003) AAM's are used to send images of faces at low bandwidth by extracting coefficients in the sending end, transmitting them to the receiver and then use the model and the coefficients to recreate the image. Extraction of features is done by minimizing the distance between the model output and the image by performing a nonlinear optimization in the model parameters.
In this work the AAM implementation by Mikkel B. Stegman `http://www.imm.dtu.dk/~aam` (Stegmann et al., 2003) is used.

Model building

To extract features a suitable subset of images in the training set is selected and annotated with points according to the MPEG-4 facial animation standard (figure 2.4). An example training set can be seen in figure 2.7. Using the annotations a 14-parameter model of the face is created; thus, with 14 parameters it is possible to create a photo realistic image. The image can be of any facial expression seen in the training set, or it can be a new unseen expression, as long as the image is a linear combination of the faces in the training set. In this example a face from the VidTimit database is used, but other models where build using the same procedure.
To validate the model a leave one out test is performed. The model was trained on all but one image and tested on the last, this procedure is repeated until all images had been left out. The result in all cases showed good correspondence between the model and the test image, indicating that the model is able to capture the important variations at least within the training set.
Once the AAM is created the model can be used to extract the lip movements in an image sequences. For each image in the sequence the 14 parameters are picked up. It should be noted that the features are not the same as the MPEG-4 control point. In figure 2.5 the result of the tracking is shown for a single

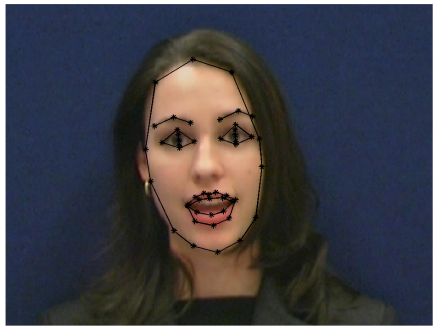


Figure 2.5: Image with automatically extracted feature points. The facial feature points used are selected from the MPEG-4 standard (figure 2.4). The points are found by adjusting the parameters in the AAM model to fit the image and then extract the shape component.

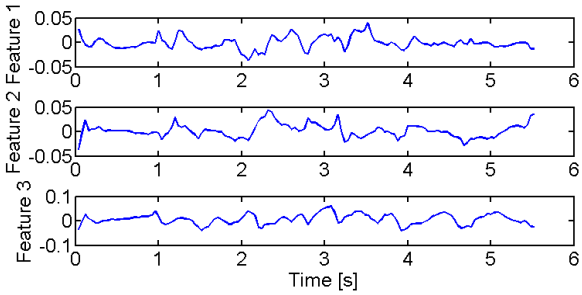


Figure 2.6: The temporal development of the first three image features while uttering the sentence: "She had your dark suit in greasy wash water all year".

representative image, the black dots represent the control points. Searching the entire image for the optimal parameter is an computational expensive and not were robust procedure. However, when the face has been identified correctly in the first image the continuity in the image sequence can be exploited. By tracking the evolution of the parameters a reasonable starting guess in the next frame can be obtained and the convergence speed of the nonlinear optimizer can be increased greatly. The evolution of the first three features is shown in figure 2.6.

2.3.2 Improvements

The way the AMM's are used in this work is to model the entire face with a single model. However, since the goal is to animate the face by using speech as input it may seem a bit optimistic to imagine that all facial movements are correlated with speech. Things like blinking are at best weakly correlated with the speech. An entity like the emotional state of the person might also be correlated with speech, but there is no evidence that emotions can be extracted by using MFCC's as is done in this work. So why try to model the entire face and not just the mouth? The answer is simply that a free floating mouth looks extremely silly!
A solution to this could be to use a hierarchical framework, generate a model of the mouth that is a sub-model of the entire face, the mouth model can then be controlled by the speech input (MFCC's) and the rest of the face can be kept fixed. To improve this further random blinking and movements of the head can be added. The ultimate goal would be to use emotion extraction techniques as described e.g. in Cowie et al. (2001) to animate the face while animating the mouth according the words.

2.3.3 Distribution of parameters

In the previous sections the choice of both sound and image features have been described, in the following chapters the choice of model will be discussed. Before choosing a model it is interesting to see how the features are distributed.
In figure 2.8(a) histograms of the features for all sentences by a single speaker are shown. Although especially the first five are either skewed or peaked, the shapes are not too far from Gaussian. The same trend is seen in the image features (figure 2.8(b)). A few features has a peak on one of the sides, this is an artifact of the binning procedure.

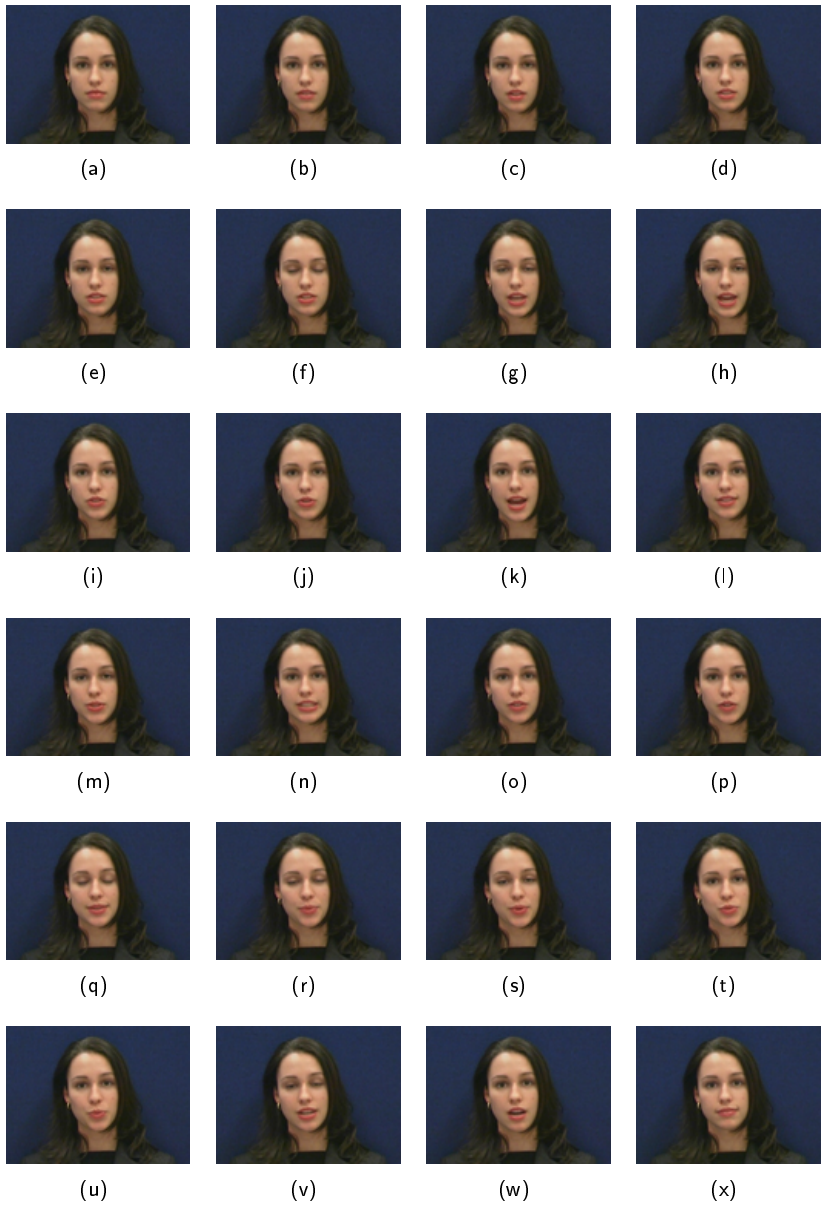
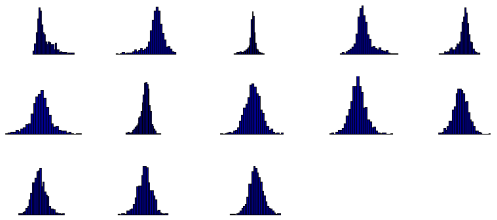
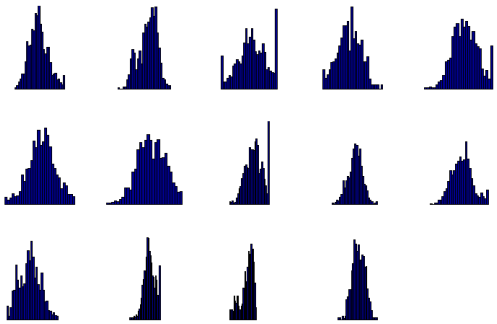


Figure 2.7: Training set for the AAM. The set contains 24 representative images. A leave one out test was performed, training the AAM on all but one images and testing it on the last. The result indicates that the training set has sufficient variation to capture the facial dynamics. This is further supported by the fact that the model was able to track all sequences accurate.



(a) Distributions of MFCC's from all sentences from a single speaker. The distributions are approximately Gaussian.



(b) Distributions of image features from all sequences from a single speaker. As for the sound features the distributions are approximately Gaussian.

Figure 2.8: When collecting sound and image features in histograms it can be seen that the distributions are approximately normal. However, this approach ignores the temporal aspect of the data and therefore the distributions cannot be used to model a single data point. Such a model should include the position of the previous point.

2.4 Vector quantization using information theoretic learning

Before proceeding to the chapters about mapping between the modalities a small de-tour is taken into the domain of vector quantization. Although not directly related to facial animation the reduction in data this method offers could be used as a preprocessing step. By discretizing data a discrete model like the Hidden Markov Model could be used for the mapping. Such an approach was adopted in Morishima et al. (1989). This work was first presented in Hegde et al. (2004); Lehn-Schiøler et al. (2005b).
The process of representing a large data set with a smaller number of vectors in the best possible way, also known as vector quantization, has been intensively studied in the recent years. Very efficient algorithms like the Kohonen Self Organizing Map (SOM) (Kohonen, 1982) and the Linde Buzo Gray (LBG) (Linde et al., 1980) algorithm have been devised. In the following a physical approach to the problem is taken, and it is shown that by considering the processing elements as points moving in a potential field an algorithm equally efficient as the before mentioned can be derived. Unlike SOM and LBG this algorithm has a clear physical interpretation and relies on minimization of a well defined cost-function. It is also shown how the potential field approach can be linked to information theory by use of the Parzen density estimator. In the light of information theory it becomes clear that minimizing the free energy of the system is in fact equivalent to minimizing a divergence measure between the distribution of the data and the distribution of the processing element, hence, the algorithm can be seen as a density matching method.

2.4.1 Background and idea

The idea of representing a large data set with a smaller set of processing elements (PE's) has been approached in a number of ways and for various reasons. Reducing the number of data points is vital for computation when working with a large amount of data whether the goal is to compress data for transmission or storage purposes, or to apply a computationally intensive algorithm.
In vector quantization, a set of data vectors is represented by a smaller set of code vectors, thus requiring only the code vector to be stored or transmitted. Data points are associated with the nearest code vector generating a lossy compression of the data set. The challenge is to find the set of code vectors (the code book) that describes data in the most efficient way. Vector quantization has application in both image processing and speech processing, in both domains it can reduce the size of the data set and it can convert continuous signals to discrete signals.
A wide range of vector quantization algorithms exist, the most extensively used are K-means (MacQueen, 1967) and LBG (Linde et al., 1980).

For other applications like visualization, a good code book is not enough. The 'code vectors', or processing elements (PE's), as they are often denoted in the self-organizing literature, must preserve some predefined relationship with their neighbors. This is achieved by incorporating competition and cooperation (soft-competition) between the PE's. Algorithms with this property create what is known as Topology Preserving Maps. The Self-Organized Map (SOM) (Kohonen, 1982) is the most famous of these. It updates not only the processing element closest to a particular data point, but also its neighbors in the topology. By doing this it aligns the PE's to the data and at the same time draws neighboring PE's together. The algorithm has the ability to 'unfold' a topology while approximating the density of the data.
It has been shown (Erwin et al., 1992) that when the SOM has converged, it is at the minimum of a cost function. This cost-function is highly discontinuous and drastically changes if any sample changes its best matching PE. As a result it is not possible to use the conventional methods to optimize and analyze it. Further more, the cost function is not defined for a continuous distribution of input points since boundaries exist where a sample could equally be assigned to two different PE's (Erwin et al., 1992). Attempts has been made to find a cost function that, when minimized, gives results similar to the original update rule (Heskes and Kappen, 1993).
Efforts have also been made to use information theoretic learning to find good vector quantifiers and algorithms for Topology Preserving Maps. Heskes (1999) introduces a cost function as a free energy functional consisting of two parts, the quantization error and the entropy of the distribution of the PE's. He also explored the links between SOM, vector quantization, Elastic nets (Durbin and Willshaw, 1987) and Mixture Modeling, concluding that the methods are closely linked via the free energy. Hulle (2002) uses an information theoretic approach to achieve self-organization. The learning rule adapts the mean and variance of Gaussian kernels to maximize differential entropy. This approach, however, leads to a trivial solution where PE's eventually coincide. To circumvent this, Hulle proposes to maximize the differential entropy and at the same time minimize the mutual information by introducing competition between the kernels. The competition is not based on information theory but rather implements an activity-based, winner-takes all heuristic. Bishop et al. (1996) proposes an algorithm (the Generative Topographic Map) in which a mapping between a lattice of PE's and data space is trained using the EM algorithm.
Ideas on interactions between energy particles have been explored previously by Scofield (1988). However, in this work the problem is approached with an information-theory perspective and the probability distributions of the particles are explicitly used to minimize the free energy of the system.
In the following sections, an algorithm for vector quantization using information theoretic learning (VQIT) is introduced. Unlike the methods described above, this algorithm is designed to take the distribution of the data explicitly into account. This is done by matching the distribution of the PE's with the distri-

bution of the data. This approach leads to the minimization of a well defined cost function. The central idea is to minimize the free energy of an information potential function. It is shown that minimizing free energy is equivalent to minimizing the divergence between a Parzen estimator of the PE's density distributions and a Parzen estimator of the data distribution.
At first, an energy interpretation of the problem is presented and it is shown how this has close links to information theory. Then, the learning algorithm is derived using the Cauchy-Schwartz inequality. After a few numerical results limitations and possible extensions to the algorithm are discussed.

2.4.2 Energy interpretation

The task is to choose locations for the PE's, so that they represent a larger set of data points as efficiently as possible. Consider two kind of particles; each kind has a potential field associated with it, but the polarity of the potentials are opposite. One set of particles (the data points) occupies fixed locations in space while the other set (the PE's) are free to move. The PE's will move according to the force exerted on them by data points and other PE's, eventually minimizing the free energy. The attracting force from data will ensure that the PE's are located near the data-points and repulsion between PE's will ensure diversity.
The potential field created by a single particle can be described by a kernel of the form $K(\cdot)$. Placing a kernel on each particle, the potential energy at a point in space $\boldsymbol{x}$ is given by

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\boldsymbol{x} - \boldsymbol{x}_i) \tag{2.1}$$

where the index i runs over the positions of all particles ($\boldsymbol{x}_i$) of a particular charge. If the kernel decays with distance ($K(x) \propto \frac{1}{(\boldsymbol{x}-\boldsymbol{x}_i)}$) the potential is equivalent to physical potentials like gravitation and electric fields. However, in the information theoretic approach, any symmetric kernel with maximum at the center can be chosen. For the sake of simplicity, Gaussian kernels are used herein.
Due to the two different particle types, the energy of the system has contributions from three terms:

1. Interactions between the data points; since the data points are fixed, these interactions will not influence minimization of the energy.

2. Interactions between the data and the processing elements; due to the opposite signs of the potentials, these particles will attract each other and hence maximize correlation between the distribution of data and the distribution of PE's.

3. Interactions between PE's; the same sign of all the PE's potentials causes them to repel each other.

In the information theoretic literature equation (2.1) is also considered a density estimator. In fact it is exactly the well known Parzen density estimator (Parzen, 1962). In order to match the PE's with the data, equation (2.1) can be used to estimate their densities and then minimize the divergence between the densities. The distribution of the data points (x_i) can be written as $f(x) = \sum_i G(x - x_i, \sigma_f)$ and the distribution over PE's (w_i) as $g(x) = \sum_i G(x - w_i, \sigma_g)$. Where $G(\mu, \sigma)$ denotes a normal distribution with mean μ and variance σ.
Numerous divergence measures exist, of which the Kullback-Leibler (K-L) divergence is the most commonly used (Kullback and Leibler, 1951). The Integrated square error and the Cauchy-Schwartz (C-S) inequality, are both linear approximations to the K-L divergence. If C-S is used, the link between divergence and energy interpretation becomes evident.
The Cauchy-Schwartz inequality

$$|ab| \leq ||a||||b|| \tag{2.2}$$

is an equality only when vectors a and b are collinear. Hence, maximizing $\frac{|ab|}{||a||||b||}$ is equivalent to minimizing the divergence between a and b. To remove the division, the logarithm can be maximized instead. This is valid since the logarithm is a monotonically increasing function. In order to minimize the divergence between the distributions $f(x)$ and $g(x)$ the following expression is minimized

$$\begin{aligned} D_{c-s}(f(x), g(x)) &= -\log \frac{(\int (f(x)g(x))\mathrm{d}x)^2}{\int f^2(x)\mathrm{d}x \int g^2(x)\mathrm{d}x} \\ &= \log \int f^2(x)\mathrm{d}x - 2\log \int f(x)g(x)\mathrm{d}x + \log \int g^2(x)\mathrm{d}x \end{aligned} \tag{2.3}$$

Following Principe et al. (2000) $V = \int g^2(x)\mathrm{d}x$ is denoted as the information potential of the PE's and $C = \int f(x)g(x)\mathrm{d}x$ the cross information potential between the distributions of data and the PE's. Note that

$$H(x) = -\log \int g^2(x)\mathrm{d}x = -logV \tag{2.4}$$

is exactly the Renyi quadratic entropy (Rényi, 1970) of the PE's. As a result, minimizing the divergence between f and g is equal to maximizing the sum of the entropy of the PE's and the cross information potential between the densities of the PE's and the data. The link between equation (2.3) and the energy formulation can be seen by comparing the terms with the items in the list above.

2.4.3 The algorithm

As described in the previous section, finding the minimum free energy location of the PE's in the potential field is equivalent to minimizing the divergence

between the Parzen estimate of the distribution of data points $f(x)$ and the estimator of the distribution of the PE's $g(x)$. The Parzen estimate for the data has a total of N kernels, where N is the number of data points, and the Parzen estimator for the PE's uses M kernels, M being the number of processing elements (typically $M << N$).
Any divergence measure can be chosen, but in the following the derivation will be carried out for the Cauchy-Schwartz divergence

$$J(w) = \log \int f^2(x)\mathrm{d}x - 2\log \int f(x)g(x)\mathrm{d}x + \log \int g^2(x)\mathrm{d}x \tag{2.5}$$

The cost function $J(w)$ is minimized with respect to the location of the PE's (w).
When the PE's are located such that the potential field is at a local minima, no effective force acts on them. Moving the PE's in the opposite direction of the gradient will bring them to such a potential minimum; this is also known as the gradient descent method. The derivative of equation (2.5) with respect to the location of the PE's must be calculated; but, since the data points are stationary only the last two terms of equation (2.5) (the cross information potential and the entropy of the PE's) have non-zero derivatives.
For simplicity, the derivation of the learning rule has been split in two parts; calculation of the contribution from cross information potential and calculation of the contribution from entropy. In the derivation Gaussian kernels are assumed, although, any symmetric kernel that obeys Mercer's condition (Mercer, 1909) can be used.
Consider the cross information potential term $(\log \int f(x)g(x)\mathrm{d}x)$; the Parzen estimator for $f(x)$ and $g(x)$ puts Gaussian kernels on each data point x_j and each PE w_i respectively, where the variances of the kernels are σ_f^2 and σ_g^2. Initially the location of the PE's are chosen randomly

$$\begin{aligned} C &= \int \hat{f}(x)\hat{g}(x)\mathrm{d}x \\ &= \frac{1}{MN}\int \sum_i^M G(x-w_i,\sigma_g^2)\sum_j^N G(x-x_j,\sigma_f^2)\mathrm{d}x \\ &= \frac{1}{MN}\sum_i^M\sum_j^N \int G(x-w_i,\sigma_g^2)G(x-x_j,\sigma_f^2)\mathrm{d}x \\ &= \frac{1}{MN}\sum_i^M\sum_j^N G(w_i-x_j,\sigma_a^2) \end{aligned}$$

where the covariance of the Gaussian after integration is $\sigma_a^2 = \sigma_f^2 + \sigma_g^2$. The gradient update for PE w_k from the cross information potential term then becomes

$$\frac{\mathrm{d}}{\mathrm{d}w_k} 2\log C = -2\frac{\Delta C}{C}$$

Where ΔC denotes the derivative of C with respect to w_k

$$\Delta C = -\frac{1}{MN}\sum_{j}^{N} G(w_k - x_j, \sigma_a)\sigma_a^{-1}(w_k - x_j)$$

Similarly for the entropy term$(-\log \int g^2(x)\mathrm{d}x)$

$$\begin{aligned} V = \int \hat{g}^2(x)\mathrm{d}x &= \frac{1}{M^2}\sum_{i}^{M}\sum_{j}^{M} G(w_i - w_j, \sqrt{2}\sigma_g) \\ \frac{\mathrm{d}}{\mathrm{d}w_k}\log V &= \frac{\Delta V}{V} \end{aligned}$$

With

$$\Delta V = -\frac{1}{M^2}\sum_{i}^{M} G(w_k - w_i, \sqrt{2}\sigma_g)\sigma_g^{-1}(w_k - w_i)$$

The update for point k consist of two terms; cross information potential and entropy of the PE's

$$w_k(n+1) = w_k(n) - \eta\left(\frac{\Delta V}{V} - 2\frac{\Delta C}{C}\right) \tag{2.6}$$

where η is the step size. The final algorithm for vector-quantization using information theoretic concepts (VQIT), consist of a loop over all w_k. Note that ΔC and ΔV are directional vectors where as C and V are scalar normalizations.
As with all gradient based methods this algorithm has problems with local minima. One of the ways local minima can be avoided is by annealing the kernel size (Erdogmus and Principe, 2002). The potential created by the particles will depend on the width of the kernels and the distance between the particles. When the distance is large compared to the width, the potential will be very 'bumpy' and have many local minima, and when the particles are close compared to the width, the corresponding potential will be 'smooth'. If, in addition, the number of particles is large the potential will have the shape of a normal distribution. Starting with a large kernel size will therefore help to avoid local minima. As with the SOM, a good starting point is to choose the kernels such that all particles interact with each other. The algorithm derived in this section uses the gradient descent method to minimize an energy function based on interactions between information particles. Each iteration of the algorithm requires $\mathcal{O}(M^2N)$ Gaussian evaluations due to the calculation of C for each PE. The parameters for the algorithm are the variances of the density estimators σ_f^2 and σ_g^2 along with the step size η. The variances can be set equal and can be annealed from a size where all particles interact. The step size should be chosen small enough to ensure smooth convergence.
An alternative approach is to use the gradient from equation (2.6) along with the value of the cost function equation (2.5) as input to a standard non-linear optimizer.

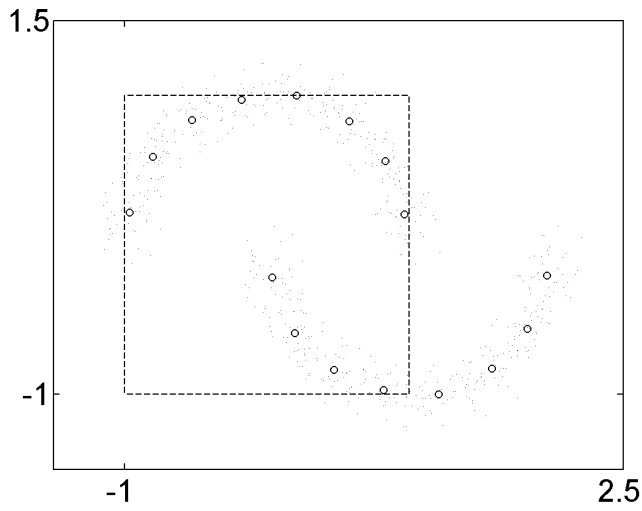
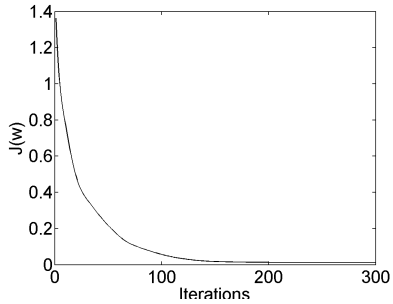


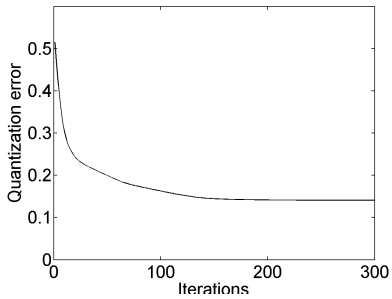
Figure 2.9: Artificial data used to evaluate performance of the VQIT algorithm. Points are chosen from two half circles distorted by Gaussian noise. Initially all processing elements (PE's) were chosen randomly from the unit square, in all simulations the algorithm converged to the same solution (indicated by circles).

2.4.4 Simulations

In this section the ability of the VQIT algorithm to perform vector quantization is illustrated on a synthetic data set consisting of two half circles with unit radius which has been distorted with Gaussian noise with variance 0.1. One of the halves is displaced in horizontal direction (figure 2.9).
The data essentially consist of two clusters, as shown in figure figure 2.9. Initially, 16 PE's are placed at random locations. The objective is to have the 16 PE's efficiently capture the structural property of the data. Using the algorithm presented above, the final locations of the PE's are shown, all in proper alignment with the data (figure 2.9).
To assess the convergence of the VQIT, 50 monte-carlo simulations were performed. Starting with different initial conditions chosen uniformly from the unit square, it was found that with the right choice of parameters the algorithm always converges to the same solution. During training mode, having an initial large kernel-size and progressively annealing it can avoid the local minima. In this simulation, the width of the kernels was adjusted to equal the data-variance on each of its individual projections. The initial kernel size for the PE's was set



(a) Development of the cost-function averaged over 50 trials. The cost-function is always non-negative but, it is not guaranteed that a cost of zero can be achieved



(b) The quantization error measure is included for comparison with other algorithms.

Figure 2.10: Convergence of the VQIT algorithm, cost-function and quantization error. The quantization error is calculated by computing the average distance between the data points and their corresponding winner Processing Element.

to be

$$\sigma_g = \sigma_n \begin{bmatrix} 0.75 & 0 \\ 0 & 0.5 \end{bmatrix}$$

where σ_n is the decaying variable. This is initially set to $\sigma_0 = 1$ and it decays after every iteration according to

$$\sigma_n = \frac{\sigma_0}{1 + (0.05\sigma_0 n)}$$

The kernel size for the data (σ_f) was set equal to σ_g. The kernel sizes were chosen such that initially all PE's interact with all data points.
The evolution of the cost-function is shown in figure 2.10(a). Note that the cost-function is always positive and that the minimum value it can obtain is zero; this optimum is achieved if the two distributions are identical. The quantization error (QE) is also calculated by computing the average distance between the data points and their corresponding winner PE. The QE convergence curve is shown in figure 2.10(b). To compare with other algorithms, the quantization error is used as a figure of merit since it is a commonly used evaluation metric. Comparison is provided with three algorithms: SOM, LBG and K-means. K-means is the only algorithm of these that does not converge to the same solution regardless of initial conditions. The result of the comparison can be seen in table 2.2. The quantization error for the VQIT, SOM, and LBG centers around 0.14 while the K-means does not perform as well. It should be noted that none of the algorithms directly minimizes QE, however, LBG includes it in the iterations.

	VQIT	SOM	LBG	K-means
QE	0.1408	0.1419	0.1393	0.1668

Table 2.2: Quantization error (QE) for the data set shown in figure 2.9, the results are the average of 50 trials with different initial conditions. The SOM, LBG and the VQIT algorithm always converges to the same solution. The four algorithms produce approximately the same results even though they do not minimize the same error-measure.

2.4.5 Issues regarding VQIT

In this section some of the critical issues regarding the algorithm are discussed. Emphasis is put on links to other algorithms and possible extensions.

The algorithm presented in this work is derived on the basis of the Cauchy-Schwartz inequality. This leads to a divergence measure based on the inner-product between two vectors in a Riemann space. As noted earlier this is not the only choice, and has in fact only been used here because of its close links to entropy. Another choice for cost-function is the Integrated Square Error which uses the quadratic distance between the distributions instead of an inner product

$$\int (f(x) - g(x))^2 \mathrm{d}x = \int f^2(x)\mathrm{d}x - 2\int f(x)g(x)\mathrm{d}x + \int g^2(x)\mathrm{d}x. \tag{2.7}$$

The terms correspond to the information potentials of the data and the PE's and the cross information potential between the two. Note that equation (2.7) is similar to equation (2.5) except for the logarithm. Derivations equivalent to those used for C-S yields the very simple update

$$w_k = w_k + \eta\,(\Delta V - \Delta C) \tag{2.8}$$

which requires $\mathcal{O}(MN)$ calculations per iteration. Annealing can also be used and the performance is similar to the VQIT.
"Density estimation is an ill posed problem and requires large amount of data to solve well" (Vapnik, 1995). Therefore, Vapnik suggests that one should not try to estimate densities in order to solve simpler problems (like vector quantization).
Even though this approach uses Parzen density estimates in its formulation, the pdf is never estimated. Instead the integral can be computed exactly through the double sum and therefore the method does not violate Vapnik's recommendations.

In a physical system, all potentials have the same form and only the magnitude (charge) can change, i.e. the same kernel type must be used for all particles.

Also, in the Parzen estimator the mixture is homoscedastic, i.e. all mixtures have the same variance. However, in many of the recent publications (Heskes, 1999; Hulle, 2002; Yin, 2001), a heteroscedastic approach is followed allowing the variance and weighting of the mixture components to change. It is easy to extend the algorithm presented in this work to include heteroscedastic mixtures. The cost-function can just as well be minimized with respect to both means, variances and mixture weights. One can then choose to use either gradient descent or the EM algorithm to find the minimum. However, introducing more free parameters also means estimating more parameters from the same data points and can therefore lead to over fitting and poor generalization performance.
Another important issue is topology preservation. This feature is important if the mapping is to be used for visualization. Unlike the SOM, the learning rule proposed in this work is not topology preserving; it does not define an ordering of the PE's. It is however important to notice that if an ordering exists, the algorithm will approximately keep this ordering during convergence. Two different alterations can ensure that neighbors in the topology are also neighbors in the mapping.
The first and simplest is to change the cost function equation (2.5). This can be done by adding a term to the update schema equation (2.8). The term should include attraction from PE's that are close on the grid, one possibility is

$$\sum_{i \in \mathcal{N}} (w_j - w_i) \tag{2.9}$$

Where $\mathcal{N}$ is the set of neighbors defined by the topology. Since the cost-function is changed, it cannot be expected that the PE's converge to the same positions. However, once the topology has unfolded, the weighting of the neighborhood term equation (2.9) can be reduced and a solution will be obtained with PE at the desired positions and this time with the desired topology. Another option more along the lines of the SOM and other algorithms (Hulle, 2002; Yin, 2001), is to change the update of the cross information potential term. If a winner PE is chosen for every data point and it only itself and its neighbors are updated, PE's close in the topology will be drawn together. Unfortunately this is not straight forward to put into the information theoretic framework.
The VQIT algorithm is based on block-computation of the data. It is possible to develop an online sample-by-sample update, which may result in a significant reduction in computational complexity. One way this can be achieved is by eliminating the second summation in equation (2.6) and computing the Kernel for only the current sample.

2.5 Final remarks

This chapter presented the data that is used through out the work. The available (and not so available) multi modal databases were described and the rationale

behind selecting the VidTimit database supplemented with own recordings was given.
A vector quantization method based on information theory was presented, the method is based on physical laws and introduces probability densities directly into the optimization. Vector quantization can be a useful way of extracting features from both sounds and images, and the initial idea behind researching this algorithm was that it might be used in this way. As it turned out, Mel Frequency Cepstral Coefficients and Active Appearance Model's were more appealing, and these representations will be used in the remainder of this thesis.

Chapter 3

State-Space Models

In this chapter, the State-Space Model is introduced and described in a general context. The state-space framework plays a major role in the approach to multimodal mapping presented in the previous chapters. However, to save a little excitement for the later chapters the explanation of exactly how the State-Space Model is used to map from speech to faces will not be revealed before chapter 5. The main purpose of this chapter is to give an introduction to the general framework of discrete State-Space Models and to present results obtained in the area of State-Space Models. These results concern the Parzen Particle Filter (Lehn-Schiøler et al., 2004) and so far unpublished work on Markov Chain Monte Carlo Filtering.

3.1 State-Space Modelling, a probabilistic approach

The model

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_k \tag{3.1a}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{3.1b}$$

describes a scenario where an unobserved (hidden) state $\boldsymbol{x}$ is progressing in time via a function $\boldsymbol{f}$ and driven by additive noise $\boldsymbol{v}_k$. At each time step an observation $\boldsymbol{z}$ is made, the observation is a function of the hidden state with added noise $\boldsymbol{w}_k$. An example of such a process is tracking of a vehicle. The

state of the vehicle at time k is complectly determined by the vector $\boldsymbol{x}_k$. The vector $\boldsymbol{x}_k$ could eg. include the position and velocity of the vehicle. The function $\boldsymbol{f}(\boldsymbol{x})$ determines how the vehicle moves. At each time step the direction to the vehicle $\boldsymbol{z}_k$ is observed by the person tracking it, the relationship between the state and the observation is given by $\boldsymbol{h}(\boldsymbol{x})$. To make the best possible guess about the location of the vehicle both the process (expected movement) and the observation must be taken into account.
In the above example the observation was of a lower dimension than the state, but the opposite might as well be the case. This could happen if the observation was a digital photograph of the scene. In that case each pixel represents a part of the observation making $\boldsymbol{h}$ a mapping from a low dimension space to a high dimensional space.
The function $\boldsymbol{f}$ is a mapping of the hidden state $\boldsymbol{x}$ from one time step $(k-1)$ to the next (k), $\boldsymbol{h}$ is function mapping the hidden state to the observation $\boldsymbol{z}$. v and w are noise contributions drawn independently at each time step. The functions are in general nonlinear and the noise can be distributed according to any distribution. In the case where $\boldsymbol{f}$ and $\boldsymbol{h}$ are linear functions and both noise distributions are Gaussian, the model reduces to the famous Kalman Filter (Kalman, 1960).
Finding the probability of the hidden state at time k depends on how many observations are available. If observations from the 'future' (w.r.t. k) are available $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k+n})$ the process is called smoothing. If observations are available up to the given time $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ the process is called filtering. And, if there are no observations available at the present time $p(\boldsymbol{x}_k|\boldsymbol{z}_{k-n})$ it is called prediction. In this work focus is on filtering and smoothing but, the extension to prediction is straight forward.
Filtering, smoothing, and prediction are all methods to find optimal hidden states $\boldsymbol{x}_k$, but often the model also has a set of parameters that can be optimized. This is e.g. the case when the transition and observation equations are linear and the noise is Gaussian (Kalman Filtering). Here the function $\boldsymbol{f}(\boldsymbol{x}, \theta_f)$ is linear and can be written as the matrix equation: $\boldsymbol{F}\boldsymbol{x}$ with $(\theta_f = \{\boldsymbol{f}_{11}, \boldsymbol{f}_{12} \ldots\})$. Parameter estimation will be dealt with in chapter 4.
There are different approaches to finding the optimal values of the hidden state. The Markov property of the model makes it possible to handle the problem sequentially, that is, find the optimal hidden state at time $k = 1$ and then use that to find the optimal at time $k = 2$ and so on. Using this technique the filtering estimate can be calculated. Once the filtering estimate is found, the chain can be traversed backwards in a similar manner and the smoothing estimate can be found. The Kalman Filter (KF) and various Particle Filter (PF) approaches all works this way. There are however other possibilities than the sequential. If a Markov Chain Monte Carlo sampling schema is applied, the smoothing density of the entire chain can be calculated simultaneous, this approach is investigated in section 3.4. In the following the sequential method is explored.

3.1.1 Filtering

First note that the equations in (3.1) can be written as the transition probability and the likelihood of the observation given the state

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = p_v\left(\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1})\right) \tag{3.2a}$$
$$p(\boldsymbol{z}_k|\boldsymbol{x}_k) = p_w\left(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{x}_k)\right) \tag{3.2b}$$

For sequential filtering the update from the distribution of the hidden state at time $k-1$ given by $p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})$ to the distribution at time k given by $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ needs to be calculated. This can be done by rewriting $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ to

$$\begin{aligned} p(\boldsymbol{x}_k|z_{1:k}) &= \frac{p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})p(\boldsymbol{z}_{1:k})}{p(\boldsymbol{z}_{1:k})} = \frac{p(\boldsymbol{x}_k, \boldsymbol{z}_{1:k-1}, \boldsymbol{z}_k)}{p(\boldsymbol{z}_{1:k-1}, \boldsymbol{z}_k)} \\ &= \frac{p(\boldsymbol{z}_k, \boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})p(\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})p(\boldsymbol{z}_{1:k-1})} \\ &= \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} \end{aligned}$$

Where the fact that $\boldsymbol{z}_k$ does not depend on $\boldsymbol{z}_{k-1}$ when $\boldsymbol{x}_k$ is known is used in the last equality and use that

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})\mathrm{d}\boldsymbol{x}_{k-1}$$

The updating formula then becomes

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)\int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})\mathrm{d}\boldsymbol{x}_{k-1}}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} \tag{3.3}$$

Unfortunately the update involves an integral that cannot be solved analytically in the general case. In section 3.2 methods to approximate the integral is described but first the linear Gaussian case is dealt with.

3.1.2 The linear Gaussian version (Kalman Filter)

When the relationship between the states $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ is linear, the function $\boldsymbol{f}(\boldsymbol{x}_{k-1})$ can be written as a multiplication between the matrix $\boldsymbol{F}$ and the state, similarly for the observation function $\boldsymbol{h}(\boldsymbol{x}_k)$. When $\boldsymbol{f}$ and $\boldsymbol{h}$ are linear ($\boldsymbol{F}$,$\boldsymbol{H}$) and the noise is Gaussian, the equations become

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1}, \boldsymbol{v} \sim N(\boldsymbol{0}, \boldsymbol{Q}) \tag{3.4a}$$
$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{w}_k, \boldsymbol{w} \sim N(\boldsymbol{0}, \boldsymbol{R}) \tag{3.4b}$$

Since everything is linear and Gaussian all distribution involved are Gaussian

$$
\begin{aligned}
p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \quad & N_{\boldsymbol{x}_k}(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}) \quad && = N(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}) && \text{(3.5a)}\\
p(\boldsymbol{z}_k|\boldsymbol{x}_k) = \quad & N_{\boldsymbol{z}_k}(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}) \quad && = N(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}) && \text{(3.5b)}\\
p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \quad & N_{\boldsymbol{x}_k}(\boldsymbol{x}_k^k, \boldsymbol{P}_k^k) \quad && = N(\boldsymbol{x}_k^k, \boldsymbol{P}_k^k) && \text{(3.5c)}\\
p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) = \quad & N_{\boldsymbol{x}_{k-1}}(\boldsymbol{x}_{k-1}^{k-1}, \boldsymbol{P}_{k-1}^{k-1}) \quad && = N(\boldsymbol{x}_{k-1}^{k-1}, \boldsymbol{P}_{k-1}^{k-1}) && \text{(3.5d)}
\end{aligned}
$$

In the middle column, the index on the N indicates what domain the distribution lives in. The objective is to find filtering estimates for the mean $\boldsymbol{x}_k^k$ and covariance $\boldsymbol{P}_k^k$ of the state. The notation is slightly complicated, the sub-index refers as usual to the discrete time, the super-index however, is used to mark what the maximum observation included in the estimate is. When the super-index is $k-1$ the estimate does not include the latest observation.
Since everything is Gaussian, the end result is normalized and there is no need to actually calculate the normalization constant. Plugging in to equation (3.3) yields

$$
N(\boldsymbol{x}_k^k, \boldsymbol{P}_k^k) \quad = \quad N(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}) \int N(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}) N(\boldsymbol{x}_{k-1}^{k-1}, \boldsymbol{P}_{k-1}^{k-1}) \mathrm{d}\boldsymbol{x}_{k-1}.
$$

Beginning with the integral

$$
N(\boldsymbol{x}_k^{k-1}, \boldsymbol{P}_k^{k-1}) \quad = \quad \int N(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}) N(\boldsymbol{x}_{k-1}^{k-1}, \boldsymbol{P}_{k-1}^{k-1}) \mathrm{d}\boldsymbol{x}_{k-1}
$$

$\boldsymbol{x}_k^{k-1}$ and $\boldsymbol{P}_k^{k-1}$ can be found to be

$$
\begin{aligned}
\boldsymbol{x}_k^{k-1} &= \boldsymbol{F}\boldsymbol{x}_{k-1}^{k-1}\\
\boldsymbol{P}_k^{k-1} &= \boldsymbol{F}\boldsymbol{P}_{k-1}^{k-1}\boldsymbol{F}^T + \boldsymbol{Q}
\end{aligned}
$$

the calculations are given in appendix B.3.
Now the only thing that remains is to multiply with the likelihood

$$
N(\boldsymbol{x}_k^k, \boldsymbol{P}_k^k) = N(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}) N(\boldsymbol{x}_k^{k-1}, \boldsymbol{P}_k^{k-1})
$$

Using the rules for multiplication of two Gaussians (equation (B.4)) and some matrix identities (appendix B.4) the mean $\boldsymbol{x}_k^k$ and the covariance $\boldsymbol{P}_k^k$ can be found

$$
\begin{aligned}
\boldsymbol{x}_k^k &= \boldsymbol{x}_{k-1}^k + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_{k-1}^k) && \text{(3.6a)}\\
\boldsymbol{P}_k^k &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^{k-1} && \text{(3.6b)}\\
\boldsymbol{K}_k &= \boldsymbol{P}_k^{k-1}\boldsymbol{H}^T(\boldsymbol{Q} + \boldsymbol{H}\boldsymbol{P}_k^{k-1}\boldsymbol{H}^T)^{-1} && \text{(3.6c)}
\end{aligned}
$$

these equations can be derived in numerous other ways e.g. using Hilbert spaces as it was done originally by Kalman (1960) or by using expectations as e.g. in Welling (2000).

3.1.3 Smoothing

In smoothing, values in the future are used as well as values in the past. Hence, a sequential schema needs both a forward and a backward pass. The filter described above is the forward pass and in the following the backward pass will be derived using the probabilistic approach. As it was the case for the filter many other derivations are possible.
Instead of calculating the conditional smoothing estimate $p(\boldsymbol{x}_k|z_{1:T})$ directly the joint probability $p(\boldsymbol{x}_k, z_{1:T})$ is used. This makes the calculations more transparent and can easily be corrected in the end by remembering that $p(\boldsymbol{x}_k, z_{1:T}) = p(\boldsymbol{x}_k|z_{1:T})p(\boldsymbol{z}_{1:T})$. The first step is to expand

$$\begin{aligned} p(\boldsymbol{x}_k, z_{1:T}) &= \int p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \boldsymbol{z}_{1:T})\mathrm{d}\boldsymbol{x}_{k+1} \\ &= \int p(\boldsymbol{z}_{1:T}|\boldsymbol{x}_k, \boldsymbol{x}_{k+1})p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})\mathrm{d}\boldsymbol{x}_{k+1} \\ &= \int p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k+1})p(\boldsymbol{z}_{k+2:T}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})\mathrm{d}\boldsymbol{x}_{k+1} \end{aligned}$$

where the last equality comes from the fact that the observations are independent given the hidden variable, and that, given the hidden state at time $k+1$, the observations are independent of the state at time k.
Now using that $p(\boldsymbol{x}_{k+1}, z_{1:T}) = p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{z}_{k+2:T}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1})$ the equation can be expanded further

$$p(\boldsymbol{x}_k, z_{1:T}) = \int p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k+1})p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})\frac{p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{k+1})}{p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1})}\mathrm{d}\boldsymbol{x}_{k+1}$$

and even further by inserting

$$p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})$$

and

$$\begin{aligned} p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1}) &= \int p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k+1})p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})\boldsymbol{x}_k \\ &= p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}) \int p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})\mathrm{d}\boldsymbol{x}_k \end{aligned}$$

into the equation

$$\begin{aligned} p(\boldsymbol{x}_k, z_{1:T}) &= \int \frac{p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k+1})p(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{k+1})}{p(\boldsymbol{z}_{1:k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1})}\mathrm{d}\boldsymbol{x}_{k+1} \\ &= \int \frac{p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k)p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{k+1})}{p(\boldsymbol{z}_{k+1}|\boldsymbol{x}_{k+1}) \int p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k}\mathrm{d}\boldsymbol{x}_{k+1} \\ &= \int \frac{p(\boldsymbol{z}_{1:k}, \boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{k+1})}{\int p(\boldsymbol{z}_{1:k}, \boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k}\mathrm{d}\boldsymbol{x}_{k+1} \\ &= \frac{p(\boldsymbol{x}_k, \boldsymbol{z}_{1:k})}{p(\boldsymbol{z}_{1:k})} \int \frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{k+1})}{\int p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k}\mathrm{d}\boldsymbol{x}_{k+1} \end{aligned}$$

After reducing both sides can be divided by $p(\boldsymbol{z}_{1:T})$ and the results arises

$$p(\boldsymbol{x}_k|z_{1:T}) = p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \int \frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{z}_{1:T})}{\int p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k}\mathrm{d}\boldsymbol{x}_{k+1} \tag{3.7}$$

It all boils down to a backwards pass that takes the smoothing estimate at time $k+1$ given by $p(\boldsymbol{x}_{k+1}|\boldsymbol{z}_{1:T})$ to the estimate at time k given by $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:T})$. Note that the filtering estimate from the forward pass ($p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$) is part of the smoothing.

3.1.4 The linear Gaussian version (Kalman Smoother)

Following the lines from the filter derivations, the Kalman smoother can be calculated in exactly the same way by inserting the Gaussians from equation (3.5) and the filtering estimates into equation (3.7). In this way the following backwards recursion arises

$$\boldsymbol{x}_{k-1}^T = \boldsymbol{x}_k^k + \boldsymbol{J}_{k-1}(\boldsymbol{x}_k^T - \boldsymbol{x}_k^{k-1}) \tag{3.8a}$$

$$\boldsymbol{P}_{k-1}^T = \boldsymbol{P}_{k-1}^{k-1} + \boldsymbol{J}_{k-1}(\boldsymbol{P}_k^T - \boldsymbol{P}_k^{k-1}) \tag{3.8b}$$

$$\boldsymbol{J}_{k-1} = \boldsymbol{P}_{k-1}^{k-1}\boldsymbol{F}^\mathsf{T}(P_k^{k-1})^{-1} \tag{3.8c}$$

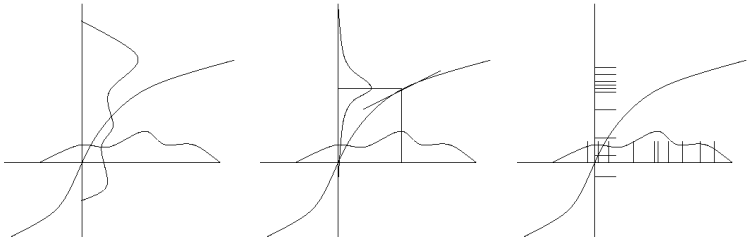
Where the superscript T denotes the smoothing estimate at the time indicated by the subscript and T denotes the transpose. A superscript k indicates a filtering estimate. The smoother is initialized with the final values of the filter $\boldsymbol{x}_T^T$ and $\boldsymbol{P}_T^T$. During smoothing the expectation of the mean and variance ($< \boldsymbol{x}_k >_{p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})}$ and $< \boldsymbol{x}_k\boldsymbol{x}_k >_{p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})}$) of the observation probability can be calculated with almost no overhead and so can the 'lag-one´ covariance smoother ($< \boldsymbol{x}_k\boldsymbol{x}_{k-1} >_{p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})}$) see e.g. Welling (2000). These quantities are used to determine the parameters in the system.

3.2 Nonlinear sequential methods

If either the transition or the observation equation is nonlinear or if the noise contributions are non-Gaussian, the Kalman Filter/smoother is not sufficient and more advanced methods are needed. For most of these methods both the filter and the smoother can be derived but smoothing often becomes extremely complicated and most applications deal with online estimation. Therefore, smoothing methods have not been developed nearly as much as filtering and here only a few methods shall be mentioned. The sequential smoothing methods include Expectation Propagation (Heskes and Zoeter, 2002; Minka, 2001), Particle Smoothing (Fong et al., 2002) and Extended Kalman smoothing (Roweis and Ghahramani, 2001).
The nonlinear filtering algorithms fall into four categories: Extended Kalman Filters, Gaussian Sum Filters, Sigma-Point Kalman Filters and Sequential Monte

Carlo Methods (Particle Filters) (Merwe and Wan, 2003). Another way to categorize the methods is Gaussian belief (Extended Kalman Filters, Sigma Point Filters, Moment Matching), Mixture Of Gaussians (Gaussian-Sum Filter, Pseudo-Bayes) and non-parametric methods (Particle Filters) [1]. In the Extended Kalman Filter, the distributions are assumed Gaussian but, the functions are not linear. The functions $\boldsymbol{f}$ and $\boldsymbol{h}$ are linearized around the previous state $\boldsymbol{x}_{k-1}$ using a second order Taylor expansion and then the standard Kalman equations are used. The result is a Gaussian distribution for $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ (see figure 3.1(b)). For nonlinear systems the solution is better than a normal Kalman Filter, and it is accurate to first order. The Unscented Kalman Filter (Sigma Point Filter) (Julier and Uhlmann, 1997) propagates points one standard deviation from the previous state $\boldsymbol{x}_{k-1}$ through the nonlinearity and then uses the points weighted appropriately (Gaussian quadrature like) to estimate mean and co-variance of a Gaussian. Finally this is used in the standard Kalman equations. It is accurate to the second order.
If the process noise distribution is approximated by a Mixture of Gaussians the family of Gaussian Sum Filters arises (Alspach and Sorensen, 1972). In the Mixture of Gaussians each mixture component is propagated through an extended Kalman Filter. The state update $\boldsymbol{f}$ is linearized around the means of each mixture component and $\boldsymbol{h}$ is linearized around the predicted value for the mean of each mixture $\boldsymbol{f}(x_{k-1})$. The resulting distribution is again a Mixture of Gaussians. If the process noise is also non-Gaussian, this too can be approximated with a Mixture of Gaussians. However, in this case the number of mixing components increases quickly.
Nonparametric methods are an entirely different approach to nonlinear filtering. In the Particle Filter it is assumed that the distributions $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ and $p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})$ from equation (3.3) can be estimated by discrete distributions (figure 3.1(c)). Samples are drawn from the posterior distribution using importance sampling and a proposal distribution. In the generic Particle Filter the transition probability ($p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$) is used as proposal, but other proposals have been proposed in e.g. the Extended Kalman Particle Filter and the unscented Particle Filter (Merwe et al., 2001), in these methods a filter (EKF or UKF) is calculated for each particle and the resulting Mixture of Gaussians is used as proposal distribution for the Particle Filter. In an attempt to combine the Particle Filter and the Gaussian Sum Filter the Gaussian Sum Particle Filtering was proposed (Kotecha and Djuric, 2001). In this approach both the density and the process noise are considered Mixture of Gaussians. In each time step samples are drawn from the mixture approximating $p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})$, these samples are propagated through the nonlinearity and used to offset the means in a mixture describing $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$. Then samples are drawn from this distribution too. In this way a discrete approximation of $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})$ is obtained and the sample mean and covariance of the new mixtures can be estimated. Unfortu-

[1] Thomas Minka http://www.stat.cmu.edu/~minka/dynamic.html



(a) A density propagated through a nonlinear function.

(b) In the extended Kalman Filter, the distribution is assumed to be a Gaussian modified by a linearization of the nonlinearity around the previous state.

(c) In the Particle Filter the densities are approximated by discrete samples. These samples can be sent through the nonlinearity to give an estimate of the output density.

Figure 3.1: Propagation of a pdf through a nonlinearity 3.1(a) and different approximations to the propagated distribution 3.1(b) and 3.1(c). This is the prediction step corresponding to equation (3.2a). Afterwards the resulting pdf is modified to match the measurements equation (3.2b).

nately the number of mixtures explode, to avoid this mixtures with small weight can be thrown away. In a similar manner, the Gaussian Mixture Sigma Point Particle Filter (Merwe and Wan, 2003) uses a bank of Sigma Point Filters to update $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})$ and then samples are drawn from the mixture and the importance weights are calculated before a Gaussian Mixture is fitted to produce the posterior estimate.
In Lehn-Schiøler et al. (2004) an algorithm based on the Parzen density estimator is presented. The algorithm is best categorized as non-parametric, since it can be seen as a direct extension to the Particle Filter. The basic concept is to improve the performance of the Particle Filter by using a better density estimate.
The algorithm is similar to the Gaussian Sum Particle Filter and the Kernel Filter (Hurzeler and Kunsch, 1998), however, it is derived in a different manner that allows use of any kernel type. The derivation of the algorithm uses a sample mean estimate of the integral $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})$ and a 'Particle Filter like´ update of the weights.

3.3 The Parzen Particle Filter

With a Parzen density estimator (Devroye and Lugosi, 2001; Parzen, 1962) a distribution can be approximated arbitrarily close by a number of identical

kernels centered on points chosen from the distribution. In the Particle Filter the kernels are delta functions but information can be gained by using a broader kernel.
The distribution at time $k-1$ can be approximated by

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) \approx \sum_{i}^{N} w_{k-1}^{i} K(\boldsymbol{A}_{k-1}^{i}(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i}))$$

where $\boldsymbol{A}^{i}$ is a transformation matrix used to keep track of distortions of the kernel. Each kernel can be propagated through the mapping $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ by using a local linearization, yielding a continuous output distribution $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$. This is again a sum of kernels but the kernels are no longer identical (in the sense that they are from the same family of functions, yet they have different parameters).
Using the kernel representation – and neglecting the normalization – equation (3.3) can be written as

$$\begin{aligned} & p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \qquad (3.9) \\ \propto\ & p(\boldsymbol{z}_k|\boldsymbol{x}_k) \int p_v(\boldsymbol{x}_k - f(\boldsymbol{x}_{k-1})) \sum_{i}^{N} w_{k-1}^{i} K(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i}) \mathrm{d}\boldsymbol{x}_{k-1} \\ =\ & \sum_{i}^{N} p(\boldsymbol{z}_k|\boldsymbol{x}_k) w_{k-1}^{i} \int p_v(\boldsymbol{x}_k - f(\boldsymbol{x}_{k-1})) K(\boldsymbol{A}_{k-1}^{i}(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i})) \mathrm{d}\boldsymbol{x}_{k-1} \end{aligned}$$

Each part of the sum can be handled individually, and under the assumption that the kernels are small compared to the dynamics in the nonlinearity, $\boldsymbol{f}$ can be locally linearized. That is, the kernels used to approximate the distribution must be narrow compared to the changes in the function. By linearizing $\boldsymbol{f}$ around $\boldsymbol{x}_{k-1}^{i}$ the jacobian $\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^{i}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{\boldsymbol{x}_{k-1}^{i}}$ is introduced and the following change of variables can be employed

$$\tilde{\boldsymbol{x}}_{k-1} = \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^{i}} (\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i})$$

Inserting the linearization and the new variable in the integral from the last line of equation (3.9) one gets

$$\begin{aligned} & \int p_v(\boldsymbol{x}_k - f(\boldsymbol{x}_{k-1})) K(\boldsymbol{A}_{k-1}^{i}(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i})) \mathrm{d}\boldsymbol{x}_{k-1} \\ =\ & \int p_v(\boldsymbol{x}_k - f(\boldsymbol{x}_{k-1}^{i}) - \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^{i}} (\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i})) K(\boldsymbol{A}_{k-1}^{i}(\boldsymbol{x}_{k-1} - \boldsymbol{x}_{k-1}^{i})) \mathrm{d}\boldsymbol{x}_{k-1} \\ =\ & \int p_v(\boldsymbol{x}_k - f(\boldsymbol{x}_{k-1}^{i}) - \tilde{\boldsymbol{x}}_{k-1}) K(\boldsymbol{A}_{k-1}^{i}(\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^{i}}^{-1} \tilde{\boldsymbol{x}}_{k-1})) \left| \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^{i}} \right|^{-1} \mathrm{d}\tilde{\boldsymbol{x}}_{k-1} \end{aligned}$$

Changing variables again such that

$$\hat{\boldsymbol{x}}_{k-1} = \boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1}^{i}) - \tilde{\boldsymbol{x}}_{k-1}$$

$$\begin{aligned} & \left|\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}\right|^{-1} \int p_v(\boldsymbol{x}_k - f(\boldsymbol{x}_{k-1}^i) - \tilde{\boldsymbol{x}}_{k-1}) K(\boldsymbol{A}_{k-1}^i (\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} \tilde{\boldsymbol{x}}_{k-1})) \mathrm{d}\tilde{\boldsymbol{x}}_{k-1} \\ = & \left|\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}\right|^{-1} \int p_v(\hat{\boldsymbol{x}}_{k-1}) K(\boldsymbol{A}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} (\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1}^i) - \hat{\boldsymbol{x}}_{k-1})) \mathrm{d}\hat{\boldsymbol{x}}_{k-1} \end{aligned}$$

This integral is an expectation over the process noise

$$E_{p_v}\left[K\left(\boldsymbol{A}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} (\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1}^i) - \hat{\boldsymbol{x}}_{k-1}\right)\right]$$

and can be approximated by a sample mean. In the extreme case a single sample drawn from the transition noise p_v can be used, and the result is a translation of the kernel by the noise sample

$$\begin{aligned} & E_{p_v}\left[K\left(\boldsymbol{A}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} (\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1}^i) - \hat{\boldsymbol{x}}_{k-1})\right)\right] \\ \approx & K\left(\boldsymbol{A}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} (\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1}^i) - \boldsymbol{v}_{k-1})\right), \boldsymbol{v}_{k-1} \sim p_v \end{aligned}$$

Returning to the expression of equation (3.9)

$$\begin{aligned} & p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \\ \approx & \sum_i^N p(\boldsymbol{z}_k|\boldsymbol{x}_k) w_{k-1}^i \left|\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}\right|^{-1} K\left(\boldsymbol{A}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} (\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_{k-1}^i) - \boldsymbol{v}_{k-1})\right), \boldsymbol{v}_{k-1} \\ = & \sum_i^N w_k^i K(\boldsymbol{A}_k^i (\boldsymbol{x}_k - \boldsymbol{x}_k^i)) \end{aligned}$$

is obtained. Here the last equality arises because the density at time k should also be expressed in terms of kernels. From this equation the component's mean, distortion and weight can be identified

$$\begin{aligned} \boldsymbol{x}_k^i &= \boldsymbol{f}(\boldsymbol{x}_{k-1}^i) + \boldsymbol{v}_{k-1} \\ \boldsymbol{A}_k^i &= \boldsymbol{A}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-1} \\ w_k^i &= w_{k-1}^i p(\boldsymbol{z}_k|\boldsymbol{x}_k^i) \left|\boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}\right|^{-1} \end{aligned}$$

This derivation holds for any kernel.
If the kernel is Gaussian, multiplying the input with $\boldsymbol{A}$ is equivalent to modifying the covariance to $\boldsymbol{A}\boldsymbol{A}^\mathsf{T}$. In that case, the covariance update is given by

$$\boldsymbol{\Sigma}_k^i = (\boldsymbol{A}_k^i)^{-1}(\boldsymbol{A}_k^i)^{-\mathsf{T}} = \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i} (\boldsymbol{A}_k^i)^{-1}(\boldsymbol{A}_k^i)^{-\mathsf{T}} \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{-\mathsf{T}} = \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i} \boldsymbol{\Sigma}_{k-1}^i \boldsymbol{J}|_{\boldsymbol{x}_{k-1}^i}^{T}$$

The transformation matrix $\boldsymbol{A}$ (or $\boldsymbol{\Sigma}$ in the Gaussian case) is distorted in each iteration. To avoid to much distortion a resampling schema can be applied. With a suitable frequency the distribution can be re-approximated by a Parzen

estimator by drawing samples from $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$, choosing $\boldsymbol{A}$ or $\boldsymbol{\Sigma}$ to take their initial values, and setting the weights equal.
The properties of the kernel are iterated through the system equations, thus it is not necessary to optimize the kernel parameters at every step. In addition, the approximation of the integral – stochastically using a sample drawn from p_v – includes an inherent resampling step at every iteration, which allows the Parzen Particle Filter accuracy to survive longer than the standard version. However, in the experiments presented in the next section resampling was employed at every iteration.
To round out the presentation of the Parzen Particle Filter pseudo code for the algorithm is presented. Real code can be found at www.imm.dtu.dk/~tls

Initialization	
Find a mean $(\boldsymbol{x}_0^i)$ for each kernel	e.g. draw from a broad distribution
Set the (common) covariance	e.g. 0.2cov means $(\boldsymbol{x}_0)$
Set all weights	$w_0^i = 1/\mathrm{N}$
For all time steps $k = 1 : T$ **and for each kernel** $i = 1 : N$	
Find the mean	$\boldsymbol{x}_k^i$ by drawing a sample from $p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)$
Find the weight	$w_k^i = w_{k-1}^i p(\boldsymbol{z}_k|\boldsymbol{x}_k^i)$
Find the covariance	$\boldsymbol{\Sigma}_k^i = \boldsymbol{J}^i \boldsymbol{\Sigma}_k^{i-1} (\boldsymbol{J}^i)^T$
Renormalize the weights	$w_k^i = w_k^i / \sum_i (w_k)$
Find the predicted state	$\hat{\boldsymbol{x}} = \sum_i w_k^i \boldsymbol{x}_k^i$
Find new kernels	resample the distribution $\sum w_k^i N(x_k^i, \boldsymbol{\Sigma}_k^i)$

3.3.1 Parzen results and conclusion

In this section the performance of the Parzen Particle Filter will be compared to the performance of the standard Particle Filter (SIR) (Arulampalam et al., 2002) The method is tested on a one-dimensional problem

$$x_k = \frac{x_{k-1}}{2} + 25\frac{x_{k-1}}{(1+x_{k-1}^2)} + 8\cos(1.2k) + v_k \tag{3.10a}$$

$$z_k = 10\arctan(\frac{x_k}{10}) + w_k \tag{3.10b}$$

Where v_k and w_k are drawn from Gaussian distributions $G(0,1)$ (figure 3.3(a)) and from gamma distributions $\Gamma(3,2)$ (figure 3.3(b)). Figure 3.2 show the transition function for $k = 0$, a similar problem was originally proposed for testing sampling methods in Carlin et al. (1992) and was used again in Arulampalam et al. (2002). Note that the fix-points move as k varies.

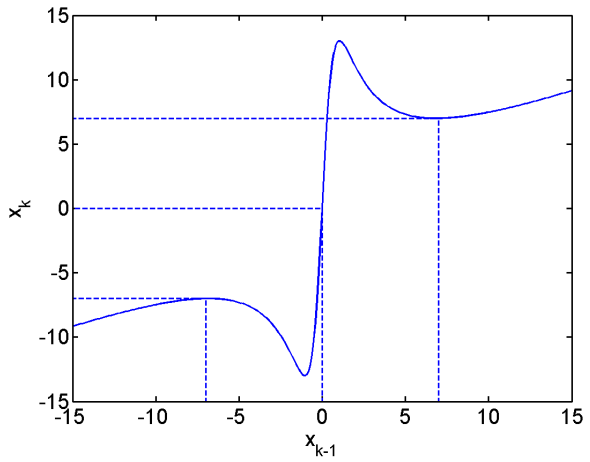
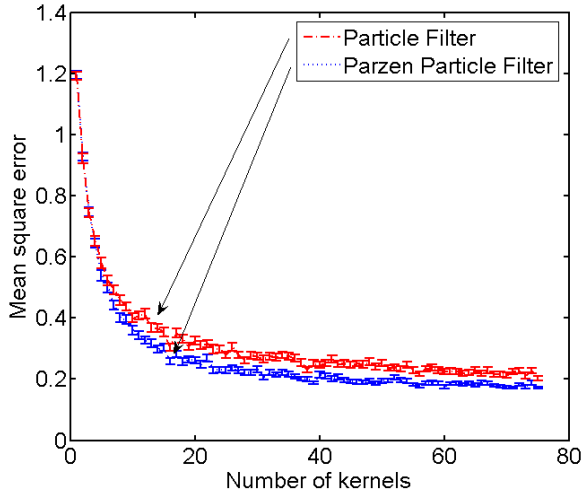


Figure 3.2: The transition function used to evaluate the performance of the Parzen Particle Filter. The functional form is given in equation (3.10) (k=0). The function has two attracting fix-points and one repelling, thus making the dynamics switch from one basin to another.

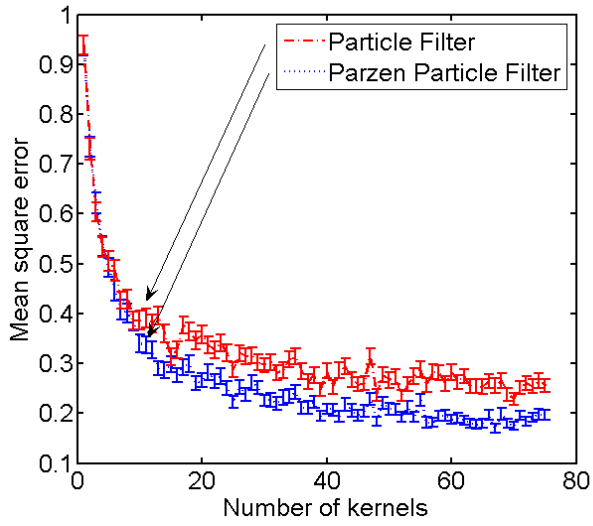
The Parzen Particle Filter and the generic Particle Filter has been used on 100 time series generated using equation (3.10).
In figure 3.3(a) the mean square error is plotted as a function of the number of kernels. It can be seen that with few kernels the methods perform equally good (or bad) but as the number of kernels increases the kernel method becomes better. It can be seen that for this one-dimensional example, the methods perform equally well but the number of particles can be reduced drastically by improving the density estimate.
The simulations show that the Parzen Particle Filter improve the performance both with Gaussian and non-Gaussian noise. In this work only the special case with a Gaussian kernel is examined. However, it is expected that a broader kernel would be well suited for long tailed noise, since it will be more likely to get the particles spread out.
The code for the Filter can be downloaded at http://www.imm.dtu.dk/~tls/code/.

3.4 Markov chain Monte Carlo

The Markov chain Monte Carlo (MCMC) method is an integration technique alternative to the Particle Filter methods. Originally the use of Monte Carlo



(a) Performance results of the Parzen Particle Filter and the standard Particle Filter. In this case with Gaussian noise. Note that the performance of a Parzen Particle Filter with $\approx$ 10 kernels equals that of a normal Particle Filter with $\approx$ 20 kernels.



(b) Performance results of the Parzen Particle Filter and the standard Particle Filter. In this case with gamma distributed noise.

Figure 3.3: Comparison of standard Particle Filter (SIR) and Parzen Particle Filter on the problem from equation (3.10). Both with Gamma distributed and Normal distributed noise the Parzen Filter requires fewer kernels.

techniques for State-Space Models was introduced by Carlin et al. (1992); Gordon et al. (1993); Shephard (1994). A thorough outline of the MCMC sampling in non-Gaussian State-Space Models are given in the review by Tanizaki and Mariano (2000).
The MCMC-method has the advantage of directly providing smoothing estimates for the state space process, but in its traditional form the method suffers from poor convergence properties. This problem has widely been held as an argument in favor for the sequential-methods, in particular emphasizing the advantages of these methods in scenarios involving real-time signal-processing (on-line filtering). However, a caveat of the sequential type of approximation to the true posterior density is that this approach is susceptible to long correlation times of the state-space process. In principle, the MCMC-algorithm is free from this problem, in keeping with the non-sequential representation of the sampling. In the following the MCMC method will be presented and a way of applying it to on-line filtering will be introduced.

3.4.1 MCMC for state-spaces

In the MCMC method, a state space, $\phi \in \Phi$ is sampled according to a given probability distribution, $\phi \sim p(\phi)$, by generating a Markov chain of states, $\{\phi^{(i)}\}_i$, through a fixed matrix of transition probabilities. Given the chain ϕ a new chain ϕ' is selected. The transition probabilities, $T(\phi \to \phi')$, are chosen so the condition of *detailed balance* is satisfied

$$p(\phi)T(\phi \to \phi') = p(\phi')T(\phi' \to \phi). \tag{3.11}$$

Let $p^{(i)}(\phi|\phi^{(0)})$ denote the probability distribution of ϕ for the i'the element of the Markov chain, when it is initialized in state $\phi^{(0)}$. According to Perron-Frobenius theorem, $p^{(i)}$ will converge to the 'true' distribution $p(\phi)$ independent of the choice of $\phi^{(0)}$;

$$p(\phi) = \lim_{i \to \infty} p^{(i)}(\phi|\phi^{(0)}),$$

provided that T is ergodic and aperiodic (see ie. Ferkinghoff-Borg (2002) for a detailed discussion).
The transition probabilities are in a computational sense constructed as a product of a proposal probability distribution $q(\phi'|\phi)$, and an acceptance rate $a(\phi'|\phi)$, i.e. $T(\phi \to \phi') = q(\phi'|\phi)a(\phi'|\phi)$. At the $(i+1)$-step in the MCMC algorithm a trial state ϕ', is drawn according to the distribution $q(\phi'|\phi^{(i)})$ and accepted as the new state $\phi^{(i+1)} = \phi'$ with the probability $a(\phi'|\phi^{(i)})$. Otherwise, one sets $\phi^{(i+1)} = \phi^{(i)}$. There is a considerable freedom in the choice of a. The standard Metropolis-Hasting algorithm (Hastings, 1970) is to use

$$a(\phi'|\phi) = \min\left\{\frac{p(\phi')q(\phi|\phi')}{p(\phi)q(\phi'|\phi)}, 1\right\}, \tag{3.12}$$

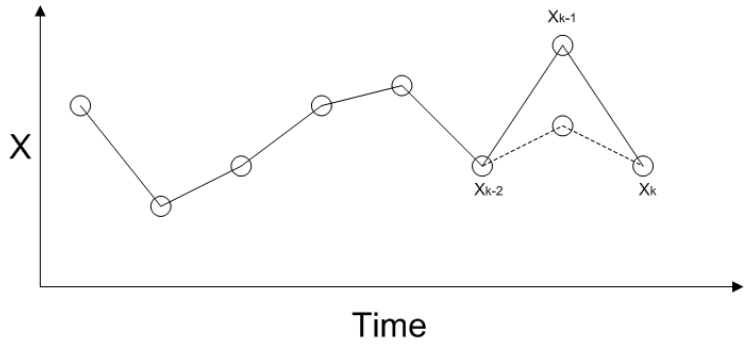


Figure 3.4: Sampling in the MCMC method. Choosing a new path involves selecting a point according to equation (3.15), in this case $\boldsymbol{x}_{k-1}$. Once the point is selected a move is proposed according to equation (3.16). The new sequence can be accepted or rejected according to equation (3.17).

This prescription automatically satisfies the condition of detailed balance, as verified by direct inspection of equation (3.11).
The main deficiency of the MCMC-method in the traditional form outlined above, is its susceptibility to slow relaxation (long correlation times) of the Markov chain. Slow relaxation reduces the effective number of samples and may lead to results which are erroneously sensitive to the particular initialization of the chain. It should be noted that the sequential methods suffers from the same problems with relaxation.

3.4.2 MCMC method for on-line filtering

In the traditional Particle Filter approach to state-space tracking, the particles represent a sample of the posterior density, $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ of the last state, $\boldsymbol{x}_k$, only. In applying the MCMC technique to the tracking problem, a state in the Markov chain, ϕ, is identified with the full history of states in the original state-space, $\phi = \boldsymbol{x}_{1:k}$. It follows from the Markov property of the state transition density and the observation likelihood, equation (3.1), that the joint posterior density $p(\phi) = p(\boldsymbol{x}_{1:k}|\boldsymbol{z}_{1:k})$ is given by

$$p(\boldsymbol{x}_{1:k}|\boldsymbol{z}_{1:k}) = \frac{1}{p(\boldsymbol{z}_{1:k})} \prod_{j=1}^{k} p(\boldsymbol{x}_j|\boldsymbol{x}_{j-1}) p(\boldsymbol{z}_j|\boldsymbol{x}_j). \tag{3.13}$$

Notice, that the normalization constant $p(\boldsymbol{z}_{1:k})$, cancels out in the Metropolis definition of the acceptance rates, equation (3.12).
One obvious advantage of sampling the joint posterior density $p(\boldsymbol{x}_{1:k}|\boldsymbol{z}_{1:k})$ rather than the marginalized posterior density $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ is the gain of statistical in-

formation. However, such extension of the state-space implies that the proposal density function commonly used in PF methods, $q(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{z}_{1:k}) = q_{PF}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{z}_k)$, must be augmented with proposal for changing past states $\boldsymbol{x}_{t<k}$. For simplicity, the extension of the proposal distribution is factorized in time (T) and space (X) in the following manner

$$q(x'_{1:k}|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}) = q_T(t|k) q_X^{(t)}(x'_{t:k}|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}) \delta(x'_{1:(t-1)} - \boldsymbol{x}_{1:(t-1)}). \tag{3.14}$$

In effect, first a time index is selected, $1 \leq t \leq k$, independent of the current state $\boldsymbol{x}_{1:k}$, according to the probability distribution $q_T(t|k)$. Then, a trial path is drawn according to the spatial proposal distribution, $q_X^{(t)}(x'_{1:k}|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k})$, which is chosen to be zero for all pairs of paths which are not identical up to time t. Since the Markov process is expected to generate states with exponentially decaying time-correlations, a natural form for $q_T(t|k)$ is the exponential distribution, $q_T(t|k) \sim \exp((t-k)/\tau)$. Here, τ equals the average size of the back-propagating step in the path-space sampling following an observation at time k. In order to make the MCMC method applicable for on-line filtering, an extra emphasis should be put on the sampling of the latest state, $\boldsymbol{x}_k$. Therefore the following definition of q_T are proposed

$$q_T(t|k) = \begin{cases} 0 & t > k \\ p_{now}\delta_{t,k} + (1 - p_{now}) \frac{1}{N_k} \exp((t-k)/\tau) & 0 < t \leq k \end{cases} \tag{3.15}$$

Here, p_{now} is the probability of attempting a change to the latest state $\boldsymbol{x}_k$ only, and N_k is a normalization constant, $N_k = \sum_{t=1}^{k} \exp(t-k)/\tau = \frac{1-\exp(-k/\tau)}{1-\exp(-1/\tau)}$.

As regards to the spatial proposal distribution $q_X^{(t)}(x'_{t:k}|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k})$, the direct approach is simply to adopt the proposal distribution applied in a given PF-method

$$q_X^{(t)}(x'_{t:k}|\boldsymbol{x}_{t:k}, \boldsymbol{z}_{t:k}) = q_{PF}(x'_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_t) \delta(x'_{(t+1):k} - x_{(t+1):k}). \tag{3.16}$$

This leads to a sampling scheme as sketched in figure 3.4. With the above choice of q_T and q_X the acceptance rates in the MCMC method, equation (3.12) takes the particular simple form

$$a(\boldsymbol{x}'_t|\boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}) = \min\left\{ \frac{p(\boldsymbol{z}_t|\boldsymbol{x}'_t) p(\boldsymbol{x}'_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t+1}|\boldsymbol{x}'_t) q_{PF}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_t)}{p(\boldsymbol{z}_t|\boldsymbol{x}_t) p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) q_{PF}(\boldsymbol{x}'_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_t)}, 1 \right\} \tag{3.17}$$

for $1 \leq t < k$. The acceptance rates for $t = k$ is obtained by omitting $\frac{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}'_t)}{p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)}$ in the above expression. In the standard Particle Filter the transition probability $p(\boldsymbol{x}'_t|\boldsymbol{x}_{t-1})$ is used as proposal distribution. This choice is also used in the MCMC method in the next section.

In essence the on-line version of MCMC selects a single sample from the sequence, propose a change of that sample and accept it according to equation (3.17). Samples near the current time are selected with higher probability

because it is expected that new observations will be more likely to influence them. Finally, knowledge about the system can be utilized to incorporate a global move. In the following bimodal example a part of the sequence e.g. $\boldsymbol{x}_{(k-20):(k)}$ can change sign with low probability. This move also has to be accepted with an acceptance rate similar to equation (3.17).

3.4.3 MCMC results

In order to compare the performance of various Particle Filtering methods with the MCMC a bimodal, one dimensional model is examined. The model is similar to the one used in section 3.3.1

$$\begin{aligned} x_k &= f(x_{k-1}) + v_{k-1} && \text{(3.18a)} \\ f(x) &= x - \frac{2h}{x_f}\left(\left(\frac{x}{x_f}\right)^3 - \left(\frac{x}{x_f}\right)\right) \\ z_k &= g(x_k) + w_k && \text{(3.18b)} \\ g(x) &= x^2 + \epsilon x \end{aligned}$$

The map $f(x)$ has two attracting fix points at $\pm x_f$ and a repulsive fix point at $x = 0$ (see figure 3.5) implying that the state-space is divided into two basins. The process will spent most of the time fluctuating around x_f or $-x_f$. The parameter h determines the potential barrier separating one basin from the other. In these experiments $x_f = 10$, $\epsilon = 1$, the noise contributions v_k and w_k are normal zero mean with variance 1. The value of h is varied between 2.5 and 4.5. The simple functional form of $f(x)$ enables analytic calculations of the transition times between the two basins of the models; these calculations are utilized in sofar unpublished work by Ferkinhoff-Borg et al. (2005)
To quantify the results two error measures where studied. The traditional root-mean-square error also used to quantify the performance of the Parzen Particle Filter is given by

$$RMSE = \sqrt{\frac{1}{T}\sum_{1}^{T}(x_k - < x_k >)^2}$$

where T is the total number of steps and $< x_k >$ is the posterior average of the state variable at time k estimated through a given algorithm. In addition to this the Basin Error (BE) defined as

$$BE = \frac{1}{2}(1 - \frac{1}{T}\sum_{1}^{T}(sgn(x_k)sgn(< x_k >))$$

was used. It quantifies the fraction of times the algorithm predicts a wrong sign for the state variable x. A value of BE = 0.5 means that the performance of the

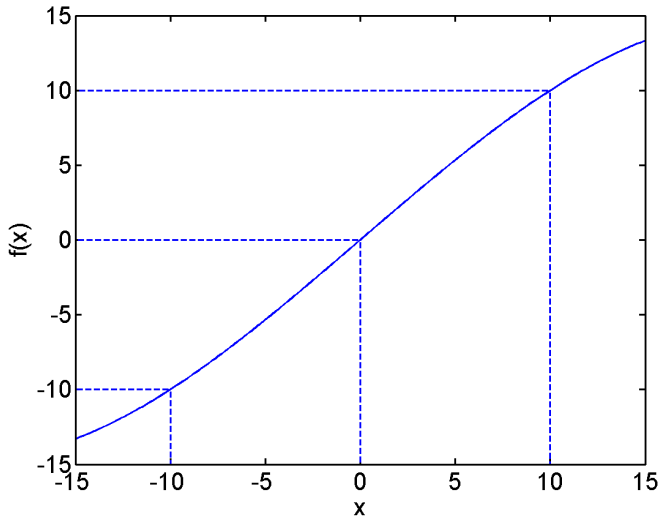


Figure 3.5: The transfer function used in the comparision of PF and MCMC methods (equation (3.18)). The function is similar to the one used to test the Parzen Particle Filter (figure 3.2). It has two stable fix-points at $x = \pm 10$ and an unstable at $x = 0$. Unlike the function used to test the Parzen Filter this function has some nice analytical properties.

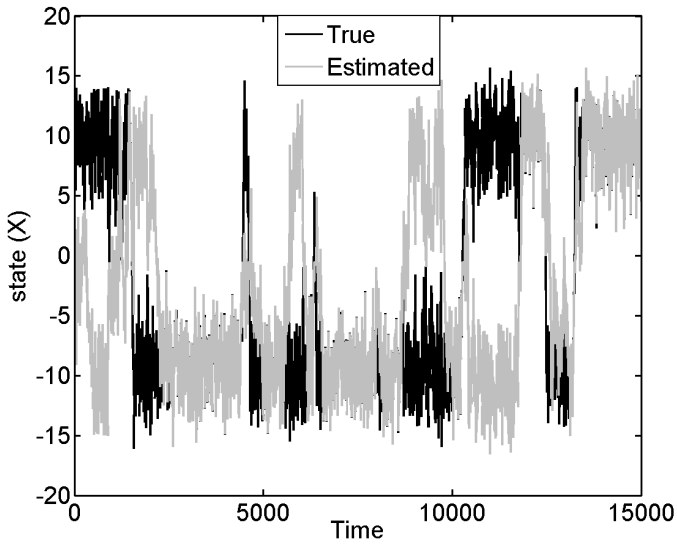


Figure 3.6: One of the instances of the sequence used to test the methods. The black line is the true path and the gray line is estimated using a PF approach. Note that often the PF chooses the wrong basin despite the numerical value being close to the true value.

Method	basin error	STD	RMS error	STD
Particle Filter	0.50	0.05	13.3	0.7
Sigma Point Particle Filter	0.47	0.04	13.0	0.3
Gaussian Sum Particle Filter	0.59	0.05	14.7	0.7
SRCDKF †	0.55	0.04	14.9	0.7
Particle Filter global move	0.44	0.05	12.3	0.7

Table 3.1: Errors obtained with different filtering methods. The ReBEL toolbox was used to perform the experiments. 1000 particles were used in the PF methods. † Three times the output was NaN

algorithm in resolving the macro-state (basin) of the system is the same as by guessing at random.
For each value of h in the model, 10 independent realizations of the state process, equation (3.18), is generated starting from $x_0 = 0$. The process in each realization is iterated T = 15000 times to ensure a non-vanishing number of transitions between the basins for all h. For each realization, a corresponding observation path $z_{1:T}$ is generated. All algorithms discussed below are tested on this fixed set of state and observation realizations.
In table 3.1 the RMSE and BE of the various sequential filtering algorithms for h = 3.0 are listed. The ReBEL toolbox (http://choosh.ece.ogi.edu/rebel/) by Rudolph van der Merwe and Eric A. Wan was used to perform the experiments. The entries give the estimated average error and the uncertainty of the estimate (STD) based on the 10 realizations and using N = 1000 particles. For this number of particles, none of the methods performs significantly better in estimating the basin than by guessing at random. This leaves to the conclusion that the accuracy of the various algorithms are more or less identical for the model at hand, and in the following focus will be on just one of these; the standard Particle Filter method (SPF). Figure 3.6 shows a typical trajectory of the state variable and the corresponding average value from the SPF method, illustrating the failure of the method in estimating the right basin of the process. As discussed in the previous section, one obvious remedy is to complement the proposal distribution with a move which explicitly carries out the transitions between the two basins. The last row of table 3.1 gives the accuracy of the SPF method when this operation is added to the sampling. The abbreviation SPF* is used for the the PF with global moves. Only a marginal improvement of the algorithm is observed, which nevertheless indicates that the failure of the method is related to the small transition probabilities between the basins.
Table 3.2 shows how the accuracy of the SPF* scales with the number of particles for various choices of h. Two interesting observations can be made. First, a very large number of particles are in general needed to reach the limiting accuracy. Secondly, the algorithm performs worse for small values of h, corresponding to larger transition probabilities between the basins. The general failure of the

	100	1000	10000	100000	1000000
2.5	0.53 ± 0.05	0.55 ± 0.03	0.30 ± 0.03	0.17 ± 0.02	0.17 ± 0.02
3.0	0.45 ± 0.05	0.44 ± 0.05	0.30 ± 0.04	0.18 ± 0.02	0.13 ± 0.02
3.5	0.60 ± 0.03	0.58 ± 0.06	0.35 ± 0.04	0.17 ± 0.02	0.12 ± 0.02
4.0	0.54 ± 0.06	0.44 ± 0.05	0.32 ± 0.05	0.14 ± 0.05	0.09 ± 0.02
4.5	0.50 ± 0.07	0.58 ± 0.07	0.30 ± 0.07	0.08 ± 0.03	0.09 ± 0.03

Table 3.2: The Basin Error for varying barrier heights (h) and number of particles. The experiments where performed with a Particle Filter using global moves. A very large number of particles are needed to reach the limiting accuracy. Also note that the algorithm performs worse for small values of h, corresponding to larger transition probabilities between the basins.

	Basin error	STD	RMS error	STD
2.5	0.24	0.005	8.51	0.38
3.0	0.140	0.002	6.41	0.05
3.5	0.090	0.001	5.18	0.04
4.0	0.056	0.002	4.14	0.08
4.5	0.079	0.002	4.95	0.07

Table 3.3: Experiments with MCMC using global moves for different barrier heights (h). Compared to the Particle Filter in table 3.2 the errors are very small given that only 1000 'particles' where used.

PF-methods arises from the fact that when a basin change occurs at some time-step i, say $x_{i-1} < 0$ to $x_i > 0$, the likelihood, $p(z_i|x)$, around $x = x_i$ is not sufficiently large to compensate for the low transition probabilities, $p(x_i|x_{i-1})$ associated with the change of basin. Consequently, the posterior distribution

$$p(x_i|z_{1:i}) \propto p(z_i|x_i) \int p(x_i|x_{i-1}) p(x_{i-1}|z_{1:i-1}) \mathrm{d}x$$

will still be much larger for the original basin. Since the typical trajectory for a basin change only involves a few number of states in the transition region, see figure 3.6, no particles are likely to occupy the new basin after filtering when the number of particles are small. In this case, the particles will be frozen in the wrong basin once the state of the process reaches the new basin around x_f. As the number of particles becomes large enough it is more likely that particles cross the barrier at the same time as the true trajectory and the filter is able to pick up the basin change.
In contrast to this the MCMC performs better. In this setup it was allowed 1000 changes to the chain which is computational similar to a PF with 1000 particles. Since these 1000 changes also effect previous time steps it is easier for the MCMC method to correct a decision to be in the wrong basin and utilize the small change in likelihood introduced by ϵ in equation (3.18). The parameter

τ determining how far back in time changes are made to the chain is set to 250. In table 3.3 the result of running the MCMC method on the data can be seen, results are much better than for the SPF*, especially considering the low number of 'particles'.
It is demonstrated that it is always possible to formulate a MCMC algorithm that uses the same proposal as a PF method. It has been shown that there are no hinderance in using MCMC in online applications and the experiments indicate that with the same computational complexity MCMC methods produces superior results. The reason for the success of the MCMC methods is the ability to accumulate evidence over several time steps, thus utilizing the small differences in posterior probabilities. On top of this there are standard ways of improving the performance of MCMC methods such as simulated annealing, parallel tempering and bridging (Ferkinghoff-Borg, 2002).

3.5 Final remarks

The State-Space Model as presented in this chapter can be used in a very wide range of applications. Whether a linear Gaussian model, a nonlinear sequential model or the Markov Chain Monte Carlo method should be applied depends on the problem at hand. However, with the approach for on-line MCMC described here there are no obvious reasons to continue using sequential sampling methods. In this chapter, it has been assumed that the model is known and that the functional relationships and noise distributions must be given before estimating the optimal hidden sequence. For physical systems this assumption is often valid. A good estimate of the functions $\boldsymbol{f}$ and $\boldsymbol{h}$ and the noise contributions can be found by observing the physical system. However, there are also a wide range of systems where the dynamics of the system cannot be modeled by hand. Fortunately, there are methods to deal with these cases. These methods will be described in the next chapter.

Chapter 4

Parameter Estimation

In the previous chapter, the State-Space Model was introduced and a variety of methods to estimating the hidden sequence were described. However, all the methods assumed the parameters of the model were known. In this chapter, parameter estimation using the Expectation Maximization (EM) algorithm will be presented along with an alternative method, based on gradients, originally proposed in Olsson et al. (2005).
The general framework of parameter estimation is applied to the linear version of the State-Space Model.

4.1 EM-Algorithm

Maximum likelihood is a popular estimator for the parameters of many different models. For State-Space Models like the one in equation (3.1) it is the most widely used method, but, the applications are much broader and extend into Independent Component Analysis and Mixture Models just to name a few. In the following the Neal and Hinton (1998) formulation of EM will be used.
Assuming a model with observed variables $\boldsymbol{z}$, state-space variables $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$, then calculation of the log likelihood, $\mathcal{L}(\boldsymbol{\theta})$ involves an integral over the state-space variables of the type

$$\mathcal{L}(\boldsymbol{\theta}) = \ln p(\mathbf{z}|\boldsymbol{\theta}) = \ln \int p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) p(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x} \tag{4.1}$$

The marginalization in equation (4.1)) is intractable for most choices of $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ and $p(\mathbf{x}|\boldsymbol{\theta})$, hence direct optimization is rarely an option – even in the Gaussian

case. Therefore, a lower bound B on the log likelihood is introduced. The bound is valid for any choice of distribution $q(\mathbf{x}|\boldsymbol{\phi})$

$$B(\boldsymbol{\theta}, \boldsymbol{\phi}) = \int q(\mathbf{x}|\boldsymbol{\phi}) \ln \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{x}|\boldsymbol{\phi})} d\mathbf{x} \leq \ln p(\mathbf{z}|\boldsymbol{\theta}) \tag{4.2}$$

Introducing this lower bound, $\boldsymbol{B}$, may seem to have complicated matters, however, the lower bound has a number of appealing properties which makes the original task of finding the parameters easier. An important fact about B becomes clear when rewriting it using Bayes theorem

$$B(\boldsymbol{\theta}, \boldsymbol{\phi}) = \ln p(\mathbf{z}|\boldsymbol{\theta}) - KL[q(\mathbf{x}|\boldsymbol{\phi})||p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})] \tag{4.3}$$

where KL denotes the Kullback-Leibler divergence between the two distributions. Thus, if the variational distribution q is chosen to be exactly the posterior of the hidden variables, B is equal to the log likelihood. For this reason one often tries to choose the variational distribution flexible enough to include the true posterior and yet simple enough to make the necessary calculations as easy as possible.
The approach is to maximize with respect to $\boldsymbol{\phi}$, in order to make the lower bound as close as possible to the log-likelihood and then maximize the bound with respect to the parameters $\boldsymbol{\theta}$. This stepwise maximization can be achieved by using the EM algorithm or by applying the *Easy Gradient Recipe* (see section 4.3).

4.1.1 The EM update

The EM algorithm, as formulated in Neal and Hinton (1998), works in a straightforward scheme which is initiated with random values and iterated until suitable convergence is reached

E: Maximize $B(\boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t. $\boldsymbol{\phi}$ while keeping $\boldsymbol{\theta}$ fixed.
M: Maximize $B(\boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t. $\boldsymbol{\theta}$ while keeping $\boldsymbol{\phi}$ fixed.

It is guaranteed that the lower bound function does not decrease on any combined E and M step. Figure 4.1 illustrates the EM algorithm. The convergence is often slow – e.g. the curvature of the bound function, B, might be much higher than that of $\mathcal{L}$, resulting in very conservative parameter updates. This is particularly a problem in latent variable models with low-power additive noise. In Bermond and Cardoso (1999) and Petersen and Winther (2005), it is demonstrated that the EM update scales with the observation noise level. That is, as the signal-to-noise ratio increases, the M-step change of the parameter decreases, and more iterations are required to converge.
To illustrate how this works the Kalman Model is a good example. In this model the E-step consists of estimating the hidden states given the parameters $\boldsymbol{\theta}(\boldsymbol{F}, \boldsymbol{H} \ldots)$ and this is done by using the smoother as presented in section 3.1.3. In the M-step the parameters are found keeping the hidden states fixed, which can be done in 'one shot´ by setting the derivatives equal to zero.

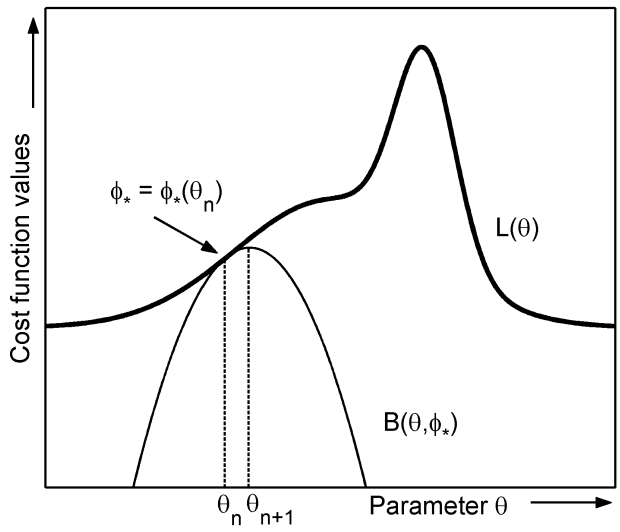


Figure 4.1: Schematic illustration of lower bound optimization for a one-dimensional estimation problem, where $\boldsymbol{\theta}_n$ and $\boldsymbol{\theta}_{n+1}$ are iterates of the standard EM algorithm. The log-likelihood function, $\mathcal{L}(\boldsymbol{\theta})$, is bounded from below by the function $B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)$. The bound attains equality to $\mathcal{L}$ in $\boldsymbol{\theta}_n$ due to the choice of variational distribution: $q(\boldsymbol{x}|\boldsymbol{\phi}_*) = p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}_n)$. Furthermore, in $\boldsymbol{\theta}_n$ the derivatives of the bound and the log-likelihood are identical. In many situations, the curvature of $B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)$ is much higher than that of $\mathcal{L}(\boldsymbol{\theta})$, leading to small changes in the parameter.

4.2 The gradient alternative

Optimization in State-Space Models based on maximizing the log-likelihood with respect to the parameters fall in two main categories based on either gradients or Expectation Maximization (EM).
The EM algorithm (Dempster et al., 1977) is probably the most important algorithm for parameter estimation. It was first applied to the optimization of linear State-Space Models by Shumway and Stoffer (1982) and Digalakis et al. (1993). A general class of linear Gaussian (state-space) models was treated in Roweis and Ghahramani (1999), in which the EM algorithm was the main engine of estimation. In the context of Independent Component Analysis (ICA), the EM algorithm has been applied, in among others Moulines et al. (1997) and Højen-Sørensen et al. (2002). In Olsson and Hansen (2004, 2005), the EM algorithm was applied to the Convolutive ICA problem.
A number of authors have reported slow convergence of the EM algorithm. In Redner and Walker (1984), impractically slow convergence in 2-component Gaussian Mixture Models is documented. This critique is, however, moderated by Xu and Jordan (1996). Modifications have been suggested to avoid the slow convergence of the basic EM algorithm, Lachlan and Krishnan (1997) and Jamshidian and Jennrich (1997) among many but most come at a high cost in terms of computational complexity or at the expense of analytical simplicity.
Another approach to maximum likelihood in State-Space Models, and more generally in complex models, is to iteratively search the space of $\boldsymbol{\theta}$ for the maximal $\mathcal{L}(\boldsymbol{\theta})$ by taking steps in the direction of the gradient, $\nabla_\theta \mathcal{L}(\boldsymbol{\theta})$. A basic *ascend* algorithm can be improved by supplying curvature information, i.e. second-order derivatives, line-search, etc. Often, numerical methods are used to compute the gradient and the Hessian, due to the complexity associated with the computation of these quantities. In Gupta and Mehra (1974) and Sandell and Yared (1978) fairly complex recipes are given for the computation of the analytical gradient in the linear State-Space Model.
In the following sections it will be documented that the *exact* gradient of the log-likelihood function can be computed using only the relatively simple math and programming of the EM algorithm. As a result, the reasonable convergence properties of the gradient-based optimizer are restored. This procedure is termed the *Easy Gradient Recipe*. Furthermore, empirical evidence, supporting the results in Bermond and Cardoso (1999), is presented to demonstrate that the signal-to-noise ratio (SNR) has a dramatic effect on the convergence speed of the EM algorithm. Under certain circumstances, i.e. in high SNR settings, the EM algorithm fails to converge in reasonable time. The central points utilized in the proposed recipe have been mentioned in, e.g. Salakhutdinov et al. (2003), but they did not comment on the relationship of the convergence properties to the SNR.

4.3 Easy Gradient Recipe

An alternative approach to the EM algorithm is to use a gradient based approach like the Easy Gradient algorithm presented in Olsson et al. (2005).
The key idea is to regard the bound, B, as a function of $\boldsymbol{\theta}$ only, instead of a function of both the parameters $\boldsymbol{\theta}$ and the variational parameters $\boldsymbol{\phi}$. As a result, the lower bound can be applied to reformulate the log-likelihood as

$$\mathcal{L}(\boldsymbol{\theta}) = B(\boldsymbol{\theta}, \boldsymbol{\phi}_*) \tag{4.4}$$

where $\boldsymbol{\phi}_* = \boldsymbol{\phi}_*(\boldsymbol{\theta})$ fulfils the constraint $q(\mathbf{s}|\boldsymbol{\phi}_*) = p(\mathbf{s}|\mathbf{x}, \boldsymbol{\theta})$. In practise, it is often complicated to find $\boldsymbol{\phi}_*$ and it cannot always be done analytically. Comparing with the general expression of the bound function in terms of the log likelihood and the Kullback-Leibler divergence (equation (4.3)) it is easy to show that $\boldsymbol{\phi}_*$ maximizes the bound. But, since $\boldsymbol{\phi}_*$ is exactly minimizing the KL-divergence, the partial derivative of the bound with respect to $\boldsymbol{\phi}$ evaluated in the point $\boldsymbol{\phi}_*$, is equal to zero. Therefore the gradient of $B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)$ is equal to the partial derivative

$$\frac{dB(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{d\boldsymbol{\theta}} = \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{\partial \boldsymbol{\theta}} + \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\Big|_{\boldsymbol{\phi}_*} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\phi}_*} = \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{\partial \boldsymbol{\theta}} \tag{4.5}$$

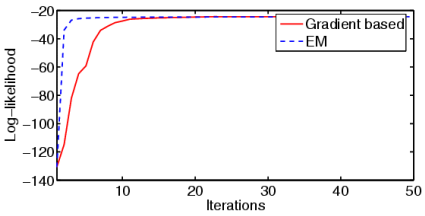
and due to the choice of $\boldsymbol{\phi}_*$, the partial derivative of the bound is the gradient of the log likelihood

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{\partial \boldsymbol{\theta}}$$

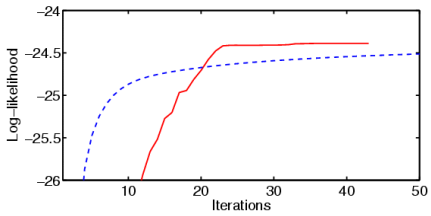
which can be realized by combining equation (4.4) and equation (4.5).
In this way exact values and gradients of the true log likelihood can be obtained using the lower bound. The observation of this is not new, since it is essentially the same that is used in Salakhutdinov et al. (2003) to construct the so-called Expected Conjugated Gradient Algorithm (ECG). The novelty of this recipe, is the practical recycling of low-complexity computations carried out in connection with the EM algorithm. This allows for a much more efficient optimization using *any* gradient-based non-linear optimizer. The recipe can be expressed in MATLAB-style pseudo-code where a function `loglikelihood` takes the parameter $\boldsymbol{\theta}$ as argument and returns $\mathcal{L}$ and its gradient $\frac{d\mathcal{L}}{d\boldsymbol{\theta}}$:

`function` $[\mathcal{L}, \frac{d\mathcal{L}}{d\boldsymbol{\theta}}] =$ `loglikelihood`$(\boldsymbol{\theta})$
1 Find $\boldsymbol{\phi}^*$ such that $\frac{\partial B}{\partial \boldsymbol{\phi}}\Big|_{\boldsymbol{\phi}^*} = 0$
2 Calculate $\mathcal{L} = B(\boldsymbol{\theta}, \boldsymbol{\phi}^*)$
3 Calculate $\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial B}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\phi}^*)$

Step **1**, and to some extend step **2**, are obtained by performing an E-step, while **3** requires only little programming to implement the gradients used to solve for the M-step. Compared to the EM algorithm, the main advantage is



(a) Convergence of EM and a gradient-based method (BFGS). EM starts out faster than the gradient-based method.



(b) Zoom-in on the log-likelihood axis. Even after 50 iterations, EM has not reached the same level as the gradient based method.

Figure 4.2: Comparing convergence of the EM algorithm and the gradient based approach. The gradient method starts out slow but converges after ≈ 40 iterations. EM is fast in the initial iterations but then slows down and even after 50 iterations it has not converged and is 'far' from the level of the gradient method. In figure 4.3 the convergence of the parameters are also provided.

that the function value and gradient can be fed to any non-linear gradient-based optimizer, which in most cases provides a substantial improvement of the convergence properties. In that sense, it is possible to benefit from the speed-ups of advanced gradient-based optimization.

The advantage of formulating the log-likelihood using the bound function, B, depends on the task at hand. In the linear State-Space Model (equation (3.1)) a brute force computation of the gradient of the log-likelihood is costly, since the computation of the gradients scales as $(d_\theta)^2$ times the cost of one Kalman Filter sweep.[1] When using the Easy Gradient Recipe, the combined computational cost depends on the optimizer of choice. Often, state-of-the-art software induces little overhead in addition to the computation of the gradient. In the case of linear State-Space Models, the total computational cost of the Easy Gradient Recipe is then dominated by steps **1** and **2**, which require a single Kalman smoothing. Sandell and Yared (1978) noted in their investigation of linear State-Space Models that a reformulation of the problem resulted in a similar reduction of the computational costs.

In this work, a quasi-Newton gradient-based optimizer has been chosen, that is the optimizer estimates the inverse Hessian using the gradient. The implementation of the BFGS algorithm is due to Hans Bruun Nielsen, (Nielsen, 2000), and has built-in line search and trust region monitoring.

To illustrate the difference in convergence speed figure 4.2 show the log likelihood as a function of iterations for the two algorithms. In the experiment a one-dimensional Kalman Model was used.

[1] The computational complexity of the Kalman Filter is $\mathcal{O}[N(d_s)^3]$, where N is the data length.

4.4 The linear Gaussian case

Before comparing the EM algorithm and the gradient alternative, parameter estimation in the linear Gaussian case i.e. the Kalman Model will be discussed. In the following iterative methods for system identification will be considered. However, there are other alternatives e.g. one-shot algorithms like the subspace method described in Van Overschee and de Moor (1996); Viberg (1995).

4.4.1 Identifiability

The unknown parameters in the Kalman Filter (equation (3.4)) are all entries in the transition matrix $\boldsymbol{F}$, all entries in the observation matrix $\boldsymbol{H}$, and the two covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. Unfortunately, these parameters are not uniquely defined from the observations. The simplest way to see this is to insert an invertible matrix in the equations

$$\boldsymbol{U}\boldsymbol{x}_k = \boldsymbol{U}\boldsymbol{F}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{x}_{k-1} + \boldsymbol{U}\boldsymbol{v}_{k-1}, \boldsymbol{v} \sim N(\boldsymbol{0}, \boldsymbol{Q}) \tag{4.6a}$$

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{x}_k + \boldsymbol{w}_k, \boldsymbol{w} \sim N(\boldsymbol{0}, \boldsymbol{R}) \tag{4.6b}$$

With the new variables $\hat{\boldsymbol{x}_k} = \boldsymbol{U}\boldsymbol{x}_k, \hat{\boldsymbol{F}} = \boldsymbol{U}\boldsymbol{F}\boldsymbol{U}^{-1}$ and so on, the output of the system ($\boldsymbol{z}$) remains the same. For a general system identification problem – where the hidden space has a physical interpretation – this is off course undesired. Even when one does not care about the hidden space the non-uniqueness can cause problems for the optimization.
In the machine learning community the un-identifiability of the parameters has in general not been considered a problem for example Welling (2000) derives the E-M algorithm for parameter estimation without even mentioning the problem. In Ghahramani and Hinton (1996); Roweis and Ghahramani (1999) it is noted that, without loss of generality, the transition noise $\boldsymbol{Q}$ can be set to the identity. However, by remembering that $cov(\boldsymbol{U}\boldsymbol{v}) = \boldsymbol{U}cov(\boldsymbol{v})\boldsymbol{U}^{\mathsf{T}}$ it is obvious that if $\boldsymbol{Q}$ is the identity, $\boldsymbol{U}$ can be any rotation matrix. Hence, restrictions on $\boldsymbol{Q}$ removes scale ambiguities, but leaves rotations.
Kevin Murphy's toolbox Bayes Net Toolbox for Matlab[2] also implements parameter estimation in Kalman Filters along these lines without taking the un-identifiability into account.
In the control community different canonical forms for the state-space system have been investigated and for single input single output (SISO) systems e.g. the observer and the controller canonical form solves the problem. However, for multiple input multiple output (MIMO) systems, the canonical forms require information about how many complex eigenvalues the system contains. The problem arises because the hidden states can have two interpretations. Either a state can describe a variable of the system or it can describe a delay. The problem is treated in e.g. Gevers and Wertz (1984); Overbeek and Ljung (1982),

[2] http://www.ai.mit.edu/~murphyk/Software/

in these papers non-unique overlapping parameterizations are suggested as well as unique representations requiring the estimation of so called 'structure indices'. In McKelvey et al. (2004) a data driven local parametrization is used.
The choice is now whether to keep the full system with all its ambiguities, fixate $\boldsymbol{Q}$ and get rid of the scaling problems, choose $\boldsymbol{F}$ to be diagonal and introduce complex states, try to estimate the structure indices, or under-parameterize the system by choosing $\boldsymbol{F}$ to be diagonal and real.
Since this work stems from the machine learning community, the initial choice was over-parametrization. However, as the work progressed this approach became increasingly unattractive due mainly to the non-uniqueness of the solution and thereby non-reproducible results. At later stages fixating $\boldsymbol{Q}$ was used, still leaving the rotation ambiguities.

4.4.2 Kalman derivatives

To find the optimal parameters in the Kalman Filter the derivatives of the lower bound $\boldsymbol{B}(\theta, \phi_*)$ (equation (4.2)) must be found with respect to the parameters θ. Here ϕ_* is the smoothing solution of the hidden state $\boldsymbol{x}_1 \dots \boldsymbol{x}_T$ thus $q(\boldsymbol{x}|\boldsymbol{\phi}_*) = p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})$. The parameter θ contain $\boldsymbol{F}$, $\boldsymbol{H}$, $\boldsymbol{Q}$, $\boldsymbol{R}$ and the initial distribution $\boldsymbol{x}_0$ and $\boldsymbol{\Sigma}_0$.
By looking at the lower bound at ϕ_*

$$B = \int q(\boldsymbol{x}|\boldsymbol{\phi}_*) \log \frac{p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{q(\boldsymbol{x}|\boldsymbol{\phi}_*)} d\boldsymbol{x} \tag{4.7}$$

$$= \int \boldsymbol{q}(\boldsymbol{x}|\boldsymbol{\phi}_*) \log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x} - \int q(\boldsymbol{x}|\boldsymbol{\phi}_*) \ln q(\boldsymbol{x}|\boldsymbol{\phi}_*) \mathrm{d}\boldsymbol{x} \tag{4.8}$$

it is seen that when $\boldsymbol{\phi}$ is fixed, only the term $\int \boldsymbol{q}(\boldsymbol{x}|\boldsymbol{\phi}_*) \log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}$ can be optimized.
Following e.g. Welling (2000) the joint probability of the hidden variables and the observation given the parameters can be factorized by using the Markov property of the model

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}|\theta) = p(\boldsymbol{x}_0|\theta) \prod_{k=2}^{T} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \theta) \prod_{k=1}^{T} p(\boldsymbol{z}_k|\boldsymbol{x}_k, \theta). \tag{4.9}$$

Since all probabilities are dependent on θ, the dependence is left out in the following to simplify the notation

$$B = \int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) \log[p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T})] \mathrm{d}\boldsymbol{x}_{1:T}$$

$$= \int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) \Big[\log[p(\boldsymbol{x}_0)] + \sum_{k=2}^{T} \log[p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})] + \sum_{k=1}^{T} \log[p(\boldsymbol{z}_k|\boldsymbol{x}_k)] \Big] \mathrm{d}\boldsymbol{x}_{1:T}$$

So far everything holds for general distributions and both linear and non-linear smoothers. Now the Gaussian distributions for the KF (equations (3.5)) can be inserted

$$\begin{aligned} B &= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\Big[\\ &\quad d_x \log[2\pi] + \log[|\Sigma_0|] + (\boldsymbol{x}_1 - \boldsymbol{x}_0)^\mathsf{T}\Sigma_0^{-1}(\boldsymbol{x}_1 - \boldsymbol{x}_0) \\ &+ \sum_{k=2}^{T}\left(d_x \log[2\pi] + \log[|\boldsymbol{Q}|] + (\boldsymbol{x}_k - \boldsymbol{F}\boldsymbol{x}_{k-1})^\mathsf{T}\boldsymbol{Q}^{-1}(\boldsymbol{x}_k - \boldsymbol{F}\boldsymbol{x}_{k-1})\right) \\ &+ \sum_{k=1}^{T}\left(d_z \log[2\pi] + \log[|\boldsymbol{R}|] + (\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)\right)\Big]\mathrm{d}\boldsymbol{x}_{1:T} \end{aligned} \tag{4.10}$$

Beginning with the transition matrix $\boldsymbol{F}$, the bound $\boldsymbol{B}$ can be differentiated with respect to all the components in $\boldsymbol{\theta}$. Note that capital T denotes the length of the time series and $^\mathsf{T}$the transpose

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{F}}B &= \frac{\partial}{\partial \boldsymbol{F}}\Big[-\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\sum_{k=2}^{T}(\boldsymbol{x}_k - \boldsymbol{F}\boldsymbol{x}_{k-1})^\mathsf{T}\boldsymbol{Q}^{-1}(\boldsymbol{x}_k - \boldsymbol{F}\boldsymbol{x}_{k-1})\mathrm{d}\boldsymbol{x}_{1:T}\Big] \\ &= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\sum_{k=2}^{T}(2\boldsymbol{Q}^{-1}\boldsymbol{F}\boldsymbol{x}_{k-1}x_{k-1}^\mathsf{T} - 2\boldsymbol{Q}^{-1}\boldsymbol{x}_k\boldsymbol{x}_{k-1}^\mathsf{T})\mathrm{d}\boldsymbol{x}_{1:T} \\ &= -\boldsymbol{Q}^{-1}\boldsymbol{F}\sum_{k=2}^{T} < x_{k-1}x_{k-1}^\mathsf{T} > + \boldsymbol{Q}^{-1}\sum_{k=2}^{T} < \boldsymbol{x}_k\boldsymbol{x}_{k-1}^\mathsf{T} > \end{aligned} \tag{4.11}$$

Where the expectation of the transition $< \boldsymbol{x}_k\boldsymbol{x}_{k-1}^\mathsf{T} > = < \boldsymbol{x}_k\boldsymbol{x}_{k-1}^\mathsf{T} >_{p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})}$ is needed. Fortunately, this quantity was derived in section 3.1.3. A similar derivation holds for the observation matrix $\boldsymbol{H}$

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{H}}B &= \frac{\partial}{\partial \boldsymbol{H}}\Big[-\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\sum_{k=1}^{T}(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_{1:T}\Big] \\ &= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\sum_{k=1}^{T}(2\boldsymbol{R}^{-1}\boldsymbol{H}\boldsymbol{x}_k x_k^\mathsf{T} - 2\boldsymbol{R}^{-1}\boldsymbol{z}_k\boldsymbol{x}_k^\mathsf{T})\mathrm{d}\boldsymbol{x}_{1:T} \\ &= -\boldsymbol{R}^{-1}\boldsymbol{H}\sum_{k=1}^{T} < x_k x_k^\mathsf{T} > + \boldsymbol{R}^{-1}\sum_{k=1}^{T}\boldsymbol{z}_k < \boldsymbol{x}_k^\mathsf{T} > \end{aligned} \tag{4.12}$$

For the covariances $\boldsymbol{R}$ and $\boldsymbol{Q}$ a few extra calculations are needed since these matrixes are symmetric and positive definite. Differentiating a symmetric matrix is different than a normal matrix. When solving for $\frac{\partial}{\partial \boldsymbol{R}}B = 0$ as is done in the EM algorithm there is no need to take the symmetry into account however, if a gradient based minimization is used it is important to use the correct gradient.

the derivative of a symmetric matrix $\boldsymbol{A}$ can be found by differentiating as usual pretending that the matrix $\boldsymbol{A}^*$ has no structure and then transform the result by: $\frac{\partial F}{\partial \boldsymbol{A}} = (\frac{\partial B}{\partial \boldsymbol{A}*} + \frac{\partial B}{\partial \boldsymbol{A}*}^{\mathsf{T}}) - diag(\frac{\partial F}{\partial \boldsymbol{A}*})$ (Petersen, 2004).
However, this does not ensure that the matrix is symmetric and positive definite after a gradient step. Two different approaches handles this. The first is to keep $\boldsymbol{A}$ diagonal and optimize the deviation σ instead of the variance σ^2. Another approach but with full covariance structure is to write $\boldsymbol{A} = \boldsymbol{A}_0\boldsymbol{A}_0^{\mathsf{T}}$ and optimize $\boldsymbol{A}_0$, this unfortunately introduces new degrees of freedom since another division of $\boldsymbol{A}$ $\boldsymbol{A}_1 = \boldsymbol{A}_0\boldsymbol{U}$ [3] could just as well have been chosen. With this new choice of parameter the derivative is changed but fortunately not much. It can be found by using the chain rule. $\frac{\partial}{\partial \boldsymbol{A}_0}B = (\frac{\partial}{\partial \boldsymbol{A}}B + \frac{\partial}{\partial \boldsymbol{A}}B)\boldsymbol{A}_0$. Using the second method the 'standard' derivative is needed

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{R}}B &= \frac{\partial}{\partial \boldsymbol{R}}\Big[-\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) \\
&\quad \sum_{k=1}^{T}\left(\log[|\boldsymbol{R}|] + (\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)\right)\mathrm{d}\boldsymbol{x}_{1:T}\Big] \\
&= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\sum_{k=1}^{T}(\boldsymbol{R}^{-T} - \boldsymbol{R}^{-T}(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)(\boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{x}_k)^{\mathsf{T}}\boldsymbol{R}^{-T}\mathrm{d}\boldsymbol{x}_{1:T} \\
&= -\frac{1}{2}(T\boldsymbol{R}^{-T} - \boldsymbol{R}^{-T}(\boldsymbol{H}\sum_{k=1}^{T} < \boldsymbol{x}_k\boldsymbol{x}_k^{\mathsf{T}} > \boldsymbol{H}^{\mathsf{T}} + \sum_{k=1}^{T}\boldsymbol{z}_k\boldsymbol{z}_k^{\mathsf{T}} \\
&\quad -\boldsymbol{H}\sum_{k=1}^{T} < \boldsymbol{x}_k > \boldsymbol{z}_k^{\mathsf{T}} - \sum_{k=1}^{T}\boldsymbol{z}_k < \boldsymbol{x}_k >^{\mathsf{T}}\boldsymbol{H}^{\mathsf{T}})\boldsymbol{R}^{-T}
\end{aligned}
\tag{4.13}
$$

In this work the second method with a parameter change is employed.
For the derivative of $\boldsymbol{B}$ with respect to $\boldsymbol{Q}$ the same considerations as for $\boldsymbol{R}$

[3] $\boldsymbol{U}$ can be any rotation matrix

applies since $\boldsymbol{Q}$ is also symmetric

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{Q}*}B &= \frac{\partial}{\partial \boldsymbol{Q}}\Big[-\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) \qquad (4.14)\\
&\quad \sum_{k=2}^{T}\left(\log[|\boldsymbol{Q}|]+(\boldsymbol{x}_k-\boldsymbol{F}\boldsymbol{x}_{k-1})^{\mathsf{T}}\boldsymbol{Q}^{-1}(\boldsymbol{x}_k-\boldsymbol{F}\boldsymbol{x}_{k-1})\right)\mathrm{d}\boldsymbol{x}_{1:T}\Big]\\
&= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\\
&\quad \sum_{k=2}^{T}(\boldsymbol{Q}^{-1}-\boldsymbol{Q}^{-1}(\boldsymbol{x}_k-\boldsymbol{F}\boldsymbol{x}_{k-1})(\boldsymbol{x}_k-\boldsymbol{F}\boldsymbol{x}_{k-1})^{\mathsf{T}}\boldsymbol{Q}^{-1}\mathrm{d}\boldsymbol{x}_{1:T}\\
&= -\frac{1}{2}((T-1)\boldsymbol{Q}^{-1}-\boldsymbol{Q}^{-1}(\boldsymbol{F}\sum_{k=2}^{T}<\boldsymbol{x}_{k-1}\boldsymbol{x}_{k-1}^{\mathsf{T}}>\boldsymbol{F}^{\mathsf{T}}+\sum_{k=2}^{T}<\boldsymbol{x}_k\boldsymbol{x}_k^{\mathsf{T}}>\\
&\quad -\boldsymbol{F}\sum_{k=2}^{T}<\boldsymbol{x}_{k-1}\boldsymbol{x}_k^{\mathsf{T}}>-\sum_{k=2}^{T}<\boldsymbol{x}_k\boldsymbol{x}_{k-1}^{\mathsf{T}}>\boldsymbol{F}^{\mathsf{T}})\boldsymbol{Q}^{-1} \qquad (4.15)
\end{aligned}$$

where the sum $\sum_{k=2}^{T}<\boldsymbol{x}_{k-1}\boldsymbol{x}_{k-1}^{\mathsf{T}}>=\sum_{k=1}^{T-1}<\boldsymbol{x}_k\boldsymbol{x}_k^{\mathsf{T}}>\neq\sum_{k=2}^{T}<\boldsymbol{x}_k\boldsymbol{x}_k^{\mathsf{T}}>$ for finite length sequences. The mean and the covariance for the initial condition can also be found, the estimate of these however, are more uncertain since they only rely on the beginning of sequence

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{x}_0}B &= \frac{\partial}{\partial \boldsymbol{x}_0}\Big[-\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})(\boldsymbol{x}_1-\boldsymbol{x}_0)^{\mathsf{T}}\Sigma_0^{-1}(\boldsymbol{x}_1-\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_{1:T}\Big]\\
&= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})2\Sigma_0^{-1}(\boldsymbol{x}_1-\boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_{1:T}\\
&= -\Sigma_0^{-1}(<\boldsymbol{x}_1>-\boldsymbol{x}_0) \qquad (4.16)
\end{aligned}$$

where $<\boldsymbol{x}_1>=<\boldsymbol{x}_1>_{p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})}$ is the expectation of the first hidden variable over the posterior distribution

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\Sigma}_0}B &= \frac{\partial}{\partial \boldsymbol{\Sigma}_0}\Big[-\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T}) \qquad (4.17)\\
&\quad \left(d_x\log[2\pi]+\log[|\boldsymbol{\Sigma}_0|]+(\boldsymbol{x}_1-\boldsymbol{x}_0)^{\mathsf{T}}\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}_1-\boldsymbol{x}_0)\right)\mathrm{d}\boldsymbol{x}_{1:T}\Big]\\
&= -\frac{1}{2}\int p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})\\
&\quad \left(\boldsymbol{\Sigma}^{-1}-\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_1-\boldsymbol{x}_0)(\boldsymbol{x}_1-\boldsymbol{x}_0)^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\right)\mathrm{d}\boldsymbol{x}_{1:T}\\
&= -\frac{1}{2}\left(\boldsymbol{\Sigma}^{-1}-\right.\\
&\quad \left.\boldsymbol{\Sigma}^{-1}\left(<\boldsymbol{x}_1\boldsymbol{x}_1^{\mathsf{T}}>-<\boldsymbol{x}_1>\boldsymbol{x}_0^{\mathsf{T}}-\boldsymbol{x}_0<\boldsymbol{x}_1>^{\mathsf{T}}+\boldsymbol{x}_0\boldsymbol{x}_0^{\mathsf{T}}\right)\boldsymbol{\Sigma}^{-1}\right)
\end{aligned}$$

with $<\boldsymbol{x}_1\boldsymbol{x}_1^{\mathsf{T}}>=<\boldsymbol{x}_1\boldsymbol{x}_1^{\mathsf{T}}>_{p(\boldsymbol{x}_{1:T}|\boldsymbol{z}_{1:T})}$.

As mentioned earlier the gradients can be used either in the EM-algorithm where the exact update can be calculated by setting them equal to zero, or in a gradient based optimizer. In the following section a comparison of the two methods on three different problems will be carried out.

4.5 EM versus gradients

In Olsson et al. (2005) The EM algorithm and the Easy Gradient Recipe were applied to three different models that can all be fitted into the linear state-space framework, the first is a one dimensional version of the Kalman model and the two others are ICA models.

Mean field ICA

In Independent Component Analysis (ICA), one tries to separate linearly mixed sources using the assumed statistical independence of the sources. In many cases elaborate source priors are necessary, which calls for more advanced separation techniques such as Mean Field ICA. The method, which was first introduced in Højen-Sørensen et al. (2002), can handle complicated source priors in an efficient approximative manner.
The model in equation (3.4) is identical to an instantaneous ICA model provided that $\mathbf{F} = \mathbf{0}$ and that $p(\mathbf{v}_t)$ is reinterpreted as the (non-Gaussian) source prior. The basic generative model of the instantaneous ICA is

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{w}_k \tag{4.18}$$

where $\mathbf{w}_t$ is assumed i.i.d. Gaussian and $\mathbf{x}_k = \mathbf{v}_k$ is assumed distributed by a factorized prior $\prod_i p(v_{ik})$, which is independent in both time and dimension. The Mean Field ICA is only approximately compatible with the Easy Gradient Recipe, since the variational distribution $q(\boldsymbol{x}|\boldsymbol{\phi})$ is not guaranteed to contain the posterior $p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{\theta})$. This, however, is not a problem if q is sufficiently flexible.

Convolutive ICA

Acoustic mixture scenarios are characterized by sound waves emitted by a number of sound sources propagating through the air and arriving at the sensors in delayed and attenuated versions. The instantaneous Mixture Model of standard ICA, equation (4.18), is clearly insufficiently describing this situation. In *convolutive* ICA the signal path (delay and attenuation) is modeled by an FIR filter, i.e. a convolution of the source by the impulse responses of the signal path

$$\mathbf{z}_k = \sum_t \boldsymbol{H}_t \boldsymbol{x}_{k-t} + \boldsymbol{w}_k \tag{4.19}$$

where $\mathbf{H}_t$ is the mixing filter matrix. Equation (4.19) and the source independence assumption can be fitted into the state-space formulation of equation (3.4),

see Olsson and Hansen (2004, 2005), by making the following model choices: 1) Noise inputs $\mathbf{v}_k$ and $\mathbf{w}_k$ are i.i.d. Gaussian. 2) The state vector is *augmented* to contain time-lagged values, i.e.

$$\bar{\mathbf{x}}_k \equiv [x_{1,k} x_{1,k-1} \ldots x_{2,k} x_{2,k-1} \ldots x_{d_s,k} x_{1,k-1} \ldots]^\top$$

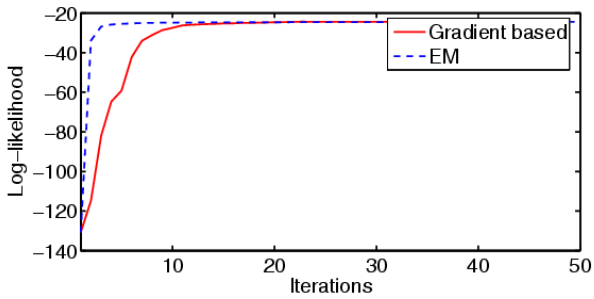
3) state-space parameter matrices (e.g. $\mathbf{F}$) are constrained to a special format (certain elements are fixed to 0's and 1's) in order to ensure the independency of the sources mentioned above.

4.6 Comparing algorithms

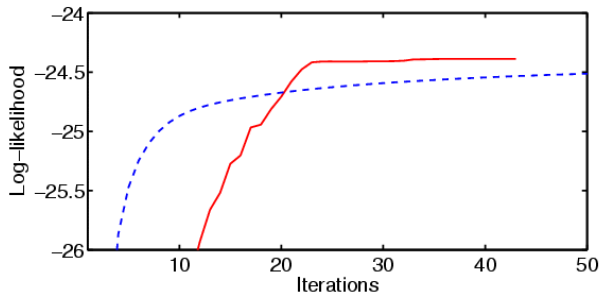
Before advancing to the more involved applications described above, the advantage of gradient-based methods over EM will be explored for a one-dimensional linear State-Space Model. In this case, $\boldsymbol{F}$ and $\boldsymbol{H}$ are scalars as well as the observation variance $\boldsymbol{R}$ and the transition variance $\boldsymbol{Q}$. The latter is fixed to unity to resolve the inherent scale ambiguity of the model. As a consequence, the model has only 3 parameters. The BFGS optimizer mentioned in section 4.3 was used.
Figure 4.3 shows the convergence of both the EM algorithm and the gradient-based method. Initially, EM is fast, i.e. it rapidly approaches the maximum log-likelihood, but slows down as it gets closer to the optimum. The large dynamic range of the log-likelihood makes it difficult to ascertain the final increase in the log-likelihood, hence figure 4.3(b) provides a closeup on the log-likelihood scale. Table 4.1 gives an indication of the importance of the final increase. After 50 iterations, EM has reached a log-likelihood value of -24.5131, but the parameter values are still far off. After convergence, the log-likelihood has increased to -24.3883 which is still slightly worse than that obtained by the gradient-based method, but the parameters are now near the generative values. Similar results are obtained when comparing the learning algorithms on the Kalman Filter Based Sensor Fusion, Mean Field ICA and Convolutive ICA problems.
As argued in section 4.1.1, it is demonstrated that the number of iterations required by the EM algorithm to converge in state-space type models critically depends on the SNR. Figure 4.4 shows the performance of the two methods on the three different problems. The plots indicate that in the low-noise case, the EM algorithm requires relatively more iterations to converge whereas the gradient-based method performs equally well for all noise levels. Note that iterations in the gradient-based approach may require more than one function evaluation. Therefore, function evaluations were counted as iterations.

4.7 Final remarks

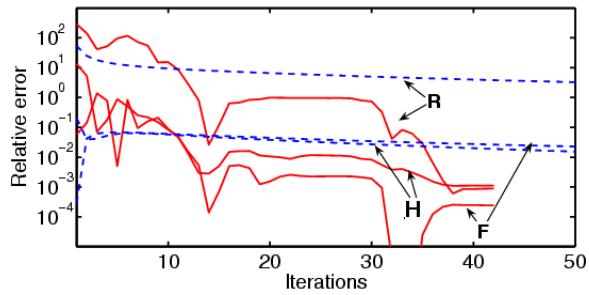
In this chapter, parameter estimation has been investigated; both in general and especially in State-Space Models.



(a) EM has faster initial convergence than the gradient-based method, but the final part is slow for EM.

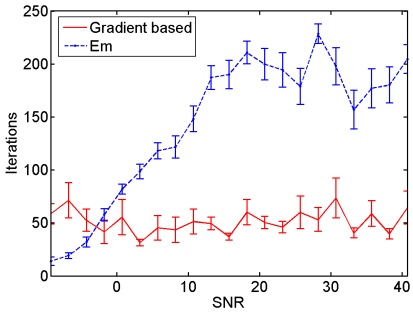


(b) Zoom-in on the log-likelihood axis. Even after 50 iterations, EM has not reached the same level as the gradient based method.

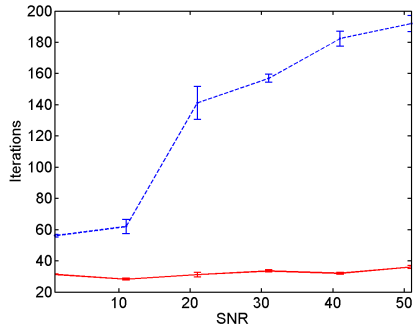


(c) Parameter estimates convergence in terms of squared relative (to the generative parameters) error. The gradient based method has a much faster but not as smooth convergence.

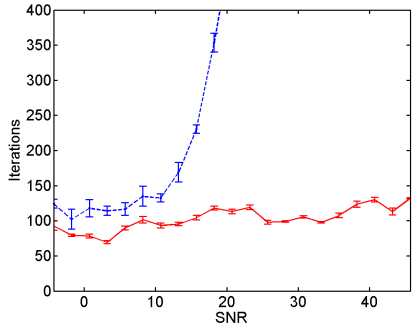
Figure 4.3: Convergence of EM (dashed) and a gradient-based method (dotted) in the linear State-Space Model. Figure 4.2(a) and 4.2(b) are repeated here for convenience. The gradient method converges much faster than the EM algorithm.



(a) Convergence of EM and the gradient based method in the one dimensional Kalman Filter.



(b) Convergence of EM and the gradient based method in Mean Field ICA.



(c) Convergence of EM and the gradient based method in Convolutive ICA.

Figure 4.4: Iterations for EM (dashed) and gradient-based optimization (solid) to reach convergence as a function of signal to noise ratio for the three different problems. Convergence was defined as a relative change in log-likelihood below 10^{-5}. It is seen that in low SNR the EM requires many iterations in all the test cases.

	Generative	Gradient	EM 50	EM ∞
Iterations	-	43	50	1800
Log-likelihood	-	-24.3882	-24.5131	-24.3883
H	0.5000	0.4834	0.5626	0.4859
F	0.3000	0.2953	0.2545	0.2940
R	0.0100	0.0097	0.0282	0.0103

Table 4.1: Estimation in the Kalman Model. The convergence of EM is slow compared to the gradient-based method. Note that after 50 EM iterations, the log-likelihood is relatively close to the value achieved at convergence, but the parameter values are far from the generative values.

It was found that if applying the EM algorithm to maximum likelihood estimation in State-Space Models, it has poor convergence properties in the low noise limit. Often a value 'close' to the maximum likelihood is reached in the first few iterations while the final increase, which is crucial to the accurate estimation of the parameters, requires an excessive amount of iterations.
A simple scheme for efficient gradient-based optimization is achieved by a transformation from the EM formulation. The simple math and programming of the EM algorithms is preserved. Following this scheme – or recipe one can get the optimization benefits associated with *any* advanced gradient based-method. In this way, the tedious and problem-specific analysis of the cost-function topology can be replaced with an off-the-shelf approach. Although the analysis provided in this thesis is limited to a set of linear Mixture Models, it is in fact applicable to *any* model subject to the EM algorithm, hence constituting a strong and general tool to be applied by the part of the machine learning community that uses the EM algorithm.
In the linear Gaussian case, the problem of identifying a unique set of parameters persists. Either the system becomes under-parameterized or a set of structural parameters must be estimated along with the 'real' parameters.
Given the simplicity of the problem it seems like the solution is hiding somewhere. Yet, even though mathematicians, statisticians and control people has been consulted along with piles of literature no final answer has been found.
For general State-Space Models with nonlinear functions and non-Gaussian noise the parameter estimation problem is much harder than in the linear case. If the function family is known, i.e. the functions is parameterized in some way, methods like the ones described in this chapter can be used to find these parameters an example is found in Andrieu and Doucet (2003). If the functional form is not known, a function approximation technique can be used to maximize the likelihood. Valpola and Karhunen (2002) uses neural network and in Roweis and Ghahramani (2001) the nonlinear mappings are modeled by radial basis functions.

Chapter 5

Making Faces

In this chapter, the methods derived in the previous two chapters are applied to the problem stated in the beginning of the thesis – the problem of mapping from speech to images.
Due to the computational advantages the linear Gaussian State-Space Model is chosen for the map. The Gaussian assumption seems reasonable given the distribution of parameters (see section 2.3.3). As for linearity it would be a strange coincidence if the true dynamics where linear, and one could argue that the more advanced models also presented in chapter 3 could be used instead. However, the linear Gaussian model is the simplest and easiest to use, if it works satisfactory everything is fine, if not, it sets the bar for more advanced models.
In the following the framework of using State-Space Models for modality mapping will be presented and the model set up will be explained. The framework has a single parameter that is not trained directly namely the dimension of the unobserved variable space. The optimal size is found and it is shown how the dimension influences what type of dynamics the model can capture. After explaining the setup and the training of the model a section is devoted to a discussion of the pros and cons of this approach to modality mapping. This chapter elaborates on the findings from Lehn-Schiøler (2004a,b); Lehn-Schiøler et al. (2005a).

5.1 Framework

As it was mentioned in chapter 1 the idea of using continuous State-Space Models for the map is in contrast to most previous work where the mapping typically is performed by either a Hidden Markov Model or a Neural Network. Previously such exchange of HMM with Kalman Filters has been proposed for speech recognition by e.g. Digalakis et al. (1993).
The advantage over a discrete approach is that is possible to control the parameters directly without using precalculated trajectories. The advantage over mapping directly from sound to image features is that the temporal aspect is considered. That is, the image sequence will contain only smooth transitions and no jerky motion.
The model is set up as follows

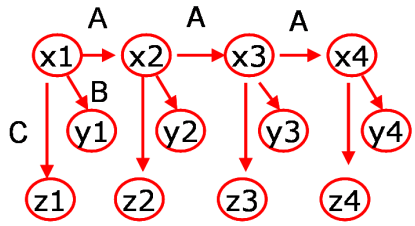
$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{n}_k^x \tag{5.1a}$$
$$\boldsymbol{y}_k = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{n}_k^y \tag{5.1b}$$
$$\boldsymbol{z}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{n}_k^z \tag{5.1c}$$

In this setting $\boldsymbol{z}_k$ represents the image features at time k, $\boldsymbol{y}_k$ represents the sound features and $\boldsymbol{x}_k$ is a hidden variable without direct physical interpretation. The hidden variable can be thought of as some kind of brain activity controlling what is said. Each equation has i.i.d. Gaussian noise component $\boldsymbol{n}$ added to it.

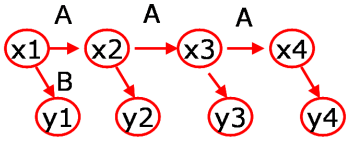
During training both sound and image features are known, and the two observation equations can be collected in one (figure 5.1(a))

$$\begin{pmatrix} \boldsymbol{y}_k \\ \boldsymbol{z}_k \end{pmatrix} = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}_k + \begin{pmatrix} \boldsymbol{n}_k^y \\ \boldsymbol{n}_k^z \end{pmatrix} \tag{5.2}$$

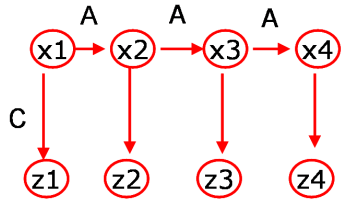
By using the parameter estimation techniques presented in chapter 4 on the training data, all parameters $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{\Sigma}^x, \boldsymbol{\Sigma}^y, \boldsymbol{\Sigma}^z\}$ can be found. As it was also discussed in chapter 4 the solution is not unique, not even with $\boldsymbol{\Sigma}^x = \boldsymbol{I}$.
When a new sound sequence arrives Kalman Filtering (or smoothing) can be applied to equation (5.1a) and equation (5.1b) to obtain the hidden state $\boldsymbol{x}$ (figure 5.1(b)). Given $\boldsymbol{x}$ the corresponding image features can be obtained by the maximum a posteriori estimate , $\boldsymbol{z}_k = \boldsymbol{C}\boldsymbol{x}_k$ (figure 5.1(c)). If the intermediate smoothing variables are available the variance on $\boldsymbol{z}_k$ can also be calculated.
Some inherent difficulties in the mapping becomes apparent when the model is posed as above. As illustrated in Figure 5.1 the hidden sequence is estimated based on the model – trained on both sound and images – and on the sound data. It is not possible to get a perfect mapping simply because not all information about lip movements is contained in the sound. This also relates to the ambiguities in the sound and image representation described in chapter 2.



(a) The Kalman Model with two different observation types.



(b) With known parameters the hidden sequence can be estimated from the sound alone.



(c) Once the hidden states are known the image sequence can be found.

Figure 5.1: The state-space approach to modality mapping. The first step is to estimate parameters of the model, then the hidden sequence can be estimated given only one of the modalities. Finally the second modality can be retrieved by mapping from the hidden sequence.

5.2 Number of hidden units

In most machine learning models there is a trade of between flexibility and generalization, this complication is often referred to as the "bias variance trade off". In Lehn-Schiøler et al. (2005a) it was showed that the bias variance predicament also applies to State-Space Models when deciding how many hidden unit to use. The results were achieved before the investigation of the convergence properties of the EM-algorithm. Therefore, the EM-algorithm was used and many iterations were needed to reach convergence. In fact with the results presented in the previous chapter it is questionable if the optimum was ever reached. However, as the likelihood comes close to the true likelihood after few iterations it is believed that the results are still valid.
The data used for the experiments was a single speaker from the VidTimit database. Nine sentences were used to train the model and a single sentence was used as test.
Figure 5.2 show, as expected, that as the number of hidden units increases the models ability to describe data in the training set also increases. Measuring the likelihood on the test set changes the picture, in figure 5.3 it is seen how adding hidden variables decreases the generalization abilities; overfitting is experienced. With few hidden units the model is not flexible enough and hence the test data can not be described by he model. With too many hidden units the model has been specialized to the training data and again does not describe the test data. When considering Figure 5.2 and figure 5.3 it should be noted that the likelihoods were not normalized and hence the absolute likelihood values cannot be compared between training and test examples.
This kind of analysis is of course heavily dependent on the size of the training set. If the training data really describes all situations well – that is the test set does not contain new information – a well trained model should not experience these problems. Unfortunately, there are never enough data! However, it is still possible to investigate the relationship between the number of training examples and the test error. Such an analysis will give an insight into the optimal training set size.
A type of analysis often used in speech processing is to look at spectrograms. Here spectrograms are used to understand what goes on in the hidden space and an analysis on the feature level rather than in the speech domain is performed. Creating the spectrogram of the hidden variables gives an idea of what dynamics are captured, especially when there are few hidden variables. In figure 5.4(c) a spectrogram from one of the hidden variables in a five variable model can be seen. It is noted that only low frequency components are present. As the hidden space gets larger it becomes possible to model more of the dynamics present in the image. The spectrogram of a representative hidden variable when using a 25 dimensional hidden space (figure 5.4(d)) has a structure very similar to what is found in one of the image features (figure 5.4(a)). When increasing the hidden units to 70, the model degrees of freedom becomes large and over fitting

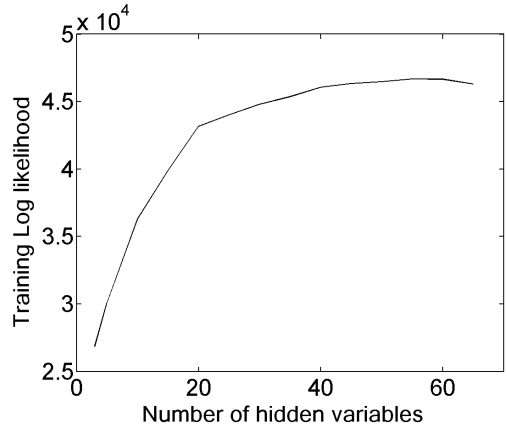


Figure 5.2: The likelihood evaluated on the training data. The more hidden units, the better the model captures the dynamics of the training data.

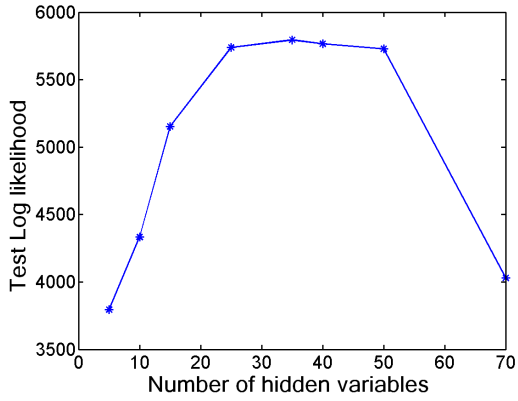
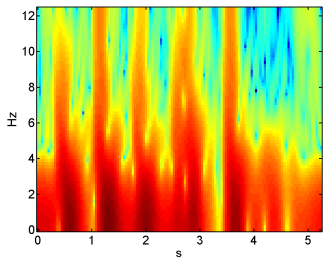
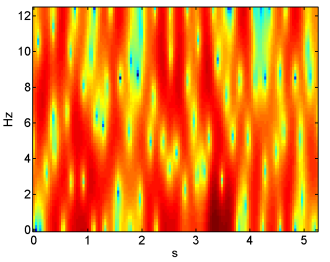


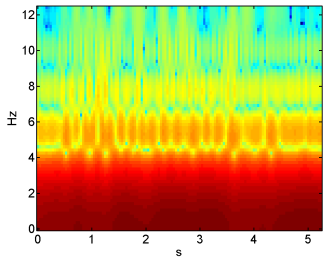
Figure 5.3: The likelihood evaluated on the test data. With few hidden variables the model is not rich enough. With too many parameters over-fitting is experienced. An optimum is found in the range 25-40 hidden variables. Note that due to normalization the test error and the training error can not be compared directly.



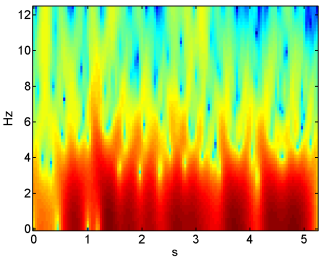
(a) Spectrogram of the first image feature.



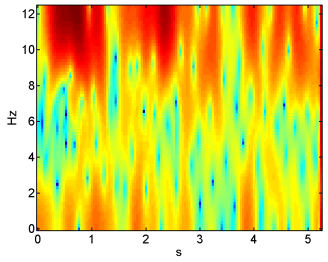
(b) Spectrogram of the first sound feature.



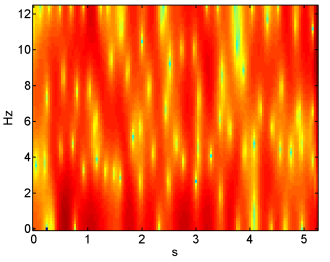
(c) Using five hidden units, only the low frequencies are captured.



(d) When using 25 states, more of the dynamic present in the image can be captured.



(e) When using 70 hidden units one of the components specializes to high frequencies.



(f) Using 70 hidden units one of the components has no direct connection to the image and sound features.

Figure 5.4: In the spectrograms of one of the predicted hidden states of on the test sequence, the effect of varying the size of the state-space can be seen. Spectrograms of the first sound and image features are provided for comparison. Note that the spectrograms are of the features and the states and not the speech signal itself.

becomes possible. Figure 5.4(e) and figure 5.4(f) show the spectrogram of two hidden variables and it is seen that the states specializes. In figure 5.4(e) high frequencies are dominant, and the spectrogram seemingly display a structure, which resembles the dynamics of the sound features as seen in figure 5.4(b). This dynamic is not relevant to the image representation due to the slower dynamics of the facial expressions. These specializations are specific to the training set and do not generalize according to figure 5.3. Given the tendency for the model to overfit to training data and the computational complexity involved in training the model a model with 25 hidden units was chosen for the mapping.

5.3 Does it work?

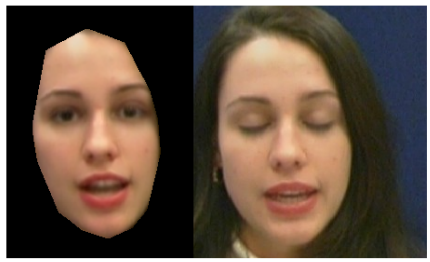
Looking at the strengths and weaknesses of the approach and the ability to create the desired mapping, a quantitative measure of performance is difficult to obtain. One of the only ways to get an idea if the system is usable is to perform subjective tests. The mapping results are not perfect and visual inspection as well as a single test with a hearing impaired person reveals that it is not possible to lip-read from the sequences. Nevertheless, fairly accurate sequences are produced and the both face and articulation looks natural. In figure 5.5 snapshots from the sequence are provided for visual inspection, but as it is the temporal aspect that makes it interesting the reader is referred to http://www.imm.dtu.dk/people/tls/code/facedemo.php for the entire sequence. Other examples can be found in the flash demo http://www.imm.dtu.dk/people/tls/code/flash/Demo_eng.swf. In the following a data-sheet like display of the system is presented, the itemized lists gives an overview of the pros and cons of the system and the accompanying text elaborates on the points.

5.3.1 Status

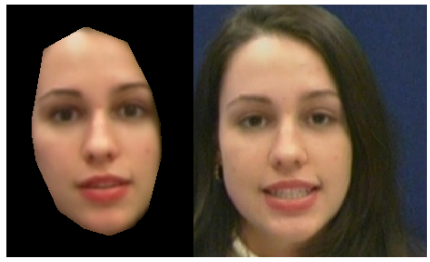
An understanding of how the State Space approach performs and how it relates to other similar approaches can be gained by examining some characteristics of the system:

- The video show natural facial animation.
- It is not possible to lipread from the video.
- It is easy to the train model for new persons.
- It is easy to train the model for new languages.
- The video has smooth transitions and no problems with jerkiness.

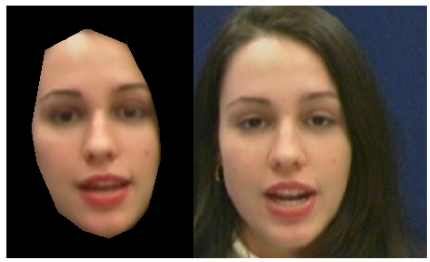
The produced sequences are natural to look at, and at a first glance it is not possible to distinguish them from form real recordings. However, once the lip-movements are inspected closely it can be seen that it is not possible to lip



(a)



(b)



(c)

Figure 5.5: Characteristic images taken from the test sequence. The predicted face is to the left and the true face to the right.

read from the sequences. There are systems that outperform this one in terms of lipreading abilities e.g. Ezzat et al. (2002) and Cohen and Massaro (2002). These systems are based on a phonetic alphabet, and hence requires a phonetic transcription of the spoken sentences.
It is no surprise that a phoneme to image mapping produces better results than a sound to image mapping, just think of the ambiguities revealed by the McGurk effect. The lipreading ability of the phonetic based systems indicates that, as speech recognition gets better phoneme representations are a more natural way to go when creating visual speech. However, besides the difficulties in transcribing speech correctly a main drawback with the phonetic approach is that there are many sounds that are not present in the phonetic alphabet. Yawning, sneezing, and 'hmm' sounds does not have a phonetic transcription. Further more, the construction of models in different languages requires different phonetic alphabets and hence extraction of different key frames.
In contrast to this, the approach proposed in this work requires only that the training data contains the desired sounds and the matching movements. The speech can be in any language and non-speech sounds are mapped as well as speech sounds. One of the greatest strength of this of mapping is the easy set up for new conditions. Creating a system for a new person, a new vocabulary or an entire new language is as easy as collecting video of the desired condition. Then it is simply extracting the features and creating the map, both of which can be set up to run automatically.
Finally, unlike other approaches the use of continuous State-Space Models ensures that the video is smooth, there are no problems with jerky motion or unnatural transitions.

Possible uses:

As discussed later there is a range of possibilities for improving the system, but even with the drawback that lipreading is not possible there are still applications where cosmetic corrections or generation of lip-movements can be used:

- Low bandwidth transmission for mobile phones and video conferences (lip reading accuracy not important).
- Correction of lip-movements in synchronized movies and commercials.
- Rough animation in cartoons (could be fine tuned by hand afterwards).
- In-game dialog.

In low-bandwidth communication it is important to get a sense of presence often on a relatively low resolution screen. Lip-movements that are time-aligned with the speech is an important factor for this. Even with a higher resolution temporal accuracy is more important than entirely correct movements. The same holds for synchronized movies, removing the most obvious miss-matches

between would improve the illusion of language change greatly. In computer games more and more in game dialog are entering, the dialog is part of the game play and even though the speech is prerecorded, lip-movements can be generated when they are needed.

Problems (and fixes):

Besides the fact that the same information is not present in the sound and image domains there are a number of other reasons that the mapping is not perfect:

- The training corpus is to small.
- The system is speaker dependent.
- The model is of entire face, not only the mouth.
- The MFCC features are probably not optimal.
- The lip movements have to small amplitude.
- The model is Linear and Gaussian.

The most prominent reason for the deficiencies in the mapping is that the training corpus is to small, the VidTimit database contain only ten examples leaving at most nine to train the system. The homemade recordings contain more examples but, still far from the 20 min. used in Ezzat et al. (2002). To improve performance a more complete training set needs to be created. Increasing the amount of data is a two edged sword. The computations are already quite time consuming as it is, and increasing the training set would only make this worse. An alternative way to get more training data is to train the system on several different speakers. The assumption would be that most of the variation is governed by the speech and that only a relatively small fraction of the face movements where person specific. Even if the face movements are highly correlated with the specific speaker a large enough number of speakers would ensure that these inter person variabilities where handled as noise. Preliminary experiment with this approach however, did not produce viable results. Trying to understand why, two main obstacles comes to mind. Firstly, the approach taken in this work models the entire face and not just the mouth. As described elsewhere, the reason for this is that a free floating mouth is very unnatural to look at. However, when trying to model several persons at a time the appearance of the rest of the face becomes an important factor. A conceptual simple but time consuming modification to the AAM, would allow a hierarchical model, where the mouth was model by itself and then pasted back onto the face. Such a hierarchical scheme would not only help in the multi-person case, but, it would also reduce the Kalman Model complexity. Such a model would also allow a separate control of the eyes, giving the possible to blink and thereby add realism.

The second factor that could also improve both the single person and multi person mapping is better sound features, even though MFCC's are used in speaker independent speech recognition they are not complectly unsensitive to the speaker.
A final problem with the mapping approach is that the lip movements typically are too small. The mouth simply does not open enough. An explanation could be that the Kalman Model favors small changes in the states. The Gaussian noise model gives large probability to small deviation from the previous and the next state. This helps ensure that the transitions are smooth, but it might also be to strict a requirement that makes fast transitions impossible. A solution could be to use a model with a broader noise distribution, or perhaps a model with nonlinear transitions.

Extensions

In the above section a list of possible ways to improve the system were presented, the improvements are based on weaknesses in the mapping and data gathering. There are also a couple of possibilities for extending the system based on the strengths:

- Emotions.
- Animating other peoples faces.

One of the most interesting extensions is perhaps to estimate emotions from speech. Emotion estimation in itself is a large research field, a good introduction is given in Cowie et al. (2001). A mapping from emotions to facial parameters could then be trained and the facial expression could be adjusted to fit the emotion. Such a system will rely on emotional expressions being in some sense orthogonal to what is being said, the emotions should provide a 'background face' where on the lip movements could be superimposed.
Another possibility would be to utilize the underlying MPEG-4 control points. In the mapping, as it is presented here, focus has been on the AAM-parameters that where extracted from the video and later reproduced by the mapping. However, from these parameters it is possible to find the MPEG control points simply by using the shape part of the AAM. Once the MPEG-control points are used it is possible to animate avatars using a MPEG-4 based model of a head e.g. as the one presented in Fedorov et al. (2003).

5.4 A noninformative face

In other parts of this thesis the goal has been to make the mapping from speech to images as accurate as possible. But, the set up has also been used in another context. Numerous experiments have shown that vision can aid in understanding

speech in noisy environments, this ability has also been used to build multimodal speech reading machines. The lip-movements carry a high degree of information about was is being said. However, little is known about how much the awareness that something is being said is utilized. It is possible that humans has the ability to use vision as a primer for hearing.
Vision can be used to determine when the interesting sounds are being heard and when the sounds are just background noise. Looking at someone's lips, gives a quite good indication of when they are speaking. If we know when the interesting person is speaking it is possible to build a model of the background noise when they are not. Having a model of the noise can perhaps aid the speech recognition. To investigate this hypothesis it is necessary to create a face that moves when something is being said, but, where the lip movements contain no (or little) information about the content of the sentence.
If this hypothesis is true it is possible that a similar approach could be used to improve speech separation algorithms. That is find some visual or other kind of indicators of when the interesting sounds are being heard, build a model of noise when they are not, use the noise model to separate the signal out.
By using the framework described in the beginning of this chapter it is possible to create a noninformative talking face. The only modification it requires is to exchange the hopefully very informative MFCC features with a noninformative feature like the power of the signal. Since the mapping works best with low dynamic range of the input the logarithm of the power can be used instead of the power. The sequences that are produced gives an impression that the mouth movements and the speech are coupled even though the only movement is lip-opening and closing. It is interesting to note that in the early attempts to create talking faces it was exactly the power of the speech that was used.
Currently the mappings described above are used in an investigation performed by Lucas C. Parra and co-workers at the City College of New York to clarify if the 'noninformative' images helps in speech understanding.

5.5 Final remarks

In this chapter, the linear State-Space Model is used to map from speech to images, and it is demonstrated how the optimal size of the hidden space can be estimated. The results presented in this chapter cover merely a fraction of all the experiments that have been performed with the setup. Even though the model in itself has only a single parameter that can be tuned, a range of other things can cause problems. Each time when they do, a new set of experiments must be set up to reveal why. Below is a list of some interesting findings that are observed during the project but have not been fully examined/documented due to the computational effort required for each of them:

- Slow convergence is a real problem. In the setting presented here, it is not uncommon to require 100.000 iterations for the EM algorithm to converge

and even the gradient approach requires thousands of iterations.

- Increasing the number of training examples does increase performance on the test set.
- The optimal dimension of the hidden state depends on the size of the training set.
- Often computational difficulties arises when the variance in some directions becomes much smaller that in others.

In conclusion: Based on State Space Modeling, a novel method for mapping from speech to images has been proposed. The method is fast and easy trainable for new persons and even for new languages. Even though the method is still at the research level, the photo realistic and natural talking faces can be applied to a wide range of real applications including synchronization of movies, computer games, and video telephony.

Chapter 6

Conclusion

In this thesis, focus is on the application of State-Space Models to modality mapping. It is proven that it is possible to produce image sequences that are natural to look at given a speech input. However, what is novel, is not the fact that such a mapping can be produced. The novelty is rather the usage of continuous State-Space Models along with a parametric representation of the face.
Work in three main directions is presented: Work done with an information theoretic approach to signal processing leading to a vector quantization algorithm, work done on the general State-Space Model and, finally, work done on applying the State-Space Model to mapping from speech to images. The main contribution of this thesis lies in the examination of State-Space Models but, small contributions have also been presented to the information theoretic research.

An alternative algorithm for vector quantization, the *VQIT*, has been derived. The algorithm provides a new way of selecting a compact representation of a data set. Like other vector quantization schemes, this algorithm can be used to compress data for storage or transmission, or it can be used to discretize a data set e.g. to make it possible to use a Hidden Markov Model afterwards.
The VQIT algorithm is derived based on concepts from information theoretic learning, and it is shown how potential fields and Parzen estimators can be used to give a physical interpretation of vector quantization. A set of Processing Elements (PEs) are to be placed optimally in relation to the data set. Both PEs and data points are considered to be information particles with associated

potentials. When releasing the PEs in the potential field, they tend towards an energy minimizing configuration. The VQIT algorithm is compared to conventional algorithms and performs equivalently. The primary novelty is that this algorithm utilizes a cost-function and its derivative to perform vector quantization.

The general State-Space Model has been described and treated in some detail, and, especially the simplest case of linear Gaussian filters has been examined and applied to the specific problem of modality mapping. Working with the general State-Space Model has led to some new ideas about filtering. By examining the class of non-linear sequential approaches, a new member, the Parzen Particle Filter, is introduced into the family of Particle Filtering algorithms. Inspired by information theory the idea of a particle as a point-shaped entity is extended, and kernels are used to increase the volume covered by a particle. The introduction of kernels with non-zero width sacrifices some of the nice computational properties of the Particle Filter in return for increased information transfer. It is demonstrated that by using the Parzen Particle Filter method, filtering can be performed with a smaller number of particles than with the standard approach.
Continuing with the general State-Space Model but entering the realm of Markov Chain Monte Carlo (MCMC) sampling methods, it is demonstrated how MCMC often proves to be superior to the sequential (Particle Filter) methods. A scheme is supplied in which one can apply MCMC methods online as data arrive and at the same time benefit from the properties of the chain. By including the history in sampling, it becomes easier to overcome basin changes since evidence for the new basin can be gathered over time.
During the investigations and use of the linear State-Space Model, the poor convergence properties of the EM-algorithm turned out to be a problem – especially in the low-noise limit. Even though values 'close' to the optimal likelihood are reached in few iterations, it is the final small changes in likelihood that ensure convergence of the parameters. To compensate for this, an alternative way of using the gradient of the lower bound function is proposed, it is termed the *Easy Gradient Recipe*. Following this recipe, one can get the optimization benefits associated with *any* advanced gradient based-method. In this way, the tedious problem-specific analysis of the cost-function topology can be replaced with an off-the-shelf approach. The gradient alternative can be used in all cases where the likelihood and the gradient of the lower bound can be calculated; that is, in most of the cases where the EM-algorithm is applied in machine learning.

One of the great strengths of State-Space Models is the ability to model data that evolve in time. This ability makes it an obvious choice when dealing with signals where the temporal aspect is of importance. Video sequences are examples of such data. The spatio temporal capacity of the model has been utilized by applying the linear version of the State-Space Model to mapping from speech

to images.
It is demonstrated that State-Space Models are indeed able to perform this task; they provide a fast on-line way of generating photo realistic talking faces that can be used in a wide range of real life applications.

Appendix A

Timeline

Time line of the advances in speech to video mapping. The criteria for gathering this list was that the contribution should either be in on the exact topic (that is, no audio-visual lip-reading etc.) or it should be of great importance to the field. Even though an effort has been made to cover as much as possible there are almost with certainty important contributors that are not been mentioned. Apologies to them. For a similar, slightly outdated, time-line gathered by Philip Rubin and Eric Vatikiotis-Bateson see http://www.haskins.yale.edu/haskins/HEADS/BIBLIOGRAPHY/biblioface.html.

Year	Event	Reference
1954	Importance of Visual information for speech perception	Sumby and Pollack
1975	First animation scheme for talking faces	Parke
1976	The McGurk Effect	McGurk and MacDonald
1985	Controlling facial expressions in cartoons	Bergeron and Lachapelle
1986	Computer graphics model of face animation	Pearce et al.
1987	Psychology of lip-reading	Dodd and Campbell
1987	Automated lip sync	Lewis and Parke
1988	Animating speech	Hill et al.

Table A.1: Time line of the advances in speech to video mapping.

Year	Event	Reference
1989	Phoneme driven facial animation	Morishima et al.
1990	Early review of talking faces	Massaro and Cohen
1991	Automated lip-sync	Lewis
1991	Conversion from speech to facial images	Morishima and Harashima
1992	Neural network for lip-reading	Stork et al.
1994	How talking faces can be used in physiological experiments	Cohen and Massaro
1994	Quality of talking faces	Goff et al.
1995	Lip sync from speech	Chen et al.
1995	Talking faces over the telephone	Lavagetto
1996	Face features for speech reading	Petajan and Graf
1997	Video rewrite, shuffling video to match new speech	Bregler et al.
1997	Speech driven synthesis of talking head sequences (neural network, MPEG)	Eisert et al.
1997	Driving synthetic mouth gestures using phonemes	Goldenthal et al.
1997	time-delay neural networks for estimating lip movements from speech analysis	Lavagetto
1997	Acoustic driven viseme identification for face animation	Zhong et al.
1998	Psychology of lip-reading (new edition)	Campbell et al.
1998	A computer graphics talking head (Mike)	DeGraph and Wahrman
1998	Mike talk, based on morphable models	Ezzat and Poggio
1998	Conversion of articulatory parameters into active shape model coefficients	Lepsoy and Curinga
1998	Psychological view on sensory integration	Massaro and Stork
1998	Active shape model for visual speech recognition	Matthews et al.
1998	Fourier based Lip-Sync	McAllister et al.
1998	Lip movement Synthesis from Speech based on HMMs	Yamamoto et al.
1999	Voice puppetry, based on Coupled HMMs	Brand
1999	User evaluation of talking faces	Pandzic et al.
2000	Lip sync using linear Predictive Analysis	Kshirsagar and Magnenat-Thalmann

Table A.1: Time line of the advances in speech to video mapping.

Year	Event	Reference
2000	Talking faces using phonemes and RBFs	Noh and Neumann
2000	Visual speech processing based on MPEG-4	Petajan
2000	HMM based lip synthesis	Tokuda et al.
2000	Neural network for talking faces	Vatikiotis-Bateson et al.
2001	Mixture of Gaussian and HMM approach to talking faces	Chen
2001	Phoneme and MPEG4 based talking face	Goto et al.
2001	Neural network for talking faces	Kakumanu et al.
2001	Study of difference between /n/ and /m/	Taylor et al.
2002	MPEG4 HMM approach to talking faces	Aleksic et al.
2002	Modeling Facial Behaviors	Bettinger et al.
2002	Talking head (Baldi)	Cohen and Massaro
2002	AAM neural network talking faces	Du and Lin
2002	Mike talk, based on morphable models	Ezzat et al.
2002	MPEG4 HMM approach to talking faces	Hong et al.
2002	HMM based lip synthesis	Nakamura
2002	A HMM based speech to video synthesizer	Williams and Katsaggelos
2003	Different types of HMM used for talking faces	Aleksic and Katsaggelos
2003	PhD Thesis on Talking Faces	Beskow
2003	Perceptual Evaluation of Video Realistic Speech	Geiger et al.
2003	Real Time Speech driven Face Animation	Hong et al.
2003	Visual Speech	Kalberer et al.
2003	Synface talking faces for the telephone	Karlsson et al.
2003	AAM face animation	Theobald et al.
2003	HMM based lip synthesis	Verma et al.
2004	State space approach to talking faces	Lehn-Schiøler et al.
2004	HMM based lip synthesis	Aleksic and Katsaggelos
2004	AAM talking face phonemes	Theobald et al.
2004	AAM based hierarchical talking face	Cosker et al.
2004	3D talking faces	Ypsilos et al.

Table A.1: Time line of the advances in speech to video mapping.

Appendix B

Gaussian calculations

B.1 Variable change

When multiplying two Gaussians it is useful to have them in the same space

$$N_{\boldsymbol{a}}(\boldsymbol{F}\boldsymbol{\mu}, \boldsymbol{A}) \tag{B.1a}$$

Can be rewritten using $\boldsymbol{a} - \boldsymbol{F}\boldsymbol{\mu} = \boldsymbol{F}(\boldsymbol{F}^{-1}\boldsymbol{a} - \boldsymbol{\mu})$

$$(\boldsymbol{a} - \boldsymbol{F}\boldsymbol{\mu})^T \boldsymbol{A}^{-1}(\boldsymbol{a} - \boldsymbol{F}\boldsymbol{\mu}) = (\boldsymbol{F}^{-1}\boldsymbol{a} - \boldsymbol{\mu})^T \boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F}(\boldsymbol{F}^{-1}\boldsymbol{a} - \boldsymbol{\mu}) \tag{B.2a}$$

$$N_{\boldsymbol{a}}(\boldsymbol{F}\boldsymbol{\mu}, \boldsymbol{A}) = N_{\boldsymbol{F}\boldsymbol{\mu}}(\boldsymbol{a}, \boldsymbol{A}) = N_{\boldsymbol{\mu}}(\boldsymbol{F}^{-1}\boldsymbol{a}, (\boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F})^{-1}) C \tag{B.2b}$$

Where $C = (\sqrt{\frac{|\boldsymbol{A}|}{|(\boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F})^{-1}|}})$ is a constant that takes care of the normalization. If $\boldsymbol{F}$ is not square a pseudo inverse can be used. (Tall $\boldsymbol{F}$ might give problems).

B.2 Multiplication of Gaussians

Let

$$N_{\boldsymbol{x}}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})) \tag{B.3}$$

Then

$$N_{\boldsymbol{x}}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) N_{\boldsymbol{x}}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = C N_{\boldsymbol{x}}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{B.4a}$$

$$C = N_{\boldsymbol{\mu}_2}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1 + \Sigma_2) \tag{B.4b}$$

$$\boldsymbol{\mu}_c = (\boldsymbol{\Sigma}_1^{-1} + \Sigma_2^{-1})^{-1}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \Sigma_2^{-1}\boldsymbol{\mu}_2) \tag{B.4c}$$

$$\boldsymbol{\Sigma}_c = (\boldsymbol{\Sigma}_1^{-1} + \Sigma_2^{-1}) \tag{B.4d}$$

see eg. Petersen (2004)

B.3 Integrating a special Gaussian

For comparisons with section 3.1 $\boldsymbol{x}_{k-1}^{k-1} = \boldsymbol{b}, \boldsymbol{x}_{k-1} = \boldsymbol{\mu}, \boldsymbol{x}_k = \boldsymbol{a}, \boldsymbol{A} = \boldsymbol{Q}, \boldsymbol{B} = \boldsymbol{P}_{k-1}^{k-1}$. Consider

$$\int N_{\boldsymbol{a}}(\boldsymbol{F}\boldsymbol{\mu}, \boldsymbol{A}) N_{\boldsymbol{\mu}}(\boldsymbol{b}, \boldsymbol{B}) \mathrm{d}\boldsymbol{\mu} = \int N_{\boldsymbol{F}\boldsymbol{\mu}}(\boldsymbol{a}, \boldsymbol{A}) N_{\boldsymbol{\mu}}(\boldsymbol{b}, \boldsymbol{B}) \mathrm{d}\boldsymbol{\mu} \tag{B.5a}$$

Changing variables as in equation B.2 and using rules for multiplication of two multidimensional Gaussian (equation B.4) the integral becomes

$$C \int N_{\boldsymbol{\mu}}(\boldsymbol{F}^{-1}\boldsymbol{a}, (\boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F})^{-1}) N_{\boldsymbol{\mu}}(\boldsymbol{b}, \boldsymbol{B}) \mathrm{d}\boldsymbol{\mu} \tag{B.6a}$$

$$= C N_b(\boldsymbol{F}^{-1}\boldsymbol{a}, \boldsymbol{B} + (\boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F})^{-1}) \int N_{\boldsymbol{\mu}}(\ldots, \ldots) \mathrm{d}\boldsymbol{\mu} \tag{B.6b}$$

The mean and covariance for the Gaussian $N_{\boldsymbol{\mu}}$ is not important, since integrating over $\boldsymbol{\mu}$ yields 1 anyway. After the integration the Gaussian $N_b(\boldsymbol{F}^{-1}\boldsymbol{a}, \boldsymbol{B} + (\boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F})^{-1})$ and the constant $\boldsymbol{C}$ remain.
This is not the desired form, and the expression can be rearranged to

$$N_b(\boldsymbol{F}^{-1}\boldsymbol{a}, \boldsymbol{B} + (\boldsymbol{F}^T \boldsymbol{A}^{-1} \boldsymbol{F})^{-1}) = N_a(\boldsymbol{F}\boldsymbol{b}, \boldsymbol{F}\boldsymbol{B}\boldsymbol{F}^T + \boldsymbol{A}) \tag{B.7a}$$

Note that the result of the integration is a scaled Gaussian. However, in the Kalman calculation the scaling factor is not of interest.

B.4 Some matrix identities

the Woodbury identity

$$(\boldsymbol{A}^{-1} + \boldsymbol{B}^T \boldsymbol{C}^{-1} \boldsymbol{B})^{-1} = \boldsymbol{A} - \boldsymbol{A}\boldsymbol{B}^T (\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{C})^{-1} \boldsymbol{B}\boldsymbol{A} \tag{B.8}$$

the following equality holds for $\boldsymbol{A}$ and $\boldsymbol{C}$ positive definite

$$(\boldsymbol{A}^{-1} + \boldsymbol{B}^T \boldsymbol{C}^{-1} \boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{C}^{-1} = \boldsymbol{A}\boldsymbol{B}^T (\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{C})^{-1} \tag{B.9}$$

Appendix C

Publications

On the following pages the papers that has been produces during the past three years are reproduced.

Vector-Quantization using Information Theoretic Concepts

Tue Lehn-Schiøler (tls@imm.dtu.dk)
Intelligent Signal Processing, Informatics and Mathematical Modelling, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

Anant Hegde (ahegde@cnel.ufl.edu), Deniz Erdogmus (deniz@cnel.ufl.edu) and Jose C. Principe (principe@cnel.ufl.edu)
Computational NeuroEngineering Laboratory, Electrical & Computer Engineering Department, University of Florida, Gainesville, FL 32611, USA

Abstract. The process of representing a large data set with a smaller number of vectors in the best possible way, also known as vector quantization, has been intensively studied in the recent years. Very efficient algorithms like the Kohonen Self Organizing Map (SOM) and the Linde Buzo Gray (LBG) algorithm have been devised. In this paper a physical approach to the problem is taken, and it is shown that by considering the processing elements as points moving in a potential field an algorithm equally efficient as the before mentioned can be derived. Unlike SOM and LBG this algorithm has a clear physical interpretation and relies on minimization of a well defined cost-function. It is also shown how the potential field approach can be linked to information theory by use of the Parzen density estimator. In the light of information theory it becomes clear that minimizing the free energy of the system is in fact equivalent to minimizing a divergence measure between the distribution of the data and the distribution of the processing element, hence, the algorithm can be seen as a density matching method.

Keywords: Information particles, Information theoretic learning, Parzen density estimate, Self organizing map, Vector-Quantization

Abbreviations: SOM – Self-organized map; PE – Processing element; C-S – Cauchy-Schwartz; K-L – Kullback-Leibler; VQIT – Vector-Quantization using Information Theoretic Concepts; QE – Quantization error LBG – Linde Buzo Gray

1. Introduction

The idea of representing a large data set with a smaller set of processing elements (PE's) has been approached in a number of ways and for various reasons. Reducing the number of data points is vital for computation when working with a large amount of data whether the goal is to compress data for transmission or storage purposes, or to apply a computationally intensive algorithm.

In vector quantization, a set of data vectors is represented by a smaller set of code vectors, thus requiring only the code vector to be stored or transmitted. Data points are associated with the nearest code vector generating a lossy compression of the data set. The challenge is

to find the set of code vectors (the code book) that describes data in the most efficient way. A wide range of vector quantization algorithms exist, the most extensively used are K-means (MacQueen, 1967) and LBG (Linde et al., 1980).

For other applications like visualization, a good code book is not enough. The 'code vectors', or processing elements (PE's), as they are often denoted in the self-organizing literature, must preserve some predefined relationship with their neighbors. This is achieved by incorporating competition and cooperation (soft-competition) between the PE's. Algorithms with this property create what is known as Topology Preserving Maps. The Self-Organized Map (SOM) (Kohonen, 1982) is the most famous of these. It updates not only the processing element closest to a particular data point, but also its neighbors in the topology. By doing this it aligns the PE's to the data and at the same time draws neighboring PE's together. The algorithm has the ability to 'unfold' a topology while approximating the density of the data.

It has been shown (Erwin et al., 1992) that when the SOM has converged, it is at the minimum of a cost function. This cost-function is highly discontinuous and drastically changes if any sample changes its best matching PE. As a result it is not possible to use the conventional methods to optimize and analyze it. Further more, the cost function is not defined for a continuous distribution of input points since boundaries exist where a sample could equally be assigned to two different PE's (Erwin et al., 1992). Attempts has been made to find a cost function that, when minimized, gives results similar to the original update rule (Heskes and Kappen, 1993).

Efforts have also been made to use information theoretic learning to find good vector quantifiers and algorithms for Topology Preserving Maps. Heskes (1999) introduces a cost function as a free energy functional consisting of two parts, the quantization error and the entropy of the distribution of the PE's. He also explored the links between SOM, vector quantization, Elastic nets (Durbin and Willshaw, 1987) and Mixture modeling, concluding that the methods are closely linked via the free energy. Van Hulle (2002) uses an information theoretic approach to achieve self-organization. The learning rule adapts the mean and variance of Gaussian kernels to maximize differential entropy. This approach, however, leads to a trivial solution where PE's eventually coincide. To circumvent this, Van Hulle proposes to maximize the differential entropy and at the same time minimize the mutual information by introducing competition between the kernels. The competition is not based on information theory but rather implements an activity-based, winner-takes all heuristic. Bishop et al. (1996) proposes an algorithm

(the Generative Topographic Map) in which a mapping between a lattice of PE's and data space is trained using the EM algorithm.

Ideas on interactions between energy particles have been explored previously by Scofield (1988). However, in this paper, we approach the same problem with an information-theory perspective and explicitly use the probability distributions of the particles to minimize the free energy between them.

In this paper, an algorithm for vector quantization using information theoretic learning (VQIT) is introduced. Unlike the methods described above, this algorithm is designed to take the distribution of the data explicitly into account. This is done by matching the distribution of the PE's with the distribution of the data. This approach leads to the minimization of a well defined cost function. The central idea is to minimize the free energy of an information potential function. It is shown that minimizing free energy is equivalent to minimizing the divergence between a Parzen estimator of the PE's density distributions and a Parzen estimator of the data distribution. In section 2, an energy interpretation of the problem is presented and it is shown how this has close links to information theory. In section 3, the learning algorithm is derived using the Cauchy-Schwartz inequality. Numerical results are presented in section 4, where performance is evaluated on an artificial data set. In section 5 limitations and possible extensions to the algorithm are discussed and it is compared to already existing algorithms. Finally, concluding remarks are given in section 6.

2. Energy interpretation

The task is to choose locations for the PE's, so that they represent a larger set of data points as efficiently as possible. Consider two kind of particles; each kind has a potential field associated with it, but the polarity of the potentials are opposite. One set of particles (the data points) occupies fixed locations in space while the other set (the PE's) are free to move. The PE's will move according to the force exerted on them by data points and other PE's, eventually minimizing the free energy. The attracting force from data will ensure that the PE's are located near the data-points and repulsion between PE's will ensure diversity.

The potential field created by a single particle can be described by a kernel of the form $K(\cdot)$. Placing a kernel on each particle, the potential

energy at a point in space $\mathbf{x}$ is given by

$$p(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{K}(\mathbf{x} - \mathbf{x_i}) \tag{1}$$

where the index i runs over the positions of all particles ($\mathbf{x_i}$) of a particular charge. If the kernel decays with distance ($K(x) \propto \frac{1}{(\mathbf{x}-\mathbf{x_i})}$) the potential is equivalent to physical potentials like gravitation and electric fields. However, in the information theoretic approach, any symmetric kernel with maximum at the center can be chosen. For the sake of simplicity, Gaussian kernels are used herein.
Due to the two different particle types, the energy of the system has contributions from three terms:

1. Interactions between the data points; since the data points are fixed, these interactions will not influence minimization of the energy.

2. Interactions between the data and the processing elements; due to the opposite signs of the potentials, these particles will attract each other and hence maximize correlation between the distribution of data and the distribution of PE's.

3. Interactions between PE's; the same sign of all the PE's potentials causes them to repel each other.

In the information theoretic literature equation (1) is also considered a density estimator. In fact it is exactly the well known Parzen density estimator (Parzen, 1962). In order to match the PE's with the data, we can use equation (1) to estimate their densities and then minimize the divergence between the densities. The distribution of the data points (x_i) can be written as $f(x) = \sum_i G(x - x_i, \sigma_f)$ and the distribution over PE's (w_i) as $g(x) = \sum_i G(x - w_i, \sigma_g)$.

Numerous divergence measures exist, of which the Kullback-Leibler (K-L) divergence is the most commonly used (Kullback and Leibler, 1951). The Integrated square error and the Cauchy-Schwartz (C-S) inequality, are both linear approximations to the K-L divergence. If C-S is used, the link between divergence and energy interpretation becomes evident. The Cauchy-Schwartz inequality,

$$|ab| \leq ||a||||b|| \tag{2}$$

is an equality only when vectors a and b are collinear. Hence, maximizing $\frac{|ab|}{||a||||b||}$ is equivalent to minimizing the divergence between a and b. To remove the division, the logarithm can be maximized instead. This

is valid since the logarithm is a monotonically increasing function. In order to minimize the divergence between the distributions $f(x)$ and $g(x)$ the following expression is minimized:

$$D_{c-s}(f(x), g(x)) = -\log \frac{(\int (f(x)g(x))\mathrm{d}x)^2}{\int f^2(x)\mathrm{d}x \int g^2(x)\mathrm{d}x} \tag{3}$$
$$= \log \int f^2(x)\mathrm{d}x - 2\log \int f(x)g(x)\mathrm{d}x + \log \int g^2(x)\mathrm{d}x$$

Following Principe et al. (2000) $V = \int g^2(x)\mathrm{d}x$ is denoted as the information potential of the PE's and $C = \int f(x)g(x)\mathrm{d}x$ the cross information potential between the distributions of data and the PE's. Note that

$$H(x) = -\log \int g^2(x)\mathrm{d}x = -logV \tag{4}$$

is exactly the Renyi quadratic entropy (Rényi, 1970) of the PE's. As a result, minimizing the divergence between f and g is equal to maximizing the sum of the entropy of the PE's and the cross information potential between the densities of the PE's and the data. The link between equation (3) and the energy formulation can be seen by comparing the terms with the items in the list above.

3. The algorithm

As described in the previous section, finding the minimum energy location of the PE's in the potential field is equivalent to minimizing the divergence between the Parzen estimate of the distribution of data points $f(x)$ and the estimator of the distribution of the PE's $g(x)$. The Parzen estimate for the data has a total of N kernels, where N is the number of data points, and the Parzen estimator for the PE's uses M kernels, M being the number of processing elements (typically $M << N$).

Any divergence measure can be chosen, but in the following the derivation will be carried out for the Cauchy-Schwartz divergence,

$$J(w) = \log \int f^2(x)\mathrm{d}x - 2\log \int f(x)g(x)\mathrm{d}x + \log \int g^2(x)\mathrm{d}x \tag{5}$$

The cost function $J(w)$ is minimized with respect to the location of the PE's (w).

When the PE's are located such that the potential field is at a local minima, no effective force acts on them. Moving the PE's in the opposite direction of the gradient will bring them to such a potential

minimum; this is also known as the gradient descent method. The derivative of equation (5) with respect to the location of the PE's must be calculated; but, since the data points are stationary only the last two terms of equation (5) (the cross information potential and the entropy of the PE's) have non-zero derivatives.

For simplicity, the derivation of the learning rule has been split in two parts; calculation of the contribution from cross information potential and calculation of the contribution from entropy. In the derivation Gaussian kernels are assumed, although, any symmetric kernel that obeys Mercer's condition (Mercer, 1909) can be used.
Consider the cross information potential term $(\log \int f(x)g(x)\mathrm{d}x)$; the Parzen estimator for $f(x)$ and $g(x)$ puts Gaussian kernels on each data point x_j and each PE w_i respectively, where the variances of the kernels are σ_f^2 and σ_g^2. Initially the location of the PE's are chosen randomly.

$$C = \int \hat{f}(x)\hat{g}(x)\mathrm{d}x \tag{6a}$$

$$= \frac{1}{MN}\int \sum_i^M G(x-w_i,\sigma_g^2)\sum_j^N G_f(x-x_j,\sigma_f^2)\mathrm{d}x \tag{6b}$$

$$= \frac{1}{MN}\sum_i^M\sum_j^N \int G(x-w_i,\sigma_g^2)G(x-x_j,\sigma_f^2)\mathrm{d}x \tag{6c}$$

$$= \frac{1}{MN}\sum_i^M\sum_j^N G(w_i-x_j,\sigma_a^2) \tag{6d}$$

where the covariance of the Gaussian after integration is $\sigma_a^2 = \sigma_f^2 + \sigma_g^2$. The gradient update for PE w_k from the cross information potential term then becomes:

$$\frac{\mathrm{d}}{\mathrm{d}w_k} 2\log C = -2\frac{\Delta C}{C} \tag{7}$$

Where ΔC denotes the derivative of C with respect to w_k.

$$\Delta C = -\frac{1}{MN}\sum_j^N G_a(w_k-x_j,\sigma_a)\sigma_a^{-1}(w_k-x_j) \tag{8}$$

Similarly for the entropy term$(-\log \int g^2(x)\mathrm{d}x)$

$$V = \int \hat{g}^2(x)\mathrm{d}x = \frac{1}{M^2}\sum_i^M\sum_j^M G(w_i-w_j,\sqrt{2}\sigma_g) \tag{9a}$$

$$\frac{\mathrm{d}}{\mathrm{d}w_k}\log V = \frac{\Delta V}{V} \tag{9b}$$

With

$$\Delta V = -\frac{1}{M^2}\sum_{i}^{M} G(w_k - w_i, \sqrt{2}\sigma_g)\sigma_g^{-1}(w_k - w_i) \tag{10}$$

The update for point k consist of two terms; cross information potential and entropy of the PE's:

$$w_k(n+1) = w_k(n) - \eta\left(\frac{\Delta V}{V} - 2\frac{\Delta C}{C}\right) \tag{11}$$

where η is the step size. The final algorithm for vector-quantization using information theoretic concepts (VQIT), consist of a loop over all w_k. Note that ΔC and ΔV are directional vectors where as C and V are scalar normalizations.

As with all gradient based methods this algorithm has problems with local minima. One of the ways local minima can be avoided is by annealing the kernel size (Erdogmus and Principe, 2002). The potential created by the particles will depend on the width of the kernels and the distance between the particles. When the distance is large compared to the width, the potential will be very 'bumpy' and have many local minima, and when the particles are close compared to the width, the corresponding potential will be 'smooth'. If, in addition, the number of particles is large the potential will have the shape of a normal distribution. Starting with a large kernel size will therefore help to avoid local minima. As with the SOM, a good starting point is to choose the kernels such that all particles interact with each other.

The algorithm derived in this section uses the gradient descent method to minimize an energy function based on interactions between information particles. Each iteration of the algorithm requires $\mathcal{O}(M^2N)$ Gaussian evaluations due to the calculation of C for each PE. The parameters for the algorithm are the variances of the density estimators σ_f^2 and σ_g^2 along with the step size η. The variances can be set equal and can be annealed from a size where all particles interact. The step size should be chosen small enough to ensure smooth convergence.

4. Simulations

In this section the ability of the VQIT algorithm to perform vector quantization is illustrated on a synthetic data set consisting of two half circles with unit radius which has been distorted with Gaussian noise with variance 0.1. One of the halves is displaced in horizontal direction (Figure 1).

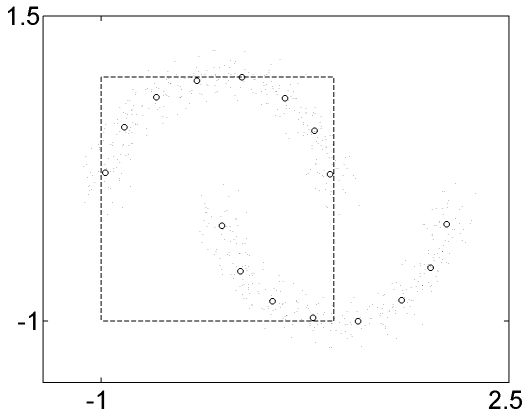
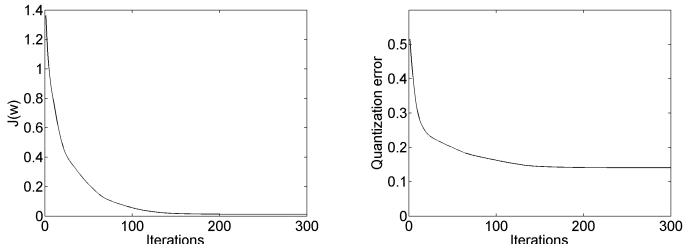


Figure 1. Artificial data used to evaluate performance, points are chosen from two half circles distorted by Gaussian noise. Initially all processing elements (PE's) were chosen randomly from the unit square, in all simulations the algorithm converged to the same solution (indicated by circles).

The data essentially consist of two clusters, as shown in Figure 1. Initially, 16 PE's are placed at random locations. The objective is to have the 16 PE's efficiently capture the structural property of the data.



a. Development of the cost-function averaged over 50 trials. The cost-function is always non-negative and has its minimum at zero but it is not guaranteed that a cost of zero can be achieved.

b. The quantization error measure is included for comparison with other algorithms.

Figure 2. Convergence of the algorithm, cost-function and quantization error.

Table I. Mean square errors for the data set shown in figure 1, the results are the average of 50 trials with different initial conditions. The Som, LBG and the VQIT algorithm always converges to the same solution.

	VQIT	SOM	LBG	K-means
QE	0.1408	0.1419	0.1393	0.1668

Using the algorithm presented above, the final locations of the PE's are shown, all in proper alignment with the data (Figure 1).

To assess the convergence of the VQIT, 50 monte-carlo simulations were performed. Starting with different initial conditions chosen uniformly from the unit square, it was found that with the right choice of parameters the algorithm always converges to the same solution. During training mode, having an initial large kernel-size and progressively annealing it can avoid the local minima. In this simulation, the width of the kernels was adjusted to equal the data-variance on each of its individual projections. The initial kernel size for the PE's was set to be:

$$\sigma_g = \sigma_n \begin{bmatrix} 0.75 & 0 \\ 0 & 0.5 \end{bmatrix}$$

where σ_n is the decaying variable. This is initially set to $\sigma_0 = 1$ and it decays after every iteration according to:

$$\sigma_n = \frac{\sigma_0}{1 + (0.05\sigma_0 n)}$$

The kernel size for the data (σ_f) was set equal to σ_g.

The evolution of the cost-function is shown in figure 2.a. Note that the cost-function is always positive and that the minimum value it can obtain is zero. The quantization error (QE) is also calculated by computing the average distance between the data points and their corresponding winner PE. The QE convergence curve is shown in figure 2.b. To compare with other algorithms, the quantization error is used as a figure of merit since it is a commonly used evaluation metric. Comparison is provided with three algorithms: SOM, LBG and K-means. K-means is the only algorithm of these that does not converge to the same solution regardless of initial conditions. The result of the comparison can be seen in Table I. The quantization error for the VQIT, SOM, and LBG centers around 0.14 while the K-means does not perform as well. It should be noted that none of the algorithms directly minimizes QE, however, LBG includes it in the iterations.

5. Discussion

In this section some of the critical issues regarding the algorithm are discussed. Emphasis is put on links to other algorithms and possible extensions.

The algorithm presented in this work is derived on the basis of the Cauchy-Schwartz inequality. This leads to a divergence measure based on the inner-product between two vectors. As noted earlier this is not the only choice, and has infact only been used here because of its close links to entropy. Another choice for cost-function is the Integrated Square Error which uses the quadratic distance between the distributions instead of an inner product:

$$\int (f(x) - g(x))^2 \mathrm{d}x = \int f^2(x)\mathrm{d}x - 2\int f(x)g(x)\mathrm{d}x + \int g^2(x)\mathrm{d}x. \quad (12)$$

The terms correspond to the information potentials of the data and the PE's and the cross information potential between the two. Note that equation (12) is similar to equation (5) except for the logarithm. Derivations equivalent to those used for C-S yields the very simple update:

$$w_k = w_k + \eta\,(\Delta V - \Delta C) \quad (13)$$

which requires $\mathcal{O}(MN)$ calculations per iteration. Annealing can also be used and the performance is similar to the VQIT.

"Density estimation is an ill posed problem and requires large amount of data to solve well" (Vapnik, 1995). Therefore, Vapnik suggests that one should not try to estimate densities in order to solve simpler problems (like vector quantization).

Even though this approach uses Parzen density densimates in its formulation, the pdf is never estimated. Instead the integral can be computed exactly through the double sum and therefore the method does not violate Vapnik's recommendations.

In a physical system, all potentials have the same form and only the magnitude (charge) can change, i.e. the same kernel type must be used for all particles. Also, in the Parzen estimator the mixture is homoskedastic, i.e. all mixtures have the same variance. However, in many of the recent publications (Van Hulle, 2002, Yin and Allinson, 2001, Heskes, 1999), a heteroskedastic approach is followed allowing the variance and weighting of the mixture components to change. It is easy to extend the algorithm presented in this work to include heteroskedastic

mixtures. The cost-function can just as well be minimized with respect to both means, variances and mixture weights. One can then choose to use either gradient descent or the EM algorithm to find the minimum. However, introducing more free parameters also means estimating more parameters from the same data points and can therefore lead to over fitting and poor generalization performance.

Another important issue is topology preservation. This feature is important if the mapping is to be used for visualization. Unlike the SOM, the learning rule proposed in this work is not topology preserving; it does not define an ordering of the PE's. It is however important to notice that if an ordering exists, the algorithm will approximately keep this ordering during convergence. Two different alterations can ensure that neighbors in the topology are also neighbors in the mapping.
The first and simplest is to add a term to the cost function equation (5). The term should include attraction from PE's that are close on the grid, one possibility is:

$$\sum_{i \in \mathcal{N}} (x_j - x_i) \tag{14}$$

Where $\mathcal{N}$ is the set of neighbors defined by the topology. Since the cost-function is changed, we cannot expect the PE's to converge to the same positions. However, once the topology has unfolded, the weighting of the neighborhood term equation (14) can be reduced and a solution will be obtained with PE at the desired positions and this time with the desired topology.
Another option more along the lines of the SOM and other algorithms (Yin and Allinson, 2001, Van Hulle, 2002), is to change the update of the cross information potential term. If we chose a winner PE for every data point and then update only itself and its neighbors, PE's close in the topology will be drawn together. Unfortunately this is not straight forward to put into the information theoretic framework.

The VQIT algorithm is based on block-computation of the data. It is possible to develop an online sample-by-sample update, which may result in a significant reduction in computational complexity. One way this can be achieved is by eliminating the second summation in equation (6) and computing the Kernel for only the current sample. However, this idea is still being explored and efforts directed at finding its similarity with the Kohonen-SOM will be addressed in a future paper.

6. Conclusion

In this paper an algorithm for finding the optimal quantization of a data set is proposed. The algorithm is derived based on concepts from information theoretic learning and it is shown how information potential fields and Parzen estimators can be used to give a physical interpretation of vector quantization. Simulations show errors equivalent to those obtained by the SOM and the LBG algorithms. However, unlike SOM and LBG, the algorithm proposed here utilizes a cost-function and its derivative. The algorithm can easily be extended to form a topology preserving map.

Future efforts will be directed towards comparing numerical properties of the algorithm and to incorporate the suggested changes. Further more, it will be interesting to see how VQIT performs on real data.

The main contribution of this work is a novel approach to vector-quantization utilizing physical laws and introducing probability densities directly into the optimization.

Acknowledgement

This work is partially supported by NSF grant ECS- 0300340

References

Bishop, C. M., M. Svensen, and C. K. I. Williams: 1996, 'GTM: a principled alternative to the self-organizing map'. *Artificial Neural Networks—ICANN 96. 1996 International Conference Proceedings* pp. 165–70.

Durbin, R. and D. Willshaw: 1987, 'An Analogue Approach of the Travelling Salesman Problem Using an Elastic Net Method'. *Nature,* **326**, 689–691.

Erdogmus, D. and J. C. Principe: 2002, 'Generalized Information Potential Criterion for Adaptive System Training'. *IEEE Transactions on Neural Networks* **13**(5).

Erdogmus, D., J. C. Principe, and K. Hild: 2002, 'Beyond second-order statistics for learning'. *Natural Computing* **1**(1), 85–108.

Erwin, E., K. Obermayer, and K. Schulten: 1992, 'Self-organizing maps: ordering, convergence properties and energy functions'. *Biological Cybernetics* **67:4755**.

Graepel, T., M. Burger, and K. Obermeyer: 1995, 'Phase Transitions in Stochastic Sef-Organizing Maps'. *Physical Review E* **56**(4), 3876–3890.

Heskes, T.: 1999, 'Energy functions for self-organizing maps'. In: S. E. Oja and Kaski (eds.): *Kohonen Maps.* Amsterdam: Elsevier, pp. 303–316.

Heskes, T. and B. Kappen: 1993, 'Error potentials for self-organization'. *Proceedings IJCNN93* **3**, 1219–1223.

Kohonen, T.: 1982, 'Self-organized formation of topologically correct feature maps'. *Biol. Cybern.* **43**, 59–69.

Kullback, S. and R. A. Leibler: 1951, 'On information and sufficiency'. *The Annals of Mathematical Statistics* **22**, 79–86.

Lampinen, J. and T. Kostiainen: 2002, 'Generative Probability Density Model in the SelfOrganizing Map'.

Linde, Y., A. Buzo, and R. M. Gray: 1980, 'An algorithm for vector quantizer design'. *IEEE Trans Commun COM* **28**, 84–95.

MacQueen, J.: 1967, 'Some methods for classification and analysis of multivariate observations'. *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability* **1**, 281–297.

Mercer, J.: 1909, 'Functions of positive and negative type and their connection with the theory of integral equations'. *Philosophical Transactions Royal Society London, A* **209**, 415–446.

Parzen, E.: 1962, 'On estimation of a probability density function and mode'. *Ann. Math. Stat* **27**, 1065–1076.

Principe, J. C., D. Xu, Q. Zhao, and J. Fisher: 2000, 'Learning from examples with information theoretic criteria'. *Journal of VLSI Signal Processing-Systems* **26**(1-2), 61–77.

Rényi, A.: 1970, *Probability Theory*. North-Holland Publishing Company, Amsterdam.

Scofield, C. L.: 1988, 'Unsupervised learning in the N-dimensional Coulomb network'. *Neural Networks* **1**(1), 129.

Sum, J., C.-S. Leung, L.-W. Chan, and L. Xu: 1997, 'Yet another algorithm which can generate topography map'. *Neural Networks, IEEE Transactions on* **8**(5), 1204–1207.

Van Hulle, M. M.: 2002, 'Kernel-based topographic map formation achieved with an information-theoretic approach'. *Neural Networks* **15**(8-9), 1029–1039.

Vapnik, V. N.: 1995, *The Nature of Statistical Learning Theory*. Springer-Verlag, New-York.

Yin, H. and N. Allinson: 2001, 'Self-organizing mixture networks for probability density estimation'. *Neural Networks, IEEE Transactions on,* **12**(2), 405–411.

Address for Offprints: Tue Lehn-Schiøler
Intelligent Signal Processing
Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads
DTU-Building 321
2800 Kgs. Lyngby
Denmark

VECTOR-QUANTIZATION BY DENSITY MATCHING IN THE MINIMUM KULLBACK-LEIBLER DIVERGENCE SENSE

Anant Hegde[1], Deniz Erdogmus[1], Tue Lehn-Schioler[2], Yadunandana N. Rao[1], Jose C. Principe[1]
[1] CNEL, Electrical & Computer Engineering Department, University of Florida, Gainesville, Florida, USA
[2] Intelligent Signal Processing, Informatics and Mathematical Modeling, Technical University of Denmark, Lyngby, Denmark

Abstract – Representation of a large set of high-dimensional data is a fundamental problem in many applications such as communications and biomedical systems. The problem has been tackled by encoding the data with a compact set of code-vectors called processing elements. In this study, we propose a vector quantization technique that encodes the information in the data using concepts derived from information theoretic learning. The algorithm minimizes a cost function based on the Kullback-Liebler divergence to match the distribution of the processing elements with the distribution of the data. The performance of this algorithm is demonstrated on synthetic data as well as on an edge-image of a face. Comparisons are provided with some of the existing algorithms such as LBG and SOM.

I. INTRODUCTION

Encoding an information source with a smaller set of code vectors is a fundamental problem in digital signal processing. There exists a huge literature on vector quantization (VQ) algorithms that use various cost functions to minimize the average distortion between the dataset and the information contained in the codebook. The K-means [1] and the LBG [2], count amongst the oldest of all VQ algorithms. The LBG mainly adopts a binary split approach that consists of splitting the centroids at each iteration, while partitioning the input space based on the centroids. The processing elements (PEs) are then updated such that they are placed at the centroids of all the partitions in the input space. Kohonen's SOM [3] is a stochastically and competitively trained vector quantizer. An important benefit of the SOM method is preservation of the topology of the input. This means, neighboring PEs in the weight space, correspond to neighboring points in the input (data) space. In summary, the SOM tries to approximate the distribution of the input data, while preserving structure.

One of the problems with the existing VQ algorithms is that they do not explicitly minimize a cost function; they are rather heuristic. Erwin [4] showed that when the SOM has converged, it is at the minimum of some discontinuous cost function. These discontinuities make the cost prone to drastic changes in some instances, which is undesirable. Heskes *et al.* [5] have made attempts to find a smooth cost function that, when minimized, gives the SOM update rule.

Efforts have also been made to design VQ algorithms using information theoretic perspectives. Heskes [5] used a cost function consisting of the quantization error and the entropy of the PEs. He also explored the links between SOM [3], elastic nets [6], and mixture modeling concluding that these methods are closely linked via the free energy point-of-view. Van Hulle [7] used a learning rule that consists of adapting the mean and variance of a Gaussian kernel, to maximize the entropy of the PEs. In order to prevent this algorithm from converging to a trivial solution where the PEs coincide, he modifies the algorithm quite heuristically to maximize entropy while minimizing mutual information by introducing competition between the kernels.

Earlier the authors approached the VQ problem from a density-matching point of view, where the statistical distributions of the data and the distribution of the PEs were matched through the maximization of the correlation, resulting in a cost function based on the Cauchy-Schwartz (CS) inequality [9]. In this paper, the VQ network weights are optimized to minimize the Kullback-Leibler (KL) divergence between the distribution of the data and the PEs. The equivalence between the minimization of KL divergence and the maximum likelihood principle is well known. Thus, the resulting optimal VQ solution can be considered equivalently as the maximum likelihood solution under the assumed distribution model. This algorithm based on KL divergence performs as well or better than the CS inequality algorithm, with reduced computational complexity.

Section II describes the proposed VQKL algorithm in detail. Section III presents simulation results using an artificial data set and a data set obtained by edge-detection of a face image. Comparisons with LBG and SOM are provided. The final section concludes the paper with remarks on possible future directions to improve the algorithm.

II. ALGORITHM

Consider the vector samples $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ from an information source in a *d*-dimensional signal space. Suppose that these samples are drawn from the distribution $g(\mathbf{x})$. Since, in practice the data distribution is generally unknown, it can be estimated using a Parzen-window estimator; this estimate of the data probability density function (pdf) is:

$$\hat{g}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i; \mathbf{\Lambda}_{\mathbf{x}}) \quad (1)$$

where $K(\xi; \mathbf{\Lambda})$ is the user-selected kernel function with $\mathbf{\Lambda}$ being the kernel size matrix and $\mathbf{x}_i$ are independent vector

samples drawn from the distribution $g(\mathbf{x})$. One of the requirements for the kernel function is that it should be symmetric, unimodal, and continuously differentiable. A Gaussian kernel meets all these requirements:

$$G(\boldsymbol{\xi};\boldsymbol{\Lambda}) = e^{-\boldsymbol{\xi}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{\xi}/2} \Big/ \left((2\pi)^{d/2} \mid \boldsymbol{\Lambda} \mid^{1/2}\right) \quad (2)$$

Similarly, let the true distribution of the PEs be $f(\mathbf{x})$. Suppose that the individual VQ weight vectors are independent samples drawn from this distribution, $\{\mathbf{w}_1,...,\mathbf{w}_M\}$. In VQ it is desirable to have $M<<N$. Using Parzen windowing with Gaussian kernels as before, the estimated density of the PEs is:

$$\hat{f}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} G(\mathbf{x}-\mathbf{w}_i, \boldsymbol{\Lambda}_{\mathbf{w}}) \quad (3)$$

The objective is to efficiently encode the data samples using a much smaller set of quantized weights without compromising the accuracy of the data representation. In other words, we wish to find a compact set of processing elements that can best represent the source data in terms of its distribution. This can be achieved by optimizing the weight vectors $\mathbf{w}_i$ such that the estimated density of the weights maximally match the estimated density of the data in accordance with some divergence criterion. Specifically, the Kullback-Leibler (KL) divergence [8], between two distributions $a(x)$ and $b(x)$, is

$$K(a \,\|\, b) = \int a(x)\log\frac{a(x)}{b(x)}dx \qquad K(b \,\|\, a) = \int b(x)\log\frac{b(x)}{a(x)}dx \quad (4)$$

All integrals are evaluated from -∞ to ∞. The KL divergence is not symmetric, i.e., $K(a \,\|\, b) \neq K(b \,\|\, a)$. Both quantities are nonnegative and become zero if and only if $a(x)=b(x)$.

A. Vector Quantization Using Kullback-Leibler Divergence

The VQKL algorithm uses the KL divergence measure as the optimality criterion. Due to the Parzen estimates of the densities using continuous and differentiable kernels, the performance surface is smooth, allowing us to use gradient-based or other iterative descent algorithms. In particular, the following cost function is minimized:

$$\begin{aligned} J(\mathbf{W}) &= \int f(\mathbf{x}) \log \frac{\hat{f}(\mathbf{x})}{\hat{g}(\mathbf{x})} d\mathbf{x} \\ &= \int f(\mathbf{x}) \log \hat{f}(\mathbf{x}) d\mathbf{x} - \int f(\mathbf{x}) \log \hat{g}(\mathbf{x}) d\mathbf{x} \\ &= E_f[\log \hat{f}(\mathbf{x})] - E_f[\log \hat{g}(\mathbf{x})] \\ &\cong \frac{1}{M}\sum_{i=1}^{M} \log \frac{1}{M}\sum_{j=1}^{M} G(\mathbf{w}_i - \mathbf{w}_j; \boldsymbol{\Lambda}_{\mathbf{w}}) \\ &\quad - \frac{1}{M}\sum_{i=1}^{M} \log \frac{1}{N}\sum_{j=1}^{N} G(\mathbf{w}_i - \mathbf{x}_j; \boldsymbol{\Lambda}_{\mathbf{x}}) \end{aligned} \quad (5)$$

where $\mathbf{W}=[\mathbf{w}_1,\ldots,\mathbf{w}_M]$. The strategy of this cost function can be intuitively understood as follows: the first term is the negative of the Shannon entropy of the weights, therefore minimizing this cost is equivalent to maximizing the entropy of the weights (similar to [7]); at the same time, minimizing the second term in (5) can be considered as maximizing the correlation between the weight distribution $f(\mathbf{x})$ and log of the data distribution $\log(g(\mathbf{x}))$. The logarithm emphasizes the contributions from the low-probability regions of the data. This emphasis of sparse regions ensures that some weights are reserved for representing these areas in the data space. This cross term ensures that the weight distribution matches the data distribution closely.

The weight vectors are optimized by minimizing (5) using gradient descent:

$$\mathbf{w}_k(n+1) \leftarrow \mathbf{w}_k(n) - \eta\, \partial J(\mathbf{W}) / \partial \mathbf{w}_k \quad (6)$$

The necessary gradient expressions with respect to each weight vector are found to be:

$$\begin{aligned} \frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} &= \frac{-2}{M}\sum_{i=1}^{M} \frac{G(\mathbf{w}_i - \mathbf{w}_k; \boldsymbol{\Lambda}_{\mathbf{w}})\boldsymbol{\Lambda}_{\mathbf{w}}^{-1}(\mathbf{w}_i - \mathbf{w}_k)}{\rho(\mathbf{w}_i, \mathbf{w}_1,..., \mathbf{w}_M; \boldsymbol{\Lambda}_{\mathbf{w}})} \\ &\quad - \frac{1}{M}\frac{G(\mathbf{w}_k - \mathbf{x}_j; \boldsymbol{\Lambda}_{\mathbf{x}})\boldsymbol{\Lambda}_{\mathbf{x}}^{-1}(\mathbf{w}_k - \mathbf{x}_j)}{\rho(\mathbf{w}_k, \mathbf{x}_1,..., \mathbf{x}_N; \boldsymbol{\Lambda}_{\mathbf{x}})} \end{aligned} \quad (7)$$

where

$$\rho(\mathbf{w}, \mathbf{x}_1,..., \mathbf{x}_N; \boldsymbol{\Lambda}) = \sum_{j=1}^{N} G(\mathbf{w} - \mathbf{x}_j; \boldsymbol{\Lambda}) \quad (8)$$

The alternative definition of KL divergence is not used because it reduces to only matching of the weight distribution to that of the data. This is easily seen by observing the explicit expression. The alternative divergence is:

$$\begin{aligned} J(\mathbf{W}) &= \int g(\mathbf{x}) \log \frac{g(\mathbf{x})}{f(\mathbf{x})} d\mathbf{x} \\ &= \int g(\mathbf{x}) \log g(\mathbf{x}) d\mathbf{x} - \int g(\mathbf{x}) \log f(\mathbf{x}) d\mathbf{x} \end{aligned} \quad (9)$$

The first term does not depend on the weights, therefore it can be dropped from the cost function. Since the entropy maximization term is lacking, it has been observed that the convergence is typically much slower, although the computational load per iteration is lower in this alternative. Therefore, we adopt the approach given in (5) in the rest of the paper.

B. Discussion of Implementation Issues in VQKL

As in all gradient-based optimization techniques, this algorithm might suffer from local minima. It has been shown in previous papers that in learning algorithms designed using the Parzen windowing technique one way to avoid local minima is to anneal the kernel size [10]. A large kernel size will stretch and smoothen the performance surface eliminating some spurious local minima and enabling the PEs to move towards the *biased* global optimum of the new surface. As training progresses, the kernel size is annealed to yield a narrower kernel and a weaker smoothening effect, thus decreasing the bias in the global optimum allowing the weights to converge to the global optimum. Therefore, in the VQKL algorithm, we propose to start with a large kernel size to enable interactions between all PE-PE and PE-data pairs. By progressively annealing the kernel size with iterations, the interactions are limited to only nearby points. This

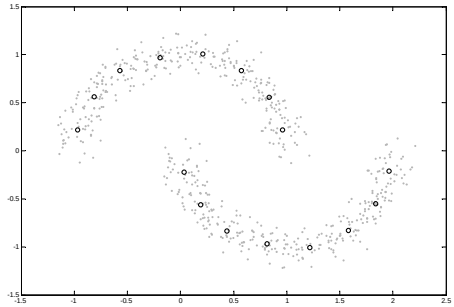


Fig. 1. Simulated data consisting of two half circles (dots). 16 PEs after convergence are shown in small circles.

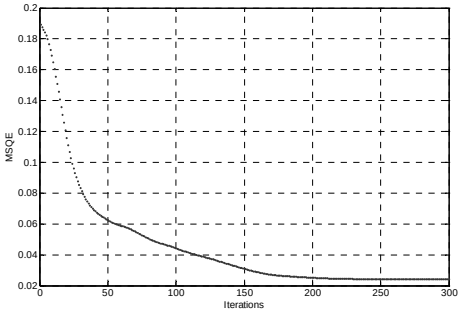


Fig. 2. Average MSQE versus iterations for VQKL in the first data set.

	VQKL	SOM	LBG
MSQE	0.024	0.024	0.023
J	2.437	3.378	2.460

Table 1. Comparison of MSQE for the three algorithms in the first data set. The standard deviations of VQKL and SOM are negligible over the Monte Carlo runs and for LBG it is zero.

progressive annealing strategy bears strong resemblance to the cooperative/competitive learning technique employed by the SOM.

Since VQKL uses batch updates, the kernel sizes are set up as follows:

a) Estimate the sample covariance matrix $\mathbf{\Sigma_x}$ of the data $\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$.

b) Set $\mathbf{\Lambda_w}(0)=\mathbf{\Lambda_x}(0)=\alpha diag(\mathbf{\Sigma_x})$, where $\alpha>0$ is a constant determined empirically (typically in the order of 10 to 100), and $diag(\mathbf{\Sigma_x})$ is a diagonal matrix consisting of the variances of the data along each dimension.

c) Anneal both kernel sizes with every iteration (where n is the iteration index) using some annealing factor λ according to

$$\mathbf{\Lambda_w}(n) = \mathbf{\Lambda_x}(n) = \alpha diag(\mathbf{\Sigma_x})/(1+\lambda n) \tag{10}$$

The kernel size is never allowed to decrease below a selected threshold $\beta diag(\mathbf{\Sigma_x})$, where $\beta \geq 0$ is a small constant on the order of 10^{-3} to 10^{-1}.

The VQKL algorithm requires $O(M^2+MN)$ Gaussian evaluations for updating the weights at every iteration. The performance of the algorithm particularly depends on how accurately the densities are estimated using the Parzen window estimator. The kernel size matrices $\mathbf{\Lambda_w}$ and $\mathbf{\Lambda_x}$ constitute the free parameters of the density estimation process. Additionally, a gradient descent step size η must be selected. The step size must be sufficiently slow compared to the annealing rate. The step size can also be annealed to ensure smoother convergence.

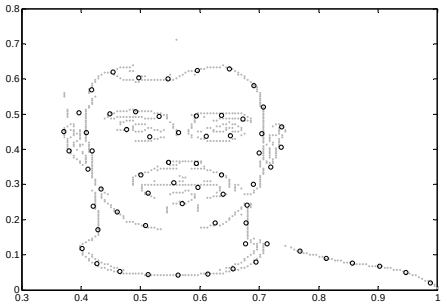
III. Results

In this section, the quantization performance of the VQKL algorithm is demonstrated on two data sets. The first data set (also used in [9]) is an artificially generated two-cluster data in 2-dimensions. The second data set is an edge-detected face image, where the positions of the edge pixels in the image constitute the data points (also 2-dimensional). The second example is especially preferred as the edges of each organ in the face constitute a *natural* clustering solution. Comparisons with LBG and SOM are presented on these two data sets using standard performance metrics.

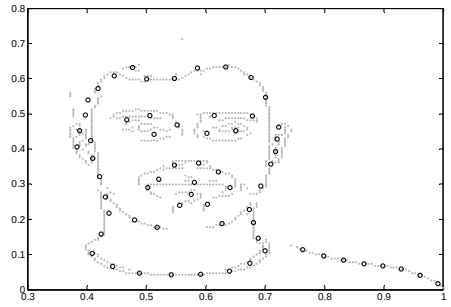
The first data set, shown in Fig. 1, essentially consists of samples drawn from two half circles with unit radius distorted with a Gaussian noise with standard deviation 0.1. Optimizing 16 randomly initialized PEs according to the KL divergence measure discussed above, the quantization solution shown in Fig. 1 is obtained consistently for all of the 20 Monte Carlo initializations. The average convergence curve of the algorithm over these Monte Carlo runs, quantified by the average mean-square-quantization-error (MSQE), is presented in Fig. 2. In this example, we set α=1.5, β=0, λ=0.08, the variances of the data in each direction were calculated as 0.75 and 0.51. The MSQE is calculated by:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}_i - \mathbf{w}_{*i}\right\|^2 \tag{11}$$

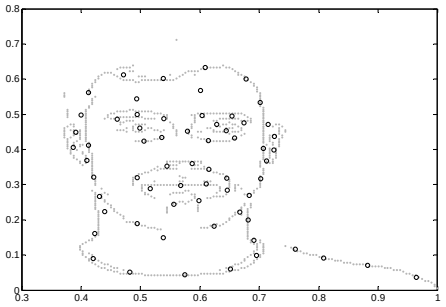
where $\mathbf{w}_{*i}$ is the nearest weight to sample $\mathbf{x}_i$ after convergence is achieved. This is a widely accepted measure in the VQ literature and has become a standard error metric for performance evaluation. The MSQE of VQKL, LBG, and SOM are provided in Table 1 for the first data set. Since LBG explicitly tries to minimize this criterion, it performs the best among the three methods. Alternatively, distortion can also be quantified by the KL divergence, $J(\mathbf{W})$, between the source and the PE distribution (5). Even though $J(\mathbf{W})$ is explicitly used as the cost function in VQKL, it appears to be a stronger measure since it directly quantifies the extent to which the distribution of PEs differ from the distribution of the data. Evidently, higher order moments are considered in $J(\mathbf{W})$ as opposed to MSQE, which merely considers second



(a)



(b)



(c)

Fig 3. Illustration of the a) VQKL, b) LBG and c) SOM algorithms to quantize an edge-detected face image. 64 PEs are shown superimposed on the face data.

	VQKL	SOM	LBG
MSQE	3.37×10^{-4}	3.52e-4	3.06e-4
J	0.101	2.205	1.774

Table 2. Comparison of MSQE and KL divergence for the three algorithms in the face data set. The standard deviations of MSQE and J over the Monte Carlo runs are not provided as they were negligible.

order statistics. In this comparison, the VQKL outperforms both the SOM and the LBG by yielding the smallest KL divergence (also shown in Table 1).

The second example is the quantization of the edges of a face image. The weights are expected to specialize in interesting areas in the face, such as the ears, the nose, the eyes, and the mouth. This VQ representation of a face finds applications in face recognition and face modeling problems. A quantizer with 64 PEs is optimized on the image shown in Fig. 3a. Using the VQKL algorithm, the optimization results varied insignificantly over the 20 Monte Carlo runs performed with random initial conditions. The parameters were set to η=0.03, α=30, λ=0.12, and β=0. The data variances in each direction were found as 0.0171 and 0.0286. For the same image, the LBG quantization result is presented in Fig. 3b.

Even though the PE assignments in Fig. 3a and Fig. 3b look very similar, certain subtle qualitative differences are also evident. The left ear and the portion just above the right ear are described better by the VQKL compared to the LBG. The VQKL saves some weights from the shoulder representation to model the eyes with more precision, for example. This is expected because, intrinsically, the LBG tries to partition the regions and place the PEs at the centroids of the partitions, regardless of the distribution of the data. The VQKL on the other hand extracts more information from the data and allocates PEs to suit their structural properties. The bias in the LBG towards the centroids can also be seen on the shoulder region, in terms of the excessive number of PEs. For a quantitative comparison, the MSQE and $J(\mathbf{W})$ for VQKL, LBG, and SOM are provided in Table 2. As before, the LBG is better in terms of MSQE, while the VQKL outperforms the other two algorithms in terms of KL divergence.

IV. Discussion

In this study, we present an information theoretic approach to the vector-quantization problem. The proposed VQKL algorithm optimizes the code vectors by using the gradient descent technique to minimize the Kullback-Liebler (KL) divergence between the data distribution and the quantization weight vector distribution. As opposed to many existing VQ algorithms, which are based on heuristic reasonings, the VQKL algorithm is based on a well defined optimization problem, which also provides an intuitive notion of how the resulting VQ models the statistical distribution of the data. Its computational complexity is higher than that of the SOM and the LBG; however, the information extracted from the data enables a better infrastructure for quantization.

Comparisons on two data sets showed that the VQKL algorithm outperforms the other two in terms of quantization error entropy, which is a direct measure of quantization uncertainty according to information theory. In the future, the face image quantization example will be extended to the important application of face recognition. Other possible applications include speech recognition using the quantized features. Finally, the sensitivity of the least-squares type

optimality criterion to outliers is well known in the statistics and signal processing literature. The LBG method is expected to be heavily biased due to the strong effects of the outliers to the centroids. Since the outliers are defined as extremely rare cases of degenerate samples, the proposed method is expected to provide reduced sensitivity of the optimal VQ solution to outliers as they will not contribute significantly to the density mismatch between the data and the code vectors. The effects of outliers on the performance will be studied in detail in the future.

ACKNOWLEDGMENTS

This work was supported by NSF grant ECS-0300340.

REFERENCES

[1] S.P. Lloyd, "Least Squares Quantization in PCM's," Bell Telephone Laboratories Paper, Murray Hill, NJ, 1957.

[2] Y. Linde, A. Buzo, R.M. Gray, "An Algorithm for Vector Quantizer Design," IEEE Transactions on Communications, vol. 28, pp. 84-95, 1980.

[3] T. Kohonen, "Self-Organized Formation of Topologically Correct Features Maps," Biological Cybernetics, vol. 43, pp. 59-69, 1982.

[4] E. Erwin, K. Obermayer, K. Schulten, "Self-Organizing Maps: Ordering, Convergence Properties and Energy Functions," Biological Cybernetics, vol. 67, pp. 47-55, 1992.

[5] T. Heskes, "Energy Functions for Self-Organizing Maps," in *Kohonen Maps*, E. Oja, S. Kaski (eds.), Elsevier, Amsterdam, pp. 303-316, 1999.

[6] R. Durbin, D. Willshaw, "An Analogue Approach of the Traveling Salesman Problem Using an Elastic Net Method," Nature, vol. 326, pp. 689-691, 1987.

[7] M.M. Van Hulle, "Kernel-Based Topographic Map Formation Achieved with an Information-Theoretic Approach", Neural Networks, vol. 15, pp. 1029-1039, 2002.

[8] S. Kullback, R.A. Liebler, "On Information and Sufficiency," The Annals of Mathematical Statistics, vol. 22, pp. 79-86, 1951.

[9] T. Lehn-Schioler, A. Hegde, D. Erdogmus, J.C. Principe, "Information Theoretic Vector-Quantization," submitted to Natural Computation, 2003.

[10] D. Erdogmus, J.C. Principe, "Generalized Information Potential Criterion for Adaptive System Training," IEEE Transactions on Neural Networks, vol. 13, no. 5, pp. 1035-1044, 2002.

PARZEN PARTICLE FILTERS

Tue Lehn-Schiøler

ISP, Technical University of Denmark
email: tls@imm.dtu.dk

Deniz Erdogmus & Jose C. Principe

CNEL, University of Florida
email: [deniz,principe]@cnel.ufl.edu

ABSTRACT

Using a Parzen density estimator any distribution can be approximated arbitrarily close by a sum of kernels. In particle filtering this fact is utilized to estimate a probability density function with Dirac delta kernels; when the distribution is discretized it becomes possible to solve an otherwise intractable integral. In this work we propose to extend the idea and use any kernel to approximate the distribution. The extra work involved in propagating small kernels through the nonlinear function can be made up for by decreasing the number of kernels needed, especially for high dimensional problems. A further advantage of using kernels with nonzero width is that the density estimate becomes continuous.

1. INTRODUCTION

The filtering problem can be formulated as

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{v}_{k-1} \quad (1a)$$
$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k \quad (1b)$$

where $\mathbf{v}$ and $\mathbf{w}$ are the process noise and the observation noise. The state transition density is fully specified by $\mathbf{f}$ and the process noise distribution and the observation likelihood is fully specified by $\mathbf{h}$ and the observation noise distribution.

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = p_v(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1})) \quad (2a)$$
$$p(\mathbf{z}_k|\mathbf{x}_k) = p_w(\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k)) \quad (2b)$$

The problem is to find an update formula from $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ to $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, where $\mathbf{z}_{1:k}$ denotes all observation $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ up to time k. The Bayesian approach [1] gives the following update:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)\int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\mathrm{d}\mathbf{x}_{k-1}}{\int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})\mathrm{d}\mathbf{x}_k} \quad (3)$$

The problem can be broken down to two subproblems. 1) Find the propagation of a probability density function (pdf) through a nonlinearity. 2) Modify the pdf according to the recorded measurements $\mathbf{z}_k$. Figure 1(a) illustrates stage one of the problem.

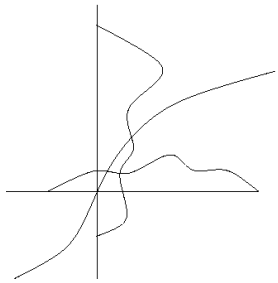
In equation (4) multiplying with $p(\mathbf{z}_k|\mathbf{x}_k)$ can be seen as stage two, and performing the multi-dimensional integration

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\mathrm{d}\mathbf{x}_{k-1} \quad (4)$$

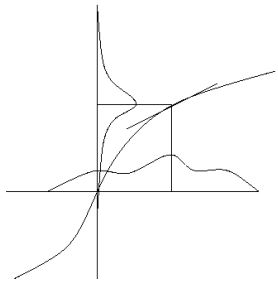
as stage one. In general the integral can not be calculated analytically, hence, we need some way of estimating the distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$.

Algorithms fall into four categories: Extended Kalman Filters, Gaussian Sum Filters, Sigma-Point Kalman Filters and Sequential Monte Carlo Methods (Particle Filters) [2]. Another way to categorize the methods is Gaussian belief (Extended Kalman filters, sigma point filters, moment matching), mixture of Gaussians (Gaussian-sum filter, pseudo-Bayes) and non-parametric methods (Particle filters) [3]. In the Extended Kalman Filter, the distributions are assumed Gaussian, but, the functions are not linear. The functions $\mathbf{f}$ and $\mathbf{h}$ are linearized around the previous state $\mathbf{x}_{k-1}$ using a second order Taylor expansion and then the standard Kalman equations are used. The result is a Gaussian distribution for $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ (see figure 1(b)). For nonlinear systems the solution is better than a normal Kalman filter, and it is accurate to first order. The Unscented Kalman Filter (Sigma Point Filter) [4] propagates points one standard deviation from the previous state $\mathbf{x}_{k-1}$ through the nonlinearity, then uses the points weighted appropriately (Gaussian quadrature like) to estimate mean and co-variance of a Gaussian. Finally this is used in the standard Kalman equations. It is accurate to the second order.

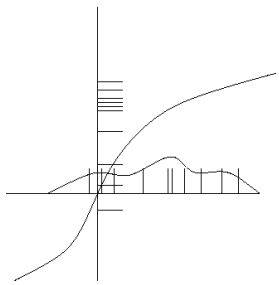
If the process noise distribution is approximated by a mixture of Gaussians the family of Gaussian sum filters arises [5]. In the mixture of Gaussians each mixture component is propagated through an extended Kalman filter. The state update $\mathbf{f}$ is linearized around the means of each mixture component and $\mathbf{h}$ is linearized around the predicted value for the mean of each mixture ($\mathbf{f}(x_{k-1})$). The resulting distribution is again a mixture of Gaussians. If the process noise is also non-Gaussian, this too can be approximated



(a) A density propagated through a nonlinear function.



(b) In the extended Kalman filter, the distribution is assumed to be a Gaussian modified by a linearization of the nonlinearity around the previous state.



(c) In the particle filter the densities are approximated by discrete samples. These samples can be sent through the nonlinearity to give an estimate of the output density.

Fig. 1. Propagation of a pdf through a nonlinearity 1(a) and different approximations to the propagated distribution 1(b) and 1(c). This is the prediction step corresponding to equation (1a), the resulting pdf is then modified to match the measurements equation (1b).

with a mixture of Gaussians. However, in this case the number of mixing components increases quickly.

Nonparametric methods are an entirely different approach to nonlinear filtering. In the Particle Filter it is assumed that the distributions $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ and $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ from equation (4) can be estimated by discrete distributions (figure 1(c)). Samples are drawn from the posterior distribution using importance sampling and a proposal distribution. In the generic particle filter the transition probability ($p(\mathbf{x}_k|\mathbf{x}_{k-1})$) is used as proposal, but other proposals has been proposed in e.g. the extended Kalman particle filter and the unscented particle filter [6], in these methods a filter (ekf or ukf) is calculated for each particle and the resulting mixture of Gaussians is used as proposal distribution for the particle filter. In an attempt to combine the particle filter and the Gaussian sum filter the Gaussian Sum Particle Filtering was proposed [7]. In this approach both the density and the process noise is considered mixture of Gaussians, in each time step samples are drawn from the mixture approximating $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$. These samples are propagated through the nonlinearity and used to offset the means in a mixture describing $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, then samples are drawn from this distribution too. In this way a discrete approximation of $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ is obtained and the sample mean and covariance of the new mixtures can be estimated. Unfortunately the number of mixtures explode, to avoid this mixtures with small weight can be thrown away. In a similar manner, the Gaussian mixture Sigma Point Particle Filter [2] uses a bank of sigma point filters to update $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ then samples are drawn from the mixture and the importance weights are calculated before a Gaussian mixture is fitted to produce the posterior estimate.

In this paper an algorithm based on the Parzen density estimator is presented. The algorithm is best categorized as non-parametric, since it can be seen as a direct extension to the particle filter. The basic concept is to improve the performance of the particle filter by using a better density estimate.

The algorithm is similar to the Gaussian Sum Particle Filter and the Kernel Filter [8], however, it is derived in a different manner that allows use of any kernel type. The derivation of the algorithm uses a sample mean estimate of the integral $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ and a particle filter like update of the weights. In section 2 the algorithm is derived and in section 3 experimental results are provided.

2. KERNEL METHOD

With a Parzen density estimator [9, 10] a distribution can be approximated arbitrarily close by a number of identical kernels centered on points chosen from the distribution. In the particle filter the kernels are delta functions, but information can be gained by using a broader kernel.

The distribution at time $k-1$ can be approximated by: $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) \approx \sum_i^N w_{k-1}^i K(\mathbf{A}_{k-1}^i(\mathbf{x}_{k-1}-\mathbf{x}_{k-1}^i))$, where $\mathbf{A}^i$ is a transformation matrix used to keep track of distortions of the kernel. Each kernel can be propagated through the mapping $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ by using a local linearization, yielding a continuous output distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, this is again a sum of kernels but the kernels are no longer identical (in the sense that they are from the same family of functions, yet they have different parameters).

Using the kernel representation equation (4) can be written as:

$$\sum_i^N w_{k-1}^i \int p_v(\mathbf{x}_k - f(\mathbf{x}_{k-1}))K(\mathbf{A}_{k-1}^i(\mathbf{x}_{k-1}-\mathbf{x}_{k-1}^i))\mathrm{d}\mathbf{x}_{k-1} \tag{5}$$

Each part of the sum can be handled individually, and under the assumption that the kernels are small compared to the dynamic in the nonlinearity, $\mathbf{f}$ can be locally linearized. By linearizing $\mathbf{f}$ around $\mathbf{x}_{k-1}^i$ the jacobian $\mathbf{J}|_{\mathbf{x}_{k-1}^i} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{\mathbf{x}_{k-1}^i}$ is introduced and the following change of variables can be employed: $\hat{\mathbf{x}}_{k-1} = \mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}^i) - \mathbf{J}|_{\mathbf{x}_{k-1}^i}(\mathbf{x}_{k-1} - \mathbf{x}_{k-1}^i)$. Inserting this in the integral from equation (5) yields:

$$\left|\mathbf{J}|_{\mathbf{x}_{k-1}^i}\right|^{-1} \int [p_v(\hat{\mathbf{x}}_{k-1}) \tag{6}$$
$$K \quad \left(\mathbf{A}_{k-1}^i \mathbf{J}|_{\mathbf{x}_{k-1}^i}^{-1}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}^i) - \hat{\mathbf{x}}_{k-1})\right) \quad] \,\mathrm{d}\hat{\mathbf{x}}_{k-1}$$

This integral is an expectation over the process noise $E_{p_v}\left[K\left(\mathbf{A}_{k-1}^i \mathbf{J}|_{\mathbf{x}_{k-1}^i}^{-1}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}^i) - \hat{\mathbf{x}}_{k-1}\right)\right]$ and can be approximated by a sample mean. In the extreme case a single sample drawn from p_v can be used, and the result is a translation of the kernel by the noise sample:

$$\begin{aligned} & E_{p_v}\left[K\left(\mathbf{A}_{k-1}^i \mathbf{J}|_{\mathbf{x}_{k-1}^i}^{-1}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}^i) - \hat{\mathbf{x}}_{k-1})\right)\right] \\ \approx \quad & K\left(\mathbf{A}_{k-1}^i \mathbf{J}|_{\mathbf{x}_{k-1}^i}^{-1}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}^i) - \mathbf{v}_{k-1})\right), \mathbf{v}_{k-1} \sim p_v \end{aligned}$$

Writing $p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \sum_i^N w_k^i K(\mathbf{A}_k^i(\mathbf{x}_k - \mathbf{x}_k^i))$ It is possible to identify the centers of the kernels $\mathbf{x}_k^i = \mathbf{f}(\mathbf{x}_{k-1}^i) + \mathbf{v}_{k-1}$ and the transformation matrix $\mathbf{A}_k^i = \mathbf{A}_{k-1}^i \mathbf{J}|_{\mathbf{x}_{k-1}^i}^{-1}$. By considering equation (4), (5) and (6) the weight update can be found to be $w_k^i = w_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^i)\left|\mathbf{J}|_{\mathbf{x}_{k-1}^i}\right|^{-1}$. This derivation holds for any kernel, however, for simplicity, in this paper the kernels are considered Gaussian.

For a Gaussian a change of variables can be employed such that the update of $\mathbf{A}_k^i$ can be replaced with an update of the covariance matrix as follows:

$$\mathbf{A}_k^i = \mathbf{A}_{k-1}^i \tag{7a}$$
$$\mathbf{\Sigma}_k^i = \mathbf{J}|_{\mathbf{x}_{k-1}^i} \mathbf{\Sigma}_{k-1}^i \mathbf{J}|_{\mathbf{x}_{k-1}^i}^T \tag{7b}$$

The transformation matrix is absorbed in the covariance matrix.

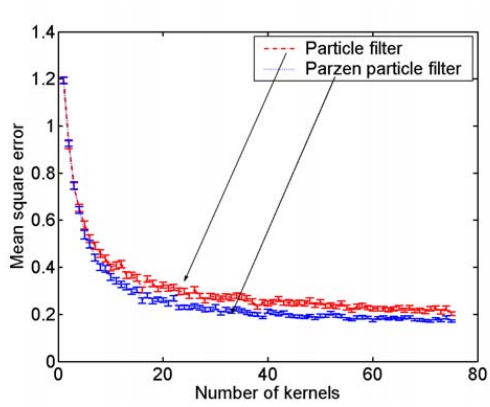


Fig. 2. Problem with nonlinear state transition, nonlinear observation process and Gaussian noise. Note that the performance of a Parzen particle filter with ≈ 10 kernels equals that of a normal particle filter with ≈ 20 kernels.

The transformation matrix $\mathbf{A}$ (or $\mathbf{\Sigma}$ in the Gaussian case) is distorted in each iteration, to avoid to much distortion a resampling schema can be applied. With a suitable frequency the distribution can be re-approximated by a Parzen estimator by drawing samples from $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, choosing $\mathbf{A}$ or $\mathbf{\Sigma}$ to take their initial values and setting the weights equal.

Earlier attempts use the kernels in the resampling phase where the shape and kernel size are selected based on the particle statistics (e.g. covariance) [11] . However, the proposed method iterates these properties of the kernel through the system equations, thus there is no need for optimization of kernel parameters at every step. In addition, the approximation of the integral stochastically using a sample drawn from p_v includes an inherent resampling step at every iteration, which allows the particle filter accuracy to survive longer than the standard version.

3. EXPERIMENTS

In this section the performance of the Parzen particle filter will be compared to the performance of the standard particle filter[1]. The method is tested on a one dimensional problem:

$$x_k = \frac{x_{k-1}}{2} + 25\frac{x_{k-1}}{(1+x_{k-1}^2)} + 8\cos(1.2k) + v_k \tag{8a}$$
$$z_k = 10\arctan(\frac{x_k}{10}) + w_k \tag{8b}$$

Where v_k and w_k are drawn from Gaussian distributions $G(0,1)$ (figure 2) and from gamma distributions $\Gamma(3,2)$ (fig-

[1]Code to reproduce the results can be found at www.imm.dtu.dk/∼tls

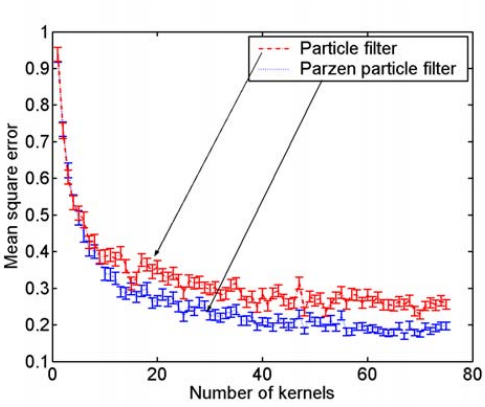


Fig. 3. Problem with nonlinear state transition, nonlinear observation process and gamma distributed noise.

ure 3).

The Parzen particle filter and the generic particle filter has been used on 100 time series generated using equation (8).

In figure 2 the mean square error is plotted as a function of the number of kernels. It can be seen that with few kernels the methods perform equally good (or bad), but as the number of kernels increases the kernel method becomes better. It can be seen that for this one dimensional example the kernel methods perform equally well, but the number of particles can be reduced drastically by improving the density estimate. It is expected that this effect will be even more impressive in higher dimensional problems.

4. CONCLUSION

A novel algorithm for nonlinear filtering is presented, the algorithm is based on Parzen density estimates and particle filter like propagation of the kernel through local linearizations of the nonlinear function.

It is shown that the improved density estimate help performance both with Gaussian and non-Gaussian noise. In this work only the special case with a Gaussian kernel is examined, however it is expected that a broader kernel would be well suited for long tailed noise, since it will be more likely to get the particles spread out.

The basic formulation for the arbitrary kernel case has been derived and performances of various kernel choices will be compared in a future publication.

5. REFERENCES

[1] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to non-linear and non-gaussian bayesian state estimation," *IEE Proceedings-F*, vol. 140, pp. 107–113, 1993.

[2] R. van der Merwe and E. Wan, "Gaussian mixture sigma-point particle filters for sequential probabilistic inference in dynamic state-space models," *Speech and Signal Processing (ICASSP)*, April 2003.

[3] T. Minka, "Bayesian inference in dynamic models – an overview, www.stat.cmu.edu/∼ minka/dynamic.html," 2003.

[4] S. J. Julier and J. Uhlmann, "A new extension of the kalman filter to nonlinear systems," 1997.

[5] D. L. Alspach and H. W. Sorensen, "Nonlinear bayesian estimation using gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, August 1972.

[6] R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan, "The unscented particel filter," *Advances in Neural Information Processing Systems 13*, Nov 2001.

[7] J.H. Kotecha and P.M. Djuric, "Gaussian sum particle filtering for dynamic state space models," *Acoustics, Speech, and Signal Processing, 2001. Proceedings. 2001 IEEE International Conference on*, vol. 6, pp. 3465 –3468 vol.6, 2001.

[8] M. Hurzeler and H.R. Kunsch, "Monte carlo approximations for general state space models," *Journal of Computational and Graphical Statistics*, vol. 7, no. 2, pp. 175–193, 1998.

[9] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat*, vol. 27, pp. 1065–1076, 1962.

[10] L. Devroye and G. Lugosi, *Combinatorial Methods in Density Estimation*, Springer, New York, 2001.

[11] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*, Springer, New York, 2001.

State-space models
- from the EM algorithm to a gradient approach

Rasmus Kongsgaard Olsson, Kaare Brandt Petersen
Tue Lehn-Schiøler

Technical University of Denmark, 2800 Kongens Lyngby, Denmark

Abstract

Slow convergence is observed in the EM algorithm for linear state-space models. A solution to the problem is to apply quasi-Newton-type optimizers that operate on the gradient of the log-likelihood function. We present a simple recipe for the computation of the exact gradient of the log-likelihood function which recycles components of the EM algorithm. We demonstrate the efficiency of the proposed method in three relevant instances of the linear state-space model. In high signal to noise ratios, where EM is particularly prone to converge slowly, we show that gradient-based learning results in a sizable reduction of the iterations required for convergence, and hence the computation time.

1 Introduction

State-space models are widely applied in cases where the data is generated by some underlying dynamics. Control engineering and speech enhancement are typical examples of applications of state-space models, where the state-space has a clear physical interpretation. The other extreme is black-box cases, where the objective is to provide in different applications a sufficiently flexible model. The state-space dynamics have no direct physical interpretation, only the generalization ability of the model matters, i.e. the prediction error on unseen data.

A fairly general formulation of *linear* state space models (without deterministic input) is:

$$\mathbf{s}_t = \mathbf{F}\mathbf{s}_{t-1} + \mathbf{v}_t \quad (1)$$

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \mathbf{n}_t \quad (2)$$

where equations (1) and (2) describe the state and observation spaces, respectively. State and observation vectors, $\mathbf{s}_t$ and $\mathbf{x}_t$, are random processes driven by i.i.d. zero-mean random inputs $\mathbf{v}_t$ and $\mathbf{n}_t$ with covariance $\mathbf{Q}$ and $\mathbf{R}$, respectively.

Optimization in state-space models based on maximizing the log-likelihood, $\mathcal{L}(\boldsymbol{\theta})$, with respect to the parameters, $\boldsymbol{\theta}$, fall in two main categories based on either gradients or Expectation Maximization (EM).

The principal approach to maximum likelihood in state-space models, and more generally in complex models, is to iteratively search the space of $\boldsymbol{\theta}$ for the maximal $\mathcal{L}(\boldsymbol{\theta})$ by taking steps in the direction of the gradient, $\nabla_\theta \mathcal{L}(\boldsymbol{\theta})$. A

basic *ascend* algorithm can be improved by supplying curvature information, i.e. second-order derivatives, line-search, etc. Often, numerical methods are used to compute the gradient and the Hessian, due to the complexity associated with the computation of these quantities. In [Gupta and Mehra, 1974] and [Sandell and Yared, 1978] fairly complex recipes are given for the computation of the analytical gradient in the linear state-space model.

The EM algorithm, [Dempster et al., 1977], is an important alternative to gradient-based maximum-likelihood. It was first applied to the optimization of linear state-space models by [Shumway and Stoffer, 1982] and [Digalakis et al., 1993]. A general class of linear Gaussian (state-space) models was treated in [Roweis and Ghahramani, 1999], in which the EM algorithm was the main engine of estimation. In the context of Independent Component Analysis (ICA), the EM algorithm has been applied in among other [Moulines et al., 1997] and [Højen-Sørensen et al., 2002]. In [Olsson and Hansen, 2004, 2005] and [Olsson and Hansen, 2005], the EM algorithm was applied to the Convolutive ICA problem.

A number of authors have reported the slow convergence of the EM algorithm. In [Redner and Walker, 1984], impractically slow convergence in 2-component Gaussian mixture models is documented. This critique is, however, moderated by [Xu and Jordan, 1996]. Modifications have been suggested to avoid the slow convergence of the basic EM algorithm, e.g. [Lachlan and Krishnan, 1997] and [Jamshidian and Jennrich, 1997] among many, but most come at a high cost in terms of computational complexity or at the expense of analytical simplicity.

The main contribution of this paper is to document that the *exact* gradient of the log-likelihood function can be computed using only the relatively simple math and programming of the EM algorithm. As a result, the reasonable convergence properties of the gradient-based optimizer are restored. This procedure is termed the *Easy Gradient recipe*. Furthermore, empirical evidence, supporting the results in [Bermond and Cardoso, 1999], is presented to demonstrate that the signal-to-noise ratio (SNR) has a dramatic effect on the convergence speed of the EM algorithm. Under certain circumstances, i.e. in high SNR settings, the EM algorithm fails to converge in reasonable time. The central points utilized in the proposed recipe has been mentioned in, e.g. [Salakhutdinov et al., 2003], but they did not comment on the relationship of the convergence properties to the SNR.

Three specializations of state-space models are investigated: 1) Sensor fusion for the black-box modelling of speech-to-face mapping problem. 2) Mean Field Independent Component Analysis (Mean Field ICA) for estimating a number of hidden independent sources that have been linearly and instantaneously mixed, 3) Convolutive Independent Component Analysis for convolutive mixtures.

In section 2, a theoretical introduction to EM and the Easy Gradient Recipe is given. In section 3, the implicated models are stated and in section 4 simulation results are presented.

2 Theory

Assume a model with observed variables $\mathbf{x}$, state space variables $\mathbf{s}$ and parameters $\boldsymbol{\theta}$, the calculation of the log-likelihood involves an integral over the state space variables:

$$\mathcal{L}(\boldsymbol{\theta}) = \ln p(\mathbf{x}|\boldsymbol{\theta}) = \ln \int p(\mathbf{x}|\mathbf{s}, \boldsymbol{\theta}) p(\mathbf{s}|\boldsymbol{\theta}) d\mathbf{s} \tag{3}$$

The marginalization in equation (3) is intractable for most choices of state prior and noise, hence direct optimization is rarely an option, even in the Gaussian case. Therefore a lower bound, B, is introduced on the log-likelihood, which is valid for any choice of $q(\mathbf{s}|\boldsymbol{\phi})$:

$$B(\boldsymbol{\theta}, \boldsymbol{\phi}) = \int q(\mathbf{s}|\boldsymbol{\phi}) \ln \frac{p(\mathbf{s}, \mathbf{x}|\boldsymbol{\theta})}{q(\mathbf{s}|\boldsymbol{\phi})} d\mathbf{s} \leq \ln p(\mathbf{x}|\boldsymbol{\theta})$$

At this point the problem seems to have been made *more* complicated, but the lower bound B has a number of appealing properties which makes the original task of finding the parameters easier. One important fact about B becomes clear when we rewrite it using Bayes theorem

$$B(\boldsymbol{\theta}, \boldsymbol{\phi}) = \ln p(\mathbf{x}|\boldsymbol{\theta}) - KL[q(\mathbf{s}|\boldsymbol{\phi})||p(\mathbf{s}|\mathbf{x}, \boldsymbol{\theta})] \tag{4}$$

where KL denotes the Kullback-Leibler divergence between the two distributions. Thus if we choose the variational distribution, q, to be exactly the posterior of the hidden variables, B is equal to the log-likelihood. For this reason one often tries to choose the variational distribution flexible enough to include the true posterior and yet simple enough to make the necessary calculations as easy as possible.

The approach is to maximize with respect to $\boldsymbol{\phi}$, in order to make the lower bound as close as possible to the log-likelihood and then maximize the bound with respect to the parameters $\boldsymbol{\theta}$. This stepwise maximization can be achieved by using the EM algorithm or by applying the *Easy Gradient Recipe*, see section 2.2.

2.1 The EM Update

The EM algorithm, as formulated in [Neal and Hinton, 1998], works in a straightforward scheme which is initiated with random values and iterated until suitable convergence is reached:

E: Maximize $B(\boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t. $\boldsymbol{\phi}$ keeping $\boldsymbol{\theta}$ fixed.

M: Maximize $B(\boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t. $\boldsymbol{\theta}$ keeping $\boldsymbol{\phi}$ fixed.

It is guaranteed that the lower bound function does not decrease on any combined E and M step. Figure 1 illustrates the EM algorithm. As mentioned in the introduction, the convergence is often slow - e.g. the curvature of the bound function, B, might be be much higher than that of $\mathcal{L}$, resulting in very conservative parameter updates. As mentioned in the introduction, this is particularly a problem in latent variable models with low-power additive noise. In [Bermond and Cardoso, 1999] and [Petersen and Winther, 2005], it is demonstrated that the EM update of the parameter $\mathbf{A}$ in equation (2) scales with the observation noise level, $\mathbf{R}$. That is, as the signal-to-noise ratio increases, the M-step change in $\mathbf{A}$ decreases, and more iterations are required to converge.

2.2 The Easy Gradient Recipe

The key idea is to regard the bound, B, as a function of $\boldsymbol{\theta}$ only, instead of a function of both the parameters $\boldsymbol{\theta}$ and the variational parameters $\boldsymbol{\phi}$. As a result, the lower bound can be applied to reformulate the log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = B(\boldsymbol{\theta}, \boldsymbol{\phi}_*) \tag{5}$$

where $\boldsymbol{\phi}_* = \boldsymbol{\phi}_*(\boldsymbol{\theta})$ fulfills the constraint $q(\mathbf{s}|\boldsymbol{\phi}_*) = p(\mathbf{s}|\mathbf{x}, \boldsymbol{\theta})$. Practically, it is often complicated to find $\boldsymbol{\phi}_*$ and it cannot always be done analytically. Comparing with equation (4), it is easy to see that $\boldsymbol{\phi}_*$ maximizes the bound. Since $\boldsymbol{\phi}_*$ is exactly minimizing the KL-divergence, the partial derivative of the bound with respect to $\boldsymbol{\phi}$ evaluated in the point $\boldsymbol{\phi}_*$, is equal to zero. Therefore the gradient of $B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)$ is equal to the partial derivative

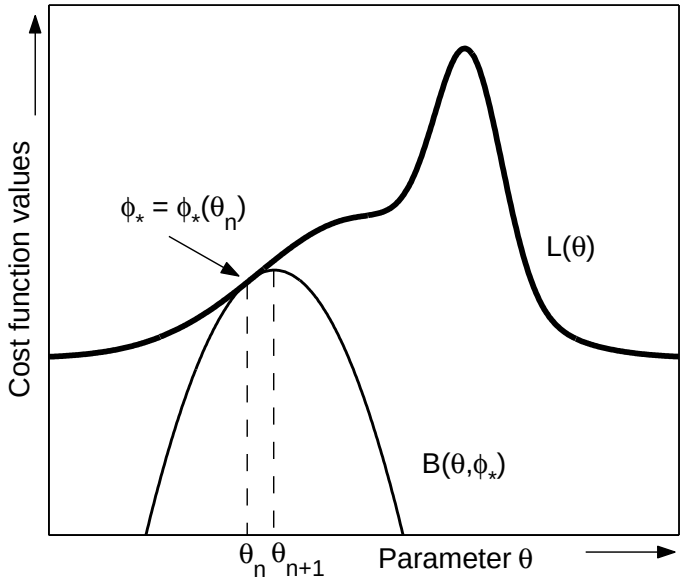
$$\frac{dB(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{d\boldsymbol{\theta}} = \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{\partial \boldsymbol{\theta}} + \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\Big|_{\boldsymbol{\phi}_*} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\phi}_*} = \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{\partial \boldsymbol{\theta}} \tag{6}$$

and due to the choice of $\boldsymbol{\phi}_*$, the gradient of the log-likelihood is the partial derivative of the bound

$$\frac{d\mathcal{L}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{dB(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{d\boldsymbol{\theta}} = \frac{\partial B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)}{\partial \boldsymbol{\theta}}$$

which can be realized by combining equations (5) and (6).

In this way we can obtain exact values and gradients of the true log likelihood using the lower bound. The observation of this is not new, since it is essentially the same which is used in [Salakhutdinov et al., 2003] to construct the so-called Expected Conjugated Gradient Algorithm (ECG). The novelty of the recipe, is the practical recycling of low-complexity computations carried out in connection to the EM algorithm for a much more efficient



(a)

Figure 1: Schematic illustration of lower bound optimization for a one-dimensional estimation problem, where $\boldsymbol{\theta}_n$ and $\boldsymbol{\theta}_{n+1}$ are iterates of the standard EM algorithm. The log-likelihood function, $\mathcal{L}(\boldsymbol{\theta})$, is bounded from below by the function $B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)$. The bound attains equality to $\mathcal{L}$ in $\boldsymbol{\theta}_n$ due to the choice of variational distribution: $q(\mathbf{s}|\boldsymbol{\phi}_*) = p(\mathbf{s}|\mathbf{x}, \boldsymbol{\theta}_n)$. Furthermore in $\boldsymbol{\theta}_n$, the derivatives of the bound and the log-likelihood are identical. In many situations, the curvature of $B(\boldsymbol{\theta}, \boldsymbol{\phi}_*)$ is much higher than that of $\mathcal{L}(\boldsymbol{\theta})$, leading to small changes in the parameter, $\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_{n+1}$.

optimization using *any* gradient-based non-linear optimizer. This can be expressed in MATLAB-style pseudo-code where a function `loglikelihood` receives as argument the parameter $\boldsymbol{\theta}$ and returns $\mathcal{L}$ and its gradient $\frac{d\mathcal{L}}{d\boldsymbol{\theta}}$:

	$\texttt{function}\ [\mathcal{L}, \frac{d\mathcal{L}}{d\boldsymbol{\theta}}] = \texttt{loglikelihood}(\boldsymbol{\theta})$
1	Find $\boldsymbol{\phi}^*$ such that $\frac{\partial B}{\partial \boldsymbol{\phi}}\big\|_{\boldsymbol{\phi}^*} = 0$
2	Calculate $\mathcal{L} = B(\boldsymbol{\theta}, \boldsymbol{\phi}^*)$
3	Calculate $\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial B}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\phi}^*)$

Step **1**, and to some extend step **2**, are obtained by performing an E-step, while **3** requires only little programming that implements the gradients used to solve for the M-step. Compared to the EM algorithm, the main advantage is that the function value and gradient can be fed to any non-linear gradient-based optimizer, which in most cases provides a substantial improvement of the convergence properties. In that sense, it is possible to benefit from the speed-ups of advanced gradient-based optimization.

The advantage of formulating the log-likelihood using the bound function, B, depends on the task at hand. In the linear state-space model, eq. (1) and (2), a brute force computation of the gradient of the log-likelihood is costly, since the computation of the gradients scales as $(d_\theta)^2$ times the cost of one Kalman Filter filter sweep.[1] When using the Easy Gradient Recipe, the combined computational cost depends on the optimizer of choice. Often, state-of-the-art software induces little overhead in addition to the computation of the gradient. In the case of linear state-space models, the total

[1]The computational complexity of the Kalman filter is $\mathcal{O}[N(d_s)^3]$, where N is the data length.

computational cost of the Easy Gradient Recipe is then dominated by steps **1** and **2**, which require only a Kalman smoothing, scaling as 2 Kalman Filter filter sweeps. [Sandell and Yared, 1978] noted in in their investigation of linear state-space models that a reformulation of the problem resulted in a similar reduction of the computational costs.

In this paper, a quasi-Newton gradient-based optimizer has been chosen, i.e. it estimates the inverse Hessian using the gradient. The implementation of the BFGS algorithm is due to Hans Bruun Nielsen, [Nielsen, 2000], and has built-in line search and trust region monitoring.

3 Models

The EM algorithm and the Easy Gradient Recipe were applied to three different models that can all be fitted into the linear state-space framework:

3.1 Kalman Filter Based Sensor Fusion

The state-space model of equations (1)+(2) can be used to describe systems where two different types of signals are measured. The signals could be, e.g., sound and images as [Lehn-Schiøler et al., 2005], where speech and lip movements were the observables. In this case, the observation equation (2) can be split into two parts.

$$\boldsymbol{x}_t = \begin{pmatrix} \boldsymbol{x}_t^1 \\ \boldsymbol{x}_t^2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}^1 \\ \boldsymbol{A}^2 \end{pmatrix} \boldsymbol{s}_t + \begin{pmatrix} \boldsymbol{n}_t^1 \\ \boldsymbol{n}_t^2 \end{pmatrix}$$

where $\boldsymbol{n}_t^1 \sim N(\mathbf{0}, \boldsymbol{R}^1)$ and $\boldsymbol{n}_t^2 \sim N(\mathbf{0}, \boldsymbol{R}^2)$. The innovation noise in the state-space equation (1) is defined as $\boldsymbol{v}_t \sim N(\mathbf{0}, \boldsymbol{Q})$. In the training phase, the parameters of the system, $\boldsymbol{F}$, $\boldsymbol{A}^1$, $\boldsymbol{A}^2$, $\boldsymbol{R}^1$,$\boldsymbol{R}^2$,$\boldsymbol{Q}$ are estimated by maximum likelihood using either EM or a gradient-based method. When the parameters have been learned, the state-space variable $\boldsymbol{s}$, which represents unknown hidden causes, can be deduced from one of the observations ($\boldsymbol{x}_1$ or $\boldsymbol{x}_2$) and the missing observation can be estimated by mapping from the state space.

3.2 Mean Field ICA

In Independent Component Analysis (ICA), one tries to separate linearly mixed sources using the assumed statistical independence of the sources. In many cases elaborate source priors are necessary, which calls for more advanced separation techniques such as Mean Field ICA. The method, which was first introduced in [Højen-Sørensen et al., 2002], can handle complicated source priors in an efficient approximative manner.

The model in equation (2) is identical to an instantaneous ICA model provided that $\mathbf{F} = \mathbf{0}$ and that $p(\mathbf{v}_t)$ is reinterpreted as the (non-Gaussian) source prior. The basic generative model of the instantaneous ICA is

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \mathbf{n}_t \tag{7}$$

where $\mathbf{n}_t$ is assumed i.i.d. Gaussian and $\mathbf{s}_t = \mathbf{v}_t$ is assumed distributed by a factorized prior $\prod_i p(v_{it})$, which is independent in both time and dimension. The Mean Field ICA is only approximately compatible with the Easy Gradient Recipe, since the variational distribution $q(\mathbf{s}|\boldsymbol{\phi})$ is not guaranteed

to contain the posterior $p(\mathbf{s}|\mathbf{x}, \boldsymbol{\theta})$. This, however, is not a problem if q is sufficiently flexible.

3.3 Convolutive ICA

Acoustic mixture scenarios are characterized by sound waves emitted by a number of sound sources propagating through the air and arriving at the sensors in delayed and attenuated versions. The instantaneous mixture model of standard ICA, equation (7), is clearly insufficiently describing this situation. In *convolutive* ICA the signal path (delay and attenuation) is modelled by an FIR filter, i.e. a convolution of the source by the impulse responses of the signal path:

$$\mathbf{x}_t = \sum_k \mathbf{C}_k \mathbf{s}_{t-k} + \mathbf{n}_t \tag{8}$$

where $\mathbf{C}_k$ is the mixing filter matrix. Equation (8) and the source independence assumption can be fitted into the state-space formulation of equations (1) and (2), see [Olsson and Hansen, 2004, 2005], by making the following model choices: 1) Noise inputs $\mathbf{v}_t$ and $\mathbf{n}_t$ are i.i.d. Gaussian. 2) The state vector is *augmented* to contain time-lagged values, i.e.

$$\bar{\mathbf{s}}_t \equiv \left[s_{1,t} s_{1,t-1} \ldots s_{2,t} s_{2,t-1} \ldots s_{d_s,t} s_{1,t-1} \ldots\right]^\top$$

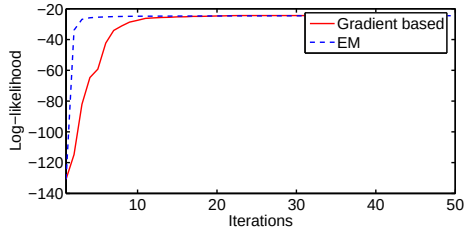
3) State-space parameter matrices (e.g. $\mathbf{F}$) are constrained to a special format (certain elements are fixed to 0's and 1's) in order to ensure the independency of the sources mentioned above.

4 Results

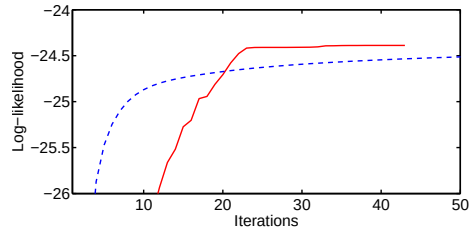
Before advancing to the more involved applications described above, the the advantage of gradient-based methods over EM will be explored for a one-dimensional linear state-space model, i.e. an ARMA(1,1) process. In this case, $\boldsymbol{F}$ and $\boldsymbol{A}$ are scalars as well as the observation variance $\boldsymbol{R}$ and the transition variance $\boldsymbol{Q}$. The latter is fixed to unity to resolve the inherent scale ambiguity of the model. As a consequence, the model has only 3 parameters. The BFGS optimizer mentioned in section 2 was used.

Figure 2 shows the convergence of both the EM algorithm and the gradient-based method. Initially, EM is fast, i.e. it rapidly approaches the maximum log-likelihood, but slows down as it gets closer to the optimum. The large dynamic range of the log-likelihood makes it difficult to ascertain the final increase in the log-likelihood, hence figure 2(b) provides a closeup on the log-likelihood scale. Table 1 gives an indication of the importance of the final increase. After 50 iterations, EM has reached a log-likelihood value of -24.5131, but the parameter values are still far off. After convergence, the log-likelihood has increased to -24.3883 which is still slightly worse than that obtained by the gradient-based method, but the parameters are now near the generative values. Similar results are obtained when comparing the learning algorithms on the Mean Field ICA and Convolutive ICA problems.

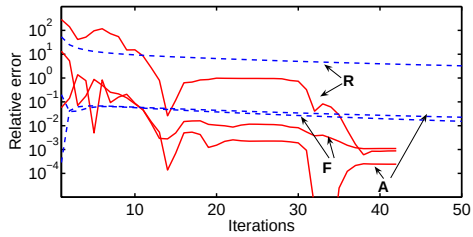
As argued in section 2, it is demonstrated that the number of iterations required by the EM algorithm to converge in state-space type models critically



(a)

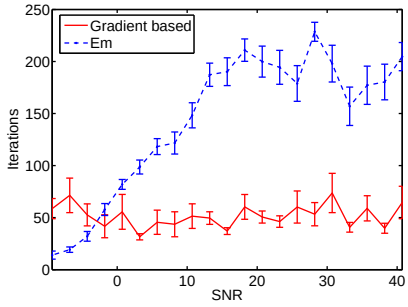


(b)

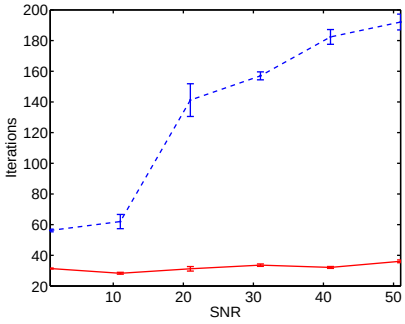


(c)

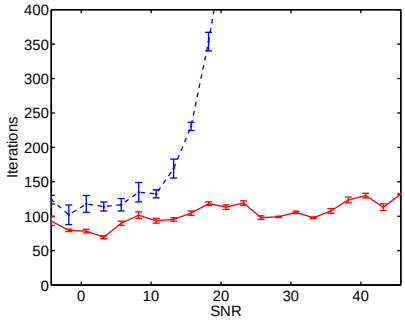
Figure 2: Convergence of EM (dashed) and a gradient-based method (dotted) in the ARMA(1,1) model. (a) EM has faster initial convergence than the gradient-based method, but the final part is slow for EM. (b) Zoom-in on the log-likelihood axis. Even after 50 iterations, EM has not reached the same level as the gradient based method. (c) Parameter estimates convergence in terms of squared relative (to the generative paramters) error.



(a)



(b)



(c)

Figure 3: Iterations for EM (dashed) and gradient-based optimization (solid) to reach convergence as a function of signal to noise ratio for the three different problems. (a) Kalman Filter model. (b) Mean Field ICA. (c) Convolutive ICA. Convergence was defined as a relative change in log-likehood below 10^{-5}.

	Generative	Gradient	EM 50	EM ∞
Iterations	-	43	50	1800
Log-likelihood	-	-24.3882	-24.5131	-24.3883
F	0.5000	0.4834	0.5626	0.4859
A	0.3000	0.2953	0.2545	0.2940
R	0.0100	0.0097	0.0282	0.0103

Table 1: Estimation in the ARMA(1,1) model. The convergence of EM is slow compared to the gradient-based method. Note that after 50 EM iterations, the log-likelihood is relatively close to the value achieved at convergence, but the parameter values are far from the generative values.

depends on the SNR. Figure 3 shows the performance of the two methods on the three different problems. The plots indicate that in the low-noise case, the EM algorithm requires relatively more iterations to converge whereas the gradient-based method performs equally well for all noise levels. Note that iterations in the gradient-based approach may require more than one function evaluation. Therefore, function evaluations were counted as iterations.

5 Conclusion

In applying the EM algorithm to maximum likelihood estimation in state-space models, we find, as many before us, that it has poor convergence properties in the low noise limit. Often a value 'close' to the maximum likelihood is reached in the first few iterations, while the final increase, which is crucial

to the accurate estimation of the parameters, requires an excessive amount of iterations.

More importantly, we provide a simple scheme for efficient gradient-based optimization achieved by transformation from the EM formulation, i.e. the simple math and programming of the EM algorithms is preserved. Following this recipe, one can get the optimization benefits associated with *any* advanced gradient based-method. In this way, the tedious, problem-specific, analysis of the cost-function topology can be replaced with an off-the-shelf approach. Although the analysis provided in this article is limited to a set of linear mixture models, it is in fact applicable to *any* model subject to the EM algorithm, hence constituting a strong and general tool to be applied by the part of the neural community that uses EM algorithm.

References

Bermond, O. and Cardoso, J.-F. (1999). Approximate likelihood for noisy mixtures. In *Proceedings of the ICA Conference*, pages 325–330.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistics Society, Series B*, 39:1–38.

Digalakis, V., Rohlicek, J., and Ostendorf, M. (1993). ML estimation of a stochastic linear system with the EM algorithm and its application to

speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442.

Gupta, N. K. and Mehra, R. K. (1974). Computational aspects of maximum likelihood estimation and reduction in sensitivity calculations. *IEEE Transactions on Automatic Control*, AC-19(1):774–783.

Højen-Sørensen, P. A., Winther, O., and Hansen, L. K. (2002). Mean field approaches to independent component analysis. *Neural Computation*, 14:889–918.

Jamshidian, M. and Jennrich, R. I. (1997). Acceleration of the EM algorithm using quasi-newton methods. *Journal of Royal Statistical Society, Series B*, 59(3):569–587.

Lachlan, G. J. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. John Wiley and Sons.

Lehn-Schiøler, T., Hansen, L. K., and Larsen, J. (2005). Mapping from speech to images using continuous state space models. In *Lecture Notes in Computer Science*, volume 3361, pages 136 – 145. Springer.

Moulines, E., Cardoso, J., and Cassiat, E. (1997). Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Proc. ICASSP*, volume 5, pages 3617–3620. The MIT Press.

Neal, R. M. and Hinton, G. E. (1998). *Learning in Graphical Models: A view*

of the EM algorithm that justifies incremental, sparse, and other variants, pages 355–368. Dordrecht: Kluwer Academic Publishers.

Nielsen, H. B. (2000). UCMINF - an algorithm for unconstrained nonlinear optimization. Technical Report IMM-Rep-2000-19, Technical University of Denmark.

Olsson, R. K. and Hansen, L. K. (2004). Probabilistic blind deconvolution of non-stationary sources. In *12th European Signal Processing Conference*, pages 1697–1700.

Olsson, R. K. and Hansen, L. K. (2005). A harmonic excitation state-space approach to blind separation of speech. In *Advances in Neural Information Processing Systems*. accepted, revised, not final.

Petersen, K. B. and Winther, O. (2005). The EM algorithm in independent component analysis. In *International Conference on Acoustics, Speech, and Signal Processing*.

Redner, R. A. and Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 2(26):195–239.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear Gaussian models. *Neural Computation*, 11:305–345.

Salakhutdinov, R., Roweis, S. T., and Ghahramani, Z. (2003). Optimization with EM and Expectation-Conjugate-Gradient. In *International Conference on Machine Learning*, volume 20, pages 672–679.

Sandell, N. R. and Yared, K. I. (1978). Maximum likelihood identification of state space models for linear dynamic systems. Technical Report Technical Report ESL-R-814, Electron. Syst. Lab, M.I.T.

Shumway, R. and Stoffer, D. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Anal.*, 3(4):253–264.

Xu, L. and Jordan, M. I. (1996). On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, (8):129–151.

Mapping from Speech to Images Using Continuous State Space Models

Tue Lehn-Schiøler, Lars Kai Hansen & Jan Larsen*

The Technical University of Denmark
Informatics and Mathematical Modelling
Richard Petersens Plads, Bld. 321.
Web: www.imm.dtu.dk (tls,lkh,jl)@imm.dtu.dk

Abstract. In this paper a system that transforms speech waveforms to animated faces are proposed. The system relies on continuous state space models to perform the mapping, this makes it possible to ensure video with no sudden jumps and allows continuous control of the parameters in 'face space'.
The performance of the system is critically dependent on the number of hidden variables, with too few variables the model cannot represent data, and with too many overfitting is noticed.
Simulations are performed on recordings of 3-5 sec. video sequences with sentences from the Timit database. From a subjective point of view the model is able to construct an image sequence from an unknown noisy speech sequence even though the number of training examples are limited.

1 Introduction

The motivation for transforming a speech signal into lip movements is at least threefold. Firstly, the language synchronization of movies often leaves the actors mouth moving while there is silence or the other way around, this looks rather unnatural. If it was possible to manipulate the face of the actor to match the actual speech it would be much more pleasant to view synchronized movies (and a lot easier to make cartoons). Secondly, even with increasing bandwidth sending images via the cell phone is quite expensive, therefore, a system that allows single images to be sent and models the face in between would be useful. The technique will also make it possible for hearing impaired people to lip read over the phone. If the person in the other end does not have a camera on her phone, a model image can be used to display the facial movements. Thirdly, when producing agents on a computer (like Windows Office Mr. clips) it would make communication more plausible if the agent could interact with lip movements corresponding to the (automatically generated) speech.

* The work is supported by the European Commission through the sixth framework IST Network of Excellence: Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL), contract no. 506778.

Lewis [1] provides an early overview paper about state of the art lip-sync in 1991. He concludes that using loudness to control the jaw is not a useful approach since sounds made with closed mouth can be just as loud as open mouth sounds. He also notes that the spectrum matching method used by MIT in the early 1980's has severe problems due to the formants independence of pitch. In this method the shape of the mouth is determined from the frequency content of the speech. The problem is illustrated by the fact that the mouth shape is the same when a sound e.g. an 'a' is spoken with a high or a deep voice. Final he mentions that it is possible to automatically generate speech from text and in this way gain control of what phoneme to visualize. In his view the speech synthesis in 1991 was not of sufficient quality to sound natural, and although progress has been made in the field automatic generated speech is still far from perfect. The suggestion in [1] is to extract phonemes using a Linear Prediction speech model and then map the phonemes to keyframes given by a lip reading chart.

The idea of extracting phonemes or similar high-level features from the speech signal before performing the mapping to the mouth position has been widely used in the lip-sync community. Goldenthal [2] suggested a system called "Face Me!". He extracts phonemes using Statistical Trajectory Modeling. Each phoneme is then associated with a mouth position (keyframe). In Mike Talk [3], phonemes are generated from text and then mapped onto keyframes, however, in this system trajectories linking all possible keyframes are calculated in advance thus making the video more seamless. In "Video rewrite" [4] phonemes are again extracted from the speech, in this case using Hidden Markov Models. Each triphone (three consecutive phonemes) has a mouth sequence associated with it. The sequences are selected from training data, if the triphone does not have a matching mouth sequence in the training data, the closest available sequence is selected. Once the sequence of mouth movements has been determined, the mouth is mapped back to a background face of the speaker. Other authors have proposed methods based on modeling of phonemes by correlational HMM's [5] or neural networks [6].

Methods where speech is mapped directly to facial movement are not quite as popular as phoneme based methods. However, in 'Picture my voice' [7], a time dependent neural network, maps directly from 11×13 Mel Frequency Cepstral Coefficients (MFCC) as input to 37 facial control parameters. The training output is provided by a phoneme to animation mapping but the trained network does not make use of the phoneme representation. Also Brand [8] has proposed a method based on (entropic) HMM's where speech is mapped directly to images. Methods that do not rely on phoneme extraction has the advantage that they can be trained to work on all languages, and that they are able to map non-speech sounds like yawning or laughing.

There are certain inherent difficulties in mapping from speech to mouth positions an analysis of these can be found in [9]. The most profound is the confusion between visual and auditive information. The mouth position of sounds like /b/,/p/ and /m/ or /k/,/n/ and /g/ can not be distinguished even though the sounds can. Similarly the sounds of /m/ and /n/ or /b/ and /v/ are very

similar even though the mouth position is completely different. This is perhaps best illustrated by the famous experiment by McGurk [10]. Thus, when mapping from speech to facial movements, one cannot hope to get a perfect result simply because it is very difficult to distinguish whether a "ba" or a "ga" was spoken.

2 Feature Extraction

Many different approaches have been taken for extraction of sound features. If the sound is generated directly from text [3], phonemes can be extracted directly and there is no need to process the sound track. However, when a direct mapping is performed one can choose from a variety of features. A non-complete list of possibilities include Perceptual Linear Prediction or J-Rasta-PLP as in [11, 8], Harmonics of Discrete Fourier Transform as in [12], Linear Prediction Coefficients as in [1] or Mel Frequency Cepstral Coefficients [2, 7, 6]. In this work the sound is split into 25 blocks per second (the same as the image frame rate) and 13 MFCC features are extracted from each block.

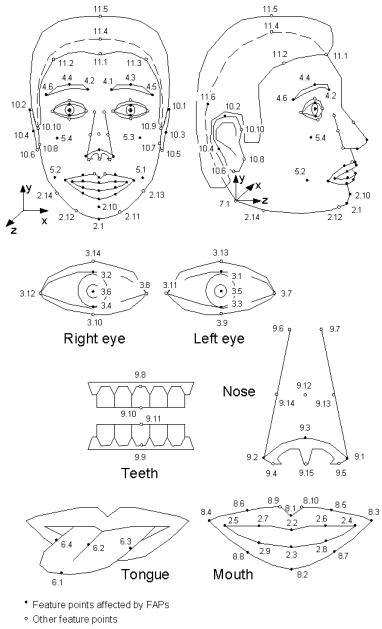


Fig. 1. Facial feature points (from www.research.att.com/projects/AnimatedHead)

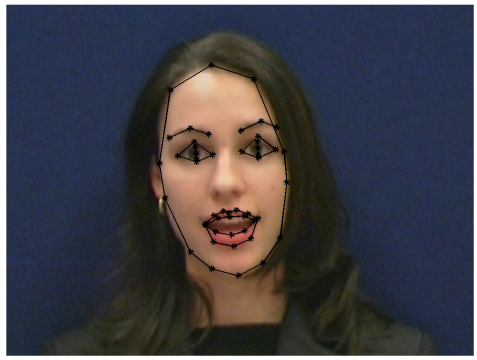


Fig. 2. Image with automatically extracted feature points. The facial feature points used are selected from the MPEG-4 standard (Fig. 1), points from main groups 2,3,4,8,10 and 11 are used.

To extract features from the images an Active Appearance model (AAM) [13] is used. The use of this model for lipreading has previously been studied by Mathews et al. [14]. In this work the implementation by Mikkel B. Stegman [15] is used. For the extraction a suitable subset of images in the training set is selected and annotated with points according to the MPEG-4 facial animation standard (Fig. 1). Using these annotations a 14-parameter model of the face is created. Thus, with 14 parameters it is possible to create a photo realistic image of any facial expression seen in the training set. Once the AAM is created the model is used to track the lip movements in the image sequences, at each point the 14 parameters are picked up. In Fig. 2 the result of the tracking is shown for a single representative image.

3 Model

Unlike most other approaches the mapping in this work is performed by a continuous state space model and not a Hidden Markov Model or a Neural Network. The reasoning behind this choice is that it should be possible to change the parameters controlling the face continuously (unlike in HMM) and yet make certain that all transitions happen smoothly (unlike NN's). Currently an experimental comparison of the performance of HMM's and the continuous state space models is investigated.

In this work the system is assumed to be linear and Gaussian and hence the Kalman Filter can be used [16]. This assumption is most likely not correct and other models like particle filtering and Markov Chain Monte Carlo are considered. However, as it will be shown below, even with the simplification the model produces useful results.

The model is set up as follows:

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{n}_k^x \tag{1}$$

$$\boldsymbol{y}_k = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{n}_k^y \tag{2}$$

$$\boldsymbol{z}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{n}_k^z \tag{3}$$

In this setting $\boldsymbol{z}_k$ is the image features at time k, $\boldsymbol{y}_k$ is the sound features and $\boldsymbol{x}_k$ is a hidden variable without physical meaning, but it can be thought of as some kind of brain activity controlling what is said. Each equation has i.i.d. Gaussian noise component $\boldsymbol{n}$ added to it.

During training both sound and image features are known, and the two observation equations can be collected in one.

$$\begin{pmatrix} \boldsymbol{y}_k \\ \boldsymbol{z}_k \end{pmatrix} = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}_k + \begin{pmatrix} \boldsymbol{n}_k^y \\ \boldsymbol{n}_k^z \end{pmatrix} \tag{4}$$

By using the EM algorithm [17, 18] on the training data, all parameters $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{\Sigma}^x, \boldsymbol{\Sigma}^y, \boldsymbol{\Sigma}^z\}$ can be found. $\boldsymbol{\Sigma}$'s are the diagonal covariance matrices of the noise components.

When a new sound sequence arrives Kalman filtering (or smoothing) can be applied to equations (1,2) to obtain the hidden state $\boldsymbol{x}$. Given $\boldsymbol{x}$ the corresponding image features can be obtained by multiplication, $\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k$. If the intermediate smoothing variables are available the variance on $\boldsymbol{y}_k$ can also be calculated.

4 Results

The data used is taken from the vidtimit database [19]. The database contains recordings of large number of people each uttering ten different sentences while facing the camera. The sound recordings are degraded by fan-noise from the recording pc. In this work a single female speaker is selected, thus 10 different sentences are used, nine for training and one for testing.

To find the dimension of the hidden state ($\boldsymbol{x}$), the optimal parameters ($\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{\Sigma}\}$) where found for varying dimensions. For each model the likelihood on the test sequence was calculated, the result is shown in Fig. 3.

With few dimensions the model is not rich enough to capture the dynamics of the image sequence. This is illustrated by the spectrogram of a hidden variable which represent the dynamics of the hidden space, as shown in Fig. 4(c). It is noted that only low frequency components are present. As the hidden space gets larger it becomes possible to model more of the dynamics present in the image. The spectrogram of a representative hidden variable when using a 25 dimensional

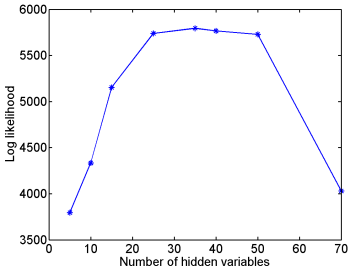
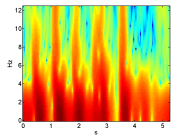
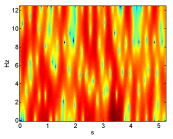


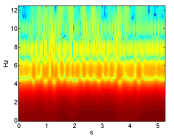
Fig. 3. The likelihood evaluated on the test data. With few hidden variables (dimesions in $\boldsymbol{x}$ space) the model is not rich enough. With too many parameters overfitting is experienced. An optimum is found in the range 25-40 hidden variables.



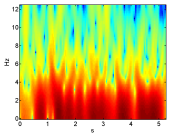
(a) Spectrogram of the first image feature.



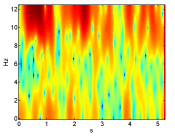
(b) Spectrogram of the first sound feature.



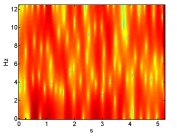
(c) Using five hidden units, only the low frequencies are captured.



(d) When using 25 states, more of the dynamic present in the image can be captured.



(e) When using 70 hidden units the states specializes.



(f) 70 hidden units part 2

Fig. 4. In the spectrograms of one of the predicted hidden states of on the test sequence, the effect of varying the size of the state space can be seen. Spectrograms of the first sound and image features are provided for comparison.

hidden space (Fig. 4(d)) has a structure very similar to what is found in one of the image features (Fig. 4(a)). When increasing the hidden units to 70, the model degrees of freedom becomes large and over fitting becomes possible. Fig. 4(e) and Fig. 4(f) show the spectrogram of two hidden variables and it is seen that the states specializes. In 4(e) high frequencies are dominant, and the other seemingly displays a structure, which resembles the dynamics of the sound features as seen in Fig. 4(b). This is not relevant due to the slower dynamics of the facial expressions. These specializations are furthermore specific to the training set and do not generalize according to Fig. 3. It should be noted that training a large model is difficult, both in terms of computations and convergence. With this analysis in mind a model with 25 hidden units is selected.

The test likelihood provides a measure of the quality of the model in feature space and provides a way of comparing models. This also allows comparison between this model and a similar Hidden Markov Model approach. However, it does not measure the quality of the final image sequence. No precise metric exist for evaluation of synthesized lip sequences. The distance between facial points in the true and the predicted image would be one way, another way would be to measure the distance between the predicted feature vector and the feature vector extracted from the true image. However, the ultimate evaluation of faces can be only provided by human interpretation. Unfortunately it is difficult to get an objective measure this way. One possibility would be to get a hearing impaired person to lipread the generated sequence, another to let people try to guess which sequence was real and which was computer generated. Unfortunately, such test are time and labor demanding and it has not been possible to perform them in this study.

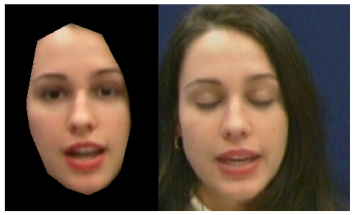
In Fig. 5 snapshots from the sequence are provided for visual inspection, the entire sequence is available at http://www.imm.dtu.dk/~tls/code/facedemo.php, where other demos can also be found.

5 Conclusion

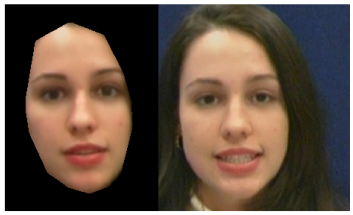
A speech to face mapping system relying on continuous state space models is proposed. The system makes it possible to easily train a unique face model that can be used to transform speech into facial movements. The training set must contain all sounds and corresponding face gestures, but there are no language or phonetic requirements to what the model can handle.

Surprisingly little attention has previously been paid to the training of state space models. In this paper it is shown that the Kalman filter is able overfit when the number of parameters are too large, similar effects are expected for the Hidden Markov Model.

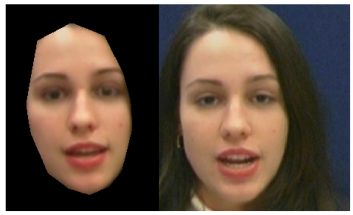
All though preliminary, the results are promising. Future experiments will show how the Kalman model and other instances of continuous state space models compares to Hidden Markov Model type systems.



(a)



(b)



(c)

Fig. 5. Characteristic images taken from the test sequence. The predicted face is to the left and the true face to the right.

References

1. Lewis, J.P.: Automated lip-sync: Background and techniques. J. Visualization and Computer Animation **2** (1991)
2. Goldenthal, W., Waters, K., Jean-Manuel, T.V., Glickman, O.: Driving synthetic mouth gestures: Phonetic recognition for faceme! In: Proc. Eurospeech '97, Rhodes, Greece (1997) 1995–1998
3. Ezzat, T., Poggio, T.: Mike talk: a talking facial display based on morphing visemes. Proc. Computer Animation IEEE Computer Society (1998) 96–102
4. Bregler, C., Covell, M., Slaney, M.: Video rewrite: driving visual speech with audio. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (1997) 353–360
5. Williams, J.J., Katsaggelos, A.K.: An hmm-based speech-to-video synthesizer. IEEE Transactions on Neural Networks **13** (2002)
6. Hong, P., Wen, Z., Huang, T.S.: Speech driven face animation. In Pandzic, I.S., Forchheimer, R., eds.: MPEG-4 Facial Animation: The Standard, Implementation and Applications. Wiley, Europe (2002)
7. Massaro, D.W., Beskow, J., Cohen, M.M., Fry, C.L., Rodriguez, T.: Picture my voice: Audio to visual speech synthesis using artificial neural networks. Proc. AVSP 99 (1999)
8. Brand, M.: Voice puppetry. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (1999) 21–28
9. Lavagetto, F.: Converting speech into lip movements: A multimedia telephone for hard of hearing people. IEEE Trans. on Rehabilitation Engineering **3** (1995)
10. McGurk, H., MacDonald, J.W.: Hearing lips and seeing voices. Nature **264** (1976) 746–748
11. Dupont, S., Luettin, J.: Audio-visual speech modelling for continuous speech recognition. IEEE Transactions on Multimedia (2000)
12. McAllister, D.F., Rodman, R.D., Bitzer, D.L., Freeman, A.S.: Speaker independence in automated lip-sync for audio-video communication. Comput. Netw. ISDN Syst. **30** (1998) 1975–1980
13. Cootes, T., Edwards, G., Taylor, C.: Active appearance models. Proc. European Conference on Computer Vision **2** (1998) 484–498
14. Matthews, I., Cootes, T., Bangham, J., Cox, S., Harvey, R.: Extraction of visual features for lipreading. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24** (2002) 198 –213
15. Stegmann, M.B., Ersbøll, B.K., Larsen, R.: FAME - a flexible appearance modelling environment. IEEE Transactions on Medical Imaging **22** (2003) 1319–1331
16. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering **82** (1960) 35–45
17. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. JRSSB **39** (1977) 1–38
18. Ghahramani, Z., Hinton, G.: Parameter estimation for linear dynamical systems. Technical report (1996) University of Toronto, CRG-TR-96-2.
19. Sanderson, C., Paliwal, K.K.: Polynomial features for robust face authentication. Proceedings of International Conference on Image Processing **3** (2002) 997–1000

Multimedia Mapping using Continuous State Space Models

Tue Lehn-Schiøler
The Technical University of Denmark
Informatics and Mathematical Modelling
Email: tls@imm.dtu.dk

***Abstract*— In this paper a system that transforms speech waveforms to animated faces are proposed. The system relies on a state space model to perform the mapping. To create a photo realistic image an Active Appearance Model is used. The main contribution of the paper is to compare a Kalman filter and a Hidden Markov Model approach to the mapping. It is shown that even though the HMM can get a higher test likelihood the Kalman filter is easier to train and the animation quality is better for the Kalman filter.**

I. Introduction

The motivation for transforming a speech signal into lip movements is at least threefold. Firstly, the language synchronization of movies often leaves the actors mouth moving while there is silence or the other way around, this looks rather unnatural. If it was possible to manipulate the face of the actor to match the actual speech it would be much more pleasant to view synchronized movies (and a lot easier to make cartoons). Secondly, even with increasing bandwidth sending images via the cell phone is quite expensive, therefore, a system that allows single images to be sent and models the face in between would be useful. The technique will also make it possible for hearing impaired people to lip read over the phone. If the person in the other end does not have a camera on her phone, a model image can be used to display the facial movements. Thirdly, when producing agents on a computer (like Windows Office Mr. clips) it would make communication more plausible if the agent could interact with lip movements corresponding to the (automatically generated) speech.

The idea of extracting phonemes or similar high-level features from the speech signal before performing the mapping to the mouth position has been widely used in the lip-sync community. Goldenthal (1) suggested a system called ”Face Me!”. He extracts phonemes using Statistical Trajectory Modeling. Each phoneme is then associated with a mouth position (keyframe). In Mike Talk (2), phonemes are generated from text and then mapped onto keyframes, however, in this system trajectories linking all possible keyframes are calculated in advance thus making the video more seamless. In ”Video rewrite” (3) phonemes are again extracted from the speech, in this case using Hidden Markov Models. Each triphone (three consecutive phonemes) has a mouth sequence associated with it. The sequences are selected from training data, if the triphone does not have a matching mouth sequence in the training data, the closest available sequence is selected. Once the sequence of mouth movements has been determined, the mouth is mapped back to a background face of the speaker. Other authors have proposed methods based on modeling of phonemes by correlational HMM's (4) or neural networks (5).

Methods where speech is mapped directly to facial movement are not quite as popular as phoneme based methods. However, in 'Picture my voice' (6), a time dependent neural network, maps directly from 11×13 Mel Frequency Cepstral Coefficients (MFCC) as input to 37 facial control parameters. The training output is provided by a phoneme to animation mapping but the trained network does not make use of the phoneme representation. Also Brand (7) has proposed a method based on (entropic) HMM's where speech is mapped directly to images. In (8) Nakamura presents an overview of methods using HMM's, the first MAP-V converts speech into the most likely HMM state sequence and the uses a table lookup to convert into visual parameters. In an extended version MAP-EM the visual parameters are estimated using the EM algorithm. Methods that do not rely on phoneme extraction has the advantage that they can be trained to work on all languages, and that they are able to map non-speech sounds like yawning and laughing.

There are certain inherent difficulties in mapping from speech to mouth positions an analysis of these can be found in (9). The most profound is the confusion between visual and auditive information. The mouth position of sounds like /b/,/p/ and /m/ or /k/,/n/ and /g/ can not be distinguished even though the sounds can. Similarly the sounds of /m/ and /n/ or /b/ and /v/ are very similar even though the mouth position is completely different. This is perhaps best illustrated by the famous experiment by McGurk (10). Thus, when mapping from speech to facial movements, one cannot hope to get a perfect result simply because it is very difficult to distinguish whether a ”ba” or a ”ga” was spoken.

The rest of this paper is organized in three sections, section II focuses on feature extraction in sound and images, in section III the model are described. Finally experimental results are presented in section IV.

II. Feature extraction

Many different approaches has been taken for extraction of sound features. If the sound is generated directly from text phonemes can be extracted directly and there is no need to process the sound track (2). However, when a direct mapping is performed one can choose from a variety of features. A non-complete list of possibilities include Perceptual Linear

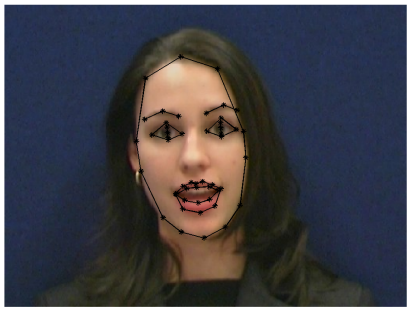


Fig. 1. Image with automatically extracted feature points. The facial feature points are selected from the MPEG-4 standard

Prediction or J-Rasta-PLP as in (7; 11), Harmonics of Discrete Fourier Transform as in (12), Linear Prediction Coefficients as in (13) or Mel Frequency Cepstral Coefficients (1; 5; 6; 8). In this work the sound is split into 25 blocks per second (the same as the image frame rate) and 13 MFCC features are extracted from each block. To extract features from the images an Active Appearance model (AAM) (14) is used. The use of this model for lipreading has previously been studied by Mathews et al. (15). AAM's are also useful for low bandwidth transmission of facial expressions (16). In this work an implementation by Mikkel B. Stegman (17) is used. For the extraction a suitable subset of images in the training set are selected and annotated with points according to the MPEG-4 facial animation standard. Using these annotations a 14-parameter model of the face is created. Thus, with 14 parameters it is possible to create a photo realistic image of any facial expression seen in the training set. Once the AAM is created the model is used to track the lip movements in the image sequences, at each point the 14 parameters are picked up. In Fig. 1 the result of the tracking is shown for a single representative image.

III. Model

In this work the mapping from sound to images is performed by two types of state space models, a HMM with a mixture of Gaussians observations and a Kalman filter. Both approaches uses the toolbox written by Kevin Murphy (http://www.ai.mit.edu/ murphyk/Software).

Normally, when using HMM's for speech to face-movement mapping a bank of HMM's are used. Each one is trained on a specific subset of data and when that model has the highest likelihood it is responsible for producing the image. In this work the entire sequence is considered at once and only a single state space model is trained. In case of the Kalman filter the model set up is as follows:

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{n}_k^x \quad (1)$$

$$\boldsymbol{s}_k = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{n}_k^s \quad (2)$$

$$\boldsymbol{i}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{n}_k^i \quad (3)$$

In this setting $\boldsymbol{i}_k$ is the image features at time k, $\boldsymbol{s}_k$ is the sound features and $\boldsymbol{x}_k$ is a hidden variable without physical meaning. $\boldsymbol{x}$ can be thought of as some kind of brain activity controlling what is said. Each equation has i.i.d. Gaussian noise component $\boldsymbol{n}$ added to it.

During training both sound and image features are known, and the two observation equations can be collected in one.

$$\begin{pmatrix} \boldsymbol{s}_k \\ \boldsymbol{i}_k \end{pmatrix} = \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}_k + \begin{pmatrix} \boldsymbol{n}_k^s \\ \boldsymbol{n}_k^i \end{pmatrix} \quad (4)$$

By using the EM algorithm (18; 19) on the training data, all parameters $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{\Sigma}^x, \boldsymbol{\Sigma}^s, \boldsymbol{\Sigma}^i\}$ can be found. $\boldsymbol{\Sigma}$'s are the diagonal covariance matrices of the noise components.

When a new sound sequence arrives Kalman filtering (or smoothing) can be applied to equations (1,2) to obtain the hidden state $\boldsymbol{x}$. Given $\boldsymbol{x}$ the corresponding image features can be obtained by multiplication, $\boldsymbol{i}_k = \boldsymbol{C}\boldsymbol{x}_k$. If the intermediate smoothing variables are available the variance on $\boldsymbol{i}_k$ can also be calculated.

In case of the Hidden Markov Model the approach is similar, the transition probabilities, the emission probabilities for the sound and image features and the Gaussian mixture parameters are estimated during training. During testing the most probable state sequence can be found from the sound features and the image feature can be found using either the mean of the emitted Gaussian or by drawing a sample from it.

IV. Results

The data used is taken from the vidtimit database (20). The database contains recordings of large number of people each uttering ten different sentences while facing the camera. The sound recordings are degraded by fan-noise from the recording pc. In this work a single female speaker is selected, thus 10 different sentences are used, nine for training and one for testing.

To find the dimension of the hidden state ($\boldsymbol{x}$), the optimal parameters for both the KF and the HMM were found for varying dimensions. For each model the likelihood on training and test sequences were calculated, the result is shown in Fig. 2 and Fig. 3.

The test likelihood provides a statistical measure of the quality of the model and provides a way of comparing models. This allows comparison between the KF and the HMM approach. Unfortunately the likelihood is not necessarily a good measure of the quality of a model prediction. If the distributions in the model are broad, i.e. the model has high uncertainty, it can describe data well, but, it is not a good generative model.

Looking at the results in Fig. 2 and Fig. 3 it is seen that the likelihood of a HMM does not increase as expected with the model complexity. The KF on the other hand has a peak in the test likelihood around 40 hidden states. The test likelihood shows that the HMM is a better model than KF. However when examining the output feature vectors controlling the face movement (Fig. 4) it is seen that the output of the HMM is varying very fast and does not follow the true feature vector. The output from the KF on the other hand is smooth and closer to the desired. Visual inspection of the video sequence shows good results from the

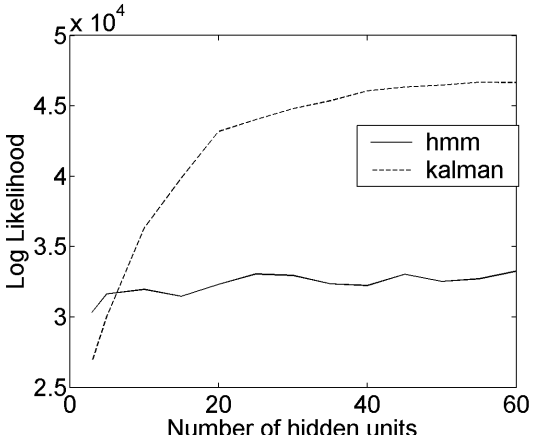


Fig. 2. The likelihood evaluated on the training data. The Kalman filter is able to utilize the extra dimension to improve the training result, whereas the HMM has almost the same performance no matter how many hidden states are used.

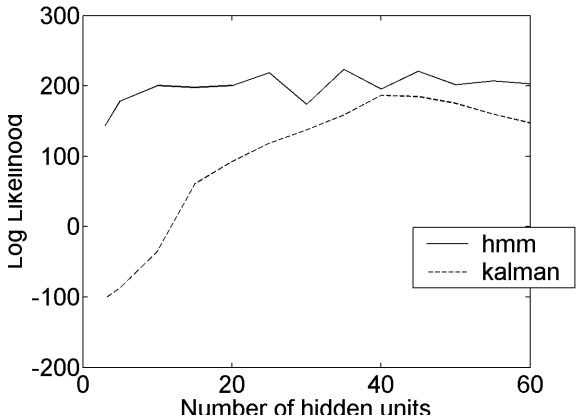


Fig. 3. The likelihood evaluated on the test data. Again the Kalman filter improves performance as more hidden dimensions are added and overfitting is seen for high number of hidden states. The HMM has the same performance independent of the number of hidden states.

KF but very jerky and unrealistic motion from the HMM. In Fig. 5 snapshots from the KF sequence are provided for visual inspection, the entire sequence is available at http://www.imm.dtu.dk/˜tls/code/facedemo.php, where other demos can also be found.

The failure of the likelihood to capture the quality of the final image sequence points to an interesting problem. No precise metric exist for evaluation of synthesized lip sequences. The distance between facial points in the true and the predicted image would be one way, another way would be to measure the distance between the predicted feature vector and the feature vector extracted from the true image. However, the ultimate evaluation of faces can be only provided by human interpretation. Unfortunately it is difficult to get an objective measure this way. One possibility would be to get a hearing impaired person to lipread the generated sequence, another to let people try to guess which sequence was real and which was computer generated. Unfortunately, such test are time and labor demanding. Further more these subjective test does not provide an error function that can be optimized directly.

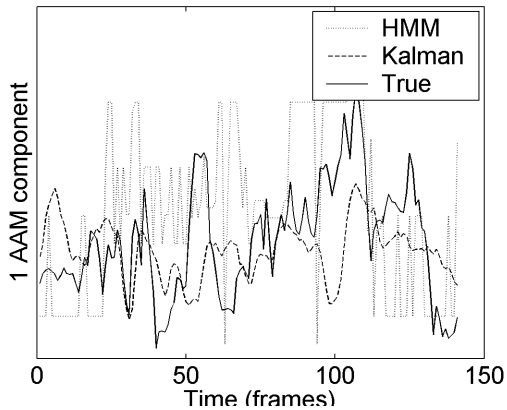


Fig. 4. Dynamics of the first AAM component true and predicted. The prediction from the Kalman model is smooth, but does not follow the curve completely. The HMM solution varies faster indicating that the uncertainty in the model is greater. On top of that the discrete nature of the model makes it jumps suddenly from frame to frame making the face movement look jerky.

V. Conclusion

A speech to face mapping system relying on state space models is proposed. The system makes it possible to easily train a unique face model that can be used to transform speech into facial movements. The training set must contain all sounds and corresponding face gestures, but there are no language or phonetic requirements to what the model can handle.

In this approach a single model is used for the entire sequence, making the problem one of system identification. For this task the Hidden Markov Model seem to be clearly inferior to the Kalman filter based on inspection of the output video. The likelihood of the HMM however shows that it is the better model. This confirms the suspicion that better error measures are needed for evaluation of lip-sync quality.

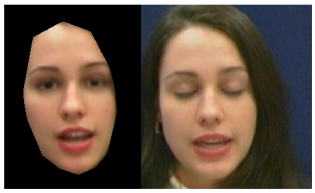
Acknowledgements

The work is supported by the European Commission through the sixth framework IST Network of Excellence: Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL), contract no. 506778.

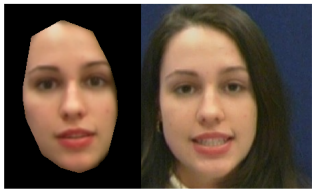
References

[1] W. Goldenthal, K. Waters, T. V. Jean-Manuel, and O. Glickman, “Driving synthetic mouth gestures: Phonetic recognition for faceme!” in *Proc. Eurospeech ’97*, Rhodes, Greece, 1997, pp. 1995–1998.

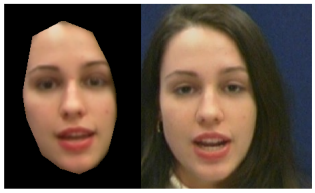
[2] T. Ezzat and T. Poggio, “Mike talk: a talking facial



(a)



(b)



(c)

Fig. 5. Characteristic images taken from the test sequence when using the Kalman filter. The predicted face is to the left and the true face to the right.

display based on morphing visemes," *Proc. Computer Animation IEEE Computer Society*, pp. 96–102, 1998.

[3] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: driving visual speech with audio," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 353–360.

[4] J. J. Williams and A. K. Katsaggelos, "An hmm-based speech-to-video synthesizer," 1998.

[5] P. Hong, Z. Wen, and T. S. Huang, "Speech driven face animation," in *MPEG-4 Facial Animation: The Standard, Implementation and Applications*, I. S. Pandzic and R. Forchheimer, Eds. Wiley, Europe, July 2002.

[6] D. W. Massaro, J. Beskow, M. M. Cohen, C. L. Fry, and T. Rodriguez, "Picture my voice: Audio to visual speech synthesis using artificial neural networks," *Proc. AVSP 99*, 1999.

[7] M. Brand, "Voice puppetry," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 21–28.

[8] S. Nakamura, "Statistical multimodal integration for audio-visual speech processing," *IEEE Transactions on Neural Networks*, vol. 13, no. 4, july 2002.

[9] L. F., "Converting speech into lip movements: A multimedia telephone for hard of hearing people," *IEEE Trans. on Rehabilitation Engineering*, vol. 3, no. 1, 1995.

[10] H. McGurk and J. W. MacDonald, "Hearing lips and seeing voices," *Nature*, vol. 264, pp. 746–748, 1976.

[11] S. Dupont and J. Luettin, "Audio-visual speech modelling for continuous speech recognition," *IEEE Transactions on Multimedia*, 2000.

[12] D. F. McAllister, R. D. Rodman, D. L. Bitzer, and A. S. Freeman, "Speaker independence in automated lip-sync for audio-video communication," *Comput. Netw. ISDN Syst.*, vol. 30, no. 20-21, pp. 1975–1980, 1998.

[13] J. P. Lewis, "Automated lip-sync: Background and techniques," *J. Visualization and Computer Animation*, vol. 2, 1991.

[14] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *Proc. European Conference on Computer Vision*, vol. 2, pp. 484–498, 1998.

[15] I. Matthews, T. Cootes, J. Bangham, S. Cox, and R. Harvey, "Extraction of visual features for lipreading," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 2, pp. 198 –213, 2002.

[16] B. Theobald, S. Kruse, J. Bangham, and G. Cawley, "Towards a low bandwidth talking face using appearance models," *Image and Vision Computing*, vol. 21, no. 13-14, pp. 1117–1124, 2003.

[17] M. B. Stegmann, "Analysis and segmentation of face images using point annotations and linear subspace techniques," Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Tech. Rep., Aug. 2002, http://www.imm.dtu.dk/pubdb/p.php?922.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *JRSSB*, vol. 39, pp. 1–38, 1977.

[19] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," Tech. Rep., 1996, university of Toronto, CRG-TR-96-2.

[20] C. Sanderson and K. K. Paliwal, "Polynomial features for robust face authentication," *Proceedings of International Conference on Image Processing*, vol. 3, pp. 997–1000, 2002.

[21] T. Lehn-Schioler, L. K. Hansen, and J. Larsen, "Mapping from speech to images using continuous state space models," in *Joint AMI/PASCAL/IM2/M4 Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, Apr. 2004.

Bibliography

Aleksic, P. S. and Katsaggelos, A. K. (2003). Speech-to-video synthesis using facial animation parameters. *ICIP*.

Aleksic, P. S. and Katsaggelos, A. K. (2004). Speech-to-video synthesis using mpeg-4 compliant visual features. *IEEE Trans. Circuits and Systems for Video Technology*, 14(5):682–692.

Aleksic, P. S., Williams, J. J., Wu, Z., and Katsaggelos, A. K. (2002). Audio-visual continuous speech recognition using mpeg-4 compliant visual features. *IEEE International Conference on Image Processing*, 1:I/960–I/963.

Alspach, D. L. and Sorensen, H. W. (1972). Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448.

Andrieu, C. and Doucet, A. (2003). Online expectation-maximization type algorithms for parameter estimation in general state space models. In *Proc. IEEE ICASSP*.

Arslan, L. M. and Talkin, D. (1998). 3-d face point trajectory synthesis using an automatically derived visual phoneme similarity matrix. *Auditory-Visual Speech Processing*, pages 175–180.

Arulampalam, S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Bengio, S. (2004). Multimodal speech processing using asynchronous hidden markov models. *Information Fusion*, 5(2):81–89.

Bengio, Y. and Bengio, S. (1996). Training asynchronous input/output hidden markov models. In *Spring Symposium on Computational Issues in Learning Models of Dynamical Systems*.

Bergeron, P. and Lachapelle, P. (1985). Controlling facial expressions and body movements in the computer generated animated short tony de peltrie. *SIGGRAPH 1985*.

Bermond, O. and Cardoso, J.-F. (1999). Approximate likelihood for noisy mixtures. In *Proceedings of the ICA Conference*, pages 325–330.

Bernstein, L. E. and Eberhardt, S. (1986). Johns hopkins lipreading corpus videodisk set.

Beskow, J. (2003). *Talking Heads - Models and Applications for Multimodal Speech Synthesis*. PhD thesis.

Bettinger, F. and Cootes, T. (2004). A model of facial behaviour. *Proc. Int.Conf on Face and Gesture Recognition*, pages 123–128.

Bettinger, F., Cootes, T. F., and C.J., T. (2002). Modelling facial behaviours. *Proc. BMVC2002*, 2:797–806.

Bishop, C. M., Svensen, M., and Williams, C. K. I. (1996). GTM: a principled alternative to the self-organizing map. In von der Malsburg, C., von Seelen, W., Vorbruggen, J. C., and Sendhoff, B., editors, *Artificial Neural Networks—ICANN 96. 1996 International Conference Proceedings*, pages 165–70. Springer-Verlag, Berlin, Germany.

Brand, M. (1999). Voice puppetry. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 21–28. ACM Press/Addison-Wesley Publishing Co.

Brand, M., Oliver, N., and Pentland, A. (1997). Coupled hidden markov models for complex action recognition. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 994. IEEE Computer Society.

Bregler, C., Covell, M., and Slaney, M. (1997). Video rewrite: driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 353–360. ACM Press/Addison-Wesley Publishing Co.

Campbell, R., Dodd, B., and Burnham, D., editors (1998). *Hearing by Eye II: Advances in the Psychology of Speechreading and Auditory-visual Speech*. Psychology Press Ltd., East Sussex, UK.

Carlin, B., Polson, N. G., and Stoffer, D. (1992). A monte carlo approach to nonnormal and non-linear state-space modelling. *Journal of the American Statistical Association*, 87(418):493–500.

Chen, T. (2001). Audiovisual speech processing. *IEEE Signal Processing Magazine*, 18(1):9 –21.

Chen, T., Graf, H. P.and Chen, H., Chou, W., Haskell, B. G., Petajan, E., and Y., W. (1995). Lip synchronization in talking head video utilizing speech information. *Proc. SPIE Conf. Visual Communication and Image Processing*.

Cohen, M. and Massaro, D. (1994). Simulation modeling: Development and experimentation with synthetic visible speech. *Behavior Research Methods, Instruments and Computers*, 26:260–265.

Cohen, M. and Massaro, D. . C. R. (2002). Training a talking head. *IEEE Fourth International Conference on Multimodal Interfaces, (ICMI'02)*.

Cootes, T., Edwards, G., and Taylor, C. (1998). Active appearance models. *Proc. European Conference on Computer Vision*, 2:484–498.

Cosker, D., Marshall, A., Rosin, P., and Hicks, Y. (2004). Speech-driven facial animation using a hierarchical model. *Vision, Image and Signal Processing, IEE Proceedings-*, 151(4):314–321.

Cowie, R., Douglas-Cowie, E. Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., and Taylor, J. G. (2001). Emotion recognition in human-computer interaction. *IEEE Signal Processing Magazine*, 18(1):32–80.

DeGraph, B. and Wahrman, M. (1998). Mike, the talking head. *Computer Graphics*, 11:15–17.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *JRSSB*, 39:1–38.

Devroye, L. and Lugosi, G. (2001). *Combinatorial Methods in Density Estimation*. Springer, New York.

Digalakis, V., Rohlicek, J., and Ostendorf, M. (1993). ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442.

Dodd, B. and Campbell, R. (1987). *Hearing by Eye: The Psychology of Lip-Reading*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Du, Y. and Lin, X. (2002). Realistic mouth synthesis based on shape appearance dependence mapping. *PRL*, 23(14):1875–1885.

Dupont, S. and Luettin, J. (2000). Audio-visual speech modelling for continuous speech recognition. *IEEE Transactions on Multimedia*.

Durbin, R. and Willshaw, D. (1987). An analogue approach of the travelling salesman problem using an elastic net method. *Nature,*, 326:689–691.

Eisert, P., Chaudhuri, S., and Girod, B. (1997). Speech driven synthesis of talking head sequences. *3D Image Analysis and Synthesis*, pages 51–56.

Erdogmus, D. and Principe, J. C. (2002). Generalized information potential criterion for adaptive system training. *IEEE Transactions on Neural Networks*, 13(5).

Erwin, E., Obermayer, K., and Schulten, K. (1992). Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:4755.

Ezzat, T., Geiger, G., and Poggio, T. (2002). Trainable videorealistic speech animation. *Proceedings of ACM SIGGRAPH 2002*.

Ezzat, T. and Poggio, T. (1998). Mike talk: a talking facial display based on morphing visemes. *Proc. Computer Animation IEEE Computer Society*, pages 96–102.

Fedorov, A., Firsova, T., Kuriakin, V., Martinova, E., Rodyushkin, K., and Zhislina, V. (2003). Talking head: Synthetic video facial animation in mpeg-4. *Proceedings of graphicon*.

Feng, L. (2004). Speaker recognition. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby. Supervised by Prof. Lars Kai Hansen.

Ferkinghoff-Borg, J. (2002). *Monte Carlo Methods in Complex Systems*. PhD thesis, Graduate School of Biophysics, Niels Bohr Institute and Risø National Laboratory, Faculty of Science University of Copenhagen.

Ferkinhoff-Borg, J., Lehn-Schiøler, T., and Winther, O. (2005). Failure of particle filters (unpublished).

Fong, W., Godsill, S., Doucet, A., and West, M. (2002). Monte carlo smoothing with application to audio signal enhancement. *IEEE Trans. on Signal Processing*, 50(2):438–448.

Geiger, G., Ezzat, T., and Poggio, T. (2003). Perceptual evaluation of videorealistic speech. In *MIT AIM*.

Gevers, M. and Wertz, V. (1984). Uniquely identifiable state-space and arma parametrizations for multivariable linear systems. *Automatica*, 20(3):333–347.

Ghahramani, Z. and Hinton, G. (1996). Parameter estimation for linear dynamical systems. Technical report.

Ghahramani, Z. and Jordan, M. I. (1995). Factorial hidden Markov models. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Proc. Conf. Advances in Neural Information Processing Systems*, volume 8, pages 472–478. MIT Press.

Goff, B., Guiard-Marigny, T., Cohen, M., and Benoit, C. (1994). Real-time analysis-synthesis and intelligibility of talking faces. *Proceedings of the Second ESCA/IEEE Workshop on Speech Synthesis*, pages 53–56.

Goldenthal, W., Waters, K., Jean-Manuel, T. V., and Glickman, O. (1997). Driving synthetic mouth gestures: Phonetic recognition for faceme! In *Proc. Eurospeech '97*, pages 1995–1998, Rhodes, Greece.

Gordon, N., Salmond, D., and Smith, A. F. M. (1993). Novel approach to non-linear and non-gaussian bayesian state estimation. *IEE Proceedings-F*, 140:107–113.

Goto, T., Kshirsagar, S., and Magnenat-Thalmann, N. (2001). Automatic face cloning and animation using real-time facial feature tracking and speech acquisition. *IEEE Signal Processing Magazine*, 18(3):17 –25.

Gross, R., Matthews, I., and Baker, S. (2004). Generic vs. person specific active appearance models. In *British Machine Vision Conference*.

Gupta, N. K. and Mehra, R. K. (1974). Computational aspects of maximum likelihood estimation and reduction in sensitivity calculations. *IEEE Transactions on Automatic Control*, AC-19(1):774–783.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109.

Hegde, A., Erdogmus, D., Lehn-Schiøler, T., Rao, Y., and Principe, J. (2004). Vector-quantization by density matching in the minimum kullback-leibler divergence sense. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 1, pages 105–109.

Heskes, T. (1999). Energy functions for self-organizing maps. In S. Oja, E. and Kaski, editors, *Kohonen Maps*, pages 303–316. Elsevier, Amsterdam.

Heskes, T. and Kappen, B. (1993). Error potentials for self-organization. *Proceedings IJCNN93*, 3:1219–1223.

Heskes, T. and Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic bayesian networks. *Proceedings UAI*, pages 216–223.

Hill, D., Pearce, A., and Wyvill, B. (1988). Animating speech: an automated approach using speech synthesised by rules. *Visual Computer*, 3(5):277–89.

Højen-Sørensen, P. A., Winther, O., and Hansen, L. K. (2002). Mean field approaches to independent component analysis. *Neural Computation*, 14:889–918.

Hong, P., Wen, Z., and Huang, T. S. (2002). Speech driven face animation. In Pandzic, I. S. and Forchheimer, R., editors, *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. Wiley, Europe.

Hulle, M. M. V. (2002). Kernel-based topographic map formation achieved with an information-theoretic approach. *Neural Networks*, 15(8-9):1029–1039.

Hurzeler, M. and Kunsch, H. (1998). Monte carlo approximations for general state space models. *Journal of Computational and Graphical Statistics*, 7(2):175–193.

Jamshidian, M. and Jennrich, R. I. (1997). Acceleration of the EM algorithm using quasi-newton methods. *Journal of Royal Statistical Society, Series B*, 59(3):569–587.

Jones, M. J. and Poggio, T. (1998). Multidimensional morphable models. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 683. IEEE Computer Society.

Julier, S. and Uhlmann, J. (1997). A new extension of the kalman filter to nonlinear systems. *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*.

Kakumanu, P., Gutierrez-Osuna, R., Esposito, A., Bryll, R., Goshtasby, A., and Garcia, O. (2001). Speech driven facial animation. In *Proceedings of the ACM conference on Perceptual User Interfaces*.

Kalberer, G., Muller, P., and Van Gool, L. (2003). Visual speech, a trajectory in viseme space. *IJIST*, 13(1):74–84.

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

Karlsson, I., Faulkner, A., and Salvi, G. (2003). Synface - a talking face telephone. *Proc of EuroSpeech 2003*.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biol. Cybern.*, 43:59–69.

Kotecha, J. and Djuric, P. (2001). Gaussian sum particle filtering for dynamic state space models. *Acoustics, Speech, and Signal Processing, 2001. Proceedings. 2001 IEEE International Conference on*, 6:3465 –3468 vol.6.

Kshirsagar, S. and Magnenat-Thalmann, N. (2000). Lip synchronization using linear predictive analysis. In *ICME00*, page TP8.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86.

Lachlan, G. J. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. John Wiley and Sons.

Lavagetto, F. (1995). Converting speech into lip movements: A multimedia telephone for hard of hearing people. *IEEE Trans. on Rehabilitation Engineering*, 3(1).

Lavagetto, F. (1997). Time-delay neural networks for estimating lip movements from speech analysis: A useful tool in audio video synchronization. *CirSysVideo*, 7(5):786–800.

Lehn-Schiøler, T. (2004a). Multimedia mapping using continuous state space models. In *IEEE 6'th orkshop on Multimedia Signal Processing Proceedings*, pages 51–54.

Lehn-Schiøler, T. (2004b). Talking faces a state space approach. In Olsen, S. I., editor, *Proceedings of DSAGM*, pages 103–111.

Lehn-Schiøler, T., Erdogmus, D., and Principe, J. C. (2004). Parzen particle filters. *Proc. ICASSP*, 5:781–784.

Lehn-Schiøler, T., Hansen, L. K., and Larsen, J. (2005a). Mapping from speech to images using continuous state space models. In Bengio, S., editor, *Lecture Notes in Computer Science*, volume 3361, pages 136 – 145. Springer.

Lehn-Schiøler, T., Hegde, A., Erdogmus, D., and Principe, J. C. (2005b). Vector-quantization using information theoretic concepts. *Natural Computing*, 4:39–51.

Lepsoy, S. and Curinga, S. (1998). Conversion of articulatory parameters into active shape model coefficients for lip motion representation and synthesis. *SP:IC*, 13(3):209–225.

Lewis, J. P. (1991). Automated lip-sync: Background and techniques. *J. Visualization and Computer Animation*, 2.

Lewis, J. P. and Parke, F. I. (1987). Automated lip-synch and speech synthesis for character animation. In *Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, pages 143–147. ACM Press.

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Trans Commun COM*, 28:84–95.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, 1:281–297.

Massaro, D. and Cohen, M. (1990). Synthesis of visible speech. *Behavior Research Methods, Instruments and Computers*, 22:260–263.

Massaro, D. W., Beskow, J., Cohen, M. M., Fry, C. L., and Rodriguez, T. (1999). Picture my voice: Audio to visual speech synthesis using artificial neural networks. *Proc. AVSP 99*.

Massaro, D. W. and Stork, D. G. (1998). Speech recognition and sensory integration - people and machines integrate auditory and visual information to understand speech. *American Scientist*, 86(3):236–244.

Matthews, I. and Baker, S. (2004). Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135 – 164.

Matthews, I., Bangham, J., Harvey, R., and Cox, S. (1998). A comparison of active shape model and scale decomposition based features for visual speech recognition. In *ECCV98*.

Matthews, I., Cootes, T., Bangham, J., Cox, S., and Harvey, R. (2002). Extraction of visual features for lipreading. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):198 –213.

McAllister, D. F., Rodman, R. D., Bitzer, D. L., and Freeman, A. S. (1998). Speaker independence in automated lip-sync for audio-video communication. *Comput. Netw. ISDN Syst.*, 30(20-21):1975–1980.

McGurk, H. and MacDonald, J. W. (1976). Hearing lips and seeing voices. *Nature*, 264:746–748.

McKelvey, T., Helmersson, A., and Ribarits, T. (2004). Data driven local coordinates for multivariable linear systems and their application to system identification. *Automatica*, 40(9):1629–1635.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions Royal Society London, A*, 209:415–446.

Merwe, R. v. d., de Freitas, N., Doucet, A., and Wan, E. (2001). The unscented particel filter. In *Advances in Neural Information Processing Systems 13*.

Merwe, R. v. d. and Wan, E. (2003). Gaussian mixture sigma-point particle filters for sequential probabilistic inference in dynamic state-space models. *Speech and Signal Processing (ICASSP)*.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc.

Morishima, S., Aizawa, K., and Harashima, H. (1989). An intelligent facial image coding driven by speech and phoneme. *ICASSP*, pages 1795– 1798.

Morishima, S. and Harashima, H. (1991). A media conversion from speech to facial image for intelligent man-machine interface. *IEEE Journal on Selected Areas in Communications*, 9(4):594–599.

Moulines, E., Cardoso, J., and Cassiat, E. (1997). Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Proc. ICASSP*, volume 5, pages 3617–3620. The MIT Press.

Nakamura, S. (2002). Statistical multimodal integration for audio-visual speech processing. *IEEE Transactions on Neural Networks*, 13(4).

Neal, R. M. and Hinton, G. E. (1998). *Learning in Graphical Models: A view of the EM algorithm that justifies incremental, sparse, and other variants*, pages 355–368. Dordrecht: Kluwer Academic Publishers.

Nielsen, H. B. (2000). Ucminf - an algorithm for unconstrained nonlinear optimization. Technical Report IMM-Rep-2000-19, Technical University of Denmark.

Noh, J.-y. and Neumann, U. (2000). Talking faces. In *IEEE International Conference on Multimedia and Expo (II)*, pages 627–630.

Olsson, R. K. and Hansen, L. K. (2004). Probabilistic blind deconvolution of non-stationary sources. In *12th European Signal Processing Conference*, pages 1697–1700.

Olsson, R. K. and Hansen, L. K. (2005). A harmonic excitation state-space approach to blind separation of speech. In *Advances in Neural Information Processing Systems*. accepted.

Olsson, R. K., Petersen, K. B., and Lehn-Schiøler, T. (2005). State-space models - from the EM algorithm to a gradient approach. *Submitted to Neural Computation*. submitted.

Overbeek, A. v. and Ljung, L. (1982). On-line structure selection for multivariable state space models. *Automatica*, 18(5):529–543.

Pandzic, I. S., Ostermann, J., and Millen, D. (1999). User evaluation: synthetic talking faces for interactive services. *Visual Computer*, 15:330–340.

Parke, F. I. (1975). A model for human faces that allows speech synchronized animation. *Journal of Computers and Graphics*, 1(1):1–4.

Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Stat*, 27:1065–1076.

Pearce, A., Wyvill, B., Wyvill, G., and Hill, D. (1986). Speech and expression: a computer solution to face animation. *Proceedings of Graphics Interface '86 and Vision Interface '86*, pages 136–40.

Petajan, E. (2000). Approaches to visual speech processing based on the mpeg-4 face animation standard. *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, pages 575–8 vol.1.

Petajan, E. and Graf, H. P. (1996). Robust face feature analysis for automatic speechreading and character animation. *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 357–362.

Petersen, K. B. (2004). The matrix cookbook. http://2302.dk/uni/matrixcookbook.html.

Petersen, K. B. and Winther, O. (2005). The EM algorithm in independent component analysis. In *International Conference on Acoustics, Speech, and Signal Processing*.

Potamianos, G., Neti, C., Gravier, G., Garg, A., and Senior, A. (2003). Recent advances in the automatic recognition of audio-visual speech. *Proceedings of the IEEE*, 91(9):1306–1326.

Principe, J. C., Xu, D., Zhao, Q., and Fisher, J. (2000). Learning from examples with information theoretic criteria. *Journal of VLSI Signal Processing-Systems*, 26(1-2):61–77.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Redner, R. A. and Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 2(26):195–239.

Rényi, A. (1970). *Probability Theory*. North-Holland Publishing Company, Amsterdam.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–45.

Roweis, S. and Ghahramani, Z. (2001). *Learning Nonlinar Dynamical Systems using the EM Algorithm.*, pages 175–220. Wiley.

Salakhutdinov, R., Roweis, S. T., and Ghahramani, Z. (2003). Optimization with EM and Expectation-Conjugate-Gradient. In *International Conference on Machine Learning*, volume 20, pages 672–679.

Sandell, N. R. and Yared, K. I. (1978). Maximum likelihood identification of state space models for linear dynamic systems. Technical Report Technical Report ESL-R-814, Electron. Syst. Lab, M.I.T.

Sanderson, C. (2002a). *Automatic Person Verification Using Speech and Face Information*. PhD thesis, School of Microelectronic Engineering, Griffith University.

Sanderson, C. (2002b). The VidTIMIT Database. IDIAP-Com 06, IDIAP.

Sanderson, C. and Paliwal, K. K. (2002). Information fusion and person verification using speech & face information.

Scofield, C. L. (1988). Unsupervised learning in the n-dimensional coulomb network. *Neural Networks*, 1(1):129.

Shephard, N. (1994). Partial non-gaussian state space. *Biometrika*, 81(1):115–131.

Shumway, R. and Stoffer, D. (1982). An approach to time series smoothing and forecasting using the em algorithm. *Journal of Time Series Analysis*, 3:253–264.

Stegmann, M. B. (2002). Analysis and segmentation of face images using point annotations and linear subspace techniques. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby. http://www.imm.dtu.dk/pubdb/p.php?922.

Stegmann, M. B., Ersbøll, B. K., and Larsen, R. (2003). FAME - a flexible appearance modelling environment. *IEEE Transactions on Medical Imaging*, 22(10):1319–1331.

Stork, D. G., Wolff, G., and Levine, E. (1992). Neural network lipreading system for improved speech recognition. *Proceedings of the 1992 International Joint Conference on Neural Networks, Baltimore, MD.*

Sumby, W. and Pollack, I. (1954). Visual contribution to speech intelligibility in noise. *J. Acoust. Soc. Am.*, 26:212–215.

Tanizaki, H. and Mariano, R. (2000). Nonlinear and non-gaussian state-space modeling with monte-carlo simulations. *Journal of Econometrics*, pages 263–290.

Taylor, J., Bitzer, D., Rodman, R.and McAllister, D., and Wang, M. (2001). Speaker independence in lip synchronization of vowels and distinguishing between /m/ and /n/. *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001) and the 7th International Conference on Information Systems Analysis and Synthesis (ISAS 2001).*

Theobald, B., Bangham, J., Matthews, I., and Cawley, G. (2004). Near-videorealistic synthetic talking faces: Implementation and evaluation. *Speech Communication Journal.*

Theobald, B., Kruse, S., Bangham, J., and Cawley, G. (2003). Towards a low bandwidth talking face using appearance models. *Image and Vision Computing*, 21(13-14):1117–1124.

Tiddeman, B. and Perrett, D. (2002). Prototyping and transforming visemes for animated speech. In *Computer Animation*, pages 248–251.

Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., and Kitamura, T. (2000). Speech parameter generation algorithms for hmm-based speech synthesis. *Proceedings of ICASSP 2000*, 3:1315–1318.

Valpola, H. and Karhunen, J. (2002). An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Comput.*, 14(11):2647–2692.

Van Overschee, P. and de Moor, B. (1996). *Subspace Identification for Linear Systems: Theory, Implementation, Applications.* Kluwer Academic Publishers, Boston.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag, New-York.

Vatikiotis-Bateson, E., Kroos, C., Kuratate1, T., Munhall, K. G., Rubin, P., and Yehia, H. C. (2000). Building talking heads: Production based synthesis of audiovisual speech. *Humanoids 2000 First IEEE-RAS International Conference on Humanoid Robots.*

Verma, A., Rajput, N., and Subramaniam, V. (2003). Using viseme based acoustic models for speech driven lip synthesis. *ICAASP Proceedings*, 5:720–723.

Viberg, M. (1995). Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, 31(12):1835–1851.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269.

Welling, M. (2000). The kalman filter. Technical report, California Institute of Technology 136-93.

Williams, J. J. and Katsaggelos, A. K. (2002). An hmm-based speech-to-video synthesizer. *IEEE Transactions on Neural Networks*, 13(4):900–915.

Xu, L. and Jordan, M. I. (1996). On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, (8):129–151.

Yamamoto, E., Nakamura, S., and Shikano, K. (1998). Lip movement synthesis from speech based on hidden markov models. In *AFGR98*, pages 154–159.

Yin, H.; Allinson, N. (2001). Self-organizing mixture networks for probability density estimation. *Neural Networks, IEEE Transactions on,*, 12(2):405–411.

Ypsilos, I., Hilton, A., Turkmani, A., and Jackson, P. (2004). Speech-driven face synthesis from 3d video. *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 58–65.

Zhong, J., Chou, W., and Petajan, E. (1997). Acoustic driven viseme identification for face animation. *Multimedia Signal Processing, 1997., IEEE First Workshop on*, pages 7 –12.