Hidden Dynamic Models for Speech Processing Applications

by

Leo Jingyu Lee

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical & Computer Engineering

Waterloo, Ontario, Canada, 2004

I hereby declare that I am the sole author of this thesis.
I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Leo J. Lee

I further authorize the University of Waterloo to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Leo J. Lee

Abstract

Human speech has a *dual* nature: the goal of speech is to convey *discrete* linguistic symbols corresponding to the intended message while the actual speech signal is produced by the *continuous* and smooth movement of the articulators with rich temporal structures. Such a dual nature has been amazingly utilized by humans in a beneficial way but has presented a big challenge for both speech science and speech technology.

This thesis starts with the observation that the continuous or dynamic aspect of human speech is inadequately modeled in current speech technology, especially in state-of-the-art speech recognition systems, while much could be learned from recent advances in speech science. This motivates a study of articulatory dynamics, based on a recently available large scale speech production database that provides simultaneous acoustic and articulatory measurements. Indeed many insights and valuable experiences have been gained from such a study and, as a result, a hidden dynamic model (HDM) that gracefully integrates the discrete and continuous nature of speech is proposed. But it also turns out that articulatory dynamics is highly complicated and can not be captured by simple models, thus the dynamics are very difficult to put into an efficient computational framework for use in speech technology.

As a continuing effort to seek internal dynamics of human speech that can reflect the continuous shape change of the vocal tract and benefit the current speech technology, the second part of the thesis turns to a study of vocal-tract-resonance (VTR) dynamics, built upon the insights and experiences gained from studying articulatory dynamics. It verifies that VTR dynamics can be captured by simple dynamic equations, and a highly accurate and efficient piecewise linear mapping from VTR dynamics to the acoustic space is also carefully designed. Two novel VTR tracking methods are developed in this part: one is based on mimicking manual tracking of VTR dynamics by human experts and uses advanced image processing methods (active contours), the other is the natural outcome of formulating a HDM for VTR dynamics and recovering the hidden dynamics by Kalman smoothing. The residual feature resulting from VTR tracking by HDM has also been used as an appended acoustic feature to improve a hidden Markov model (HMM) based phone recognizer on the TIMIT database.

The final part of the thesis is dedicated to arguably the most difficult and compre-

hensive speech processing application: automatic speech recognition (ASR). It first casts the HDM formulated for speech application under the general framework of probabilistic graphical models in machine learning. However, it also becomes clear that exact inference and parameter learning for such a model is NP hard. In order to use HDM for speech recognition, this final part concentrates on developing novel and powerful variational EM algorithms. The effectiveness of the new algorithms invented has been demonstrated by extensive simulation experiments, and special concerns for speech recognition are also discussed.

Acknowledgements

I wish to express my deep appreciation to Prof. Li Deng, who leads me into the fascinating research area of speech processing. Li has helped me develop interest in speech problems and gain solid background in both speech technology and speech science. Two fruitful internships at Microsoft Research (MSR), under Li's close guidance, is what makes many breakthroughs in this thesis possible.

I feel that I am extremely lucky to have Prof. Paul Fieguth as my supervisor after Li's left of Waterloo to pursue industrial interests at MSR. Paul has set me an example of being an excellent lecturer, researcher and supervisor. His encouragement and continuing emphasis on seeing the big picture is what makes this broad yet focused thesis possible.

Dr. Hagai Attias (previously with MSR, currently working on his own start-up company) is a great mentor with great patience that leads me into the exciting new field of probabilistic graphical models, especially variational methods. It has been a great pleasure and memorable experience to work with Hagai. I also wish to thank him for always standing by my side when difficulty and pressure seem to prevail.

My committee members, Prof. Mari Ostendorf, Prof. Brendan Frey, Prof. Daniel Miller and Prof. En-Hui Yang, have taken time to carefully read my thesis and attend the defense from their extremely busy schedules. The valuable suggestions they provided and the good questions asked at the defense help to further improve the quality of this work.

Ontario graduate scholarship (OGS) has provided most of the financial support during my PhD journey, with further help from NSERC research grants and Ontario student assistance program (OSAP). I have also been surrounded by love and support from family and friends in Waterloo during the last few years so that this journey has never been lonely.

To my beloved wife and my parents...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Human Speech Communications

The capability of speaking a language is one of the most amazing skills humans possess. It serves as a very effective way of communication, sharing experiences, feelings, thoughts and ideas among people. However, such an ability has also been taken for granted for a long time and serious research started only fairly recently. The discipline that aims to understand the human speech communication process is speech science, which usually includes the study of the physiology of speech production, the acoustical characteristics of speech and the process by which listeners perceive speech [22]. Although some pioneering work in these areas can be dated back to the 19th century or even earlier, for example, Hermann Von Helmholtz's study on acoustics [110], Henry Sweet's study on phonetics [232, 233] and Alexander Graham Bell's effort to make speech visible to deaf people [15] (besides his many profitable inventions including telephone), modern collaborative research on human speech communication doesn't start until around 1930's [71], along with the development of telephony communication systems. During this early period, speech science and speech engineering, which refers to the practical applications dealing with speech signals, are closely coupled and many achievements important to both fields are summarized in Flanagan's classical book [77].

Among the many valuable studies in speech science, the modeling aspect of human behavior is of major interest in this thesis, since it helps both to understand the speech

process and to suggest ways of simulating human speech tasks by machine. Over the last forty years, speech scientists have developed increasingly detailed and sophisticated models for human speech production and perception. Speech production, which concerns the conversion from an intended linguistic message, e.g., the phrase "so cool", to the acoustic waveform radiated mainly from one's mouth, is the focus of this thesis. Recent and comprehensive reviews on speech production models and theories can be found in the book chapters [79, 131]. Some important scientific findings and theories related to speech production are briefly presented below to serve as a background for readers who are not familiar with this area.

A phoneme sequence is usually taken as the input to human speech production systems. Phonemes are the minimal contrastive units of speech sounds in a given language, and are typically specified by a set of *distinctive features* based on articulatory, acoustic or perceptual properties [39, 117]. Phonemes are defined in an abstract sense while the acoustic realizations of them are called phones or allophones. After defining phoneme as the basic unit of spoken language, it is tempting to assume that the speech production system merely produce each phoneme sequentially to generate the intended linguistic message. Such a "beads on the string" approach has been the basis of traditional linear phonology but it has also been shown to be far from adequate in capturing important aspects of human speech. At the phonological level speech is symbolic or discrete, but it becomes smooth and continuous at the phonetic level, reflected by the smooth movement of the speech production organs, also known as articulators, or the continuous change of the vocal tract shape. In continuous speech, it is often impossible to determine the phone boundaries precisely from the acoustic signal[1], although a person can clearly perceive these discrete speech units. The realization of a phoneme can also be very different from when it is realized in isolation, heavily depending on the neighboring phonemes. Such a phenomenon is generally called *coarticulation*, which very broadly refers to the fact that a phonological segment is not realized identically in all environments, but often apparently varies to become more like an adjacent or nearby segment [103]. Coarticulation *cannot* be explained as an imperfection in the way language is realized due to the physical constraint of the articulators. On the contrary, coarticulation significantly aids human speech perception beyond the efficient in-

[1]Some speech scientists even argue that such a procedure itself is ill-defined.

tegration of realizing successive phonemes [65, 103]. However, coarticulation does present a big challenge if we want to faithfully model human speech production.

There are many other subtle phenomena that make modeling human speech a very difficult problem, but further introduction is beyond the scope of this thesis. Generally speaking, as an effort to account for all or at least most of the complexities involved in human speech production, speech scientists typically take a multi-level approach. As mentioned previously, the top level is usually a sequence of phonemes, each of which is represented by a set of distinctive features or *feature bundles*. However, unlike the unstructured features used in traditional linear phonology, features in modern nonlinear phonology[2] are allowed to extend over domains greater or less than one single phoneme [94, 129], hierarchically organized based on their inter-relationships [41], or directly related to the action of articulators [27, 28, 29]. A computational model for American English based on the theoretical foundations of these nonlinear phonological theories was also developed recently [57, 58, 66, 230], known as *overlapping articulatory features*. This first step effectively turns a phoneme sequence into a set of multi-tier overlapping features. The next level is *task dynamics* or *task-variables* [130, 213], for example they can be defined as vocal tract constriction locations and degrees for each feature. These task-variables are usually realized by the coordination of an ensemble of articulators, resulting in the articulatory dynamics measured in many scientific studies [107, 249, 250, 254]. The movements of the articulators drive the smooth and continuous change of the vocal tract shape, which can be mainly characterized by the location and bandwidth of its resonances. Finally the glottis and the vocal tract interacting as source and filter to produce the noisy acoustic signal that we perceive as speech. To complete the human speech communication chain, the acoustic waveform is subsequently received by the intended listener, processed by the highly robust peripheral auditory system against environmental variations, and cognitively decoded by the human brain using all sources of speech-related knowledge accumulated in the past, where the details at this end are also subject to substantial amount of further research work [175].

Although this thesis is mainly engineering and application oriented, it has been mo-

[2]Here "linear" and "nonlinear" refer to whether the strict sequential order of phonemes are followed, which is quite different from the concept of "linear" systems in math and engineering communities.

tivated and benefited from many scientific studies on human speech. In particular, the thesis has paid much attention to the dynamic nature of speech, *e.g.*, at the articulatory or vocal tract resonance (VTR) level, besides its symbolic nature. It is the author's belief that this is one important aspect that hasn't been addressed enough in the current speech technology while much can be learned from advances in speech science.

1.2 Machine Speech Processing

Along with the development of telephony communication systems, the first important machine speech processing application is speech coding: to store and transmit speech signals efficiently [70]. In the early days, it was the center of speech applications and a driving force under both speech engineering and speech science [77, 199]. Today the need to conserve bandwidth or bit rate persists even with the ever increasing capacity of transmission medias, especially in areas such as cellular phone communication systems, live audio feed over the internet and secure (encrypted) voice communications. However, as computers and other "smart devices" penetrate more and more into the daily life of ordinary people, communicating with machines via natural speech has become a focus in recent years. There are three key steps that machines need to perform in order to have a *conversation* with human beings: speech recognition (or speech to text), speech understanding (understand the content of speech and act appropriately), and speech synthesis (or text to speech). As security concerns grow in our society, speaker recognition (including both speaker identification and verification) [31, 84, 205] also provides a cheap and convenient way for preliminary access control. Speech enhancement is often used to improve speech quality under adverse conditions for human listeners or further machine processing. There are fundamental speech processing techniques important for different applications as well, such as speech analysis and feature extraction. A few recent books provide excellent and up-to-date coverage on these basic techniques and application areas [92, 115, 178]. In summary, it is reasonable to expect that as natural and ubiquitous as speech is to humankind, so will be machine speech processing applications in the future.

Among the various speech processing applications, it is fair to say that automatic speech recognition (ASR) remains to be the most difficult and comprehensive one so far.

It has long been touted as the Holy Grail of artificial intelligence; it also subsumes many other speech processing techniques and important speech-specific knowledge as necessary components. To further illustrate, let us view the human speech communication chain as an encoding-decoding process, which is more familiar to electrical engineers and computer scientists. As an encoder, a talker uses the knowledge of words, phrases, grammars and the sound representation of the intended linguistic messages to generate a phonetic plan; such a phonetic plan is subsequently executed by the human speech production system to produce speech waveforms. At the decoder end, the listener uses extensive knowledge about speech accumulated over the years to interpret the speech signal received and transformed by the human auditory system. ASR plays the role of a listener to decode the underlying linguistic message from the speech waveform received via a (typically) noisy channel. Such a view point not only requires an ASR system to possess the power of human auditory system but also indicates that it needs to be equipped with a *key*, or an internal generative model, that is compatible with (but does not have to be identical to) the human speech production system in order to decode successfully. This makes the design of ASR systems both comprehensive (requiring the knowledge of human speech perception and production systems and related processing techniques) and difficult (due to the complexity of both systems).

The generative model used in state-of-the-art speech recognition systems is the hidden Markov model (HMM) and its extensions [16, 179, 196, 197]. Although details of HMM does not belong here as introduction to the whole thesis[3], the basic structure of this statistical model is presented in Fig. 1.1, which is a typical phone model used in speech recognition [259]. There are five hidden states in this model with the permissible transition probabilities a_{ij} drawn as arrows. Notice that this is a *left-to-right* model since as time increases the state index increases or stays the same. This is a simple constraint to reflect the temporal flow of speech. The hidden state sequence is supposed to account for the observed acoustic feature vectors $\mathbf{o}_j$ (computed from the acoustic waveforms) via probability distributions $b_i(\mathbf{o}_j)$. Given the hidden state sequence, the observation vectors are assumed to be distributed independently. Of course in practical training and testing

[3]A detail introduction of HMM including various key algorithms will be given in Part III, where ASR is focused.

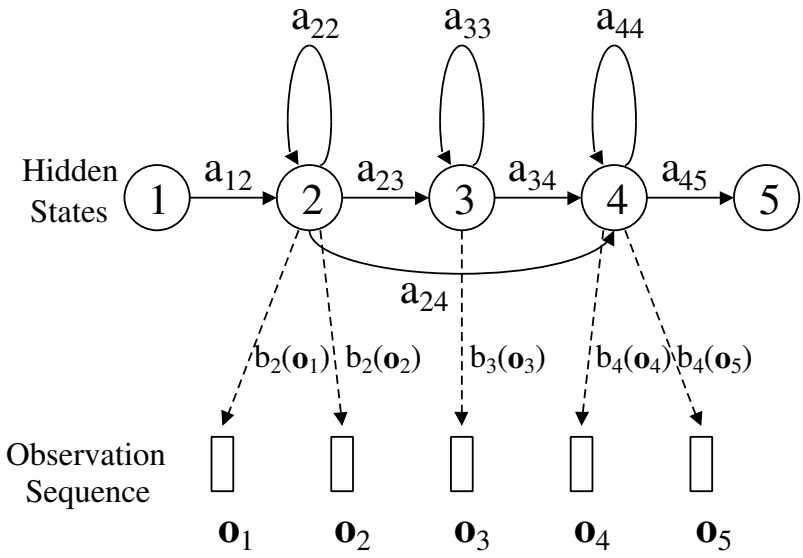

Figure 1.1: A five state HMM used in ASR as a phone model.

scenarios the hidden state sequence is unknown and has to be inferred from the data. Also notice that the first and last hidden states are *non emitting*, *i.e.*, they do not generate any observable acoustic measurements. Such a design is to facilitate the concatenation of phone models. Therefore, the above model is usually referred to as a *three-state* HMM instead of a *five-state* one in the speech literature.

1.3 Motivations

From the overview of human speech communication and machine speech processing presented in the previous sections, it is not hard to observe that speech engineers are using a much simpler model structure to account for the human speech production mechanism than speech scientists. Typical approaches of both are illustrated in Fig. 1.2.

Is the HMM good enough as a generative model for speech processing applications, especially for speech recognition? As increasingly difficult, real-world speech recognition tasks are being attacked [85, 62], such as the recent AURORA and EARS project, more and more researchers realize that the answer is clearly no. Although current speech recognizers

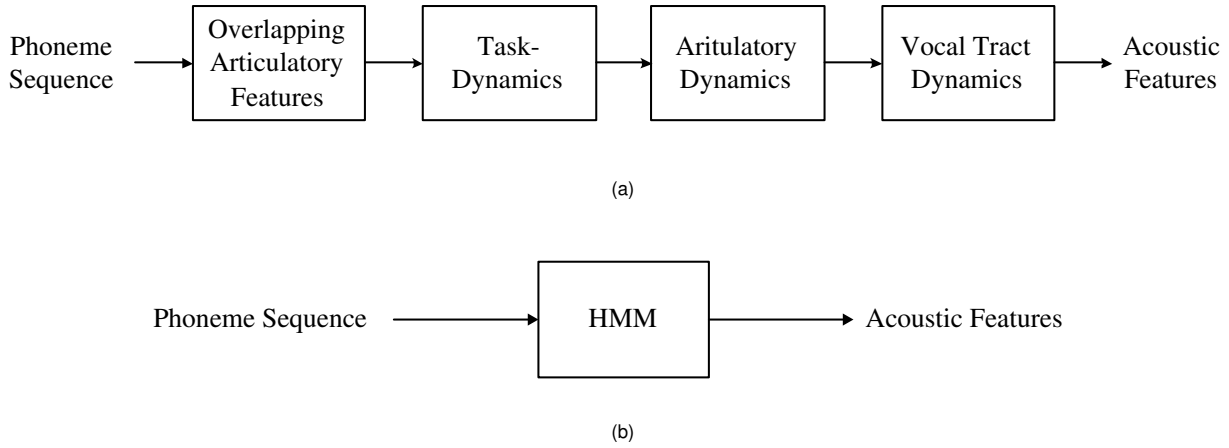

Figure 1.2: Modeling comparison in speech science (a) and speech technology (b).

are quite successful in some limited yet useful applications [50, 96, 114], their word error rates are still at least an order of magnitude worse than those of human listeners on many speech tasks [156, 192]. Even worse is that under more challenging conditions, such as high level of noise or significant change of speaking style, most speech recognizers will completely break down. Moreover, typical speech recognizers today have a few million trainable parameters while there is evidence that the language competency humans are born with is encoded with about 5,000 bytes of information on the human genome [193]. All these indicate that the success of current speech recognizers is mainly achieved by their rigorous statistical formalism and computational efficiency to learn huge number of parameters from ever increasing amount of training data, while the underlying model structure remains to be weak, demonstrated by their inherent lack of robustness. Further analysis of HMM also shows that it is far from compatible with the true human speech production mechanism, comparing to knowledge gained from scientific studies of human speech production (more details will be provided in Part III). It is clear that in order to further improve the current speech technology and to bring it closer to human performance, a very promising direction is to enrich the underlying model structure while reducing the number of trainable parameters. This is exactly the goal of this thesis.

On the other hand, although increasingly detailed and sophisticated speech production models have been developed by speech scientists over the past forty years, none of them

can be directly "ported" to speech technology. This is mainly due to the following reasons:

- These scientific models are deterministic in nature, resulting from the study and simulation of a very limited number of subjects under constrained conditions, and not able to cover the complicated variability among a large number of speakers, or even the systematic variability of the same speaker under different conditions.

- Although considerable efforts have been devoted to model human speech in accuracy and detail, most of the models lack the comprehensiveness in covering all classes of speech sounds, which is essential for speech technology.

- Most importantly, speech science and speech engineering have very different research goals. Speech science aims to understand the human speech communication process and concentrates on the explanatory power of the models to account for diverse and complicated speech phenomena, while computational issues, which are crucial for engineering implementation, play very little or no role at all.

This thesis is strongly motivated by observing the respective strengths and weaknesses of speech engineering and speech science after broad literature survey in both fields. It aims to combine the knowledge-based modeling approach in speech science and the data-driven approach in speech engineering in a graceful way to achieve novel, compact and efficient modeling of human speech production to benefit the current speech technology. It also aims to bridge the gap between the largely distinct communities of speech scientists and speech engineers (despite their common origin in the 1930's) to stimulate further collaborations among researchers of these two communities. More specifically, this thesis studied two kinds of "hidden" dynamics that are completely missing in HMM and its extensions used in the state-of-the-art speech technology, namely the articulatory dynamics and the vocal tract resonance (VTR) dynamics which largely characterize the continuous change of the vocal tract shape. These dynamics are called "hidden" because they are usually not directly observable from acoustic measurements. This thesis not only proposed new models based on detailed studies of these hidden dynamics but also developed powerful and advanced statistical algorithms to facilitate their use in speech technology. A brief outline of the thesis is given in the next section.

1.4 Thesis Organization

The thesis is largely divided into three parts. Part I concentrates on the study of articulatory dynamics based on a recently available speech production database that measures the movement of the articulators and the produced acoustic signal simultaneously. A graphic user interface (GUI) analysis tool is first developed to facilitate the study on the database. Important insights about speech are developed by scrutinizing the data and shown by a few typical examples. Next, the nonlinear articulatory-to-acoustic (ATA) mapping is studied by using two neural network architectures and some standard machine learning techniques, followed by further attempts to model the movement of the articulators during speech production. Finally these two studies are combined to form a novel articulatory dynamic model of human speech production.

Part II naturally shifts to the study of VTR dynamics based on insights and results obtained from studying articulatory dynamics. It starts with another GUI tool to ease the task of manually tracking VTR dynamics, followed by proposing and testing a number of powerful models to capture VTR dynamics during speech production. Then a novel VTR tracking method is developed by simulating the way human experts do so based on spectrograms, using image processing methods, namely, active contours or snakes. A knowledge-based piecewise linear mapping is carefully constructed next to describe the relationship between the VTRs and the observed acoustic features. Based on a VTR dynamic equation and the VTR-to-acoustic (VTA) mapping, another novel VTR tracking method is developed by formulating the relationship between VTR dynamics and acoustic features as a state space model to do Kalman filtering and smoothing. At the end, the importance of VTR dynamics is demonstrated by appending the resulting residual vector as a new feature to improve a well-trained baseline TIMIT phone recognizer.

Finally a specific form of the hidden dynamic model (HDM) formulated as switching state space (SSS) model aiming at ASR application is fully developed in Part III. Due to the intractability of exact inference in the SSS model, novel variational methods are first developed to approximate the exact inference in a principled way, and the powerful expectation-maximization (EM) framework is subsequently adopted to estimate model parameters. These new algorithms are thoroughly studied and tested by extensive simulation examples. A few practical considerations of building a speech recognizer with the original

variational EM algorithms are also listed and discussed, followed by some simple speech examples.

The thesis is concluded in Part IV with summaries and discussions of future work.

Part I

A Study on Articulatory Dynamics

Chapter 2

Fundamentals of Speech Production and Analysis

This chapter briefly presents some background material about the speech production mechanism and related analyzing techniques. These theories and methods will be used extensively in the remaining chapters of the thesis. It also defines important concepts and sets up notations that will be used consistently throughout the thesis.

2.1 Source-Filter Model of Speech Production

At a global and functional level, human speech production can be viewed as an excitation source filtered by an acoustic tube [75, 224]. In modeling speech, effects of the excitation source and the acoustic filter are often considered independently. While in reality the source and filter do interact, their interdependence only causes minor secondary effects and is ignored in most speech analysis methods. A cross sectional view of major speech production organs is sketched in Fig. 2.1.

The source of most speech occurs in the larynx where vocal folds can obstruct airflow from the lung. For *sonorant* sounds (mainly vowels), the sound pressure is originated from the periodic vibration of the vocal folds[1]. The rate of vibration is called the *fundamental*

[1]More accurately, the vibration is only quasi-periodic over intervals of tens of milliseconds: the oscillation of the vocal folds is a highly complicated nonlinear phenomenon known as *Bernoulli* oscillation.

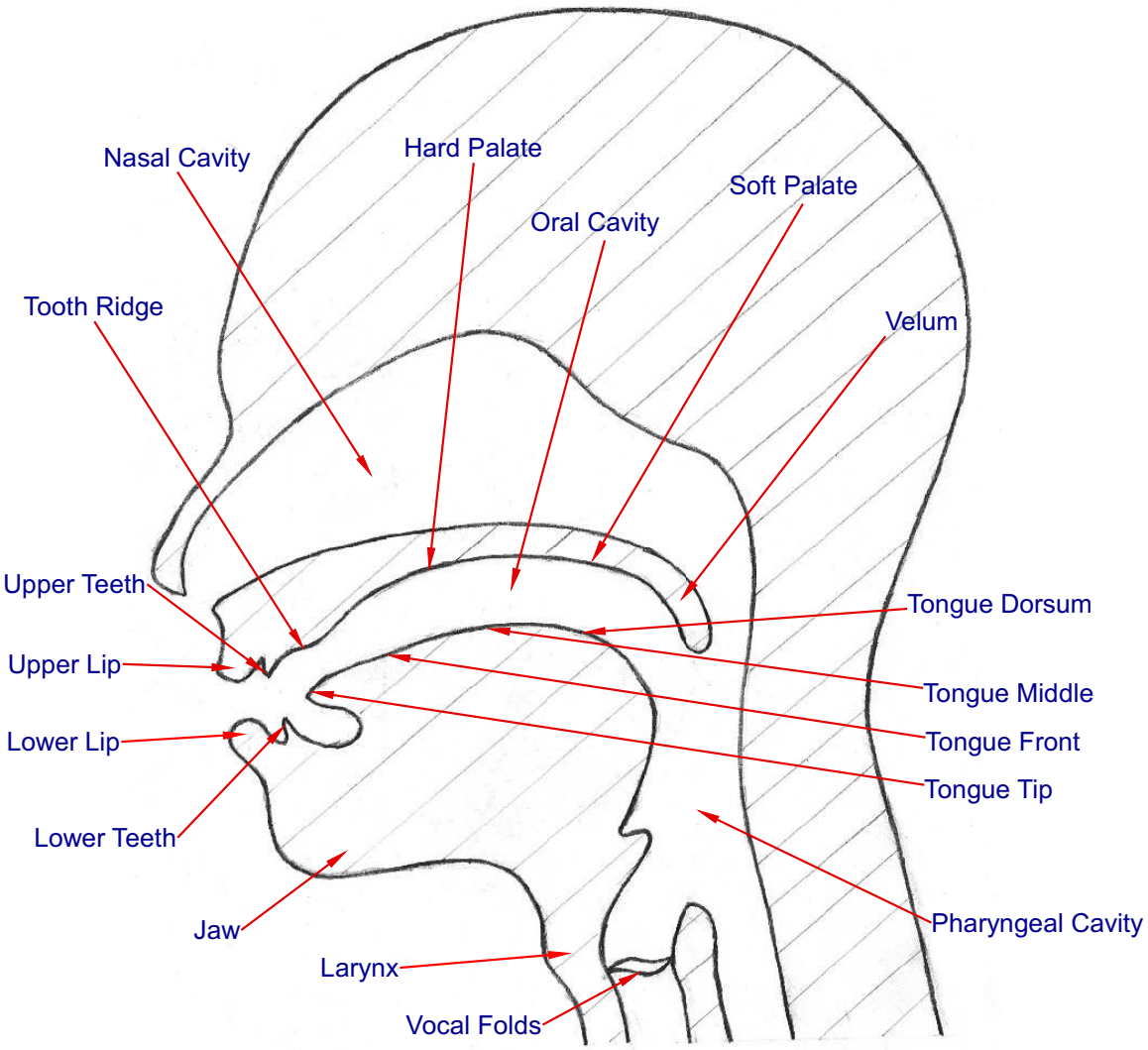

Figure 2.1: Major speech production organs.

frequency or $F0$, which ranges from 60 Hz for a large man to 300 Hz for a small woman or child. When the vocal folds are further relaxed, they can form a narrow constriction to produce turbulent noise at the glottis. Speech arising from such noise is called *aspiration*, such as the phoneme /h/ in English or when people speak in a whisper. When the vocal folds are totally relaxed, such as in normal breathing or the generation of unvoiced fricatives, the air expelled by the lungs passes through the larynx unobstructedly and creates little audible sound.

The acoustic filter for most speech sounds is the vocal tract, which consists of the pharyngeal cavity and oral cavity. For example, all the vowels in English use the same excitation source and are solely distinguished by different vocal tract shapes that amplify certain sound frequencies while attenuating others. When the velum is lowered (opened), the vocal tract couples with the nasal tract to produce nasals. For *obstruent* (stop and fricative) sounds, the vocal tract also contributes as the sound source by generating turbulent noise near its narrowest constriction. The point of narrowest vocal tract constriction is also known as the *place of articulation*. For obstruent sounds, the filter is mainly formed by the portion of the vocal tract in front of the place of articulation, resulting in larger gains for high frequency components than aspiration due to the shorter cavity.

While properties of the sound source help to define broad phoneme classes, it is the acoustic filter that provides most of the contrast among different phonemes. Therefore, the vocal tract, which mainly acts as the time-varying acoustic filter, plays a central role in human speech production. In this thesis, much effort has also been devoted to studies of the vocal tract properties. It is very difficult to directly model the time-varying vocal tract shape in speech, although recently there have been some complex physiological models aiming at achieving that [47, 48, 214, 255]. For most practical applications, the modeling is done at the frequency domain, which is both easier to work with and more directly related to human speech perception. Typically the excitation source is simply modeled as a white noise source, an impulse train with a flat spectrum or a combination of both depending on the sound class, while all the frequency shaping is accounted for by the acoustic filter, which will be further discussed in the next section.

2.2 All-Pole Filter Model

As with any filter in signals and systems, the property of the acoustic filter is fully characterized by its *frequency response* or *transfer function*, which is the *z-transform* of its impulse response[2]. As a reasonable approximation, the frequency response of the acoustic filter for human speech production (which mainly consists of the vocal tract, but may also include effects caused by the glottis, nasal tract and sound radiation) is popularly modeled as having poles only. If we denote the z-transform of the speech signal as $S(z)$, the excitation source as $E(z)$ and the frequency response as $H(z)$, we have

$$H(z) = \frac{S(z)}{E(z)} = \frac{G}{\prod_{n=1}^{N}(1 - p_n z^{-1})} = \frac{G}{1 - \sum_{n=1}^{N} a_n z^{-n}} \ , \tag{2.1}$$

where G is the gain of the filter and the p_n's are its pole locations (N in total). Alternatively, the denominator can also be expressed as an N^{th} order polynomial of z^{-1}.

The above all-pole model has worked quite well in many speech applications and may be considered reasonable for speech production due to the following reasons:

- When a sound source has only one acoustic path to the output, the frequency response has only poles (resonances) and no zeros. This is the case when the vocal tract solely acts as the acoustic filter.

- When the glottis, nasal, radiation and other minor effects are considered, zeros will be introduced into the frequency response. However, typically these zeros only play a secondary role in human speech perception since humans pay much more attention to spectral poles than zeros.

- A spectral zero can be expressed by an infinite number of poles through the following series expansion:

$$1 - az^{-1} = \frac{1}{\sum_{i=0}^{\infty} a^i z^{-i}} \quad \text{when} \ \ |a| < |z| \, . \tag{2.2}$$

  In practice a few extra poles are often good enough to represent a spectral zero (by truncating the infinite series expansion).

[2]In modern speech processing, the original continuous-time speech signal is always properly sampled to obtain a discrete-time counter-part first. Therefore z-transform is commonly used.

- There exist fast and efficient algorithms to calculate the parameters of the all-pole model from observed speech signals.

It is not hard to see that the all-pole filter model has a very clear meaning in the time domain as well. By taking inverse z-transform, we have

$$s[n] = \sum_{i=1}^{N} a_i s[n-i] + e[n]. \tag{2.3}$$

This equation is generally known as an *autoregressive* (AR) model, where the current sample is predicted by a linear combination of previous N samples plus an error term (the excitation source). Such a time domain explanation gives the all-pole model another popular name: *linear predictive coding* (LPC).

2.3 Short-Time Analysis and Spectrogram

As mentioned previously, the frequency domain properties of speech are much more closely related to human speech perception than time domain properties, but conventional frequency analysis methods can not be directly applied to speech signals since these methods are only defined for the entire signal while speech signals are time-varying or *non-stationary*. In order to reveal the time-varying spectral content of speech, a set of techniques known as *short-time analysis* have been developed for speech analysis, and they are equally applicable to any other time-varying signals. These techniques decompose the speech signal into a series of short segments, referred to as *analysis frames*, and analyze each one independently. It is implicitly assumed that the speech signal within each analysis frame is approximately stationary. The analysis frames are typically obtained by multiplying the original signal by a window function which is zero everywhere except a small region centered at the time of interest (the window length is around 10-30ms for speech analysis). The window function can be a simple rectangular function, or more commonly a smoother function such as the Hamming window whose nonzero portion is a raised cosine defined by

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad \text{for } 0 \le n \le N, \tag{2.4}$$

or the closely related Hanning window to reduce edge effects and spectral leakage. Each analysis frame is further processed by one or more signal analysis techniques, such as Fourier and similar transforms, filterbank analysis and cepstral analysis, thus forming a diverse array of speech features suitable for different applications. Just to name a few, popular speech features include mel-frequency cepstral coefficients (MFCCs), linear predictive coding derived cepstral coefficients (LPC-cep), line spectral frequencies (LSF) and perceptual linear prediction (PLP) coefficients [115, 259].

A *spectrogram* is a critical visual tool for speech analysis since its debut in the 1940's [139]. It is computed based on the *short-time Fourier transform* (STFT), *i.e.*, by applying fast Fourier transform (FFT) to each analysis frame which is a few milliseconds apart and keep the amplitude or energy of the spectrum only. It displays time in its horizontal axis, frequency in its vertical axis and usually uses a gray scale to indicate the energy at each point (t, f), with white representing low energy and black high energy. Sometimes a color scale or even a 3-D representation may also be used for display to achieve better visual effects. Typically a few extra processing steps are needed to produce a "good" spectrogram: the speech signal is first *pre-emphasized* by a first-order difference FIR filter to counter the *roll-off* of natural speech, which refers to the typical power fall-off of voiced speech with frequency at about -12dB/octave; the computed energy from STFT is displayed at a log scale to better match the sensitivity of human auditory perception; and some thresholding is usually applied when using a gray or color scale.

There are two types of spectrograms depending on the span of the window function $w(n)$ used when constructing the analysis frames. If $w(n)$ is of a short duration (< 10ms), then its Fourier transform will be of wide bandwidth (> 200 Hz) and the resulting spectrogram is called a *wideband* spectrogram. It is able to achieve good time resolution and track the rapid spectral change during speech, but has poor frequency resolution and cannot display fine spectral details, *e.g.*, the individual harmonics in voiced speech due to glottal vibration (multiples of $F0$) are often completely smeared. On the other hand, if a relatively long window is used (> 20 ms) which leads to a filter with narrow bandwidth (< 100 Hz), the resulting spectrogram is called a *narrowband* one. It achieves good spectral resolution at the expense of poor time resolution, *e.g.*, it is capable of resolving the spectral harmonics but often fails to track the spectral change quickly enough. A typical example of both is

shown in Fig. 2.2. Ideally one would like to achieve good time resolution and frequency resolution simultaneously, but this is impossible due to the well-known uncertainty principle in signal analysis [42]. To better handle this dilemma, more advanced signal processing methods, such as wavelet analysis [188], could be explored.

It has been long observed that visual cues in the spectrogram (usually wideband spectrogram for this purpose) are closely related to much of the significant acoustic-phonetic properties of speech and aspects of human speech perception [65, 178, 265]. Most researchers also agree that the human auditory system extracts some perceptual relevant information directly from a spectrogram-like representation of speech. Indeed, a well-trained human expert is able to decode the underlying linguistic units by carefully examining the time-frequency properties of speech signals visually displayed by spectrograms. Such a process, known as *spectrogram reading* [266], plays a special educational role for speech researchers. While we as human beings acquire the skill of translating auditory information into meaningful messages at an early age, it is almost impossible to consciously analyze this process. Learning to read a spectrogram allows our conscious mind fully involved in examining the time-frequency pattern of speech signals to gain insights into the encoding-decoding process of human communication and sparks to improve the current speech technology.

Looking at the example in Fig. 2.2(b), the most prominent feature is the smooth movement of the dark bands during voiced speech. The center of these dark bands indicate the *formant* frequencies, which are usually defined to be the resonance frequencies or poles of the acoustic filter in the "filter" part of the source-filter model. Formant frequencies, commonly denoted as $F_1, F_2, \ldots, F_N$, are the most important property that characterizes the acoustic filter (others are the formant bandwidths $B_1, B_2, \ldots, B_N$ and formant amplitudes $A_1, A_2, \ldots, A_N$). For example, it is well-known that the first three formant frequencies are sufficient to discriminate all English vowels and semivowels [190]. Formants also play an important role in speech coding (formant vocoders) and speech synthesis (formant synthesizers [32, 137, 138]). A closely related but different concept is the vocal tract resonance (VTR) frequencies. While the formant frequencies are typically defined in the measurable acoustic domain or on an abstract modeling sense ("filter" part of the source-filter model of speech production), VTR frequencies are the intrinsic physical property of the vocal tract.

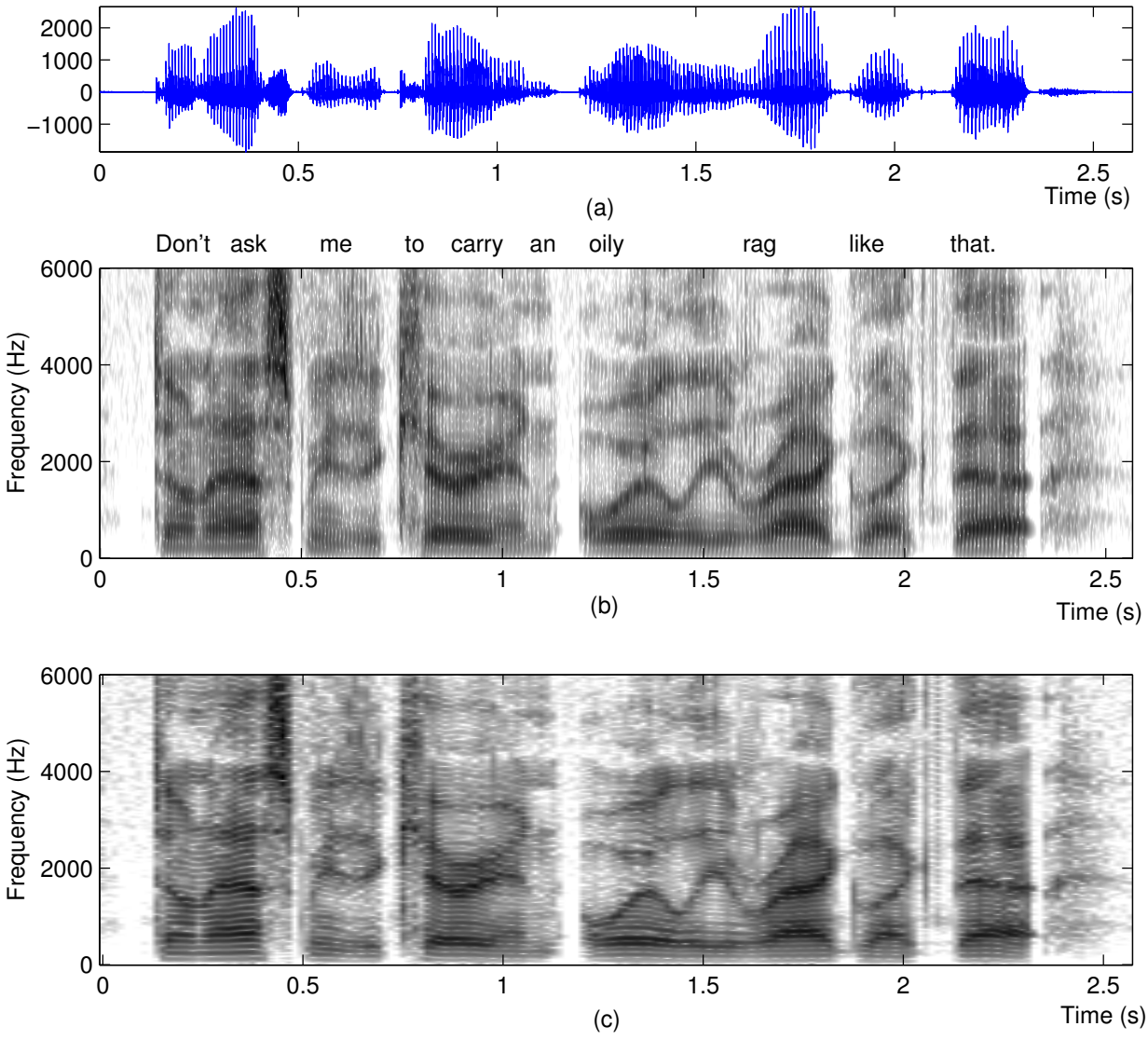

Figure 2.2: Acoustic waveform (a) and the corresponding wideband (b) and narrowband (c) spectrograms of a sentence.

When the vocal tract solely acts as the acoustic filter, *e.g.*, for nonnasalized vowels, the formant and VTR frequencies are identical, but they can be different otherwise. Moreover, formant frequencies can change abruptly, *e.g.*, due to the sudden lowering of the velum which introduces nasal coupling, but the time-evolvement of the VTR frequencies, as a result of the relatively slow shape change of the vocal tract, are always continuous and smooth during the production of all sound classes. Such a property of the VTR plays a vital role in later chapters of the thesis where VTR is chosen as the underlying dynamics characterizing human speech production.

There are many more important concepts and techniques of speech that are impossible to cover in such a short chapter. As a matter of fact, speech is inherently a multidisciplinary research area which requires the joint effort from electrical engineers, computer scientists, linguists, physiologists and psychologists in order to progress. It is often necessary to have a general understanding about all these fields as well in order to successfully carry out research on a specific topic. Interested readers are referred to a few excellent and up-to-date books that cover the topics of speech research and applications from one or more different angles [65, 92, 115, 127, 178, 224]. However, the background material provided in this chapter should be adequate to help readers understand most of the thesis with little problem. Later chapters of the thesis will emphasis automatic speech recognition (ASR) in particular and an introduction to this specific application area will be provided in Chapter 8.

Chapter 3

Data Description and Processing

This chapter describes the University of Wisconsin X-ray microbeam (UW-XRMB) speech production database used in the study of articulatory dynamics, a graphic user interface (GUI) tool developed for data analysis and some simple processing to prepare the data for further study.

3.1 The UW-XRMB Database

Being able to obtain accurate measurements of the articulatory motion and the simultaneously generated acoustic waveform has long been a goal for speech scientists. However, collecting such highly useful data is no easy task. Early attempts involve generating full X-ray movies or X-ray cineradiography of the vocal tract but were quickly abandoned due to serious health concerns [77], although old data (with relatively low image and audio quality) is still available for analysis [171]. More recently, magnetic resonance imaging (MRI) and ultrasound are able to provide three dimensional, high spatial resolution image about the shape of the vocal tract during the production of various sounds. But these techniques cannot achieve good enough time resolution (20 ms or less) to truly capture the complex articulatory movements during continuous speech. Nonetheless, some important "quasi-static" studies of human speech production have been done based on these techniques [6, 226, 229, 238, 262]. Another class of techniques have good temporal or time resolution but can only provide limited spatial information about the configuration

of the mouth, such as electropalatography (EPA/EPG) [102], electromagnetic midsagittal articulography (EMMA) [189] and X-ray microbeam (XRMB) tracking [81, 235, 134]. While EPA only indicates points of contact between the tongue and palate[1], EMMA and XRMB techniques can accurately track a few selected points on the articulators, usually restricted to the midsagittal plane (the plane which bisects the speaker's head vertically). But this is not a very serious limitation since it is often possible to infer the shapes that the tongue assumes during speech production from its mid-line profile only [227, 228]. The UW-XRMB database is a large-scale multi-speaker database collected by the XRMB tracking technique, after more than a decade of diligent work by John Westbury and his team [247, 248]. More details about this database is described below.

3.1.1 Data Acquisition

The XRMB tracking technique was invented by Osamu Fujimura and his colleagues in 1970's [81] when they built the first machine of this kind for speech research at the University of Tokyo. The current machine at the University of Wisconsin–Madison is a second generation and improved system over the original one. A schematic of this system, which is used as a device for recording motions of articulators during speech production, is shown in Fig. 3.1. First an electron beam is generated by a 600 kV, 5 mA (maximum) power supply and carefully directed to a specific point on the surface of the water-cooled tungsten target, by the accelerating column and beam-line components, to generate high energy X-rays. The deflection and focusing of the electron beam, and consequently, the direction of the incident narrow X-ray beam (roughly 0.4 mm in diameter), can be accurately and rapidly controlled by a built-in computer. The X-ray beam then passes through a pinhole and strikes desirable areas of a speaker's head. Such a narrow beam significantly reduces the radiation dosage a subject has to suffer comparing to early full X-ray scans so that it becomes possible to collect large amount of speech data over many speakers. Finally the X-ray beam passes through the speaker's head and reaches the X-ray detector at the far right side, which consists of a sodium iodide (NaI) crystal, a photon-multiplier tube and an integration circuit. The entire machine is further encased in 2-4 inch lead plates, making

[1]Such information is nonetheless valuable for speech production since other techniques often fail to detect the contact accurately enough.

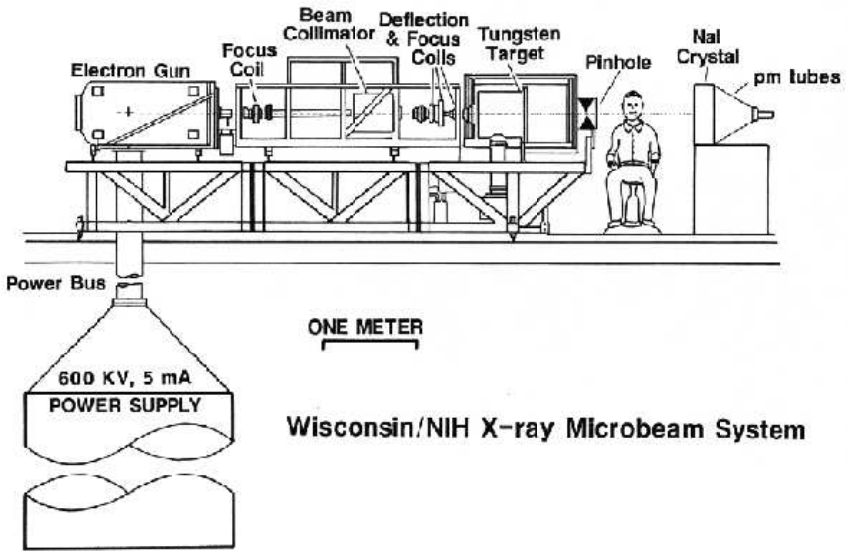

Figure 3.1: A schematic of the UW-XRMB system. Adapted from a UW-XRMB webpage `http://www.medsch.wisc.edu/ubeam/gen/generator.html`

its total weight approximately 15 tons.

A number of 2.5-3.0 mm diameter gold pellets are glued to some selected points on the speaker's midsagittal plane. The system performs its primary, pellet-tracking function by producing local X-ray scans roughly 6 mm square, circumscribing the expected location of each pellet. When a pellet falls within such a search square, it casts a recognizable shadow on the X-ray detector so that its position can be accurately calculated. Current and previous locations of a pellet are used to predict its future position and to determine the required area of a subsequent local scan of the X-ray beam for that particular pellet. The series of x-y positions for a pellet so determined during speech production constitutes the pellet's trajectory recorded in the database.

The placements of gold pellets used in the UW-XRMB database are shown in Fig. 3.2, where a total of eleven pellets are used. Among the eleven pellets, eight are used for tracking articulatory motions while the remaining three are used as reference points. Jaw articulation is tracked via pellets on the mandibular incisor (MANI) and mandibular molar (MANM). Soft tissue pellets are placed on upper lip (UL), lower lip (LL), tongue tip (TT),

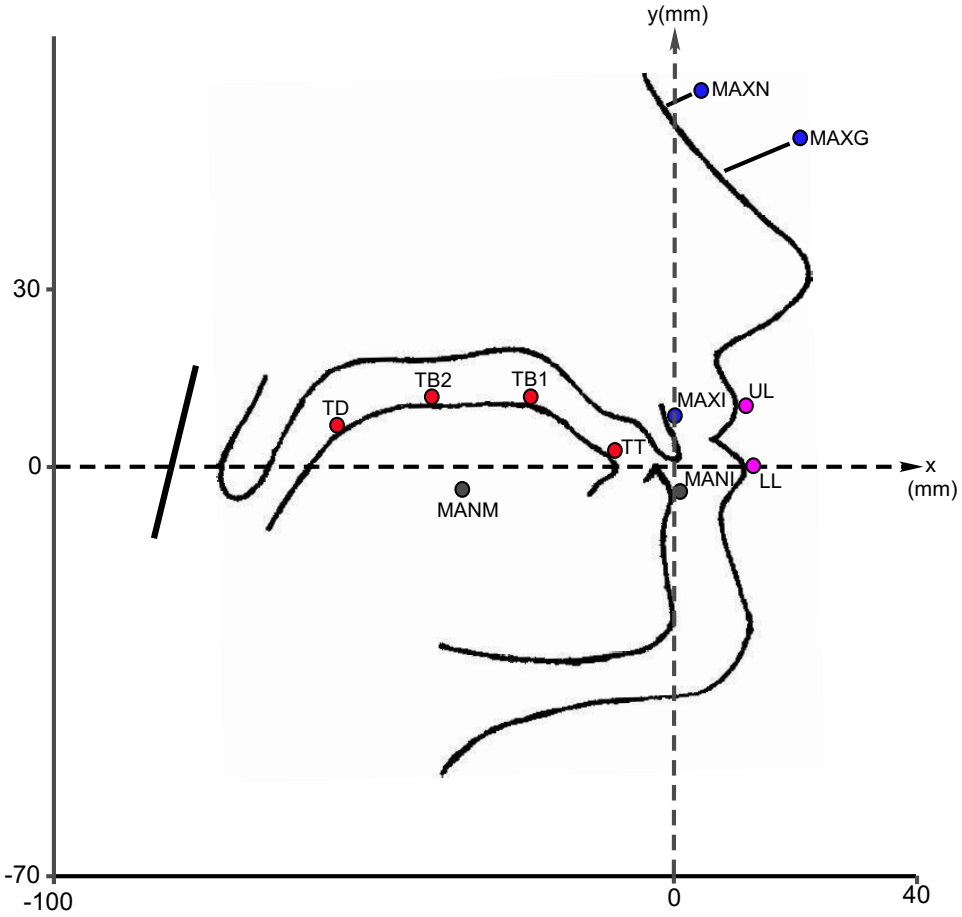

Figure 3.2: Pellet placements and coordinate system for articulator tracking; a total of 11 points are tracked, with three serving as reference points and the remaining eight as measurement points.

two on the tongue blade (TB1, TB2), and tongue dorsum (TD). Two nose pellets (MAXN, MAXG) and a maxillary incisor (MAXI) pellet are used as references to help setting up a coordinate system suitable for each speaker (also shown in the figure) and to correct for head movements during tracking. These pellets are tracked with different frequencies (40 to 160 Hz) based on typical moving speed of the point being attached to. In final distributions of the database, however, all pellets are re-sampled to have a sampling period of 6.866 ms (this somewhat odd number is chosen for technical conveniences) to facilitate further analysis.

It is very difficult to accurately determine the spatial resolution or tracking error of the system due to various technical reasons. A rough estimate according to John Westbury is 0.15-0.6 mm [248]. The tracking error can also be estimated from the data itself based on the interesting fact that there are two pellets (MANI and MANM) tracking the same bony structure — the jaw. If there is no tracking error, the distance between these two pellets should be a constant for a given speaker. Therefore, an error estimate can be obtained by calculating simple statistics of this distance throughout the database and it is roughly 0.7 mm [211]. This estimate is slightly larger than the one given by Westbury possibly because such a method also takes into account the imperfection of the head motion correcting calculation. Nevertheless, these error estimates indicate that the accuracy of articulator tracking in UW-XRMB database is good enough for most scientific studies on human speech production.

Sound pressure wave is simultaneously recorded by a high quality directional microphone with a sampling period of 46 μs (approximately 21739 Hz) for most speakers. There is considerable machine and room noise present during the experiment so that the signal to noise ratio (SNR) is no better than 30 dB for loud speakers and even worse for quite ones. The timing or synchronous error between sound recording and articulatory tracking is no more than 2 ms. There are also other measurements recorded synchronously during the experiments but not contained in the CD distribution of the database, such as the neck wall vibration signal which roughly indicates voicing during speech and lateral and frontal video images taken throughout each speaker's session mostly for monitoring purposes.

3.1.2 Corpus Organization

The UW-XRMB database is a remarkable achievement in high quality data collection for speech science and related studies. It vastly increases the extent of speaker sample and the richness of speech tasks performed comparing to previously available data, where both are very limited (typically no more than three speakers performing one or two types of speech tasks) [248]. The UW-XRMB database contains articulatory and acoustic data from 57 native American English speakers, 32 females and 25 males, each performing about 20 minutes of speech tasks. The task inventory include isolated phones, connected phone sequences, isolated words, sentences, paragraph and some special motor oral tasks, organized into 118 utterances, and is weighted more towards continuous natural speech, which is also the main focus of this study. A speaker also performs a subset of required speech tasks before the attachment of gold pellets: both as an opportunity for the speaker to practice some sample utterances and for the examiner to detect the effect of pellet placement on the quality of speech. Such audio-only data is also included in the distribution of the database and the effect of pellet placement seems to be negligible from the author's own listening experience. The database is organized into subdirectories denoted by each speaker's subject number and filenames indicating task numbers.

3.2 A Data Segmentation and Analysis Tool

At the beginning of this study, very limited tools were available to manipulate the articulatory and acoustic data contained in the database. As a result, a GUI tool coded in Matlab was developed from scratch to suit the needs of this study. The tool was originally written under Matlab 5.0 and subsequently updated to take advantage of the enhancements in newer versions (Matlab 6.5 at the time of writing). The choice of Matlab is mainly due to its powerful signal processing capacity and rich graphical support coupled with an adequate set of GUI programming commands. Moreover, the developed tool is able to behave consistently across multiple platforms as Matlab itself does. A few nice examples of GUI programming with Matlab can be found in [100], which also inspired the original development of this tool.

A screen shot of the GUI tool displaying some typical acoustic and articulatory data is

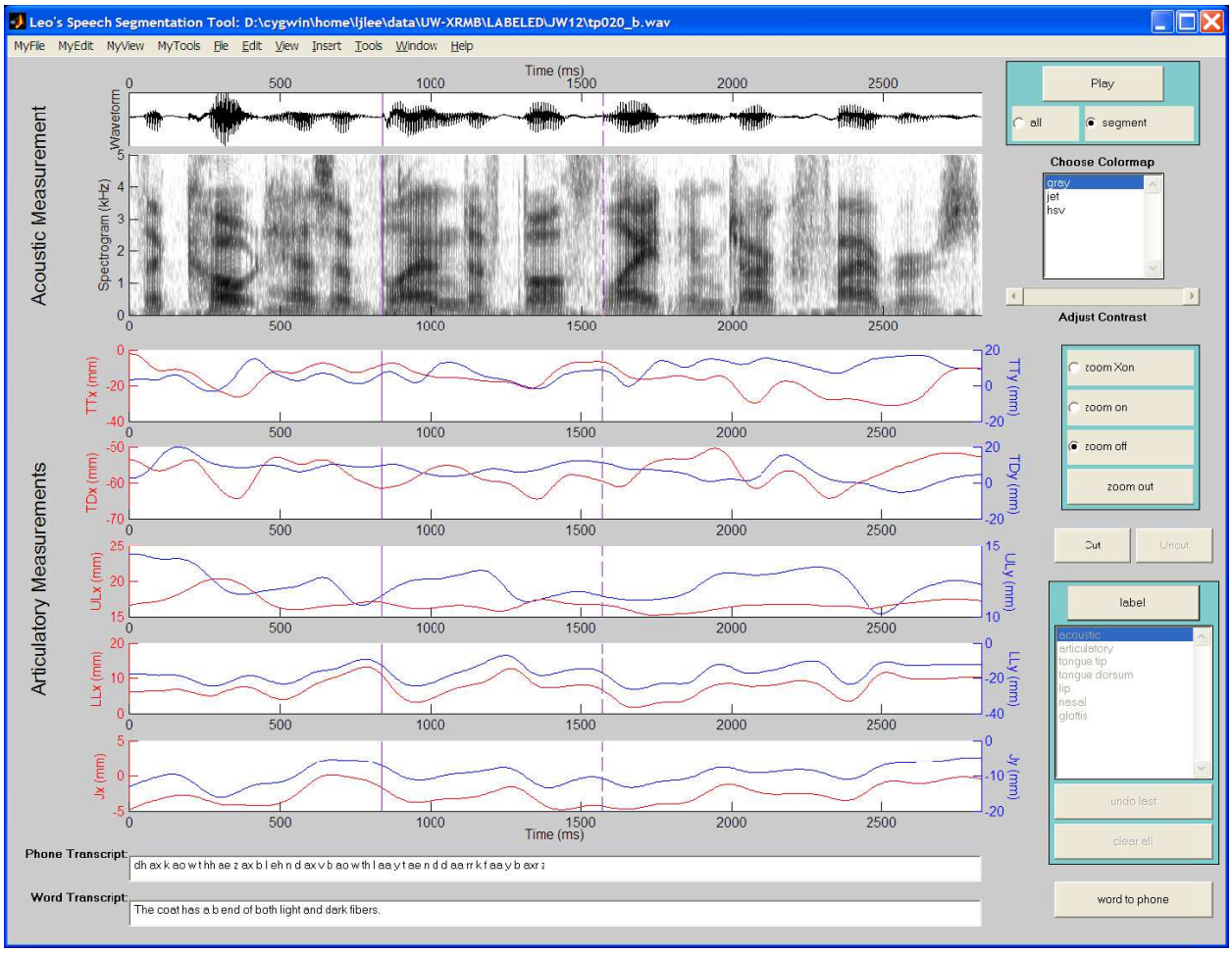

Figure 3.3: A GUI data segmentation and analysis tool.

shown in Fig. 3.3. The top two plots are the acoustic waveform and corresponding wideband spectrogram while the remaining plots display trajectories of some key articulators. Among the displayed articulatory trajectories, the jaw trajectory (`Jx` and `Jy`) corresponds to the measurements of "MANI" pellet in Fig. 3.2. This tool provides not only a compact and informative view of the acoustic and articulatory measurements but also various useful functions in visualizing and analyzing the data. For example, it allows you to zoom into a specific portion of the original data to have a better view, play a selected segment of the original waveform (indicated by vertical purple lines in Fig. 3.3), cut a long utterance into smaller segments and save them separately (this is particularly useful for the UW-XRMB database since one utterance often contains a number of sentences with pauses in between), type in the word-level or phone-level transcripts of the utterance after listening to them (it can also convert a word transcript to phone transcript based on the CMU pronouncing dictionary[2]) and save these labeling results in appropriate file extensions and formats for future analysis. Most importantly, it supports convenient manual segmentation of the data based on acoustic or articulatory features. These features will be elaborated with examples in later sections where details of data processing and analysis are described.

3.3 Preliminary Processing and Data Preparation

For a general-purpose database, it is often necessary to do some simple processing first to suit a particular research goal. This is especially true for the UW-XRMB database since the raw data distributed on the CDs has gone through very little post-processing. The preprocessing steps to suit the goal of the current study are described in detail in this section.

3.3.1 Selecting a Subset of Measured Articulatory Points

The main goal of this study is not a purely scientific one. It is rather to gain insight and motivate the use of articulatory dynamics to benefit the current speech technology. As a result, computational issues are considered up-front. An effective way to reduce

[2]The dictionary is available for free download at `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

potential computational cost is to reduce the dimensionality of the articulatory data being studied and/or modeled, since computational effort often grows exponentially with data dimensionality.

Originally there are eight articulatory points being tracked which results in a 16×1 measurement vector for each sample (x and y coordinates for each point). Since pellets "MANI" and "MANM" track the same rigid jaw structure, only one of them needs to be kept without loss of information (ignoring tracking error). Here "MANI" is chosen to represent jaw position. Moreover, four points on the tongue surface are tracked in the database, but from my previous experience of working with physiologically based articulatory models, controlling two points on the tongue (tip and dorsum) is capable of generating almost all speech sounds [49, 147, 150]. Separate factor analysis (FA) results based on different sets of articulatory data also indicate that two factors are sufficient to represent the tongue shape during speech production [104, 262]. Therefore, I choose two tongue points, TT and TD, to represent the tongue shape while discarding the other two (TB1 and TB2) in this study. The overall dimensionality of the measurement vector is thus reduced from 16 to 10 by keeping five out of the eight measured articulatory points.

The selection of two tongue points is further verified by simple analysis on the data itself. For simplicity, one speaker's data (JW15) is chosen for analysis. First a simple principal component analysis (PCA) [122] is performed on the measurement of four tongue points, which is an 8×1 vector. The result is summarized in Fig. 3.4. It is clear that the tongue measurements have a high redundancy and can be well explained by 3-5 principal components, with the first three accounting for 93.1% of the variance and the first five accounting for 98.6%. Therefore it is reasonable to expect that the selected two tongue points, TT and TD, which corresponds to a 4×1 vector, may be able to well represent the tongue shape during speech production.

Next the positions of TT and TD are used to predict the positions of TB1 and TB2. A very simple linear model is adopted and 100 out of the total 120 utterances[3] are used for training and the remaining 20 utterances are reserved for testing. Every coordinate of TB1 and TB2 is estimated by a linear combination of the coordinates of TT and TD plus

[3] Two utterances are repeated for this speaker so that the total number of utterances is 120 instead of the usual 118.

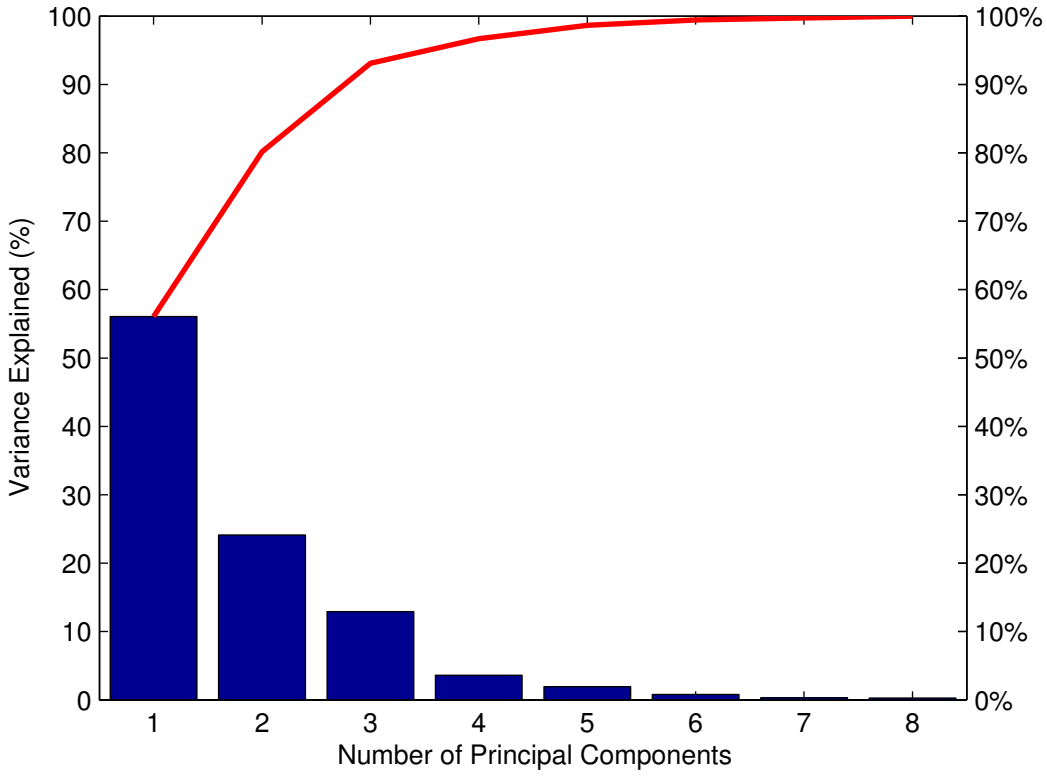

Figure 3.4: Principal component analysis (PCA) of the four measured tongue points for speaker JW15. Blue bar: variance explained by individual principal components; red line: variance explained by the principal components accumulated.

a constant bias term, *i.e.*,

$$\mathrm{TB1}_x = a\mathrm{TT}_x + b\mathrm{TT}_y + c\mathrm{TD}_x + d\mathrm{TD}_y + e, \tag{3.1}$$

where a, b, c, d, e are the parameters to be estimated. The effectiveness of such a rudimentary linear regression model can be evaluated by calculating the R^2 statistics [180] when fitting the model to data in the training set, and the results are listed in table 3.1. The

Articulatory Position	$\mathrm{TB1}_x$	$\mathrm{TB1}_y$	$\mathrm{TB2}_x$	$\mathrm{TB2}_y$
R^2 Statistics Value	0.9352	0.7314	0.8364	0.8431

Table 3.1: Regression R^2 statistics when estimating TB1 and TB2.

value of R^2 is the amount of variance in the predictors ($\mathrm{TB1}_x$, $\mathrm{TB1}_y$, $\mathrm{TB2}_x$ and $\mathrm{TB2}_y$) explained by the regressor variables (TT_x, TT_x, TT_x, TT_x), and these relatively high values indicate that the simple linear model is doing a good job . This is further confirmed by testing on untrained utterances. The estimation result for a test sentence is plotted in Fig. 3.5, where reasonably good estimates of TB1 and TB2 trajectories are obtained. From this example it can also be observed that the test result is consistent with the training one, *i.e.*, $\mathrm{TB1}_x$ is estimated with the highest accuracy while $\mathrm{TB1}_y$ is the lowest and $\mathrm{TB2}_x$, $\mathrm{TB2}_y$ are somewhere in between. Of course it is reasonable to expect that more general and powerful nonlinear estimators (some of them will be used in the next chapter) will be able to achieve better results, but the simple results obtained here, combined with knowledge gained from independent speech scientific studies, are sufficient to validate the selection of two tongue points out of the four being measured in the database.

It is also quite straightforward to further reduce the dimensionality of the data by PCA or FA methods. This is not carried out, however, in order to preserve the clear physical meanings of the original measurement points so that their behavior can be better understood and analyzed at the current stage. As a result, a 10×1 measurement vector,

$$\mathbf{z} = [\mathrm{J}_x, \mathrm{J}_y, \mathrm{UL}_x, \mathrm{UL}_y, \mathrm{LL}_x, \mathrm{LL}_y, \mathrm{TT}_x, \mathrm{TT}_y, \mathrm{TD}_x, \mathrm{TD}_y]^T, \tag{3.2}$$

is consistently used throughout the remaining of the study.

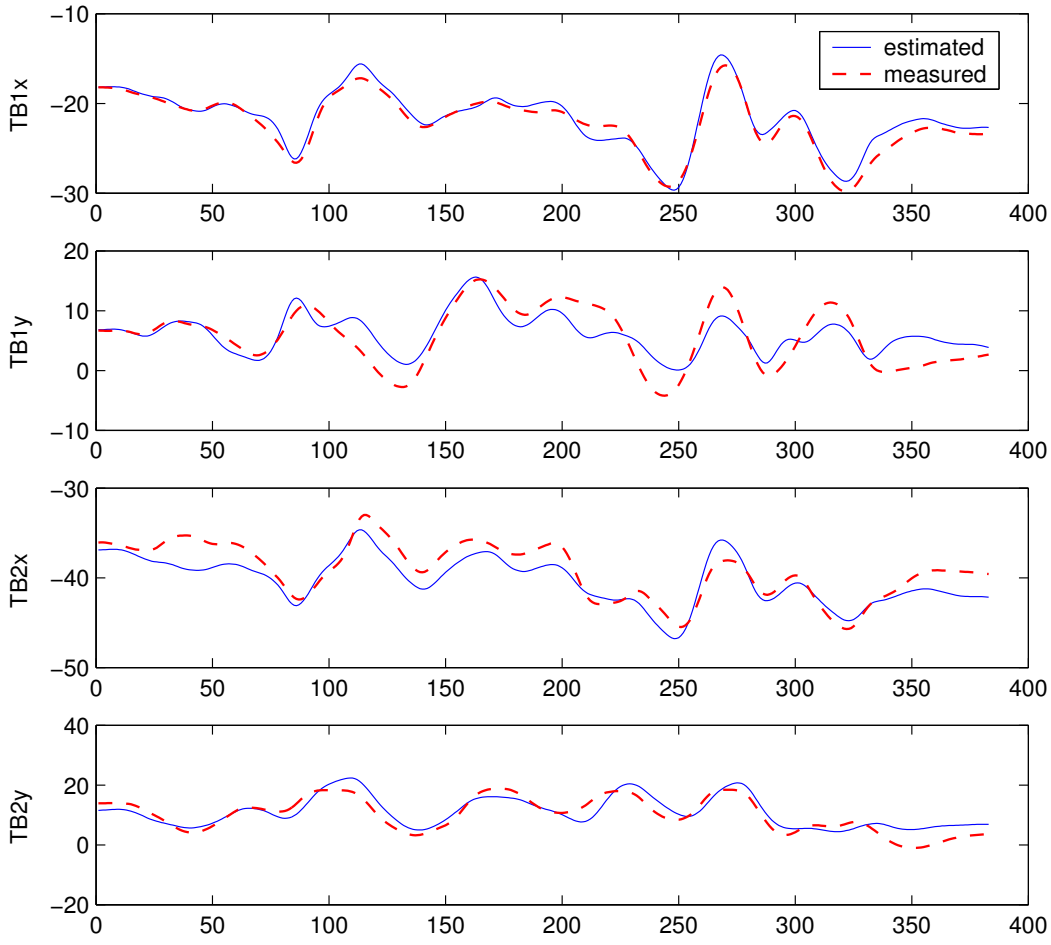

Figure 3.5: Linear regression estimates of TB1 and TB2 for a test sentence.

3.3.2 Hand Labeling and Segmentation

There are a number of important issues need to be solved before further analysis on the data, especially the relationship between articulatory dynamics and the corresponding acoustic waveform, can be carried out. These include:

- Mistracked pellets: A technical limitation of the XRMB tracking system is that a pellet may be lost by the X-ray beam at certain times . This means that for certain periods the position information of a particular pellet (especially one of the tongue pellets due to the relatively swift and irregular movement of the tongue) may be unavailable. These mistracked intervals are marked in the original data file, which account for about 2% of the total time that the pellets are tracked. However, the proportion of the utterances that contains at least one mistracked pellet is close to one half. These mistracked points create little problem when studying static properties of the articulators where time ordering of the points is irrelevant, such as in the previous section where the correlation among four tongue points is studied, since the mistracked data can be simple ignored. But they become quite troublesome when dynamic properties of the articulators are to be studied.

- The actual phone level transcripts: Although *nominal* word level transcripts are provided with the database, which is the same for all speakers, there is no record about what is actually being said by each speaker. The nominal word transcript is merely the text that subjects were instructed to read from, but even a casual listening of the audio files reveals that what is actually said is often different from these ideal word transcripts. High level errors, such as deleting, adding or repeating entire words or phrases, are not uncommon, nor do all speakers choose the same phonetic pronunciation of the words.

- Acoustic phone boundaries: As is customary in many speech scientific studies, it is often desirable to know the acoustic phone boundaries so that the relationship between the discrete (or symbolic) and continuous nature of human speech can be studied in detail. These phone boundaries are not provided by the UW-XRMB database either.

All the above issues are handled by the self-developed GUI tool described in Section 3.2. Since pellet mistracking typically occurs only during a short interval of the affected utterance, while each utterance usually contains concatenated words or sentences with pauses in between, only affected portions of the utterance are cut out by the GUI tool. The mistracking problem is also less severe for the current study due to the exclusion of two tongue points. As a result, only a very small portion of the original measurement has to be discarded. In general, long utterances, *e.g.*, those that contain a few sentences, are also cut into shorter ones (typically one sentence long) to facilitate further analysis. The only way to correct for individual speakers' deviation from nominal word transcripts appears to be labeling the acoustic files by a human listener. Therefore, each utterance is loaded into the GUI tool and carefully listened to, so that the actual word level and phone level transcripts can be typed in by consulting the nominal word transcript and CMU pronunciation dictionary (the conversion from word transcript to phone transcript is done automatically by a Perl script, but manual correction is needed from time to time). The acoustic phone boundaries are also hand-labeled by the GUI tool, and an example of a labeled sentence displaying all phone boundaries is shown in Fig. 3.6. Notice that although the nominal word transcript is "Combine all the ingredients in a large bowl", what is actually said is "Combine all the gredients in a large bowl", *i.e.*, the first syllable of word "ingredients" has been deleted. Such a mistake by the speaker is not recorded in the word level transcript (since "gredients" is not a dictionary word) but is reflected in the phone level transcript.

Admittedly, this kind of hand-labeling is *extremely* tiring and time-consuming and it is simply impossible to manually label all speakers in the database. As a result, only two speakers (JW12 and JW15) have been hand-labeled, which is still a formidable task that takes months. It should be pointed out that automatically obtaining acoustic phone boundaries with a fairly high accuracy is possible, *e.g.*, by training a HMM on the data and performing *forced alignment* or *Viterbi Alignment*, which is a well-known intermediate step in training automatic speech recognition (ASR) systems [120, 259]. However, this method requires a correct word level transcript at least, which again needs to be done

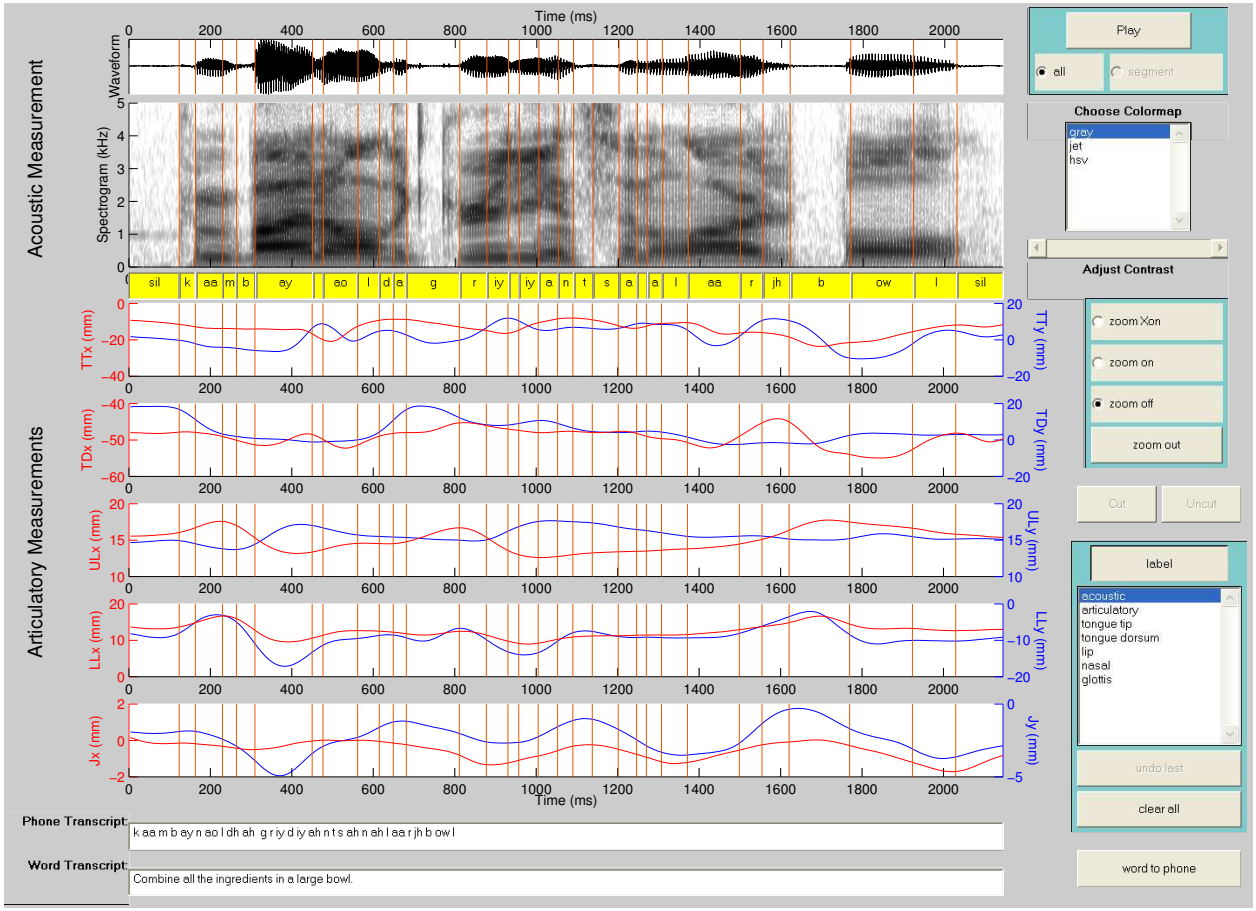

Figure 3.6: Example of a hand-labeled sentence.

manually[4]. Since the mistracked portion of the articulatory data also has to be handled manually, automatic phone segmentation doesn't provide that much help in preparing data for further analysis. Due to all these limitations, further analysis of the UW-XRMB data is concentrated on the two speakers whose data has been hand-labeled.

[4]The deletion of a syllable as shown in the previous example cannot be handled correctly by this method unless the nondictionary word "gredients" is manually added to the pronunciation dictionary.

Chapter 4

Data Analysis

This chapter focuses on results and insights derived from the UW-XRMB database. It starts by illustrating a few important properties of the human speech production process with typical examples of articulatory and acoustic measurements in Section 4.1, followed by a study on the static mapping from articulatory to acoustic space in Section 4.2. In Section 4.3 some attempts are made to model the highly complicated articulatory movements at a global level. The studies of Sections 4.2 and 4.3 are combined to formulate a hidden dynamic model (HDM) of speech in Section 4.4, geared towards practical applications, and finally some conclusions are drawn in Section 4.5.

4.1 Examples of Interesting Speech Phenomena

Let us first look at the measurement of a typical sentence in the UW-XRMB database, shown in Fig. 4.1, to illustrate some important and interesting phenomena in human speech. First it can be observed that although the acoustic signal can be divided into distinct regions, articulatory movements are smooth and continuous throughout the whole utterance. The distinction of different acoustic regions can be crudely determined by looking at the amplitude and shape of the waveform over time, or much more accurately, by *reading* the rich visual cues contained in the corresponding spectrogram. In fact, spectrogram reading is the most reliable way to determine acoustic phone boundaries, although sometimes perceptual verification may be needed. However, if only the articulatory trajectories are

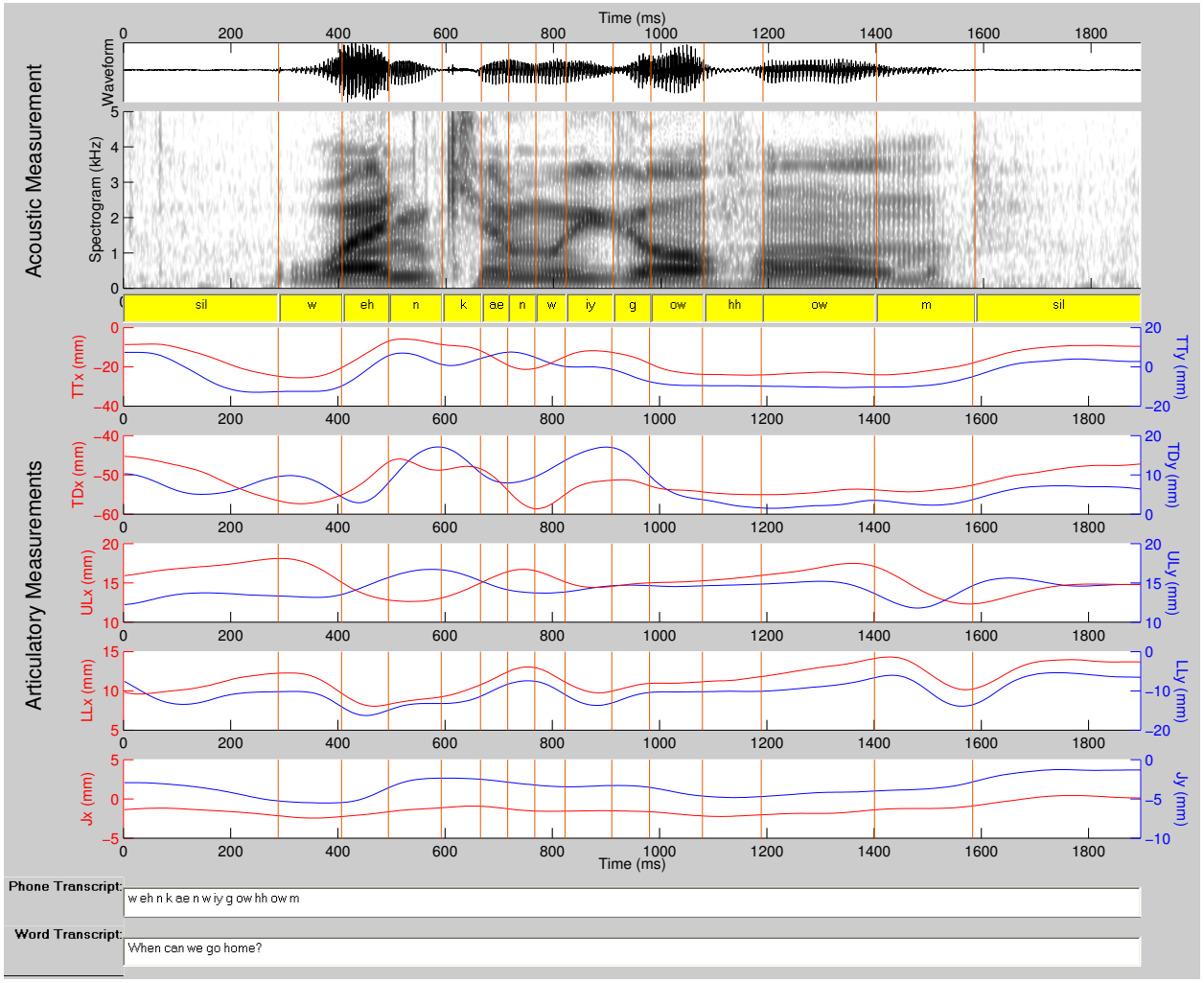

Figure 4.1: Acoustic and articulatory data: Example 1.

given, there is no way whatsoever to segment an utterance into perception-related acoustic regions. Displaying the acoustic and articulatory measurements simultaneously allows us to clearly see the *dual* nature of speech: on the one hand, the movement of the articulators is always smooth and continuous, on the other hand, some distinct discrete regions can be observed from the acoustic signal[1]. Furthermore, the bearing of this dual nature of speech on auditory perception is subtle yet important. On the one hand, it allows us to clearly perceive the discrete units of speech, such as phones or syllables; on the other hand, it also makes the perception of a phone heavily depend on the context or its neighboring phones. One clear evidence is that it is impossible to label acoustic phone boundaries based on perception alone, *i.e.*, by listening to small segments of the speech waveform. When cut into small segments, some phones just don't sound like what they were intended to be anymore. More commonly, for two adjacent phones, there is an interval of overlap. If you place the acoustic phone boundary within this interval, both phones can be perceived from either side. This actually reflects a big advantage coarticulation offers to speech perception, *i.e.*, we can usually perceive a phone by only listening to a very small segment of the actual realization, sometimes even by subtle cues in the neighboring phones. Such an amazing ability contributes greatly to the robustness of human speech perception under noisy or adverse conditions.

Secondly it can be observed that the articulators don't simply move independently of each other. On the contrary, their movements are often closely related or strongly correlated. For this particular sentence, it can be most easily observed by looking at the movements of `ULx` and `LLx`, which appears to be very similar. Moreover, the correlation among articulators also depends heavily on the speech sound intended to be produced, which means that for a different utterance, the similar movements between `ULx` and `LLx` will likely not occur. This is demonstrated by a second example of acoustic and articulatory measurements shown in Fig. 4.2. Of course it is impossible to get a detailed and complete picture of the correlation among articulators by simply looking at examples. Sophisticated data driven methods, such as a few simple ones used in 3.3.1, or careful examination of the underlying physical connections among articulators, will be needed.

[1]Nonetheless, the continuous or dynamic nature can be observed from the acoustic signals as well, *e.g.*, manifested by the smooth movement of the formants in vowel regions.

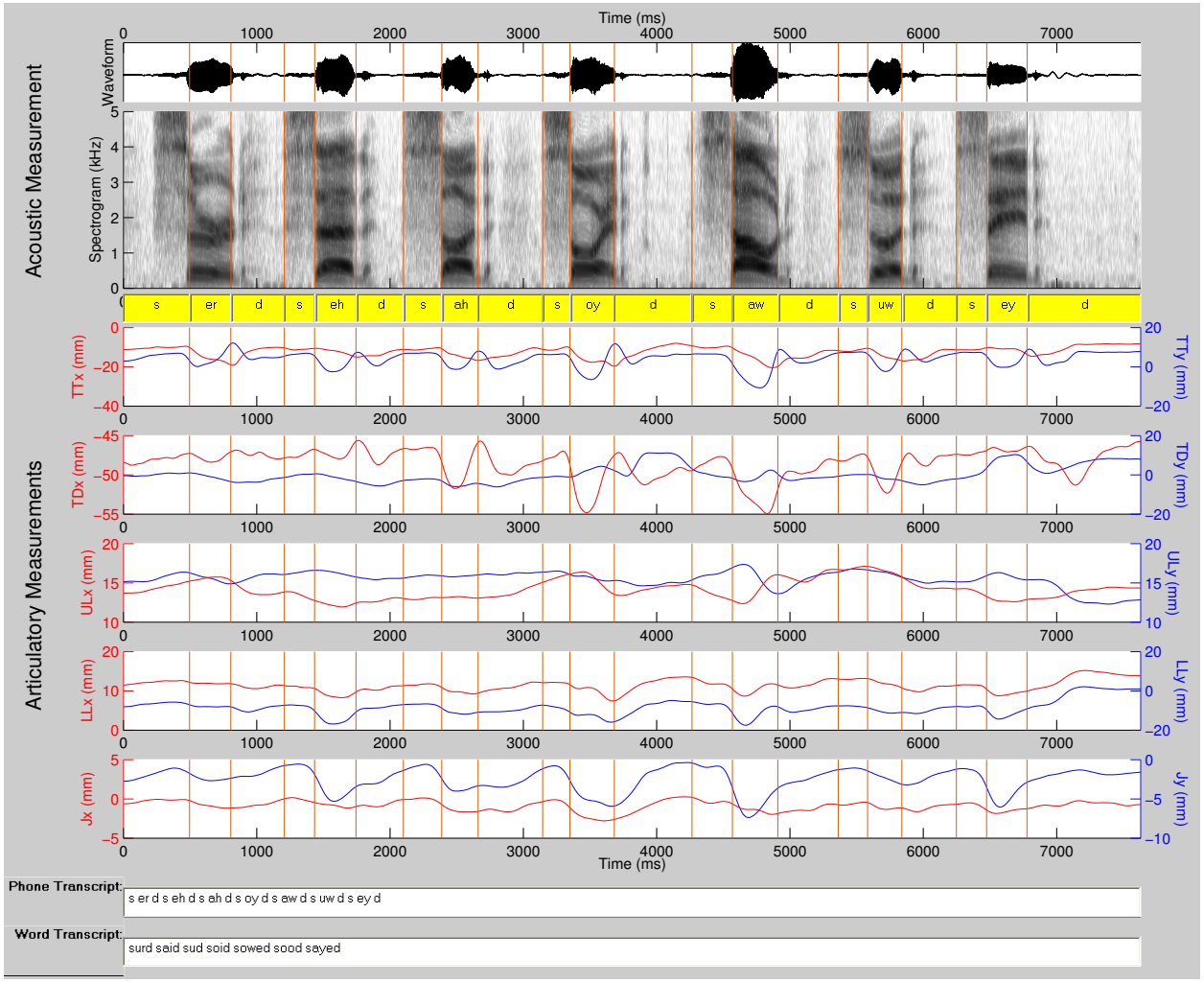

Figure 4.2: Acoustic and articulatory data: Example 2.

Fig. 4.2 also presents an excellent example to explain the concept of *active* and *inactive* articulators for the implementation of a phoneme. The active articulators of a phoneme refer to those that have to reach one or more well-defined positions (also called targets or goals) in order to produce the specific phone; while inactive articulators, on the other hand, are largely irrelevant to the production of the phone since their positions have a fairly large degree of freedom throughout the production of the phone[2]. Although such a definition may not be so rigorous, it helps to identify two different roles an articulator may play in generating different sounds. For example, `TDy` is the active articulator for all vowels while `TTy` is inactive for most of them. This concept is better illustrated by looking at real data, such as the production of phoneme /d/ in Fig. 4.2. Since during the initial closing phase of generating /d/, the tongue tip has to touch the back of the upper teeth, `TTy` is an active articulator for this phone. This is clearly demonstrated by examining the trajectory of `TTy`, where it reaches a fixed target value (approximately 10 mm) at the beginning of each /d/ production. In contrast, `TDy` is inactive for /d/ since its positions vary quite a lot depending on the neighboring phones (where some of them have `TDy` as an active articulator).

Another important and common property of the articulators is their anticipatory behavior, which refer to that fact that they often move towards the target of a phoneme well ahead of its acoustic realization. This can be clearly observed in both examples. For example, in producing /d/, the movement of `TTy` towards its target all starts in the previous phone as shown in Fig. 4.2. Note that this anticipatory behavior can be observed from the movement of the formants (especially $F2$) in the spectrogram as well. Sometimes the anticipation of articulators can also occur more than one phone ahead, *e.g.*, the movement of `TDy` in producing /k/ and /iy/ in Fig. 4.2. This is possible since in the generation of /n/ (the phone preceding /k/), /n/ and /w/ (the two phones preceding /iy/), `TDy` is an inactive articulator.

As a quick summary, this section has illustrated the dual nature (continuous versus discrete) of speech, the correlation among articulators, the concept of active versus inactive

[2]This concept is closely related to the feature values in the overlapping articulatory features theory [66]: roughly speaking, active articulators correspond to nonzero feature values while inactive articulators correspond to a feature value of zero.

articulators and the important anticipatory behavior in speech production via two typical examples. They by no means cover all the important phenomena in speech, in fact, not even most of them. Instead, these are only a few that can be easily demonstrated through measurements in the UW-XRMB database so that the complexity and subtlety of speech science research as well as the challenge of studying articulatory dynamics can be better appreciated. Many behaviors of articulators during speech production, especially at a more detailed level, are still far from being well understood and is the subject of a large amount of ongoing research in speech science [53, 106, 107, 249, 250, 252, 251, 253, 254].

4.2 Learning the Articulatory-to-Acoustic Mapping

This section studies a somewhat simple problem, which is from some point measurements in the articulatory space to features in the acoustic space. It is simple in the sense that it can be characterized by a static nonlinear mapping, i.e., without taking into account the dynamic behavior of the articulators, as opposed to the study of articulatory dynamics to be carried out in the next section. The static nature of this mapping is because of the underlying physical laws which govern the generation of speech sounds, and is exemplified by sophisticated yet well-developed transmission line models in speech science to calculate acoustic waveforms from static vocal tract configurations [161]. If detailed measurements along the mid-sagittal plane are available, one can use an α-β model [109] to predict the vocal-tract area functions for a given speaker and input them to a transmission line acoustic model to compute the corresponding acoustic output. However, such a knowledge-based scientific approach is not applicable for the UW-XRMB database since we only have a few point measurements and there are no well-established methods to reconstruct the entire vocal tract configuration based on a few points. Therefore, a data-driven approach, as is typical in machine learning, is adopted in this section.

4.2.1 A Simple Linear Articulatory-to-Acoustic (ATA) Mapping

Although knowledge from speech science and similar past studies [133, 211] indicate that such a mapping could be highly nonlinear, it still makes sense to start from a simple linear model. Linear mapping is very easy to implement; it can serve as a baseline to compare

with more advanced but complicated methods and also gives a rough estimate on the difficulty of the problem.

The articulatory measurement used here is the ten dimensional vector shown in equation (3.2). The acoustic feature is the mel-frequency cepstral coefficients (MFCCs) which are most popularly used in speech and speaker recognition applications [204, 259]. MFCCs are computed from speech waveforms using analysis frames 10 ms apart and 24 mel scale filter banks, and all 24 coefficients are retained as the acoustic feature. In order to be synchronous with the calculated MFCCs, the articulatory measurement is resampled to have the same period of 10 ms (from the original 6.866 ms) using a spline interpolation. One speaker's data (JW15), which has been hand-labeled, is used in this study. Since this thesis focuses on continuous natural speech, only this type of utterance is included in the training and test set, which consists of 50 and 10 sentences respectively. In terms of data points, the training set contains 13,685 and the test set has 2,518. This amount of data is large enough to set up a very well-defined machine learning problem to suit the purpose here, although it may be considered little data in many speech applications since it corresponds to only a few minutes of speech.

As in Section 3.3.1, a simple linear regression model is used and the values of R^2 statistics for the first 12 MFCCs on the training set are listed in table 4.2.1. Based on the

Acoustic Feature	MFCC1	MFCC2	MFCC3	MFCC4
R^2 Statistics Value	0.2378	0.3302	0.2297	0.3996
Acoustic Feature	MFCC5	MFCC6	MFCC7	MFCC8
R^2 Statistics Value	0.2820	0.1724	0.2013	0.0581
Acoustic Feature	MFCC9	MFCC10	MFCC11	MFCC12
R^2 Statistics Value	0.0592	0.2314	0.0958	0.0986

Table 4.1: R^2 statistics for a linear ATA mapping.

very low R^2 values in the table, it is clear that such a rudimentary linear model fails to capture the ATA mapping and further test is unnecessary. It also indicates that such a mapping is highly nonlinear and may be quite difficult. Therefore, the remainder of this study focuses on using powerful nonlinear regression methods, namely two popular neural network architectures: multilayer perceptrons (MLPs) and radial basis functions (RBFs),

to model the ATA mapping. Mathematical background of neural networks will not be presented in this thesis due to space constraint, if necessary readers may refer to a number of good books [19, 97, 108] or the documentation of Matlab's neural network tool box [55] for more details.

4.2.2 ATA Mapping Approximated by MLPs

MLPs with one hidden sigmoid layer have been widely used in many difficult function approximation (regression) problems. Such a network architecture has been proven to be a general function approximator under very mild conditions [46, 83, 113]. Furthermore it also scales nicely with the dimensionality of the input space [10] so that it is much less sensitive to the *curse of dimensionality* than many other function approximators (such as an n^{th} order polynomial), making it suitable for approximating multidimensional input-output relationships. However, the universal function approximation theorem is only an existence proof. In practice the number of hidden neurons has to be determined through experiments. This is carried out first.

Choosing the number of hidden neurons

First a small development set with five training sentences (1452 data points) and one test sentence (277 data points) is used in order to save computational time and to get a rough idea about the complexity of the mapping. The Levenberg-Marquardt algorithm with Bayesian regularization is chosen as the training algorithm for this small set since it possesses a number of desirable properties and is also an efficient fast training algorithm for MLPs. A little more detail about this algorithm is provided below.

As with any standard regularization techniques, the cost function J for training a neural net under Bayesian regularization is a weighted sum of two terms: the usual error function which is the mean of the sum of squares error (MSE) and a regularization term which is the mean of the sum of squares of the network weights and biases (MSW), i.e.,

$$J = \text{MSE} + \gamma \cdot \text{MSW}. \tag{4.1}$$

For MLPs, smaller weights and biases will force the network response to be smoother and less likely to overfit. But unlike traditional regularization techniques where the regulariz-

ing parameter γ has to be chosen heuristically, under the Bayesian framework it can be determined automatically [159]. More specifically for MLPs, the algorithm is developed under the assumption that both the noise in the training data set and prior distribution of the network parameters (weights and biases) are Gaussian while γ has a noninformative uniform prior [78]. As a by-product of calculating γ, it also provides a measure of how many parameters are effectively used to reduce MSE during training, which is very helpful when proper network size, *i.e.*, the number of hidden neurons, has to be determined for a specific regression problem. Estimation of γ involves the calculation of Hessian matrix, whose complexity is at least $\mathcal{O}(NW^2)$ where N is the total number of training examples and W is the total number of network parameters, but this expensive computational overhead can be avoided by using an approximate Hessian matrix as provided by quasi-Newton optimization methods [194]. Therefore, it is natural to integrate Bayesian regularization with a quasi-Newton method for MLP network training. The Levenberg-Marquardt algorithm [162], which is a quasi-Newton method specifically designed to minimize MSE and has been demonstrated to be an efficient algorithm for MLP training [98], provides a reasonably accurate approximate Hessian matrix and fits this purpose well. This has been implemented in Matlab as the function `trainbr`.

Initially 20 hidden neurons are tried and sample output from Matlab is as follows:

```
TRAINBR, Epoch 290/300, SSE 19957.8/1e-05, SSW 332.902, Grad 2.43e+01/1.00e-10, #Par 7.12e+02/724
TRAINBR, Epoch 295/300, SSE 19957.6/1e-05, SSW 332.992, Grad 2.37e+01/1.00e-10, #Par 7.12e+02/724
TRAINBR, Epoch 300/300, SSE 19957.4/1e-05, SSW 333.092, Grad 2.35e+01/1.00e-10, #Par 7.12e+02/724
```

The output shows that the training has virtually converged with 300 epochs since both the sum squared error (SSE) and sum squared weights (SSW) remained almost constant over the last ten iterations. It also shows that 712 out of 724 parameters are effectively used in the training, which indicates that no overfit has occurred with 20 hidden neurons. Therefore, 30 hidden neurons are used and a sample output is:

```
TRAINBR, Epoch 290/300, SSE 17136.4/1e-05, SSW 484.05, Grad 2.43e+01/1.00e-10, #Par 1.06e+03/1074
TRAINBR, Epoch 295/300, SSE 17136.4/1e-05, SSW 484.045, Grad 2.43e+01/1.00e-10, #Par 1.06e+03/1074
TRAINBR, Epoch 300/300, SSE 17136.4/1e-05, SSW 484.044, Grad 2.43e+01/1.00e-10, #Par 1.06e+03/1074
```

Again it shows that training converged with 300 epochs and no overfit occurs (1060 out of 1074 parameters are effectively used). One could keep increasing the number of hidden neurons and monitor the output to eventually choose the appropriate number of hidden

neurons for this small development set. However, it turns out that the memory requirement of this algorithm for larger MLPs is prohibitive: with 20 hidden neurons the algorithm requires up to 600MB RAM; with 30 hidden neurons it requires up to 900MB. Therefore, this simple case study shows that Levenberg-Marquardt algorithm with Bayesian regularization is not applicable for large scale problems (even the small development set for ATA mapping) in spite of its very desirable properties. To continue further study, training algorithms coping well with large MLPs as well as alternative methods of detecting overfit are needed.

After a few comparisons and also heavily based on the author's previous experience of working with MLPs [146], the scaled conjugate gradient algorithm [167] is chosen to train the large MLPs (with more than a few thousand parameters) in this study. The scaled conjugate gradient algorithm strikes a good balance between training speed and memory requirement, and has been demonstrated to be the most efficient for training large MLPs (where quasi-Newton methods such as Levenberg-Marquardt are not applicable) in many situations. Overfit is detected by *early stopping*, but for simplicity no separate validation set is used. The complete 50 sentence training set is used for training while test error on the 10 sentence test set is directly monitored throughout the training phase to detect overfit. Initially 100 hidden neurons are tried and the training and test errors as a function of the training epochs are shown in Fig. 4.3. The result indicates no overfit. The number of hidden neurons are subsequently increased to 200 and 300 and the corresponding training and test errors are shown in Fig. 4.4 and Fig. 4.5. Again, no obvious overfit occurs in either case. Comparing Fig. 4.3, Fig. 4.4 and Fig. 4.5, it can also be observed that although the training errors decrease as the number of hidden neurons increase from 100 to 300, the test errors stay about the name. Keep in mind that MLP training algorithms are local optimization algorithms, and the performance of trained MLP networks does depend on initial values of the parameters. Therefore a number of different random initializations have been tried on each network architecture but it turns out that the effect of initialization is very minor, and the training and test errors shown in Fig. 4.3 to Fig. 4.5 are typical cases[3]. Although the MSE as shown in the figures is a good indicate of the performance of

[3]The long computing time of training MLPs also prevent this from carrying out too extensively: training a MLP network with 300 hidden neurons takes a whole day on a Pentium 4 2.4 GHz PC.

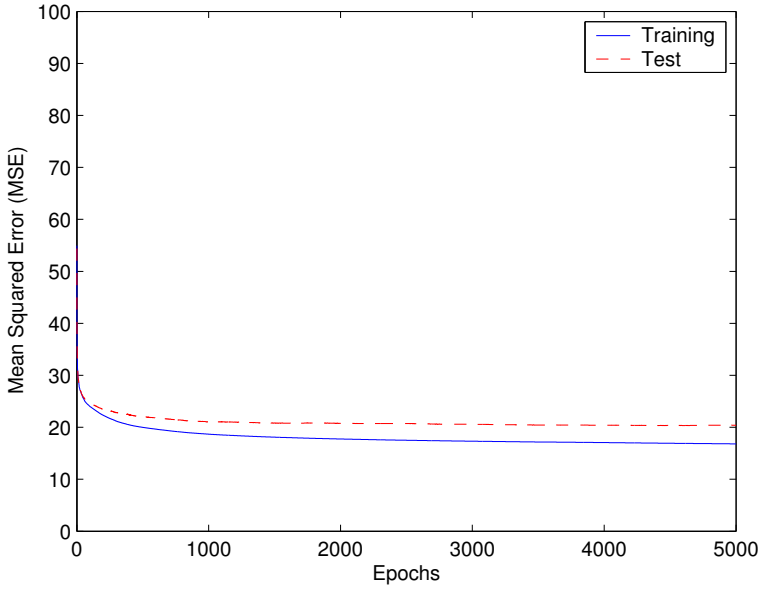

Figure 4.3: Training and test error of a MLP network with 100 hidden neurons.

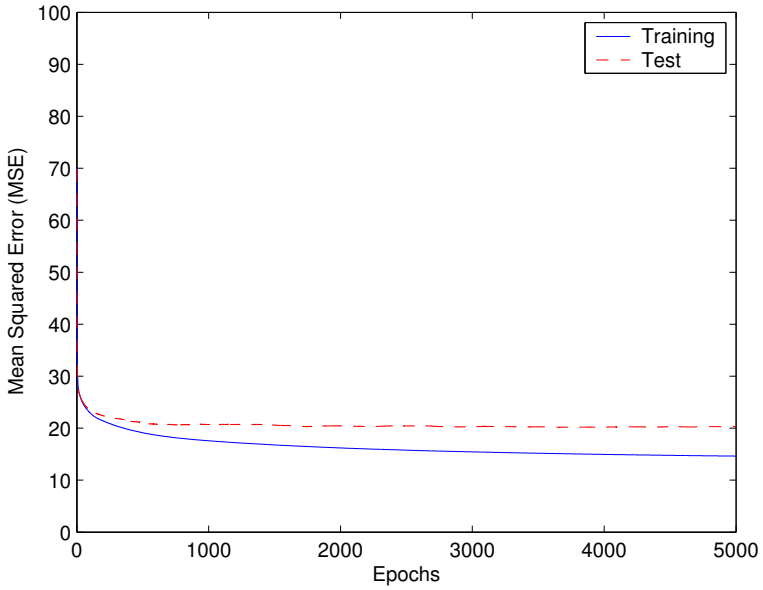

Figure 4.4: Training and test error of a MLP network with 200 hidden neurons.

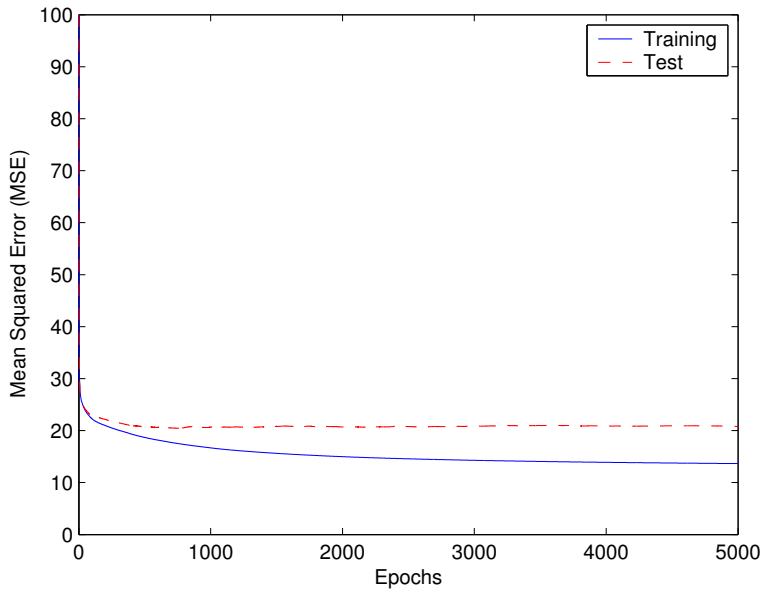

Figure 4.5: Training and test error of a MLP network with 300 hidden neurons.

a MLP network, in many cases such a single number doesn't provide enough information to facilitate detailed analysis and guide further improvement. Therefore more careful analysis on the results obtained so far is performed before further refinements are attempted.

Result analysis

Fig. 4.6 shows the test result of a trained MLP network with 200 hidden neurons on one test sentence. Out of the 24 predicted MFCCs, 12 are plotted in the figure. It can be seen that the overall result is not very good: reasonably good estimation of lower order MFCCs is obtained but the result is very poor for higher order ones. The poor estimation of higher order MFCCs could be attributed to their relatively small magnitude and high sensitivity to environmental noise, while the recording noise in the UW-XRMB database is not negligible as explained in Section 3.1.

A more comprehensive and informative way of evaluating the performance of a neural network for function approximation problems is to perform a linear regression analysis between the network response and the corresponding targets. If the network is trained

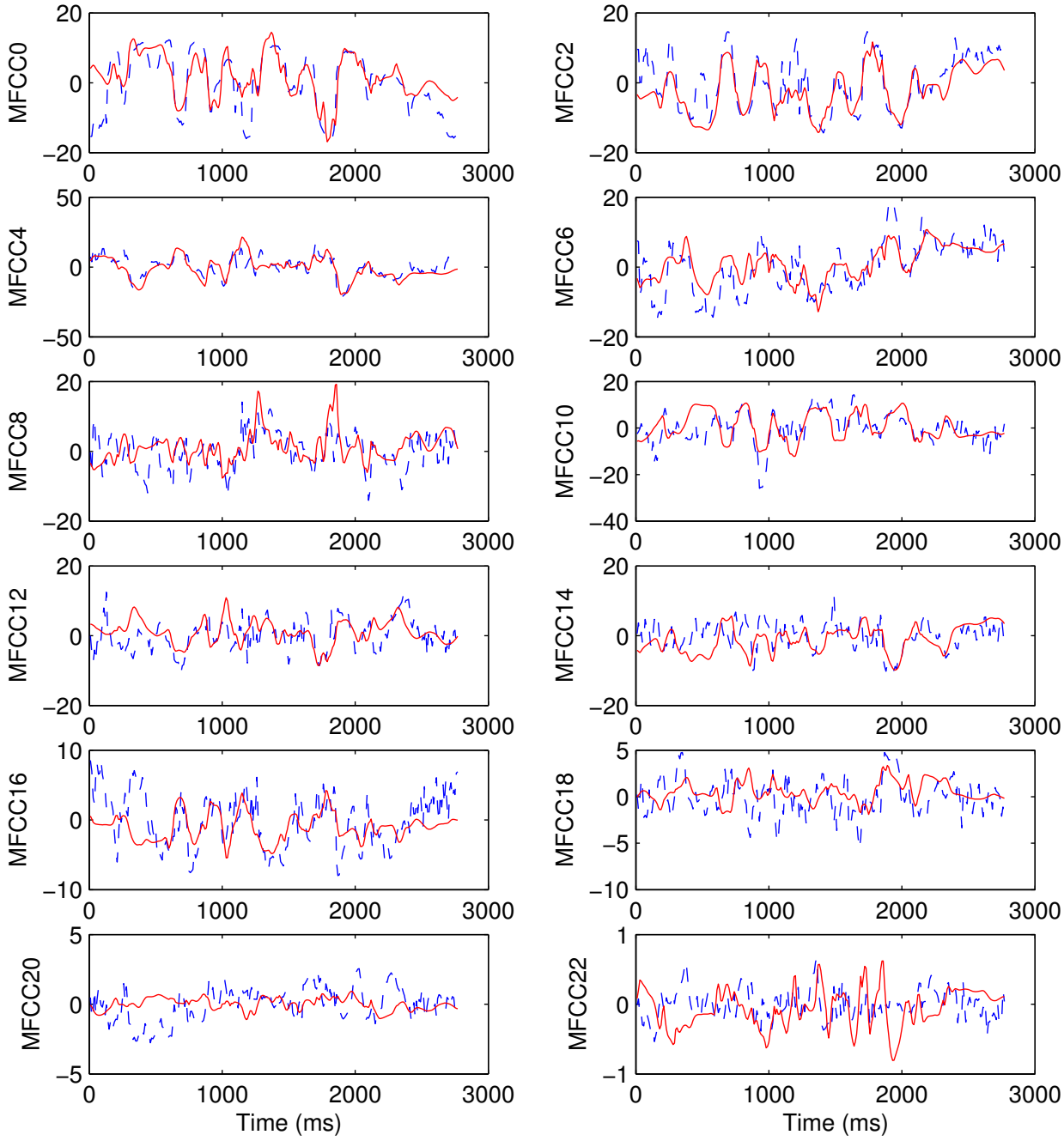

Figure 4.6: True and estimated MFCCs for a test sentence. Dash blue line: true MFCCs; solid red line: estimated MFCCs from a MLP network with 200 hidden neurons.

perfectly, the R-values of such a linear regression analysis for all the output variables will be 1 when feeding the training data to the neural net; if the network can also predict perfectly, the R-values will be 1 on the test set as well. In general R is a number between -1 and 1, and the closer it is to 1 the better is the network performance[4]. Such a linear regression analysis has been done on the training and test sets separately and the results are listed in Table 4.2. By looking at the R-values on the training set, it shows the same trend

Features	R_{train}	R_{test}	Features	R_{train}	R_{test}
MFCC0	0.7811	0.6573	MFCC12	0.5537	0.3892
MFCC1	0.7971	0.6720	MFCC13	0.6468	0.5521
MFCC2	0.7707	0.6760	MFCC14	0.6398	0.5393
MFCC3	0.7627	0.6622	MFCC15	0.5137	0.3651
MFCC4	0.8477	0.7642	MFCC16	0.6248	0.4848
MFCC5	0.8071	0.6781	MFCC17	0.3056	0.2579
MFCC6	0.6840	0.5718	MFCC18	0.5407	0.4468
MFCC7	0.7303	0.6456	MFCC19	0.4460	0.3278
MFCC8	0.6839	0.4701	MFCC20	0.3927	0.2697
MFCC9	0.5834	0.4714	MFCC21	0.3320	0.2365
MFCC10	0.7238	0.5896	MFCC22	0.1371	0.0757
MFCC11	0.6020	0.4154	MFCC23	0.0402	0.0104

Table 4.2: R-Values of the linear regression analysis for a MLP network with 200 hidden neurons.

as the one test example reveals: R-values are reasonably high (around 0.8) for lower order MFCCs while quite poor for higher order ones. However, by comparing with table 4.2.1, it can be observed that this nonlinear mapping does provide a significant improvement over a simple linear one[5]. R-values obtained from the test set shows the same trend, but are also marginally lower than those obtained from the training set as is typically the case.

[4]Readers are cautioned that MSE and linear regression R-values are different measures on the quality of the mapping and they don't always agree; when inconsistency occurs, MSE is a more direct measure for function approximation problems.

[5]R^2 statistics is simply the square of R-values for linear regression analysis [112].

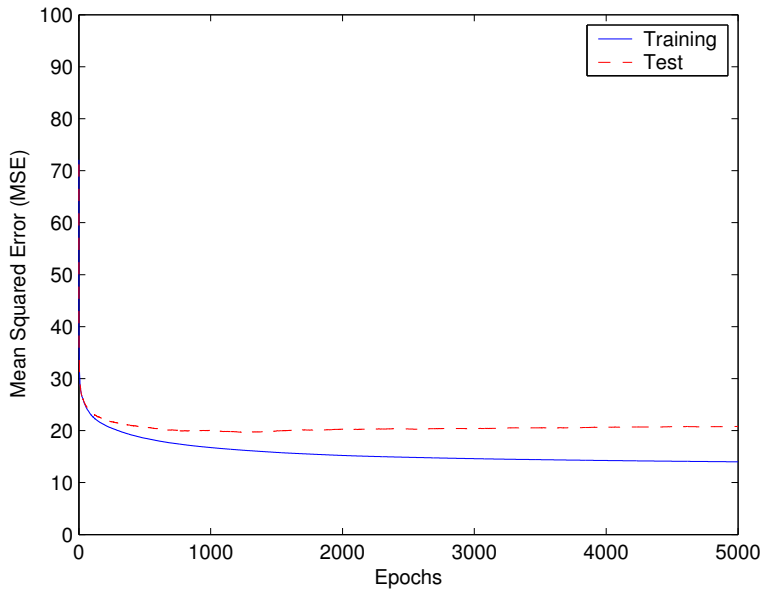

Figure 4.7: Training and test errors of a MLP network with 200 hidden neurons on cleaned data.

Enhancing input data quality

It is always possible that network output cannot be accurately predicted from network input due to lack of information. Do articulatory measurements in the UW-XRMB database provide enough information to predict MFCCs? Unfortunately the answer is no based on knowledge about human speech production. First of all, there is usually a short period of leading and/or trailing silences for most sentences in the data set. During these periods, the movement of the articulators does not produce any speech while the target MFCCs merely reflect the environmental noise (which is relatively small but not negligible in this database). Therefore to enhance input data these silence periods should be removed. This is carefully done by hand and the resulting *cleaned* training and test set have 11,920 and 2,210 data points respectively. Training and test MSEs on this cleaned data set for a MLP network with 200 hidden neurons are shown in Fig. 4.7, where no obvious improvement over Fig. 4.4, similar experiment on the *raw* data set, is achieved.

Moreover, there are two pieces of important acoustic-related information missing from

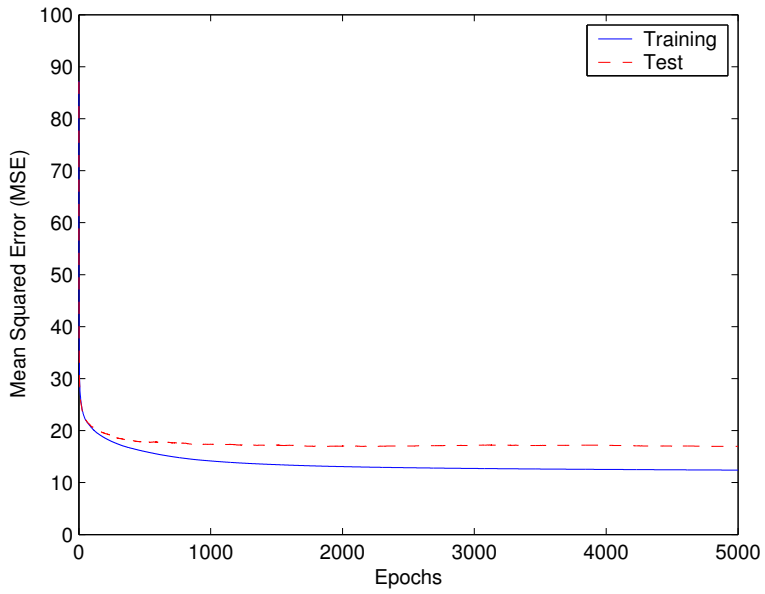

Figure 4.8: Training and test errors of a MLP network with 200 hidden neurons on appended data.

the articulatory measurements: the glottal source and the velum. The status of the glottal source controls whether a speech sound is voiced or unvoiced, and the lowering of the velum introduces nasal coupling to produce nasal sounds. One way to infer this information is from the generated acoustic waveform. Since all the sentences have been hand-labeled, binary values for the glottal and nasal features are derived based on the phone boundaries and identities. The dimensionality of input vectors is thus augmented from 10 to 12 to include these two extra binary variables. It should be pointed out that such a phone boundary based labeling method is only an approximation due to the well-known vocalization and nasalization phenomena in speech, i.e., the voice and nasal features tend to spread to neighboring phones. But this is a big improvement over the complete missing of information and saves extra (and potentially large amount of) manual work to get more accurate boundaries for the glottal and nasal features. Again a MLP network with 200 hidden neurons is used and the training and test errors on this *appended* data set is shown in Fig. 4.8. Considerable improvement over Fig. 4.4 is obtained this time, with almost 20% reduction on the test error (16.97 versus 20.34), and it confirms that the glottal and velum

features are quite important for the prediction of MFCCs.

Result analysis on enhanced input data

Similar analysis is carried out on the MLP network trained on the appended data. First the test result on the same test sentence is shown in Fig. 4.9, where noticeable improvements can be observed. Next linear regression analysis on the training and test set are performed and the results are summarized in Table 4.3. It can be observed that R-values for all the

Features	R_{train}	R_{test}	Features	R_{train}	R_{test}
MFCC0	0.8198	0.7374	MFCC12	0.6467	0.5304
MFCC1	0.8658	0.8364	MFCC13	0.7056	0.6320
MFCC2	0.8262	0.7506	MFCC14	0.7096	0.5961
MFCC3	0.8117	0.7206	MFCC15	0.5893	0.4442
MFCC4	0.8856	0.8109	MFCC16	0.6683	0.5284
MFCC5	0.8569	0.7510	MFCC17	0.3924	0.3440
MFCC6	0.7347	0.6343	MFCC18	0.5850	0.4968
MFCC7	0.7921	0.7333	MFCC19	0.4931	0.3875
MFCC8	0.7234	0.5987	MFCC20	0.4685	0.3878
MFCC9	0.6667	0.5412	MFCC21	0.4371	0.2802
MFCC10	0.8016	0.6890	MFCC22	0.2612	0.1407
MFCC11	0.6778	0.5265	MFCC23	0.1246	0.0942

Table 4.3: R-Values of the linear regression analysis for a MLP network with 200 hidden neurons on the appended data.

MFCCs on both the training and test set are improved noticeably. However, the same difficulty of predicting higher orders of MFCCs persists. The overall result suggests that the nonlinear mapping learned by MLP networks may be good enough to predict lower order MFCCs, which is still quite useful since most practical applications only use lower order ones, such as the first twelve used in many state-of-the-art speech recognizers.

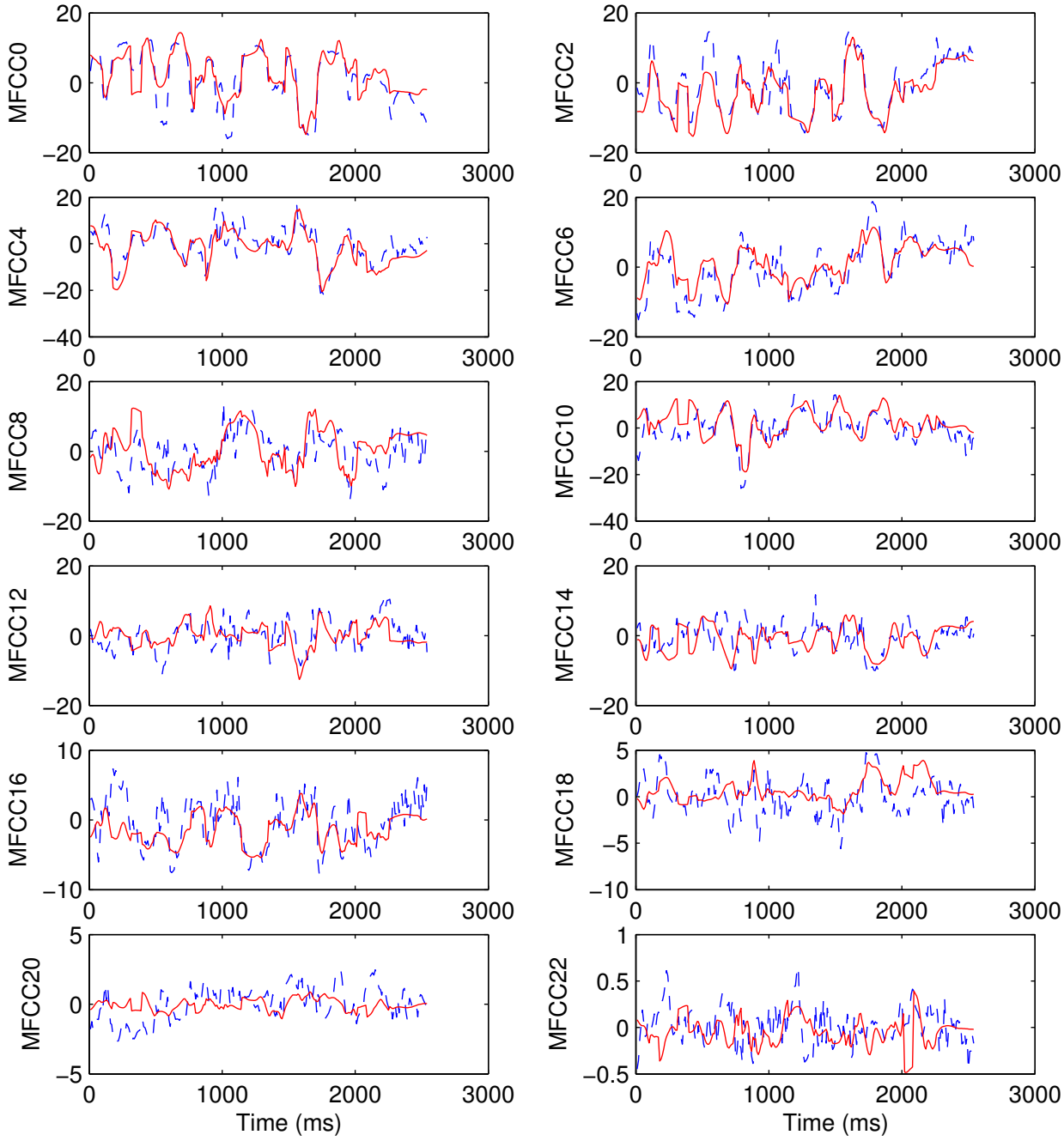

Figure 4.9: True and estimated MFCCs for a test sentence on appended data. Dash blue line: true MFCCs; solid red line: estimated MFCCs from a MLP network with 200 hidden neurons.

4.2.3 ATA Mapping Approximated by RBFs

Another popular neural network architecture that has been used to solve difficult nonlinear regression problems is radial basis functions (RBFs). Despite their very different origin and mathematical structures, RBFs and MLPs share similar universal function approximation properties [183, 191]. RBFs also scale nicely with the dimensionality of the input space [174] so that it is a good candidate for approximating multidimensional input-output relationships. In this section, Gaussian function is used as the transfer function of the radial basis neurons, which is one of the most popular choices. Two types of RBF networks, namely regularization RBFs and generalized RBFs, are studied[6]. This section runs parallel to the previous section and also serves as a comparative case study between MLPs and RBFs for a practical function approximation problem.

Using Regularization RBF Networks

A regularization RBF network has the same number of radial basis (or hidden) neurons as the number of training samples, each centering at a training sample. It essentially performs a multi-dimensional interpolation that passes through all the training points. Such a RBF network will reach zero training error, but typically results in a large number of hidden neurons (for a large training set), is subjective to numerical problems and also highly sensitive to noise presented in the training data. Nevertheless, it can be designed very quickly and the training algorithm is readily available in Matlab. Therefore, it serves as the first step to apply RBFs for the ATA mapping.

Since the number of hidden neurons and their centers are fixed, the only adjustable parameter is the width or spread of the Gaussian basis function, which controls the influence of a training sample on its vicinity and the smoothness of the interpolation. This may be automatically determined from the training data by some heuristic methods, or simply tuned by monitoring the error on a validation set or the test set. For the ATA mapping problem, the spread is adjusted manually from 0.5 to 5 with a step size of 0.1 and test error is monitored directly. Fig. 4.10 shows the result of such an experiment, where the

[6]Please note that the terminologies used here is very different from those used in Matlab's neural network toolbox but follows typical literatures in the neural network community [108, 191].

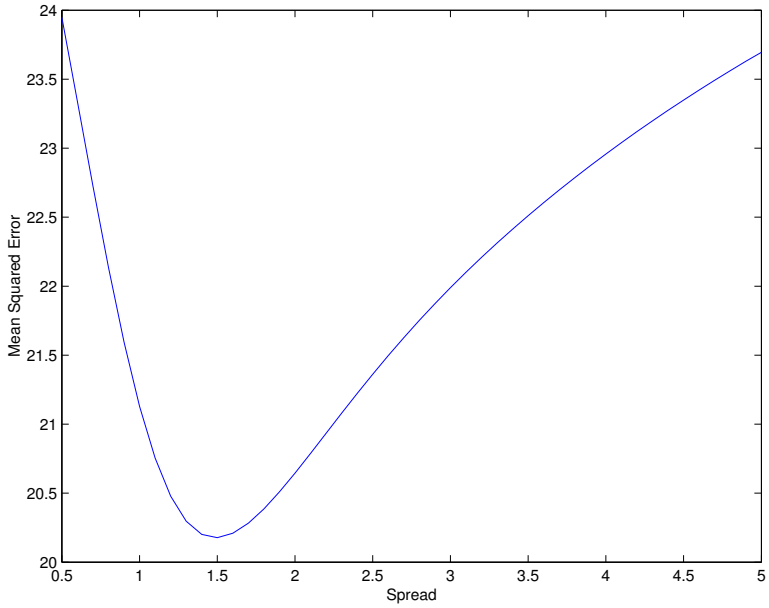

Figure 4.10: Test error of a regularization RBF network versus the spread of the Gaussian basis function.

network test error versus the spread of the Gaussian basis function is plotted. The optimal spread so determined is 1.5, but within a certain range the test error is not very sensitive to the spread of the basis function.

Next a regularization RBF network with the optimal spread (1.5) for its basis functions is trained and tested on the raw data set, and similar result analysis as in the MLP case is performed. Fig. 4.11 shows the test result on the same test sentence as in the MLP case. The figure shows the same trend: lower order MFCCs can be predicted much more accurately than higher order ones. The overall result seems to be quite comparable to that obtained from a MLP network. A linear regression analysis between network output and the corresponding target values are also performed. Since a regularization RBF network achieves zero training error, the analysis only needs to be done on the test set. The result is summarized in Table 4.4, along with the R values of a MLP network with 200 hidden neurons on the same test set to facilitate comparison. It can be observed that the R-values of the regularization RBF network is consistently higher than those of the MLP network,

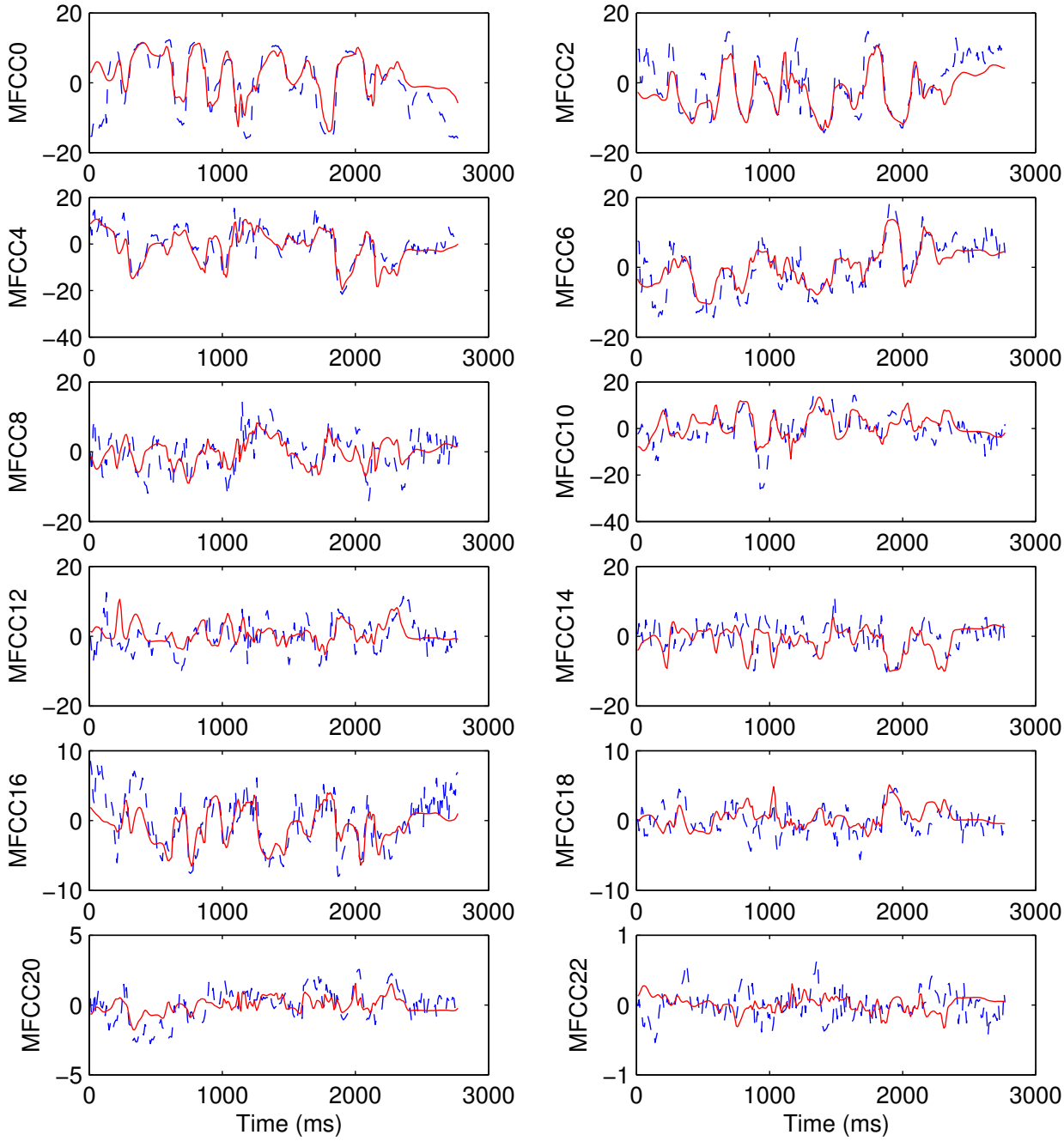

Figure 4.11: True and estimated MFCCs for a test sentence on raw data. Dash blue line: true MFCCs; solid red line: estimated MFCCs from a regularization RBF network.

Features	R_{RBF}	R_{MLP}	Features	R_{RBF}	R_{MLP}
MFCC0	0.7248	0.6573	MFCC12	0.4805	0.3892
MFCC1	0.7597	0.6720	MFCC13	0.6296	0.5521
MFCC2	0.7190	0.6760	MFCC14	0.6215	0.5393
MFCC3	0.7256	0.6622	MFCC15	0.4894	0.3651
MFCC4	0.8079	0.7642	MFCC16	0.5726	0.4848
MFCC5	0.7437	0.6781	MFCC17	0.4221	0.2579
MFCC6	0.6549	0.5718	MFCC18	0.5473	0.4468
MFCC7	0.6970	0.6456	MFCC19	0.4011	0.3278
MFCC8	0.5741	0.4701	MFCC20	0.4927	0.2697
MFCC9	0.5566	0.4714	MFCC21	0.3473	0.2365
MFCC10	0.6709	0.5896	MFCC22	0.3890	0.0757
MFCC11	0.5386	0.4154	MFCC23	0.3515	0.0104

Table 4.4: R-Values of the linear regression analysis for a regularization RBF network and a comparable MLP network on raw data.

and the improvement is especially significant for higher orders of MFCCs (although the absolute value is still quite low). However, these improved R-values don't translate into a equally significant MSE reduction on the test set (20.18 for regularization RBF versus 20.34 for MLP), which is a more direct measure on the quality of function approximation.

The regularization RBF network is further trained and tested on the cleaned and appended data set. The reduction of test error on the cleaned data is negligible but it is quite significant on the appended data, similar to the MLP case. This can be observed by carefully examining the testing result on the same test sentence shown in Fig 4.12. Similar linear regression analysis is also preformed and the result is summarized in Table 4.5, along with the R values of a MLP network with 200 hidden neurons under the same condition. It can again be observed that R-values of the regularization network are consistently higher than those of the MLP network, although the improvement is a bit smaller than that on the raw data set. Judging by the more direct measure MSE however, MLPs and regularization RBFs perform very similar on both the raw and appended data set (the actual values are summarized in Table 4.7 at the end of this section on page 70).

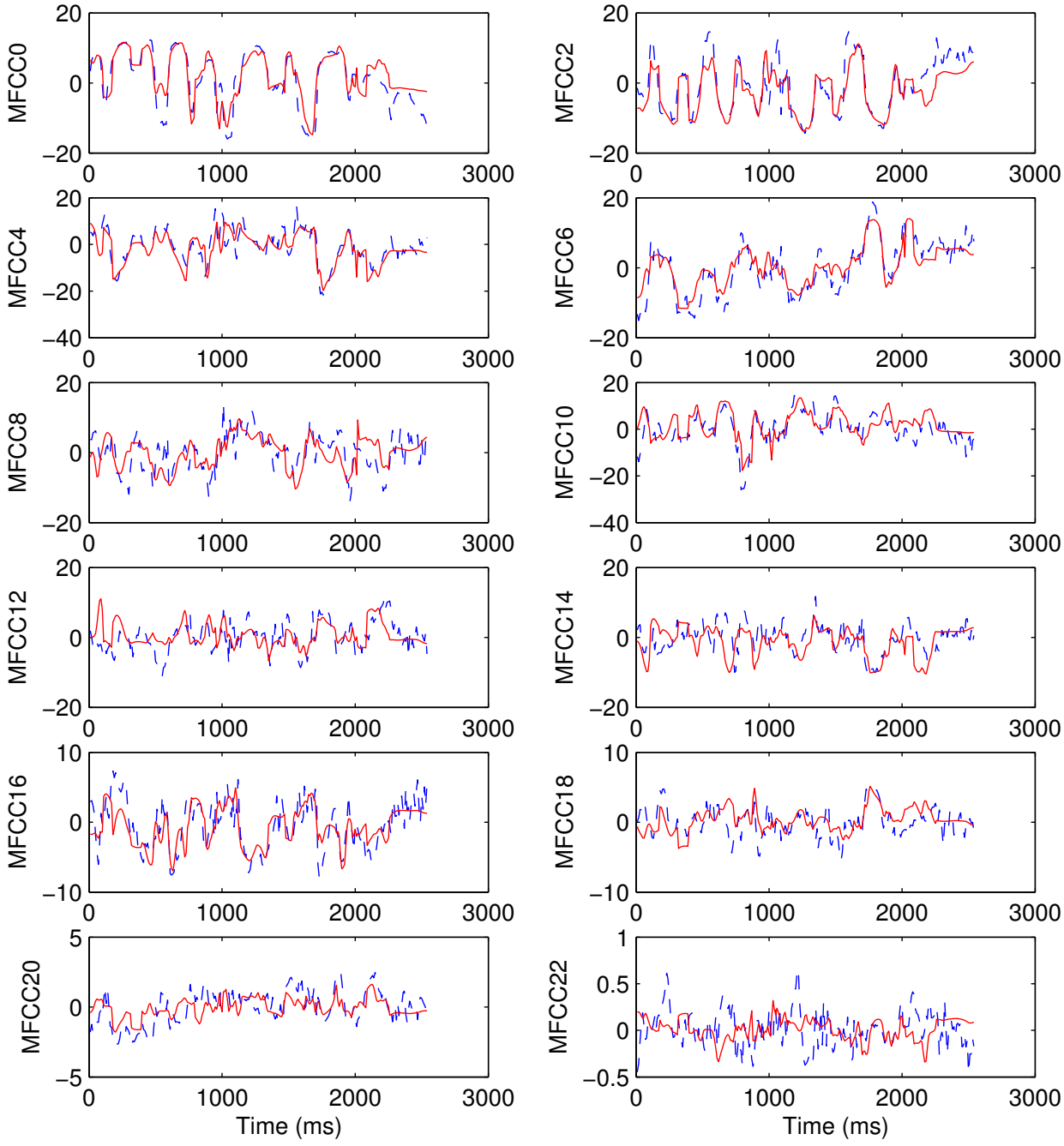

Figure 4.12: True and estimated MFCCs for a test sentence on appended data. Dash blue line: true MFCCs; solid red line: estimated MFCCs from a regularization RBF network.

Features	R_{RBF}	R_{MLP}	Features	R_{RBF}	R_{MLP}
MFCC0	0.7699	0.7374	MFCC12	0.5505	0.5304
MFCC1	0.8695	0.8364	MFCC13	0.6680	0.6320
MFCC2	0.7771	0.7506	MFCC14	0.6325	0.5961
MFCC3	0.7737	0.7206	MFCC15	0.5255	0.4442
MFCC4	0.8419	0.8109	MFCC16	0.6108	0.5284
MFCC5	0.8072	0.7510	MFCC17	0.4457	0.3440
MFCC6	0.6920	0.6343	MFCC18	0.5851	0.4968
MFCC7	0.7649	0.7333	MFCC19	0.4362	0.3875
MFCC8	0.6478	0.5987	MFCC20	0.5006	0.3878
MFCC9	0.6121	0.5412	MFCC21	0.3932	0.2802
MFCC10	0.7131	0.6890	MFCC22	0.4165	0.1407
MFCC11	0.5752	0.5265	MFCC23	0.4012	0.0942

Table 4.5: R-Values of the linear regression analysis for a regularization RBF network and a comparable MLP network on appended data.

Using Generalized RBF Networks

A generalized RBF network refers to one that has less (and usually much less) hidden neurons than the number of training samples. The number of hidden neurons needed is supposed to represent the internal complexity of the mapping instead of simply growing with the size of training data as in regularization networks, which is similar to the role hidden neurons play in a MLP network. When appropriately designed, such a network can not only represent the desired input-output mapping more efficiently, but also cope well with noisy input data. However, the number of hidden or basis neurons again have to be determined from experiments. There are many variants of generalized RBF networks and many ways to design them accordingly. For example, Matlab supports an iterative procedure which adds one neuron at a time (centered at one of the training samples) based on orthogonal least squares criterion [37] until a pre-specified error goal or a maximum number of neurons is reached. Unfortunately, such an algorithm runs too slow for the size of the current training set. In this study, a much faster hybrid supervised/unsupervised

training algorithm proposed by Moody and Darken [168] is adopted and implemented in Matlab. This method first performs a *K-means clustering* on the training data to determine the center and spread of the radial basis functions, where the number of clusters K is pre-selected, then use a supervised learning method to determine the weights and biases of the output layer by performing a fast linear optimization (essentially a pseudo-inverse on the outputs of radial basis neurons).

Besides the number of hidden neurons, we also have to determine the spread of the basis functions as in regularization RBF networks. The K-means clustering algorithm not only returns centers of the clusters, which will serve as centers of radial basis neurons in the RBF network, but can also calculate the sample variance of the clusters. Intuitively the optimal spread of the basis functions should be related to these sample variances somehow. Here a simple method is adopted to determine the optimal spread: we let all the hidden neurons have the same spread and assume that the spread is proportional to the average sample mean of the clusters. The ratio between the spread and the average sample variance, referred to as the *spread ratio* hereafter, is determined through experiments. Parameter tuning to obtain the optimal network architecture is further complicated by the fact that K-means clustering is a local optimization algorithm and centers of the K clusters returned are sensitive to the initial values, which are usually picked randomly. Therefore, for a fixed number of hidden neurons, a number of training and test sessions should be run and the average test error should be taken as an appropriate measure. However, experiments show that the effect of initialization is quite minor for this mapping problem. Furthermore, it also turns out that the test error is not very sensitive to the number of hidden neurons as long as it is large enough. Networks that have the number of hidden neurons ranging from 500 to 3500 are tried on a fairly coarse scale and it is observed that test errors stay about the same for networks with 1500 or more hidden neurons (when each network also has its own optimal spread ratio).

The dependency of test error on the spread ratio is much stronger, however, and examples of two generalized RBF networks with 1500 and 2000 hidden neurons respectively are shown in Fig. 4.13 and Fig. 4.14, where the test error versus the spread ration is plotted. It can be observed that the optimal spread ratio is 10 for a generalized RBF network with 1500 hidden neurons and 12 for one with 2000 hidden neurons. The network that has

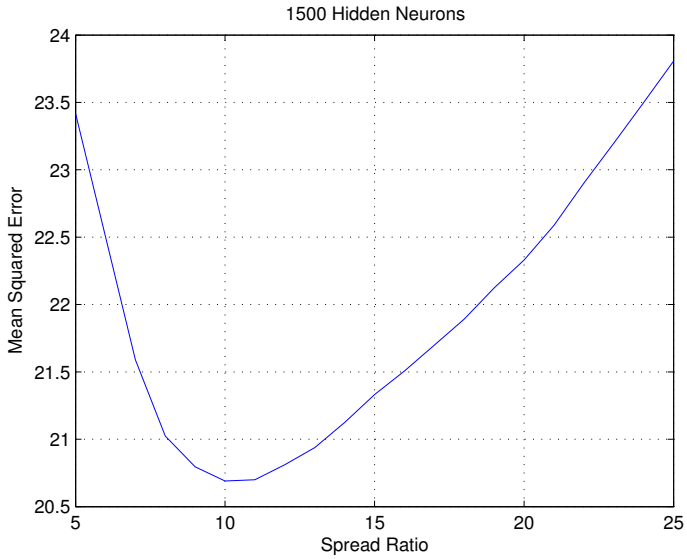

Figure 4.13: Test error of a generalized RBF network with 1500 hidden neurons as a function of the spread ratio.

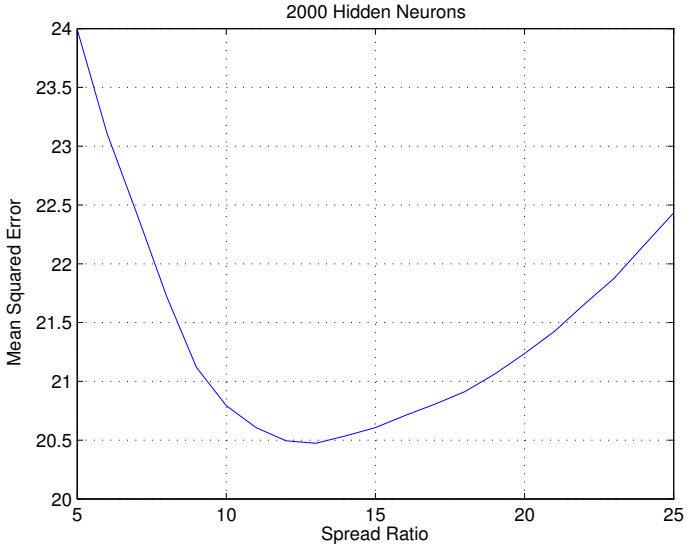

Figure 4.14: Test error of a generalized RBF network with 2000 hidden neurons as a function of the spread ratio.

2000 hidden neurons also performs slightly better than the one with 1500 hidden neurons. Adding more hidden neurons only provide very little gain in terms of test error (if at all), but with a big price to pay in terms of computational cost: when the number of hidden neurons is too large, the memory requirement is prohibitive (caused by the pseudo-inverse of a very large matrix), and the Linux PC used for this experiment (with 1GB RAM and 2GB swap space) can only handle hidden neurons up to 3500. Of course the motivation of using a generalized RBF network in the first place is to select a RBF network with the minimum number of hidden neurons that can approximate the desired mapping well. Therefore, a generalized RBF network with 2000 hidden neurons and a spread ratio of 12 is used in the following experiments.

Training and testing of the generalized RBF network is again performed on raw, cleaned and appended data sets as before. Test errors on the raw and cleaned data set are very similar while some gain is achieved on the appended data set. Test results on the same test sentence are plotted in Fig. 4.15 and Fig. 4.16 for the raw and appended data respectively. The overall quality is quite similar to those achieved by the MLP networks and regularization networks. Similar linear regression analysis on the test set is also carried out and the results are summarized in Table 4.6. The R-values are somewhat lower than those obtained from the regularization RBF network but are comparable with those obtained from the MLP network with 200 hidden neurons. They also show the same trend: in general R-values of lower order MFCCs are much higher than those of higher order MFCCs. The overall performance of generalized RBFs is very close to that of MLPs and regularization RBFs judging by MSEs on the test set.

4.2.4 Further Improvements by Ensemble Learning

In machine learning, ensemble methods [68, 176] refer to learning algorithms that construct a set of classifiers/regressors and determine the output on test data by taking a (weighted) vote of their predictions. The original ensemble method is Bayesian averaging, but two simple yet powerful ensemble methods recently developed are bagging [24] and boosting [80]. Given the difficulty of ATA mapping, it makes sense to try these ensemble learning methods to see if further improvements can be made by combining the power of individual regressors.

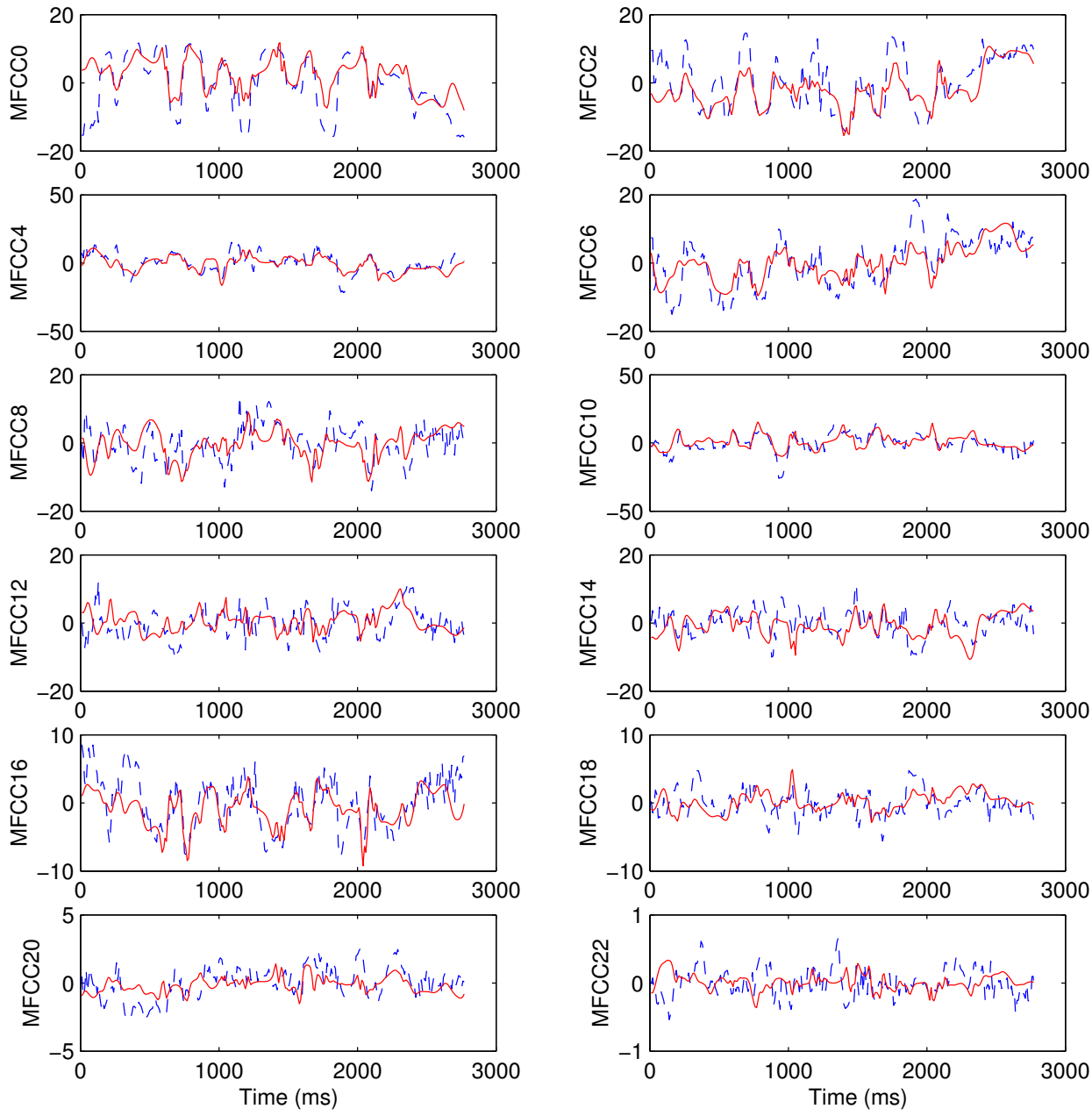

Figure 4.15: True and estimated MFCCs for a test sentence on raw data. Dash blue line: true MFCCs; solid red line: estimated MFCCs from a generalized RBF network.

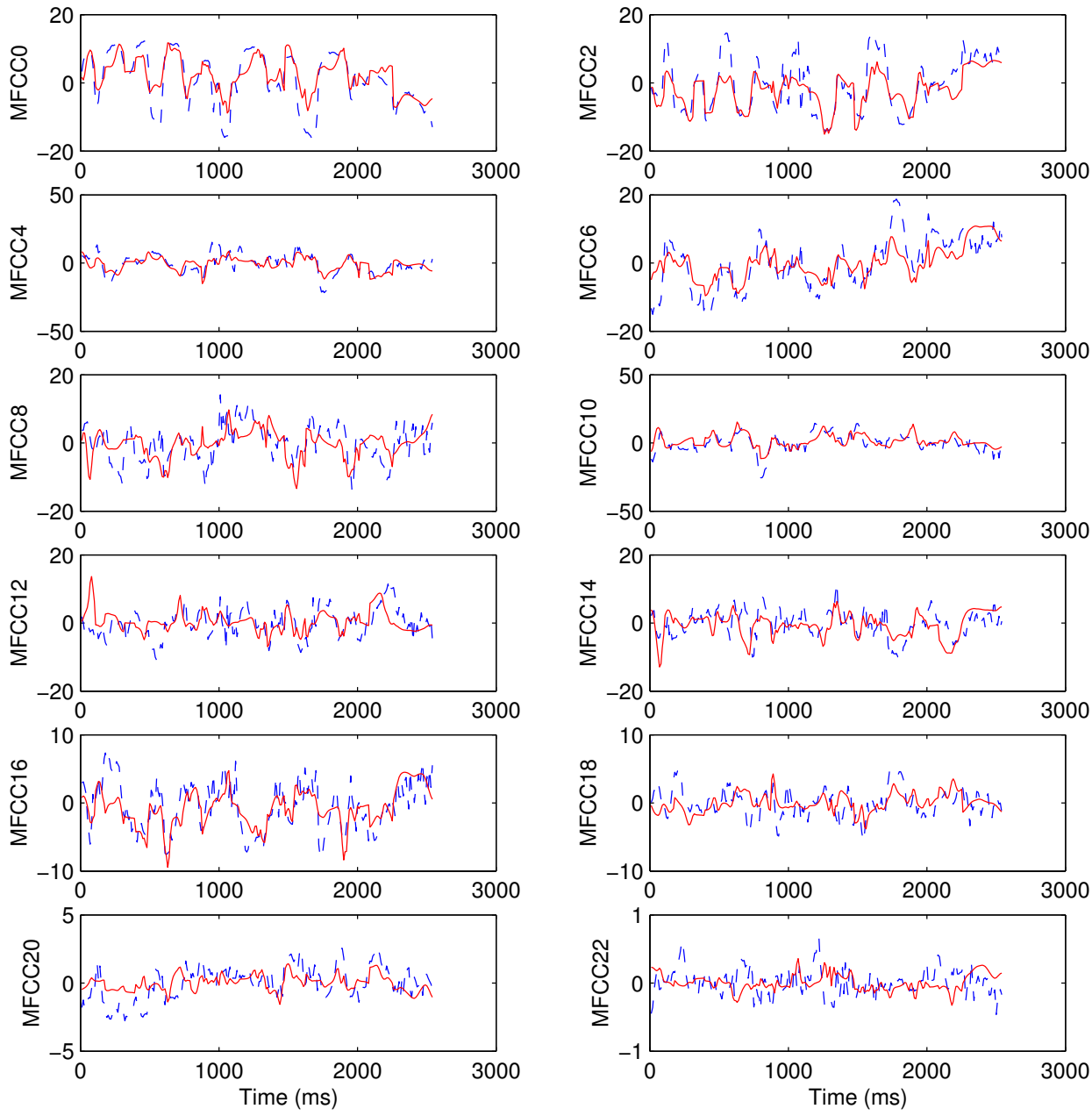

Figure 4.16: True and estimated MFCCs for a test sentence on appended data. Dash blue line: true MFCCs; solid red line: estimated MFCCs from a generalized RBF network.

Features	R_{raw}	R_{label}	Features	R_{raw}	R_{label}
MFCC0	0.6937	0.7467	MFCC12	0.4490	0.5426
MFCC1	0.7398	0.8539	MFCC13	0.6122	0.6492
MFCC2	0.6825	0.7577	MFCC14	0.5993	0.6247
MFCC3	0.6940	0.7435	MFCC15	0.4471	0.4707
MFCC4	0.7881	0.8174	MFCC16	0.5472	0.5632
MFCC5	0.7146	0.7731	MFCC17	0.3934	0.3863
MFCC6	0.6229	0.6755	MFCC18	0.5068	0.5492
MFCC7	0.6953	0.7446	MFCC19	0.3879	0.4082
MFCC8	0.5466	0.5966	MFCC20	0.4274	0.4565
MFCC9	0.5278	0.6248	MFCC21	0.2693	0.3434
MFCC10	0.6136	0.6734	MFCC22	0.3843	0.3946
MFCC11	0.4885	0.5351	MFCC23	0.3401	0.3764

Table 4.6: R-Values of the linear regression analysis for generalized RBF networks with 2000 hidden neurons on raw and appended data.

Generally speaking, bagging is good at variance control but weak at bias control, while boosting is very effective at bias control and provides some variance control as well. For MLP networks, since no overfit is observed so far, boosting should be more effective. Applying boosting to classification problems (especially binary class problems) is very straightforward, but it is considerably more involved for regression problems. The algorithm implemented here follows closely to the one proposed by Zemel and Pitassi [261], with a natural extension to vector outputs. An exponentiated squared error of weighted training distributions replaces the standard MSE to serve as the cost function for MLP training, where the weighting coefficients (between 0 and 1) are determined by a simple line search algorithm and the scaled conjugate gradient method is again used for optimization. Up to 10 training distributions are used to train MLPs with 200 hidden neurons but no significant reduction on test error has been achieved. Since the algorithm is very expensive to run (takes more than a week), no further efforts are made along this direction.

For generalized RBF networks, the same boosting algorithm will significantly increase computational complexity since modification of the cost function will turn the current

linear optimization problem into a nonlinear one, and the computational efficiency enjoyed by generalized RBFs will be completely lost. Therefore, boosting is not implemented on RBFs. Instead, bagging as described in [24] is tried on a generalized RBF network with 3000 hidden neurons and a spread ratio of 14 (the optimal spread ratio for such an architecture). The reason of choosing 3000 hidden neurons is trying to create a little bit overfit so that the effect of bagging may be more obvious. Bootstrap replicates up to 25 are tried, but it also fails to provide any significant improvement on the test error over baseline systems described in the previous two sections.

4.2.5 Summary

A formal and systematic study on the ATA mapping is carried out in this section. Three neural network architectures (MLP, regularization RBF and generalized RBF) are carefully applied and ensemble learning methods are also attempted to further improve the result. It is observed that the added glottal and nasal features are effective at improving the accuracy of the mapping. However, the final result is still not very satisfactory and this seems to reveal that the difficulty lies in the intrinsic complexity of the mapping rather than the technique used to learn it. It could be caused by the lack of information or inaccuracy of the input data (such as the glottal and nasal features) or the complicated nature of the acoustic feature (MFCCs) chosen. The considerable amount of effort required to apply neural networks for this large scale function approximation problem also suggests that an alternative knowledge-based approach is worth pursuing. Actually, the lessons learned here has strongly motivated the use of analytical mapping from the VTR domain to the acoustic domain in Section 7.1.

Among different neural network architectures, the final accuracies they can reach are quite similar. The mean squared errors on both the raw and appended test set for different architectures are summarized in Table 4.7. However, there are some practical concerns when different network architectures are used. MLPs are very expensive to train but very fast to operate once it is trained. Regularization RBFs are very fast to train (since it essentially remembers all the training examples) but slow and expensive (in terms of memory requirement) to operate. The generalized RBFs seem to strike a balance between time and space complexities in both training and testing, but the gain in training is significantly

Network Architecture	Raw Data	appended Data
MLP with 200 hidden neurons	20.34	16.97
Regularization RBF	20.18	17.38
Generalized RBF with 2000 hidden neurons	20.29	17.12

Table 4.7: Test error (MSE) of different neural network architectures.

reduced due to the tuning of two free parameters instead of one in MLP and regularization RBF networks. For most speech applications where such a mapping can be trained off-line, MLPs seem to be an appropriate choice.

4.3 Modeling the Articulatory Dynamics

Due to the highly complicated articulatory movements during speech production, so far no successful models have been developed in speech science to account for their behavior at a global level. In this section a simple statistical model aiming at this goal is developed and tested.

4.3.1 A Functional Articulatory Dynamic Model

As illustrated in Section 4.1, when an articulator is *active* during the production of a phoneme, it will move towards or reach one or more well-defined positions. These well-defined positions are called targets or goals of the active articulators when producing the phoneme. Most phonemes have a single target for each active articulator, but some phonemes may have more than one target for an active articulator depending on the context. A well-known example is the two tongue dorsum (`TD`) targets for phoneme /k/, such as in words "car" and "can", where one is relatively backward and the other is relatively forward. From a statistical point of view, most targets can be modeled by a unimodal distribution (such as a Gaussian) but some may need a multimodal distribution (such as a mixture of Gaussians) to be modeled accurately; while the position of the inactive articulators can be modeled by a relatively flat distribution.

However, accounting for all these complexities at the same time requires a fairly com-

plicated model. As a first attempt to capture key properties of the articulatory dynamics and their relationship to phoneme production globally, the following simple target-directed statistical linear dynamic equation is proposed:

$$\mathbf{z}(k) = \mathbf{\Phi}_s \mathbf{z}(k-1) + \mathbf{\Psi}_s \mathbf{u}_s + \mathbf{w}_s(k). \tag{4.2}$$

The vector $\mathbf{z}(k)$, as defined in (3.2), represents the positions of measured articulators at time k, and will be referred to as the state vector or simply states from time to time. The phoneme that is currently being realized by the articulators is indexed by s. When phonemes are produced sequentially, the initial state of the current phone is the last state of the previous phone, thus imposing a simple continuity constraint across the whole utterance. The key parameters of the model are the quantities $\mathbf{\Phi}_s$, $\mathbf{\Psi}_s$ and $\mathbf{u}_s$: matrix $\mathbf{\Phi}_s$ encodes the time interaction among the articulatory components; vector $\mathbf{u}_s$ is the target position of the articulators; and matrix $\mathbf{\Psi}_s$ describes the control effect of the targets on the articulatory movement. All three quantities are phone dependent, although for implementation purposes the values $\mathbf{\Phi}_s$ and $\mathbf{\Psi}_s$ may each be tied for broader classes of phones. Due to the well-known forward-anticipatory property of the articulators as illustrated in Section 4.1, the boundaries for these parameters (especially the target $\mathbf{u}_s$) should be in advance of the actual acoustic boundaries. The quantitative degree of anticipation is one additional parameter to be learned from the articulatory data. Finally, $\mathbf{w}_s(k)$ is a phone-dependent white Gaussian noise process with time-invariant covariance matrices $\mathbf{Q}_s$ to account for model inaccuracy.

By definition, (4.2) must satisfy the assumed asymptotic target-directed property when no state noise is present, that is, $\mathbf{z}(k) = \mathbf{z}(k-1) \to \mathbf{u}_s$ as $k \to \infty$ if $\mathbf{w}_s(k) = 0$ at all times[7]. First for $\mathbf{z}(k)$ to asymptotically reach a steady state, matrix $\mathbf{\Phi}_s$ must be *convergent*, *i.e.*, all its eigenvalues must lie within the unit circle. Second if we substitute $\mathbf{z}(k) = \mathbf{z}(k-1) = \mathbf{u}_s$ and $\mathbf{w}_s(k) = 0$ into (4.2), we have

$$\mathbf{\Psi}_s = \mathbf{I} - \mathbf{\Phi}_s, \tag{4.3}$$

which is the constraint between $\mathbf{\Phi}_s$ and $\mathbf{\Psi}_s$. This indicates that $\mathbf{\Psi}_s$ is not a free parameter of the model if the asymptotic target-directed property is to be satisfied. Therefore, (4.2)

[7]When noise is present, the mean of $\mathbf{z}(k)$ should converge to $\mathbf{u}_s$.

can be rewritten as

$$\mathbf{z}(k) = \mathbf{\Phi}_s \mathbf{z}(k-1) + (\mathbf{I} - \mathbf{\Phi}_s)\mathbf{u}_s + \mathbf{w}_s(k), \tag{4.4}$$

which is the form used throughout the remaining of the study.

Notice that under noise-free conditions, if $\mathbf{z}(k)$ becomes a scalar or more generally $\mathbf{\Phi}_s$ is a diagonal matrix, (4.4) can only generate exponential curves asymptotically reaching a constant (the target), which is a very simple type of dynamics. When $\mathbf{\Phi}_s$ is not restricted to be diagonal, slightly more complicated dynamics can be generated through the interactions among components of $\mathbf{z}$, or equivalently, among different articulators. Therefore, the structure of matrix $\mathbf{\Phi}_s$, which represents the interrelationships between $\mathbf{z}(k)$ and $\mathbf{z}(k+1)$ and the interactions among articulators, is of great importance. In particular, we can identify approximate conditional independencies among articulators: for example the movement of the upper lip, related to that of the lower lip, is largely independent of the jaw position. One possible $\mathbf{\Phi}_s$ matrix, determined after exploring all such conditional independence relations, follows:

$$\mathbf{\Phi}_s = \begin{bmatrix} \phi_{11} & \phi_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \phi_{21} & \phi_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{33} & \phi_{34} & \phi_{35} & \phi_{36} & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{43} & \phi_{44} & \phi_{45} & \phi_{46} & 0 & 0 & 0 & 0 \\ \phi_{51} & \phi_{52} & \phi_{53} & \phi_{54} & \phi_{55} & \phi_{56} & 0 & 0 & 0 & 0 \\ \phi_{61} & \phi_{62} & \phi_{63} & \phi_{64} & \phi_{65} & \phi_{66} & 0 & 0 & 0 & 0 \\ \phi_{71} & \phi_{72} & 0 & 0 & 0 & 0 & \phi_{77} & \phi_{78} & \phi_{79} & \phi_{7,10} \\ \phi_{81} & \phi_{82} & 0 & 0 & 0 & 0 & \phi_{87} & \phi_{88} & \phi_{89} & \phi_{8,10} \\ \phi_{91} & \phi_{92} & 0 & 0 & 0 & 0 & \phi_{97} & \phi_{98} & \phi_{99} & \phi_{9,10} \\ \phi_{10,1} & \phi_{10,2} & 0 & 0 & 0 & 0 & \phi_{10,7} & \phi_{10,8} & \phi_{10,9} & \phi_{10,10} \end{bmatrix}, \tag{4.5}$$

and this structure is retained for all phones s. Note that asserting this structure simultaneously reduces the number of parameters and makes the parameter estimation more robust. To facilitate model parameter learning, which will be presented shortly, it is even more desirable for $\mathbf{\Phi}_s$ to be block diagonal. This is approximately achieved through a change of basis,

$$\mathbf{z} = [\mathrm{J}_x, \mathrm{J}_y, \mathrm{UL}_x, \mathrm{UL}_y, \mathrm{LL}_x - \mathrm{J}_x, \mathrm{LL}_y - \mathrm{J}_y, \mathrm{TT}_x - \mathrm{J}_x, \mathrm{TT}_y - \mathrm{J}_y, \mathrm{TD}_x - \mathrm{J}_x, \mathrm{TD}_y - \mathrm{J}_y]^T, \tag{4.6}$$

leading to a revised $\mathbf{\Phi}_s$ matrix

$$\mathbf{\Phi}_s = \begin{bmatrix} \phi_{11} & \phi_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \phi_{21} & \phi_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{33} & \phi_{34} & \phi_{35} & \phi_{36} & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{43} & \phi_{44} & \phi_{45} & \phi_{46} & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{53} & \phi_{54} & \phi_{55} & \phi_{56} & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_{63} & \phi_{64} & \phi_{65} & \phi_{66} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_{77} & \phi_{78} & \phi_{79} & \phi_{7,10} \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_{87} & \phi_{88} & \phi_{89} & \phi_{8,10} \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_{97} & \phi_{98} & \phi_{99} & \phi_{9,10} \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_{10,7} & \phi_{10,8} & \phi_{10,9} & \phi_{10,10} \end{bmatrix}. \tag{4.7}$$

The model presented here is termed *functional* since the underlying physiological mechanism which governs the movement of articulators is not directly described, although clearly the model must reflect key constraints imposed by the physiological system. Rather, a target-oriented, parameterized linear state equation is proposed, where the parameters are learned automatically from real speech data. Such a model is also strongly motivated by a similar target-directed model well-developed in speech science [213]. The difference is that the original model is deterministic, continuous-time and defined in the abstract task space, where the dynamics is supposed to be quite simple; here the highly complicated articulatory movement is dealt with directly and a statistical model is indispensable to account for model inaccuracy. The parameter estimation strategy is presented next.

4.3.2 Model Parameter Learning

First assume that the articulatory phone boundaries, *i.e.*, the boundaries where phone-dependent model parameters switch values, are known. Suppose we have a set of articulatory measurements $\mathbf{Z} = \{\mathbf{z}(0), \mathbf{z}(1), \ldots, \mathbf{z}(K)\}$ belonging to the same phone, and the model parameters to be estimated are $\mathbf{\Theta} = \{\mathbf{\Phi}, \mathbf{u}, \mathbf{Q}\}$[8]. The formulas for estimating these parameters are derived based on the maximum likelihood (ML) criterion.

[8]The dependency of model parameters on phone index s is dropped for notational clarity: the derived results apply to all phones.

Observe that the state equation (4.4) implicitly encodes a Markov property on $\mathbf{z}(k)$, and the total log likelihood of the time series $\mathbf{Z}$ can be expressed as

$$L(\mathbf{Z} \mid \mathbf{\Theta}) \triangleq \log p(\mathbf{Z}) = \log \left\{ \prod_{k=1}^{K} p[\mathbf{z}(k) \mid \mathbf{z}(k-1)] \cdot p[\mathbf{z}(0)] \right\}. \tag{4.8}$$

Assume $\mathbf{z}(0)$ is known and does not contribute to the maximization of L, and define the more compact notation $\mathbf{z}_k \triangleq \mathbf{z}(k)$, we have

$$\begin{aligned} L &= \sum_{k=1}^{K} \left\{ \frac{1}{2} \log \left| \frac{1}{2\pi \mathbf{Q}} \right| - \frac{1}{2} [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}]^T \mathbf{Q}^{-1} [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}] \right\} \\ &= \sum_{k=1}^{K} \left\{ \frac{1}{2} \log \left| \frac{\mathbf{Q}^{-1}}{2\pi} \right| - \frac{1}{2} [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}]^T \mathbf{Q}^{-1} [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}] \right\}. \end{aligned} \tag{4.9}$$

Taking derivatives of L w.r.t. all the parameters,

$$\frac{\partial L}{\partial \mathbf{\Phi}} = \sum_{k=1}^{K} \mathbf{Q}^{-1} [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}](\mathbf{z}_{k-1} - \mathbf{u})^T, \tag{4.10}$$

$$\frac{\partial L}{\partial \mathbf{u}} = \sum_{k=1}^{K} (\mathbf{I} - \mathbf{\Phi})^T \mathbf{Q}^{-1} [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}], \tag{4.11}$$

$$\frac{\partial L}{\partial \mathbf{Q}^{-1}} = \sum_{k=1}^{K} \left\{ \mathbf{Q} - [\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}][\mathbf{z}_k - \mathbf{\Phi}\mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}]^T \right\}. \tag{4.12}$$

ML estimates of the parameters are obtained by setting the above three partial derivatives to zero and solving for the parameters. After some careful algebraic manipulation

(especially for $\mathbf{\Phi}$ and $\mathbf{u}$), the following results can be obtained

$$\mathbf{\Phi} = \left(\frac{1}{K} \sum_{k=1}^{K} \mathbf{z}_k \sum_{k=1}^{K} \mathbf{z}_{k-1}^T - \sum_{k=1}^{K} \mathbf{z}_k \mathbf{z}_{k-1}^T \right) \cdot \left(\frac{1}{K} \sum_{k=1}^{K} \mathbf{z}_{k-1} \sum_{k=1}^{K} \mathbf{z}_{k-1}^T - \sum_{k=1}^{K} \mathbf{z}_{k-1} \mathbf{z}_{k-1}^T \right)^{-1}, \tag{4.13}$$

$$\mathbf{u} = \frac{1}{K} (\mathbf{I} - \mathbf{\Phi})^{-1} \left(\sum_{k=1}^{K} \mathbf{z}_k - \mathbf{\Phi} \sum_{k=1}^{K} \mathbf{z}_{k-1} \right), \tag{4.14}$$

$$\mathbf{Q} = \frac{1}{K} \sum_{k=1}^{K} \left\{ [\mathbf{z}_k - \mathbf{\Phi} \mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}][\mathbf{z}_k - \mathbf{\Phi} \mathbf{z}_{k-1} - (\mathbf{I} - \mathbf{\Phi})\mathbf{u}]^T \right\}. \tag{4.15}$$

A few further notes follow the above derived results:

1. ML parameter estimation typically can only achieve local optimum. However, since the likelihood function L of this simple linear dynamic model has a quadratic form, its local optimum is the same as its global optimum. Therefore, the estimated parameters are also globally optimal for this particular model.

2. The above result applies to estimating a general, unconstrained matrix $\mathbf{\Phi}$. In the block-diagonal case (4.7) the above estimator can be applied separately to each block. Notably, the above estimator does *not* apply to the more interesting constrained case (4.5); such a case forces us to work on elements of $\mathbf{\Phi}$ explicitly and the much more complicated result cannot be expressed in a succinct, closed form. Therefore, the block-diagonally structured $\mathbf{\Phi}$ is studied first.

3. The same linear dynamic model has been proposed and used in related work by previous group members under different contexts [64, 158, 237]. However, the closed form solution as presented in (4.13) and (4.14) was not obtained in those previous publications due to less careful derivations; such a result first appeared in a previous publication of the author [151].

4. The above derivation assumes that there is only one training example for a given phone. In practice multiple training examples (popularly termed *tokens* in the speech community, especially for speech recognition) are available for each phone and the

above formulas need to be slightly modified by adding one more summation over the training tokens. For example, if there are M training tokens for a phone, each represented by a time series of length K_m ($m = 1, \ldots, M$), then the modified parameter estimation formulas for $\mathbf{\Phi}$ and $\mathbf{u}$ are

$$\mathbf{\Phi} = \left[\frac{1}{\sum_{m=1}^{M} K_m} \left(\sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_k \right) \left(\sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_{k-1}^T \right) - \sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_k \mathbf{z}_{k-1}^T \right] \cdot \left[\frac{1}{\sum_{m=1}^{M} K_m} \left(\sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_{k-1} \right) \left(\sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_{k-1}^T \right) - \sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_{k-1} \mathbf{z}_{k-1}^T \right]^{-1}, \quad (4.16)$$

$$\mathbf{u} = \frac{1}{\sum_{m=1}^{M} K_m} (\mathbf{I} - \mathbf{\Phi})^{-1} \left(\sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_k - \mathbf{\Phi} \sum_{m=1}^{M} \sum_{k=1}^{K_m} \mathbf{z}_{k-1} \right). \quad (4.17)$$

The estimated matrix $\mathbf{\Phi}$ also has to satisfy the further constraint of being convergent. This is done by checking the eigenvalues of the estimated matrix $\mathbf{\Phi}$ and project it to the closest convergent matrix if it is not already one. $\mathbf{u}$ and $\mathbf{Q}$ are reestimated by (4.14) and (4.15) using the new value of $\mathbf{\Phi}$ if necessary. The actual projection step of $\mathbf{\Phi}$ is carried out as follows. First do a eigenvalue decomposition of $\mathbf{\Phi}$ and collect its eigenvectors in matrix $\mathbf{V}$ and eigenvalues in a diagonal matrix $\mathbf{D}$, such that

$$\mathbf{D} = \mathbf{V}^{-1} \cdot \mathbf{\Phi} \cdot \mathbf{V}, \quad (4.18)$$

i.e., $\mathbf{D}$ is obtained from $\mathbf{\Phi}$ via a change of basis. Clamp each diagonal elements of $\mathbf{D}$ to have a modulus less than one (the value 0.99 is used in the actual implementation) but keep the original angle. Call this new matrix $\mathbf{D}'$ and the projected $\mathbf{\Phi}'$ is obtained via another change of basis

$$\mathbf{\Phi}' = \mathbf{V} \cdot \mathbf{D}' \cdot \mathbf{V}^{-1}. \quad (4.19)$$

Typically we don't know the *articulatory* phone boundaries where the targets and other parameters switch values, as assumed in deriving the estimation formulas. In many cases the acoustic phone boundaries may be available, such as in the TIMIT database to be described in the next chapter or the two hand-labeled speakers in UW-XRMB, or they can be estimated by the forced-alignment procedure as mentioned in Section 3.3.2. Since each articulatory boundary typically lies within neighboring acoustic boundaries (especially for

tongue, but less so for lips), we can search through all frames within two adjacent acoustic boundaries μ_{p-1} and μ_p for the optimal articulatory boundary ν_p by maximizing the total likelihood of the data given all the model parameters. However, jointly optimizing all the articulatory boundaries under such a constraint is still an intractable problem: basically we have to enumerate all the possible combinations of articulatory boundaries for each phone to achieve the global maximum of the total likelihood. A reasonable approximation is to sequentially adjust one articulatory boundary at a time and choose the optimal one while fixing all others in an utterance. Such a heuristic approximation is adopted in the experiments of this study for simplicity.

Notice that the estimation of articulatory phone boundaries will affect the values of estimated model parameters and vice versa, so iterations between estimating model parameters and phone boundaries need to be performed until the total likelihood converges (or a fixed number of iterations may be preformed in practice). Admittedly, the current heuristic approximation of optimizing articulatory phone boundaries and the associated model parameters iteratively can be quite crude, but more rigorous treatment presents a big challenge. Such a problem is tackled later and rigorous treatment of the more general switching dynamics problem is the central topic of Chapter 9.

4.3.3 Articulatory Trajectory Fitting Experiments

A number of articulatory fitting experiments are performed in this section to see how effective the proposed statistical linear dynamic model is at capturing articulatory dynamics. In all experiments, model parameters (including articulatory phone boundaries) are first learned according to the above described algorithms on some training sentences, then some predicted or fitted articulatory trajectories are generated based on the learned model parameters to compare with measured ones. The fitted trajectories are generated by running the linear dynamic equation (4.4) under a noise-free mode, using the learned model parameters (including the articulatory phone boundaries) and starting with the true initial position of the articulators. Such a fitted trajectory is the same as the time evolution of $E[\mathbf{z}_k]$ given the true initial value $\mathbf{z}_0$. If the model can capture articulatory dynamics perfectly, then the fitted trajectory $E[\mathbf{z}_k]$ will exactly match the measured trajectory and the noise covariance matrix $\mathbf{Q}_s$ will be zero. However, if the model is very poor at capturing

the true dynamics, then $E[\mathbf{z}_k]$ will show little resemblance to the measured trajectory while most variabilities of the articulatory movements will be accounted for by the noise process $\mathbf{s}_k$. In the latter case, time correlation among articulatory positions is totally lost since the noise process is white, and this is highly undesirable.

A trajectory fitting experiment is first performed on a single sentence, where it serves as both the training and test set. The fitted and measured articulatory trajectories, as well as the estimated articulatory boundaries, are plotted in Fig. 4.17. It can be observed that the overall fitting quality is quite good, especially considering that the model, containing about 500 degrees of freedom (parameters), is being used to fit more than 12,000 articulatory data points. The model has the biggest trouble of capturing jaw movement, where two obvious model-data mismatches happen during the production of phones /b/ and /ae/[9]. These mismatches demonstrate the inherent limitations of the proposed simple linear model. Since the jaw moves independently of all other articulators, the trajectories we can get from (4.4) are only target-directed exponential curves, and these exponential curves may be quite far from the true dynamics of the articulators. This is exactly the case in regions of /b/ and /ae/ of the jaw movement, and similar mismatches can be observed in the fitted trajectories of other articulators as well, such as the tongue tip movement during /iy/, although to a much lesser degree. It can also be observed that the fitted trajectory shows jumps at some phone boundaries. These jumps stem from the fact that equation (4.4) only imposes zeroth-order (continuity) constraints on $\mathbf{z}_k$, rather than having constraints on higher order derivatives as well. Nevertheless, the overall quality of this initial fitting seems to suggest that the model is promising at reflecting true dynamics of articulatory movements at a global level during speech production.

The true power of the model needs to be evaluated on a large training and test set, since the model will *only* be useful if it can capture systematic variations of articulatory dynamics under different contexts. Therefore, as the next step, a 50 sentence training set and a 10 sentence test set of one speaker, which are the same sets used in the study of ATA mapping, are used in the experiment here as well. An example of trajectory fitting and estimated articulatory phone boundaries on a training sentence is plotted in Fig 4.18. It is clear that the model fails to capture the true dynamics well under this

[9]Notice the very small range of `Jx` movement: the mismatch is exaggerated by the scale of the plot.

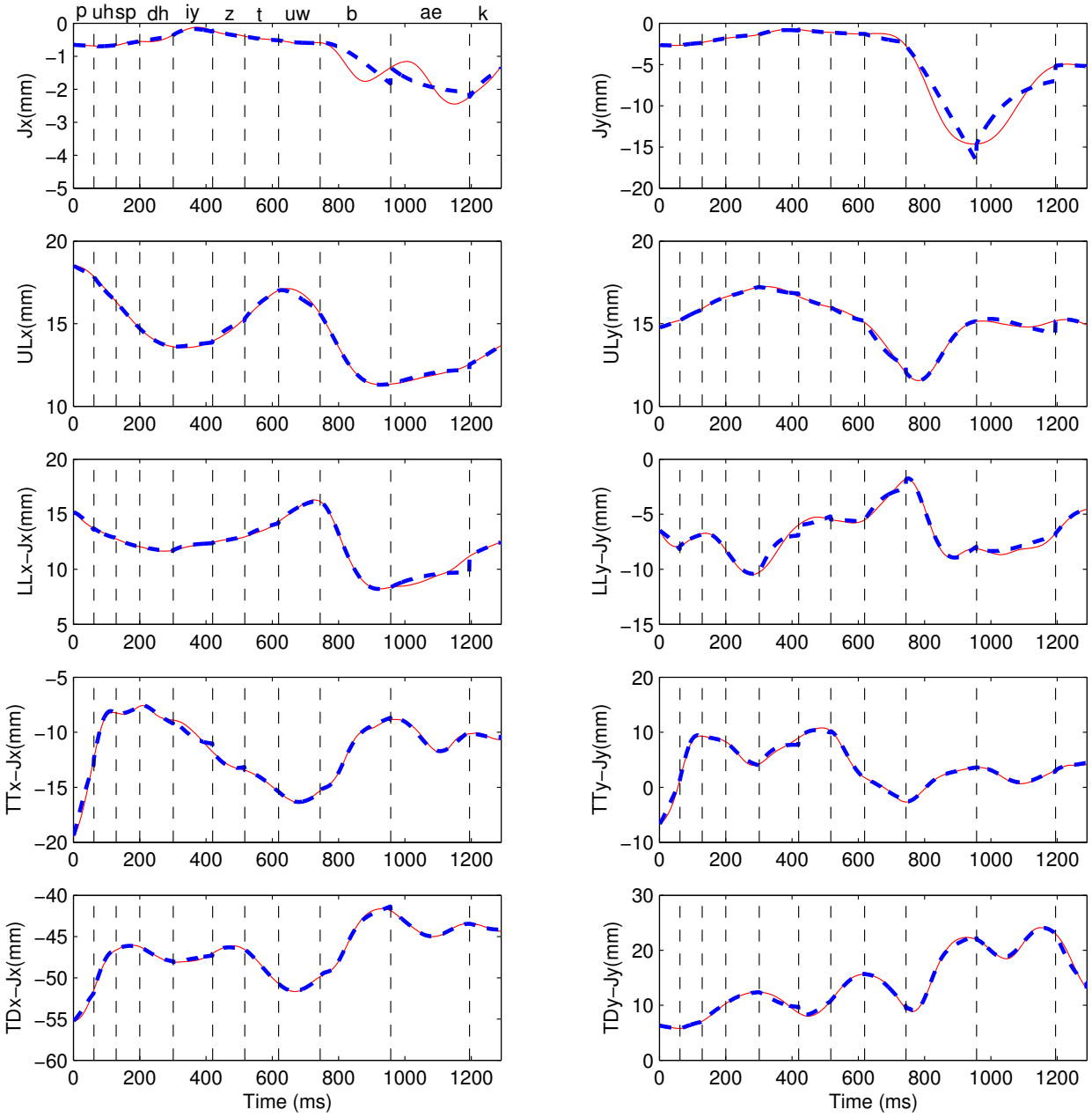

Figure 4.17: Articulatory trajectory fitting for: *Put these two back*. Solid red line: real trajectory; dash blue line: fitted ones.

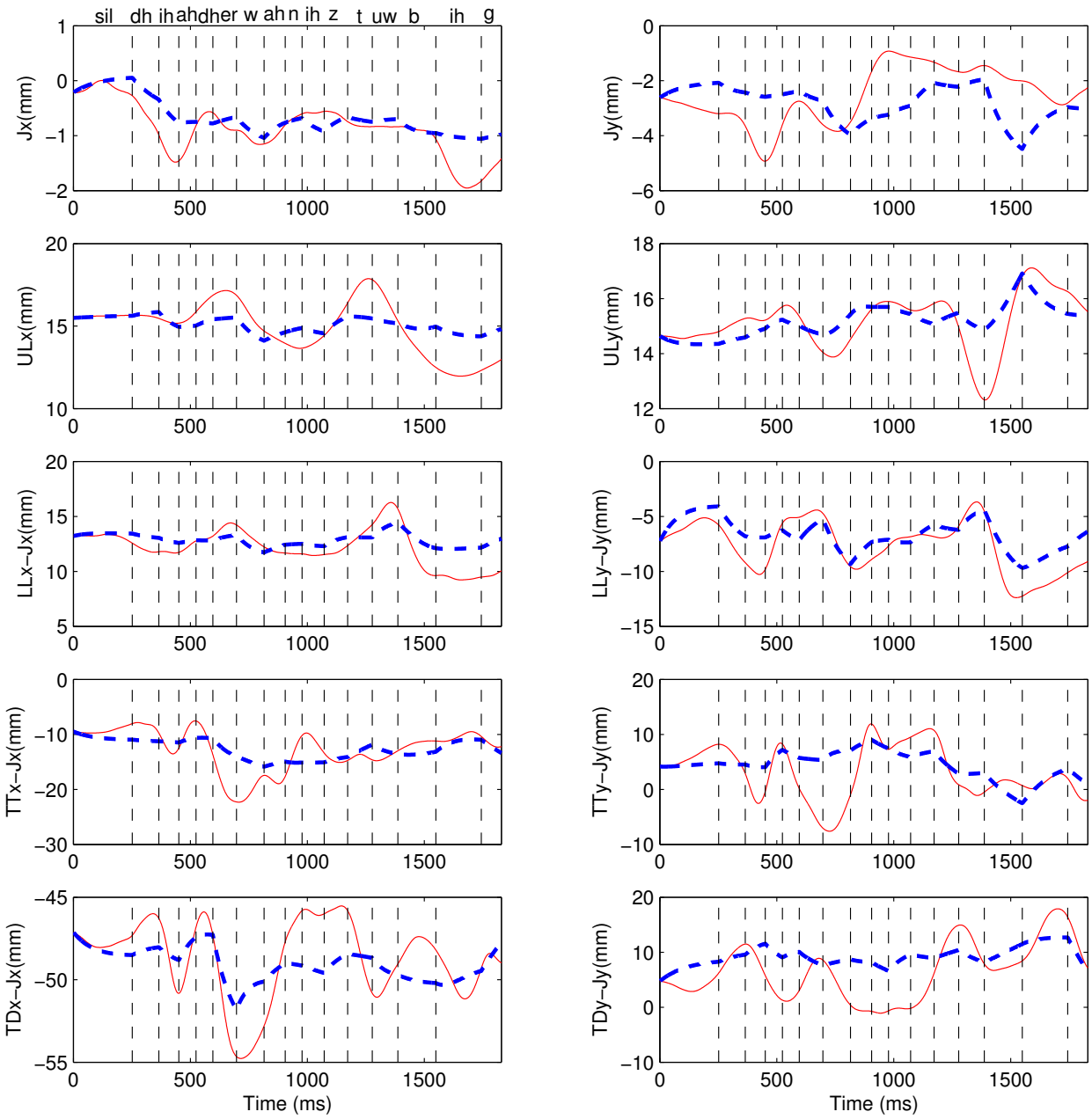

Figure 4.18: Articulatory trajectory fitting for: *The other one is too big.* Solid red line: real trajectory; dash blue line: fitted ones.

more challenging situation. Although at some regions the fitted trajectory shows some tendency of following the real trajectory, most of the time it simply moves around the initial position of the articulators with a very small amplitude comparing to the true trajectory. This is caused by the large variations of articulatory movement when producing the same phoneme under different contexts combined with the inability of this simple first-order linear model to describe the rather complicated articulatory dynamics. As a compromise, $E[\mathbf{z}_k]$ simple moves around some mean positions of the articulators while leaving most of the uncertainties to be handled by the process noise $\mathbf{s}_k$. The performance of articulatory trajectory fitting shown in Fig 4.18 is typical for all training and test sentences, and it demonstrates that our proposed simple linear dynamic model is inadequate in capturing the complicated articulatory dynamics.

4.3.4 Further Observations and Possible Improvements

This section provides some further observations and suggestions on how to model the articulatory dynamics and their relationship with the intended acoustic entities more accurately.

First it is sensible to increase the order of the model so that it can capture more complicated dynamics, such as using second or third-order difference equations instead of the simple first order one used in trajectory fitting examples. But how to choose the order of the model or how many orders are enough? This can be empirically determined by examining the derivatives (with respect to time) of the measured articulatory trajectories. For example, for a first order system, the second derivative of the articulatory trajectory should be zero, or behave like white noise when measurement error is taken into account. This has been carefully examined on the articulatory data and two typical examples, on the measurement of `TTy` and `TDy` of an utterance, are shown in Fig. 4.19 and Fig. 4.20 respectively. It can be observed in both figures that there are fair a number of nonrandom structures up to the third derivative of the trajectories while the fourth derivative acts more or less like white noise. Therefore, a third-order system is necessary to faithfully model the articulatory dynamics.

Second it has been assumed that different articulators share the same articulatory boundaries where model parameters (especially the targets) switch values according to phone identities. However, this is not the case in real speech production. As can be

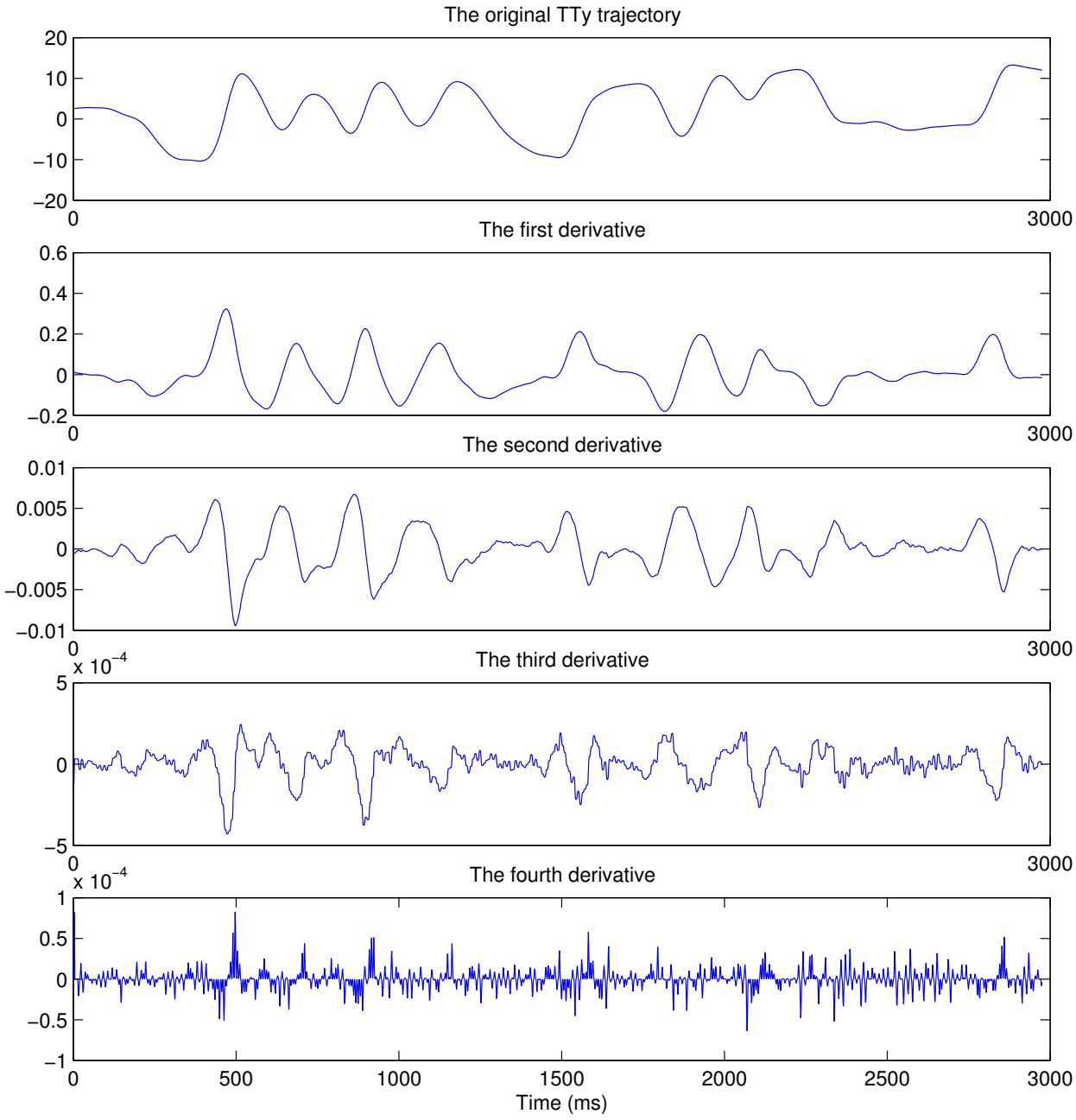

Figure 4.19: The original and derivatives of TTy trajectory for the sentences: *The coat has a blend of both light and dark.*

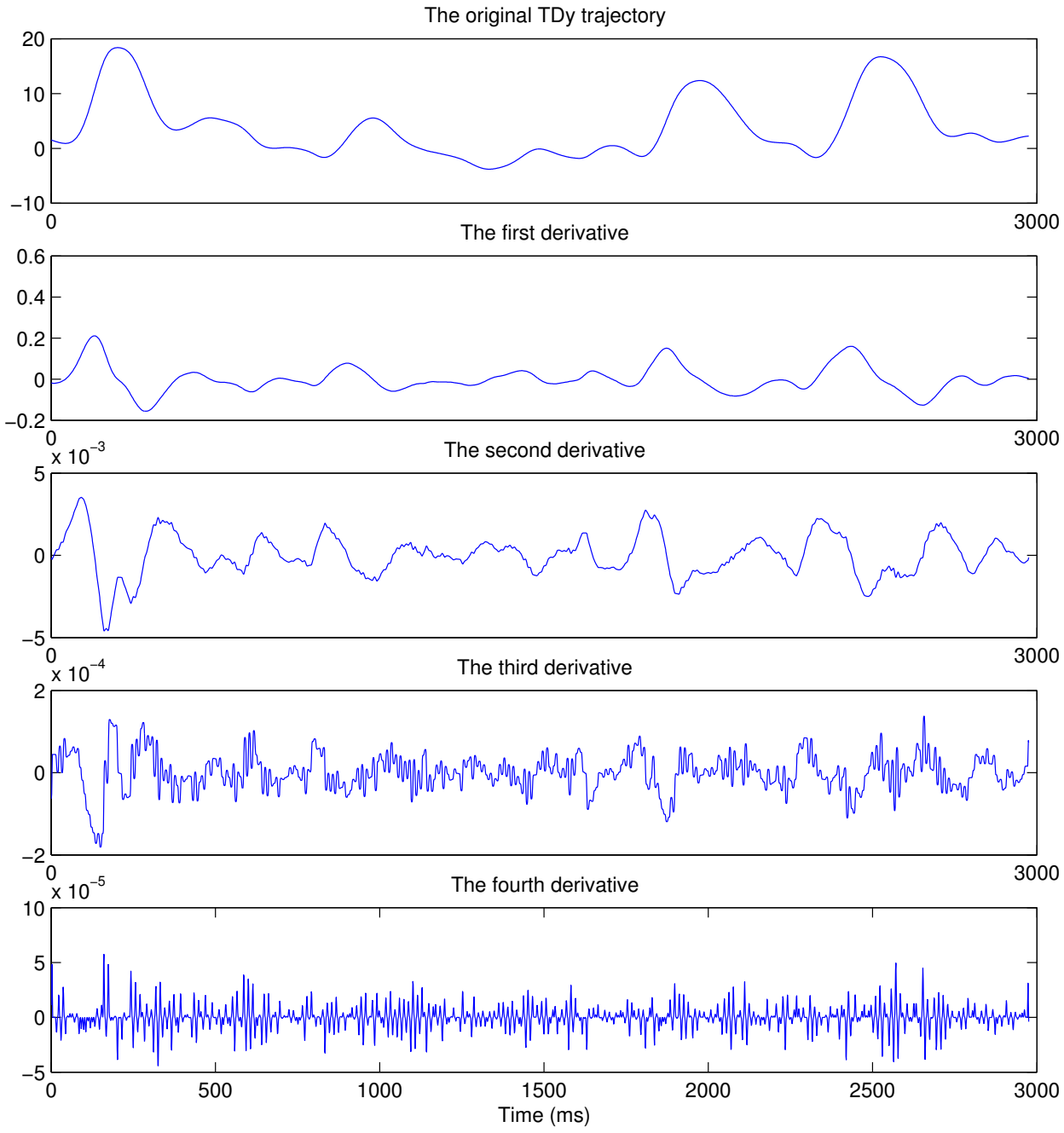

Figure 4.20: The original and derivatives of TDy trajectory for the sentence: *The coat has a blend of both light and dark.*

observed from Fig. 4.17 and 4.18, articulatory dynamics can switch targets asynchronously. These observations are also consistent with past work on overlapping articulatory features [57, 58, 66, 230]. A more accurate model should allow the articulatory boundaries to be learned separately for each articulator, or for groups of closely coupled articulators.

Third, targets of each phoneme should really be modeled as having a multi-variant prior distribution to better account for compensatory articulation, context-dependent targets and the behavior of inactive articulators. Using only a deterministic unknown variable as in the simple linear dynamic model is a big simplification.

Finally, since the true articulatory dynamics are highly involved, some transformation of the original measurements may have simpler dynamics. A good candidate could be the *task dynamics*, which is the dynamics of the vocal tract constriction variables widely used in speech science [27, 28, 29, 213]. However, it may be hard to find such a desired transformation, or for task dynamics, it may be difficult to learn or describe the mapping from a few point measurements of the articulators to vocal tract constriction variables.

All the above suggested improvements either add considerable complexity to the original linear target-directed dynamic model or may be difficult problems themselves. As a result, none of them are seriously pursued in this thesis, and for a good reason: the goal of the thesis is *not* to preform a purely scientific study on speech, *e.g.*, to study the detailed behavior of articulatory movements during speech production, rather it is to derive insights and experience from scientific studies to benefit speech technology. When computational issues for practical applications, especially for speech recognition, are taken into account, even the simple linear dynamic model presents a huge challenge, which will become clear in Part III of the thesis. Therefore, the thesis proceeds by seeking alternative forms of internal dynamics of speech that exhibits simpler dynamic behaviors, which leads to Part II of the thesis. But before that happens, a unified modeling and computational framework is first presented in the next section to demonstrates how to use the articulatory dynamics (as well as other internal dynamics of speech) for practical speech applications.

4.4 An Articulatory Speech Production Model

The previous two sections studied the articulatory dynamics and articulatory-to-acoustic (ATA) mapping separately. This section describes how these two studies can be combined to formulate a novel and powerful model of human speech production towards practical applications.

Mathematically, they are combined via the following state-space model:

$$\mathbf{z}_k = \mathbf{\Phi}_s \mathbf{z}_{k-1} + (\mathbf{I} - \mathbf{\Phi}_s)\mathbf{u}_s + \mathbf{w}_k, \tag{4.20}$$

$$\mathbf{o}_k = \mathcal{H}_s[\mathbf{z}_k] + \mathbf{v}_k. \tag{4.21}$$

The state equation (4.20) is the same as the linear dynamic model used for fitting articulatory trajectories in Section 4.3, and has been described in detail previously. The observation equation (4.21) describes the relationship between observable acoustic features $\mathbf{o}_k$ and the (usually) hidden articulatory movement $\mathbf{z}_k$, where $\mathcal{H}_s$ can be any suitable function characterizing the mapping, such as the neural network architectures used in Section 4.2, and $\mathbf{v}_k$ is a white Gaussian noise process accounting for inaccuracy of the mapping. In general $\mathcal{H}_s$ can be a function depending on the phone index s, although in practical implementations it may be phone-independent or only depend on broad phone classes.

A block diagram of how such a state space model can be used in speech synthesis is shown in Fig. 4.21. A time aligned phoneme sequence is used to drive the state equation to produce realistic articulatory trajectories and to provide extra velum and glottis features (which are not modeled in the state equation but important for generating acoustic signals). Both the articulatory trajectories and the velum and glottis features are fed into the observation equation to generate MFCCs or other acoustic features. Finally the acoustic feature vectors can be converted to speech waveforms as a separate step [86].

The inverse problem of speech synthesis is speech recognition. Conceptually we only have to invert the above process to get the desired phoneme sequence (and subsequently words or sentences), although careful thinking will reveal that the inverse problem is much harder. When such a model is used for speech recognition, it belongs to a family of models termed *hidden dynamic models* (HDMs) [25, 148, 206], which have been proposed recently to overcome inherent limitations of the state-of-the-art HMMs. The name deduces from the fact that these models all describe some form of unobservable internal dynamics of

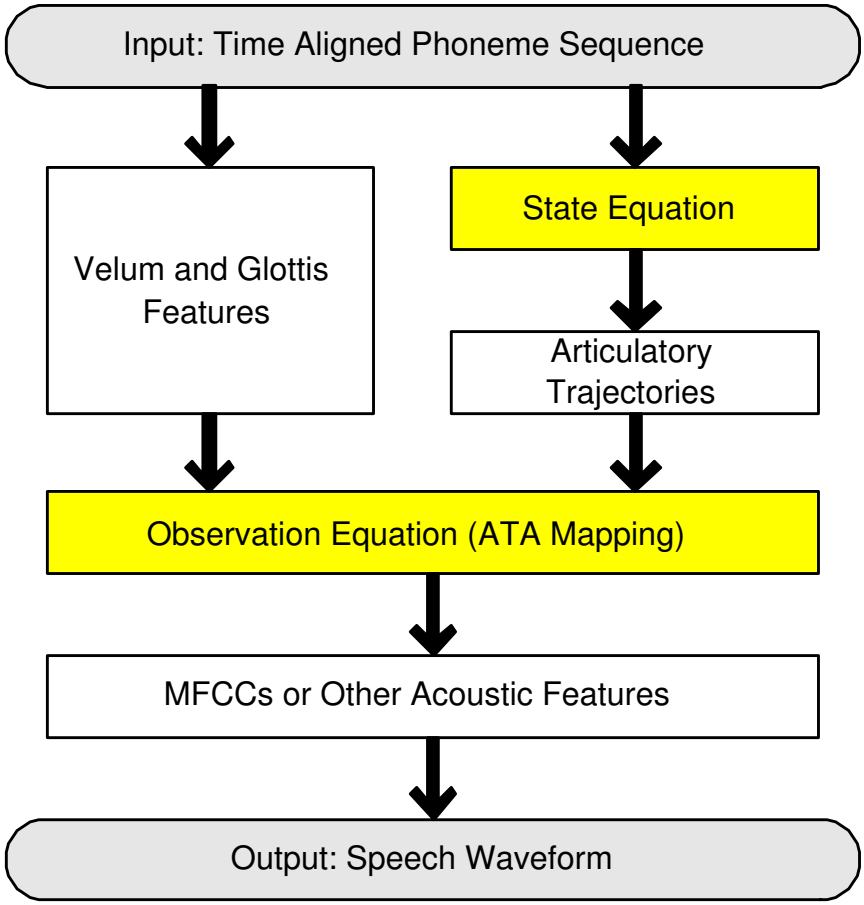

Figure 4.21: Articulatory dynamic model used in speech synthesis.

speech, either physically meaningful or abstract, besides the measurable acoustic features. The topic of speech recognition and detailed descriptions about how such a model can be used in speech recognition will be delayed until Part III of the thesis.

4.5 Concluding Remarks

This chapter has described in-depth data analysis on the UW-XRMB database, more specifically, the ATA mapping and the articulatory dynamics have been studied in detail. Unfortunately, both problems turn out to be very difficult, and the results are not so satisfactory. The ATA mapping approximated by neural networks may still provide enough accuracy for some speech applications, especially for those only need lower order MFCCs. But the articulatory movements exhibit highly complicated dynamic behavior and are not subject to simple modeling. However, lots of insights on speech have been gained through these analyses and by scrutiny of the UW-XRMB data itself, and they are of greater importance than the results themselves that guide and benefit the remaining thesis work. A very important observation that leads to the following chapters of the thesis is that, through numerous spectrogram readings, it is observed that the formant or closely related vocal-tract-resonance (VTR) dynamics seem to exhibit simple phone-dependent, target-directed dynamics and have a close relationship to the underlying shape change of the vocal tract. This naturally leads to the next part of the thesis, where properties of VTR dynamics and its use for speech applications are studied in detail.

Part II

A Study on VTR Dynamics

Chapter 5

Introduction to VTR Dynamics

The concept of vocal-tract-resonance (VTR) and its relationship to formants have been introduced in Section 2.3. From this chapter on, the thesis will focus on using the time evolvement of VTRs as the "hidden dynamics" to characterize the continuous aspect of speech besides its discrete nature.

5.1 Are VTR Dynamics Really Hidden?

In the speech community, it is well accepted that articulatory dynamics or the underlying continuous shape change of the vocal tract is "hidden". Not only can they not be directly observed during normal speech, but they are also very difficult to recover from the acoustic signal alone. The problem of recovering articulatory movements from acoustic signals, known as *acoustic to articulatory inversion*, has attracted many research efforts in both speech science and speech engineering and is a well-known hard and open problem [73, 218]. However, given the close relationship between VTRs and formants, can VTR dynamics be regarded as "hidden" as well?

To clarify this point, let us start with the well-known concept of formants. Formants are usually considered observable during vowel portions of speech. But even under this ideal situation, how well formants can be observed/estimated still critically depends on the fundamental frequency $F0$. Based on the source-filter model of speech, the speech spectrum can be treated as the product of an excitation spectrum (source) and a vocal tract

spectrum (filter). The formants correspond to peak locations of the spectrum envelope, whose amplitude can only be measured at integer multiples or harmonics of $F0$, which can be observed from Fig. 2.2, a narrowband spectrogram example presented earlier in Section 2.3. When $F0$ is high, such as in some child or female speech, it may exceed formant bandwidths and accurate estimation of formant frequencies becomes very difficult or even impossible based on the observed acoustic signal alone. However, if the underlying vocal tract shape is known, there will be no difficulty in the accurate calculation of formant frequencies. Therefore, even in the ideal vowel regions, formants (or equivalently VTRs) are more directly related to the underlying vocal tract shape than the surface acoustic signal.

Unfortunately, the exact definition of formants is still far from being unanimous in the speech community [263], especially for unvoiced sounds. Some researchers define formants entirely in the acoustic domain and equate them with peaks of the power spectrum. Thus it is common to talk about the "onset" of F_2 following a stop release [231] in the phonetics literature, and some papers even call the peak frequency of a fricative spectrum its "formant". A different articulatory based definition is used in formant-based speech synthesis: systems such as *Klattalk* [137, 138] assume that every formant frequency is a continuous function of time, with a defined value even when the corresponding formant resonator is completely decoupled from the output acoustic signal. This is echoed by Stevens who defines formants to be the eigenfrequencies of the airway from glottis to lips, bounded by appropriate termination impedances, regardless of whether or not any given formant is coupled to the acoustic spectrum [225, 224]. In this thesis, the very purpose of using and defining a relatively rare term: vocal tract resonance (VTR), is to avoid the potential confusion about formants used among different speech researchers. The definition of VTR is very similar to the articulatory based definition of formants, but nasal coupling has also been explicitly excluded for simplicity[1]. Of course when nasal coupling is introduced by lowering of the velum, the resonance of the vocal tract blocking nasal tract becomes an approximation, but such a simplification also makes VTR dynamics smooth and continuous across all sound classes, which is a desirable feature that facilitates simple modeling. The widely used convention $F_1, F_2, \ldots, F_N$ and $B_1, B_2, \ldots, B_N$, typically representing formant

[1]Some researchers do so for formants as well, see [224] for example.

frequencies and bandwidths, will also be used to denote VTR frequencies and bandwidths in this thesis.

From the above elaboration, it should be clear that VTR dynamics truly qualify as the "hidden dynamics" of speech since it is directly defined in the hidden articulatory domain. Although it is usually much easier to recover VTR dynamics than articulatory dynamics from acoustic signals, this is still a highly nontrivial problem, demonstrated by the fact that despite of the availability of a large number of VTR (formant) trackers based on different techniques, none of them are really satisfactory so far. More comprehensive review of existing VTR tracking techniques will be provided in the next chapter, where one of two novel VTR tracking techniques developed in this study is presented.

5.2 Hand Labeling of VTRs

Manual tracking by a human expert based on spectrograms or sometimes aided by other signal processing techniques remains to be the most accurate way to track formants as well as VTRs so far. This section first briefly introduces the TIMIT database on which most of the experimental results in the remaining chapters of the thesis are based, then describes a convenient GUI tool to facilitate the hand-labeling of VTR frequencies.

5.2.1 The TIMIT Database

The TIMIT corpus of read speech, distributed by the US National Institute of Standards and Technology (NIST) in 1990, is designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. It contains a total of 6,300 sentences, 10 spoken by each of 630 speakers (438 male and 192 female) from 8 major dialect regions of the United States. Out of the ten sentences spoken by each speaker, two are dialect "shibboleth" sentences read by all speakers[2], five are phonetically-compact sentences selected from a pool of 450 carefully designed ones, and the remaining three are phonetically-diverse sentences picked from a total of 1890 sentences which have appeared in previous speech databases. The utterances

[2]These sentences are designed to expose the dialect of a speaker.

are recorded under broadband (a sampling rate of 16 kHz with 16 bits per sample) and ideal (noise-free) laboratory conditions. The database provides not only word-level and phone-level transcripts but also hand-labeled acoustic phone boundaries for each sentence. For speech recognition tasks, TIMIT has also been divided into standard training and test sets to facilitate comparison among different systems. Since its debut, TIMIT has been widely used in the research community of speech and speaker recognition, and is especially popular in the academic world for testing new ideas due to the rich information provided by the database and the moderate computational requirement to work with the relatively small amount of data.

5.2.2 A VTR Hand-Tracking Tool

A Matlab-based GUI tool has been developed to help manually tracking VTR dynamics on TIMIT speech data. This tool can be easily modified to work on any other speech databases as well. The development is based on the segmentation and analysis tool for the UW-XRMB database and it inherits many convenient features from that tool. A screenshot of the GUI tool is shown in Fig. 5.1, along with the first three labeled VTR frequencies of a TIMIT sentence. The hand-labeled phone boundaries provided by the database are also plotted to provide extra information for manual tracking. This tool works by letting the user decide on a number of key positions of a VTR (by left-mouse clicks) and do a spline interpolation to get its dynamics over the whole utterance. Since the position and number of the labeled points usually need to be adjusted a few times before getting a satisfactory trajectory, the tool provides a number of convenient features to do so, such as deleting an undesirable labeled point by right-clicking on it and moving a labeled point by left-clicking and dragging with the mouse. Comparing to labeling the VTR frequency at each frame (usually 10 ms apart) of the speech signal, this is substantially less work. Slightly modified versions of this tool have also been shared with Dr. Roberto Togneri's group at the University of Western Australia and researchers at Microsoft Research. This tool helps to prepare some limited yet accurate data for the initial study and modeling of VTR dynamics.

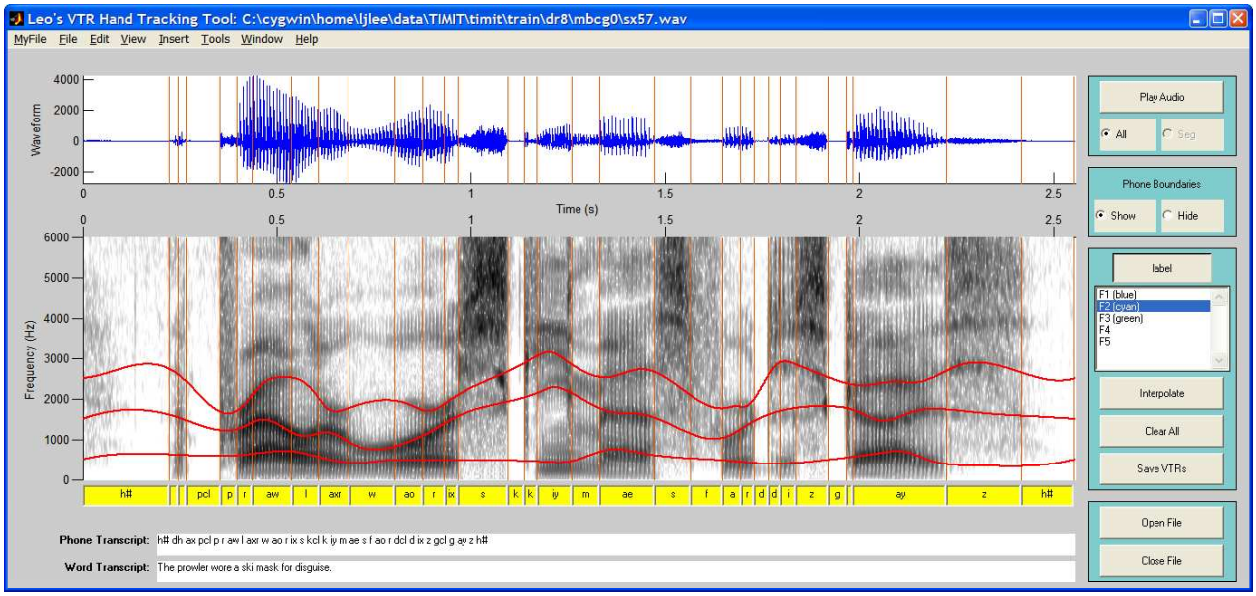

Figure 5.1: A GUI VTR hand tracking tool.

5.3 Modeling VTR Dynamics

This section models the VTR dynamics by target-directed statistical dynamic equations, and will confirm that the time evolution of VTR frequencies indeed follow simple dynamics, unlike those of the articulators studied previously.

Properties of VTRs have been well studied both in speech science and speech engineering [178, 224]. Especially well-known are the target values of VTR frequencies during steady vowel productions [190], where different vowels (in English and also in many other languages) can be distinguished based on target values of the first three VTR frequencies. VTR targets of consonants are less well-known, but have also been extensively studied in formant-based speech synthesizers. Therefore, unlike in the study of articulatory dynamics where target values of the articulators have to be learned from data, target values of VTR frequencies and bandwidths can be obtained from reliable prior knowledge, *i.e.*, by the large number of previous studies as well as personal spectrogram reading experiences. When it is desirable to learn the targets from data, *e.g.*, to better fit the characteristics of a particular speaker, the values based on prior knowledge can serve as good initial points

for parameter estimation algorithms. For simplicity, these knowledge-based (untrained) target values are used in this section, and they are listed in Table 5.1 and Table 5.2. These two tables will also be used in various later parts of the thesis.

The listed targets are mean values of adult male speakers, and are mostly adapted from Klatt's formant-based speech synthesizer [4], with slight modifications based on experiences of spectrogram reading on the TIMIT database. Notice that there are two classes of phonemes based on the property of their VTR targets: one is context-independent and the other is context-dependent. Targets of phonemes belonging to the context-independent class don't change values according to surrounding phones, while targets of those belonging to the context-dependent class are conditioned on whether the following phone is a front vowel, *i.e.*, one of { /ae/, /eh/, /ih/, /iy/, /y/}. For example, VTR frequencies of phoneme /b/ will assume the targets of $[180, 1800, 2300]$ if the following phone is a front vowel, but the targets of $[180, 1100, 2300]$ if the following phone is not. This is essentially caused by the anticipatory tongue positions associated with the following phone. The CMU 39 phone table is adopted in this thesis, and the phonemes listed in Tables 5.1 and 5.2 (44 in total) cover most of them, with two noticeable exceptions:

- Diphthongs (including /aw/, /ay/, /ey/, /ow/ and /oy/) are not listed in the table since they are essentially dynamic vowels that assume two targets sequentially. For example, the realization of /ay/ can be decomposed as /aa/ $\rightarrow$ /iy/.

- The phoneme /hh/ is not listed since the main feature of this phone is aspiration at the glottis while it has no fixed targets for VTR frequencies. It simply takes the targets of the following phone as its own. Same is true when short pauses (/sp/) appear in an utterance.

Next we present and test a number of ideas on modeling the time evolvement of VTR frequencies, which are closely related to the underlying articulatory movements. VTR bandwidths, on the other hand, are not directly related to the continuous shape change of the vocal tract and don't possess true "dynamics" to be modeled. The inclusion of VTR bandwidths in this study is merely to improve the quality of VTR-to-acoustics (VTA) mapping to be discussed in Chapter 7.

Phoneme	F_1 (Hz)	F_2 (Hz)	F_3 (Hz)	B_1 (Hz)	B_1 (Hz)	B_1 (Hz)
aa	730	1090	2440	130	70	160
ae	660	1720	2410	70	130	300
ah	640	1190	2390	80	50	140
ao	570	840	2410	90	100	80
ax	500	1500	2500	80	50	140
ch	300	1700	2400	200	110	270
d	180	1800	2700	70	115	180
dh	250	1300	2500	60	95	185
eh	530	1840	2480	60	90	200
el	450	1000	2700	65	60	80
en	500	1500	2500	120	70	110
er	490	1350	1690	100	60	110
ih	390	1990	2550	50	100	140
iy	270	2290	3010	50	200	400
jh	300	1700	2400	70	110	280
l	450	1060	2640	50	100	280
n	250	1800	2700	40	300	260
r	460	1240	1720	70	100	120
s	250	1900	2700	200	95	220
sh	250	1700	2500	200	110	280
t	180	1800	2700	300	180	220
th	250	1300	2500	225	95	200
uh	440	1020	2240	80	100	80
uw	300	870	2240	65	110	140
w	350	770	2340	50	80	60
y	360	2270	2920	40	250	500
z	250	1900	2700	70	85	190
zh	250	1900	2500	220	140	250

Table 5.1: Context-independent target values for the first three VTR frequencies and bandwidths.

Phoneme	F_1 (Hz)	F_2 (Hz)	F_3 (Hz)	B_1 (Hz)	B_1 (Hz)	B_1 (Hz)
b	180	1100	2300	65	90	125
b_f	180	1800	2300	65	150	125
g	180	1500	2200	70	145	190
g_f	180	2200	2800	70	185	190
k	180	1500	2200	280	220	250
k_f	180	2200	2800	280	220	250
p	180	1100	2300	300	190	185
p_f	180	1800	2300	300	220	185
f	250	1100	2300	225	120	175
f_f	250	1800	2300	225	150	175
m	250	1100	2300	40	175	120
m_f	250	1800	2300	40	200	120
ng	250	1500	2200	160	150	100
ng_f	250	2200	2800	160	170	100
v	250	1100	2300	55	95	125
v_f	250	1800	2300	55	100	125

Table 5.2: Context-dependent target values for the first three VTR frequencies and bandwidths. Which one of the two listed target values of the same phoneme will be taken depends on whether the following phone is a front vowel, where the subscript f denotes when it is followed by a front vowel. Front vowels include { /ae/, /eh/, /ih/, /iy/, /y/}.

We again start with the first-order linear dynamic equation (4.4), but with some further simplifications to facilitate the development of more delicate yet powerful models to be presented shortly. Recall that matrix $\mathbf{\Phi}_s$ encodes the interaction among different articulators in (4.4), and now it will represent the inter-dependency among VTR frequencies. However, we make the simple assertion that $\mathbf{\Phi}_s$ is a diagonal matrix as a first step, so that VTR frequencies are decoupled and each one can be modeled separately by the following scalar equation[3]:

$$z_k = \phi_s z_{k-1} + (1 - \phi_s)u_s + w_k. \tag{5.1}$$

Furthermore, only the dynamics of F_2 are studied as an example since it is the most representative: its value changes the most during speech production thus is the most difficult one to be modeled accurately.

The quality of the model is again judged by trajectory fitting experiments, where the fitted or predicted trajectory refers to one generated by running (5.1) at a noise-free mode given the true initial point, or equivalently $E[z_k]$ given the correct initial value. Fig. 5.2 (a) shows applying equation (5.1) to fit the dynamics of F_2 on a typical TIMIT sentence, where both predicted (dashed blue line) and hand-labeled (solid red line) F_2 trajectories are plotted on top of the spectrogram. The predicted F_2 trajectory is generated by using untrained target values taken from Tables 5.1 and 5.2, phone transcript and boundaries provided by the TIMIT database, and a tied time constant $\phi_s = 0.85$ for all phones. Considering the fact that none of the model parameters are directly trained on the sentence (only a tied ϕ is slightly adjusted), the figure shows a good fit, in stark contrast with the unsuccessful modeling of articulatory dynamics in Fig. 4.18. There are a number of obvious mismatches though, such as those between 2 and 2.5 seconds where the predicted F_2 trajectory seems to "lag behind" of the hand-labeled ones. This is mainly due to the forward-anticipatory property of VTR dynamics, which is linked to the forward-anticipation of the underlying articulatory movements. To counter this anticipation effect, the original phone boundaries are moved 50% forward so that F_2 will move towards its next phone target before the

[3]Notice the change of notations: this thesis typically uses upper case bold letters to represent matrices, lower case bold letters to represent vectors and normal letters (mostly lower case, but will be upper case occasionally) to represent scalars; there will be no distinctions between a deterministic variable and a random variable, but it should be clear from the context.

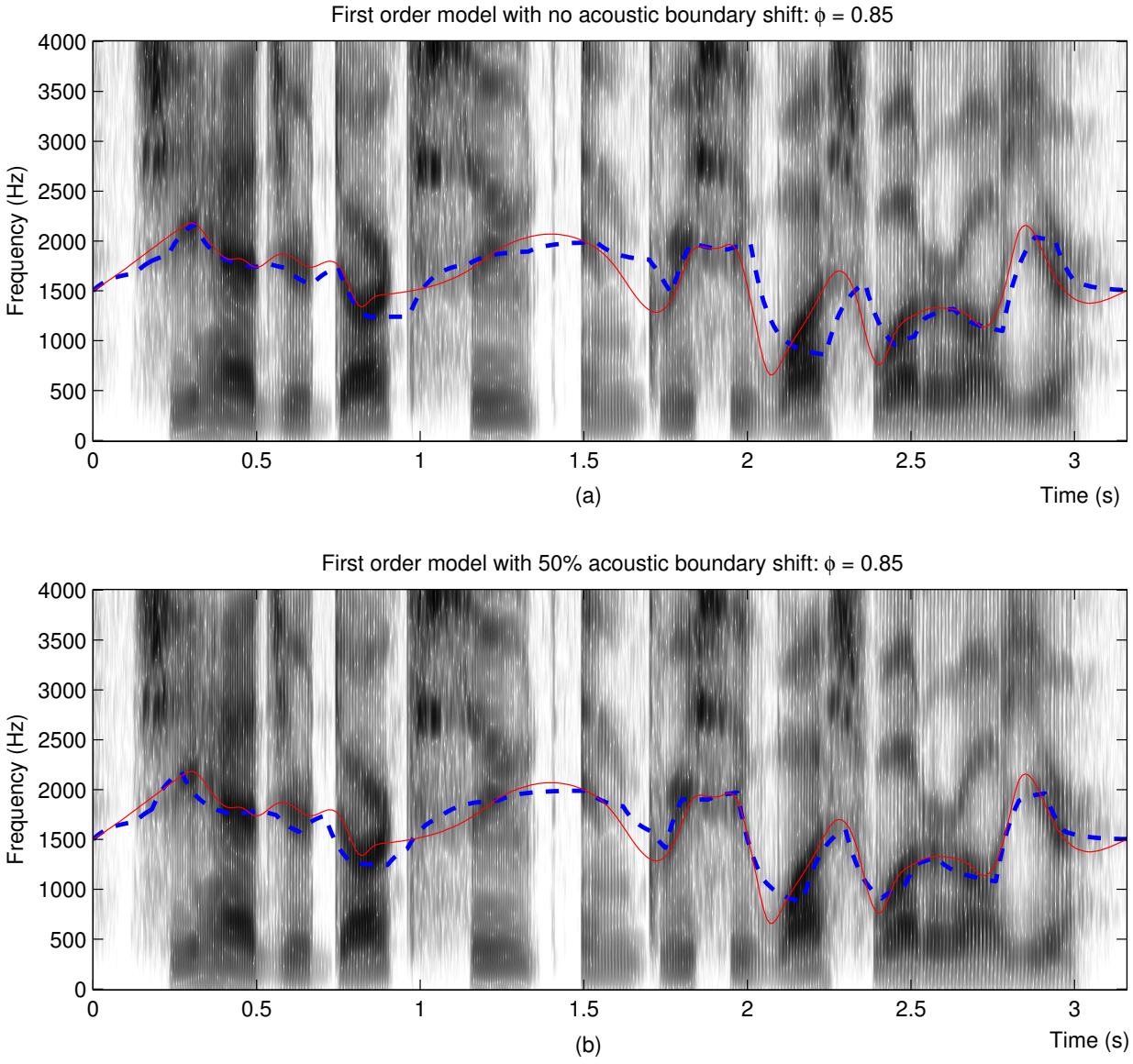

Figure 5.2: Fitting VTR dynamics (F_2) on a TIMIT sentence "*She had your dark suit in greasy wash water all year*" by first order models with original (provided by the database) (a) and 50% forward shifted (b) acoustic phone boundaries. Dashed blue line: fitted F_2; solid red line: manually tracked F_2.

acoustic realization of that phone, and the result is shown in Fig. 5.2 (b), which gives a even better fit. The successful fitting of this example supports the following two points:

1. The targets obtained from prior knowledge are quite accurate and it can truly reflect the trend of VTR movements in real speech.

2. The movement of VTR frequencies exhibit simple dynamics and it can be captured by a first-order linear dynamic system reasonable well.

As an attempt to further improve the fitting, a number of models with increasing complexity are developed. First it can be observed that due to the first-order nature of model (5.1), VTR dynamics at phone boundaries are not smooth. A straightforward way to improve the smoothness is to use a second order model, with the following form

$$z_k = 2\gamma_s z_{k-1} + \gamma_s^2 z_{k-2} + (1-\gamma_s)^2 u_s + w_k, \tag{5.2}$$

which is a discrete-time second-order critically-damped system described in standard signal processing textbooks [177]. It has the property of reaching the target u_s with time constant γ_s without overshooting. The time constant γ_s ($0 < \gamma_s < 1$) plays the same role as ϕ_s in the first order system, where it controls how fast z moves towards its target: the closer γ_s (as well as ϕ_s) is to zero, the faster z_k can change over time.

Secondly it is desirable to parameterize the forward-anticipation behavior of VTR dynamics instead of heuristically moving the acoustic phone boundaries forward. One possible way to parameterize is to have a time-varying target combining the effect of the current target and the next target. Suppose the current phone has a target of u_s and starts at time l_s, while the next phone has target $u_{s'}$ and starts at $l_{s'}$, the effective target u_k at time k ($l_s < k \leq l_{s'}$) could be

$$u_k = u_s + \left(\frac{k-l_s}{l_{s'}-l_s}\right)^{\zeta_s} (u_{s'} - u_s), \tag{5.3}$$

where ζ_s is the anticipation parameter. When $\zeta_s = 0$, $u_k = u_{s'}$ which corresponds to the case of total anticipation; when $\zeta_s \to \infty$, $u_k = u_S$ and there is no anticipation[4]. This time-varying effective target may be incorporated into both the first-order and second-order

[4]The value of ζ_s may seem counter-intuitive; alternatively, we can define $\zeta_s' = \frac{1}{\zeta_s}$.

linear dynamic equations, resulting in the following two new models

$$z_k = \phi_s z_{k-1} + (1 - \phi_s)u_k + w_k, \tag{5.4}$$

$$z_k = 2\gamma_s zk - 1 + \gamma_s^2 z_{k-2} + (1 - \gamma_s)^2 u_k + w_k, \tag{5.5}$$

where u_k is calculated according to (5.3) at each k. Notice that the time-varying targets not only capture forward-anticipation but also at the same time improve the smoothness of the predicted trajectories.

All these models are fitted on the same TIMIT sentence and the results are shown in Fig. 5.3. For simplicity the time constants (ϕ_s and γ_s) and the anticipation parameter ζ_s are tied across all phones, and the actual values for each model are listed in the figure. Phone boundaries of the first and second-order models without anticipatory targets are shifted forward by 50%, while original boundaries are kept for those with anticipatory targets. It can be observed that all four models achieve excellent fit: most of the disagreements with the hand-labeled F_2 are within the limit of a bandwidth, and large discrepancies only occur at consonant regions where the uncertainty of hand-labeling F_2 is also high. Looking at the fitted trajectories carefully, the second-order model does provide a little smoother VTR trajectory than the first-order one, with more smoothness obtained by using anticipatory targets. As expected, the anticipatory targets also take care of the forward-anticipation automatically without artificial boundary shifting. Arguably the most complicated model, *i.e.*, the second-order model with anticipatory targets as described in (5.5), achieves the best fit, but the improvement over other models, even the simple first-order model, is quite small. As a reference, the average absolute differences between the fitted F_2 trajectories and the hand-labeled one are also listed in Table 5.3.

To further demonstrate the power of the proposed models, VTR fitting on a fast speaking sentence is performed[5], and the results are plotted in Fig. 5.4. It can be observed that good fits are once again achieved by all four models: the predicted F_2 trajectories have high agreement with hand-labeled ones in most areas while large discrepancies only occur in consonant regions. Notice that only a single parameter of each model, *i.e.*, the time constant (ϕ or γ) which controls how fast a VTR frequency can reach its target, is

[5]This sentence is not in the TIMIT database but is from private data collected at Microsoft; notice that its duration is about the same as the TIMIT sentence but has many more words spoken.

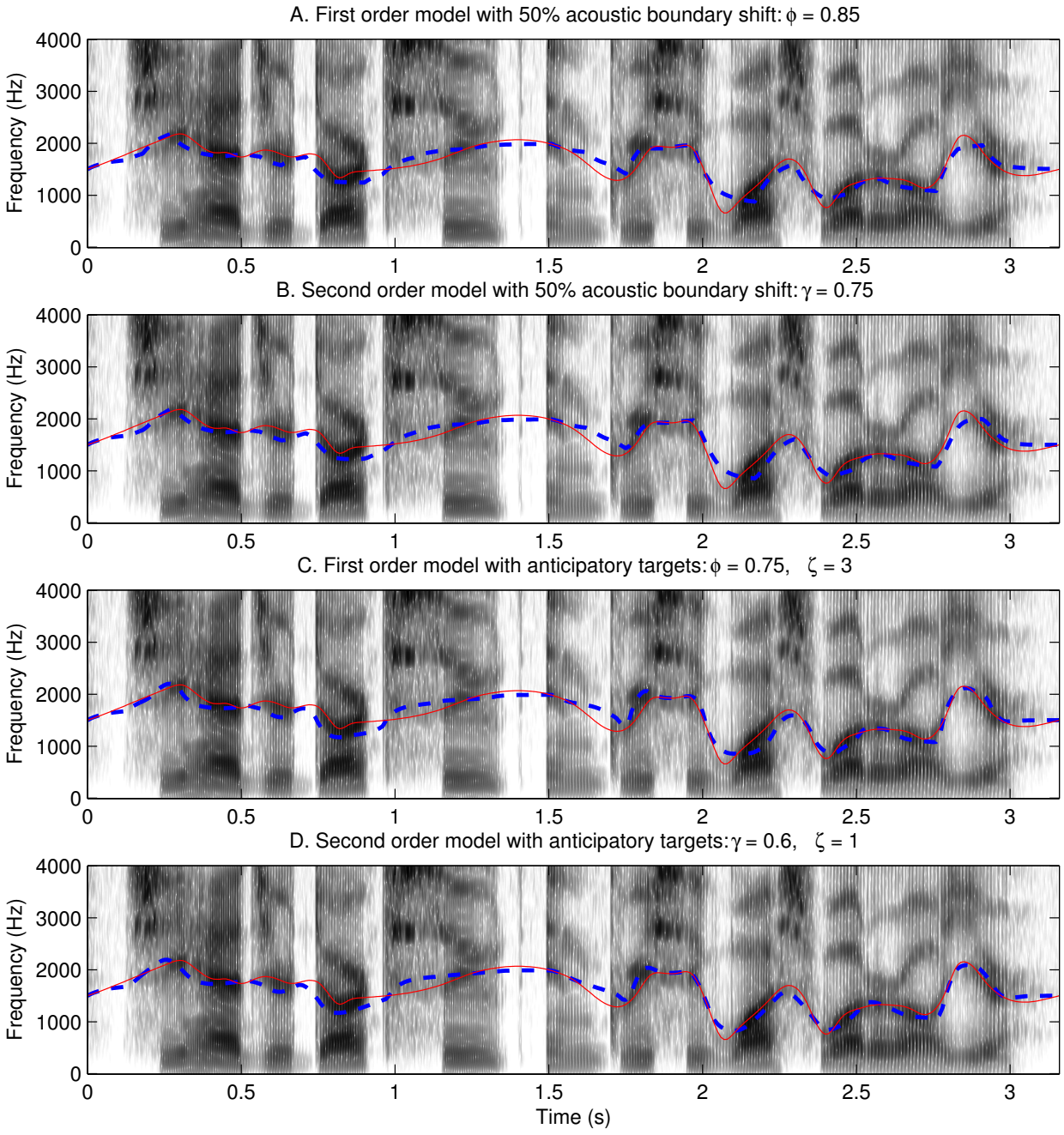

Figure 5.3: Fitting VTR dynamics (F_2) on a TIMIT sentence "*She had your dark suit in greasy wash water all year*". Dashed blue line: fitted F_2; solid red line: manually tracked F_2.

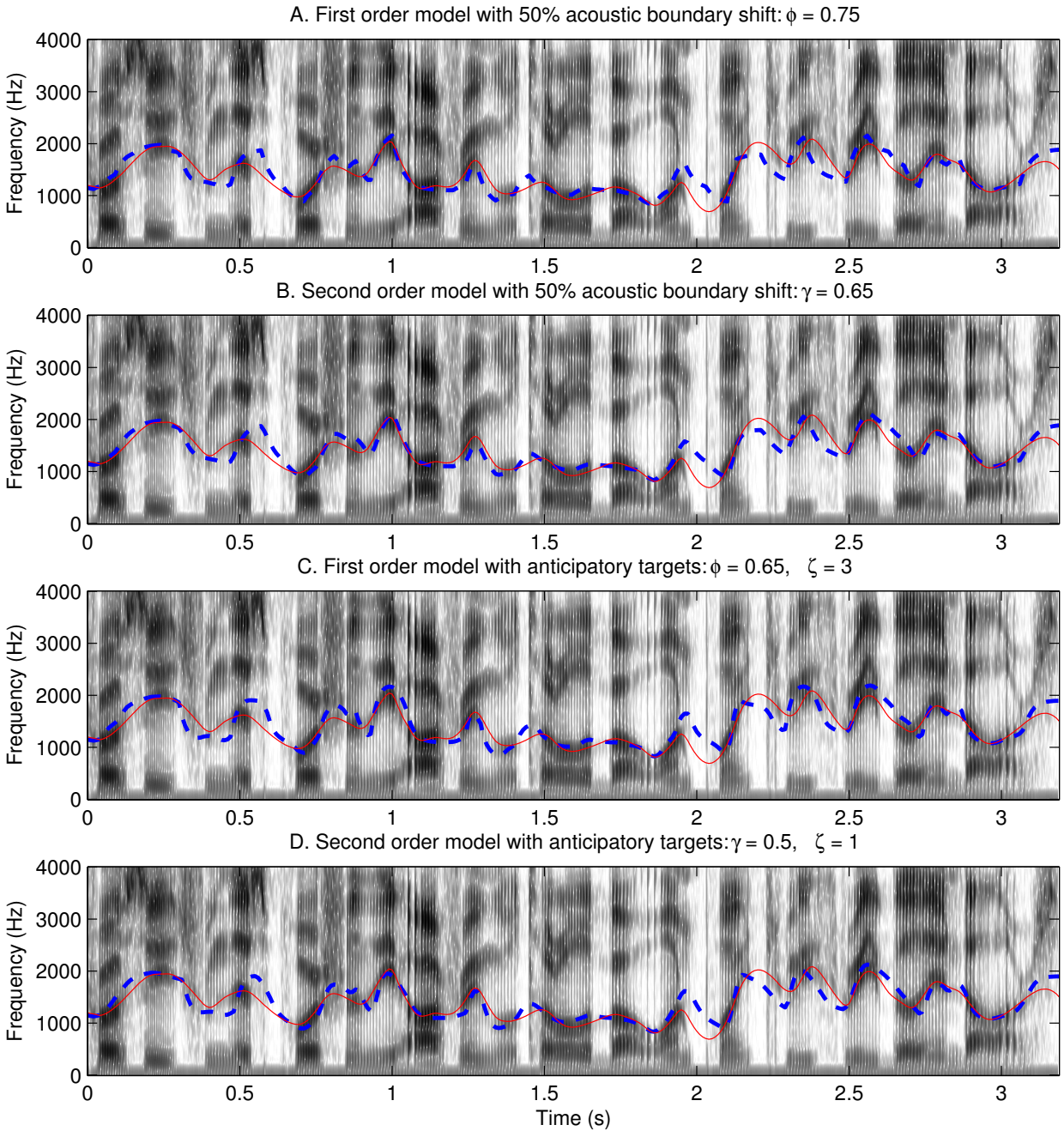

Figure 5.4: Fitting VTR dynamics (F_2) on a fast speaking sentence "*Proceeds from this portion of the offering will total about one point three three billion dollars*". Dashed blue line: fitted F_2; solid red line: manually tracked F_2.

Model	A	B	C	D
Normal Speech (Fig. 5.3)	109	114	112	104
Fast Speech (Fig. 5.4)	132	123	143	131

Table 5.3: Average absolute differences (in Hz) between hand-labeled and fitted F2 trajectories: the different models are defined in Fig. 5.3 and 5.4.

slightly adjusted to account for the much faster speaking rate. This example demonstrates the great potential VTR dynamic models hold in describing different styles/rates of speech under a unified modeling framework. This is one area that the current speech technology, especially speech recognizers, is having great trouble with while models incorporating VTR dynamics may excel[6]. Careful scrutiny may reveal that the fitting quality is slightly worse than that of the previous normal speaking-rate example, but since fast, casual speech is known to be much harder to model than formal, read speech, the result achieved here is quite impressive.

It can also be observed that the improvement on fitting quality provided by more complicated models is very minor, as in the previous normal-rate speaking example, judging from the visual inspection as well as the fitting errors listed in Table 5.3. Overall, the simple first-order linear dynamic model seems to be sufficient in capturing key characteristics of VTR dynamics. As an early effort to incorporate VTR dynamics in speech modeling, the remaining chapters of this thesis will concentrate on the simple first-order model of (5.1), which simplifies model evaluation and algorithm development comparing to other more complicated models proposed in this section.

[6]Again, this topic will be revisited and elaborated in Part III of the thesis.

Chapter 6

VTR Tracking by Active Image Contours

To successfully apply VTR dynamics in machine speech processing, they have to be automatically extracted from acoustic signals somehow, either directly or indirectly. The problem of VTR (or formant) tracking is not new; on the contrary, it has been an important topic for speech processing since the early days of speech research and many different techniques exist [1, 5, 61, 144, 219, 257, 264]. However, it has remained to be a difficult problem and none of the automatic methods give satisfactory results under all conditions. This and the next chapter present two novel VTR tracking methods that suit our purpose of obtaining smooth VTR trajectories across the whole utterance. Both methods extract VTR dynamics directly from acoustic signals, without any other extra information (such as word or phone transcripts). The first method, to be described in this chapter, is to mimic the way human experts track VTR dynamics. It treats VTR tracking completely as an image processing problem based on the spectrogram, with prior speech knowledge properly incorporated during the tracking. The second method is based on setting up a state-space model for the hidden VTR dynamics and will be described in the next chapter.

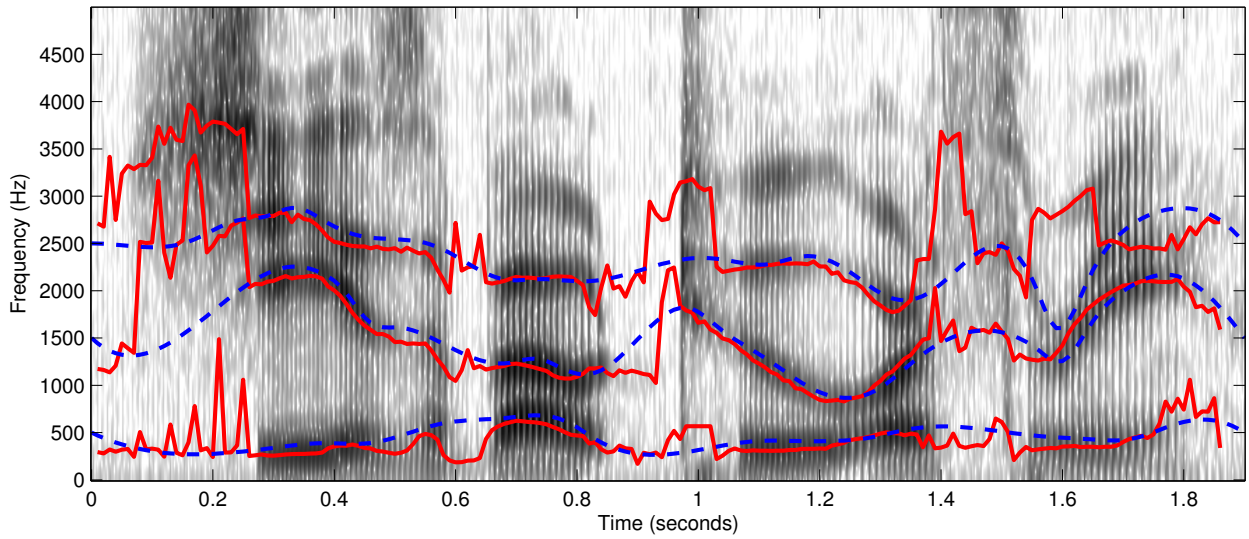

Figure 6.1: VTR tracking of a UW-XRMB sentence: *She is about two or three*. Solid red lines: VTR tracking by *waves+*; dash blue lines: manually tracked VTR.

6.1 Existing VTR Tracking Methods

Basic VTR tracking methods are based on LPC analysis [5, 163, 219, 223]. One way of obtaining VTR candidates at a frame level is to solve for the complex roots of a N^{th} order LPC polynomial (real roots are discarded), and each root can be expressed as

$$z_i = exp(-\pi B_i + j2\pi F_i), \tag{6.1}$$

where F_i and B_i are the VTR frequency and bandwidth associated with the i^{th} root. Another common method consists of finding the peaks on a smoothed spectrum at each frame, where the smoothed spectrum is also usually obtained through an LPC analysis. These frame-level VTR candidates are then heuristically adjusted, such as imposing some kind of continuity constraint and discarding roots with large bandwidths (which indicates a weak spectrum peak), before generating the VTR trajectories of the whole utterance. Fig. 6.1 shows tracked VTR frequencies (solid red lines) of a typical sentence based on the traditional LPC analysis methods, which are computed by *waves+*, a popular commercial speech processing software. As a reference, manually tracked VTR trajectories for the

same sentence are also plotted (dash blue lines). In spite of the fact that VTR frequencies at vowel regions (where they are visible as dark bands in the spectrogram) are generally well tracked, two serious problems can be observed from this example.

1. Estimated VTR trajectories at unvoiced regions are highly rugged and inaccurate. This is not surprising since VTR frequencies are largely decoupled from the acoustic output in these regions. The hidden VTR frequencies can only be inferred by carefully examining the VTR dynamics at neighboring vowel regions and performing a smooth interpolation/extrapolation (or maybe obtained with a higher accuracy if the underlying phone identity is known).

2. When two VTR frequencies are very close to each other (or merge together), e.g., around 1.6 sec in the sample sentence, they can be mistakenly regarded as one. Therefore, the automatic VTR tracker will track a wrong VTR (F_4 as F_3 in this example) at least for a short period.

Many improvements and alternative techniques have been developed to overcome the limitations of traditional LPC-based methods [30, 101, 201, 257], usually to fit a specific purpose. A common theme is to impose stronger global constraints, such as by using a extended Kalman filter [207], a hidden Markov model (HMM) [1, 141] or even extra phonetic information [152]. This chapter looks at the VTR tracking problem from a different angle, *i.e.*, by treating it as an image processing problem as human spectrogram readers would do. Methods appeared in the literature with a spirit closest to the current approach is the work of Laprie and Berger [144], where image processing techniques have been used as a regularization step after VTRs are tracked by a standard LPC based method[1]. A VTR tracking problem so formulated is also a very interesting image processing problem in its own right.

[1]Unfortunately their work was not noticed by the author when the current method was first developed.

6.2 Problem Formulation and Algorithm Development

From an image processing point of view, the task of VTR tracking is simply to detect image features of interest subject to certain constraints imposed by prior knowledge of the problem. *Snakes*, also known as *active contours*, which is introduced by Kass *et al.* [128] as energy minimizing deformable curves guided by external and internal forces, fit this purpose well.

The detection of basic image features, such as lines and edges, has been the subject of research for image processing since its early days [95]. Snakes also serve the same purpose but differ from many low-level methods in one important aspect: instead of completely relying on the image itself to provide information about interesting features, prior knowledge is directly built into snakes from the very beginning of problem formulation. A careful review of the existing VTR tracking methods convince us that the incorporation of prior knowledge is also crucial to realize accurate and robust VTR tracking, which is why snakes or active contours are applied to solve this problem. Snakes have enjoyed wide acceptance and great successes in many difficult image processing problems since its debut [21, 20, 43, 87], and is especially popular in biomedical image processing applications [74, 200]. As a first attempt to solve the VTR tracking problem by image processing methods, the algorithm developed in this section follows closely to the original idea of Kass *et al.*, which is easy to understand even for people with little background in image processing.

The first step of using snakes to solve a practical problem is to establish the energy function of snakes. The energy function of a snake usually consists of two parts:

$$E_{snake} = E_{int} + E_{ext}, \tag{6.2}$$

where E_{int} is the internal energy that incorporates prior knowledge of the problem, and E_{ext} is the external energy provided by the input image. In the VTR tracking problem, two simple yet important properties of VTRs are treated as prior knowledge:

1. VTR frequencies have strict ordering ($F_1 < F_2 < F_3$ *etc*) so that they never cross each other.

2. VTR trajectories are continuous and smooth across the whole utterance.

This breaks the internal energy of each VTR snake (for each VTR trajectory) into three terms:

$$E_{int} = E_{continue} + E_{smooth} + E_{cons}, \tag{6.3}$$

where $E_{continue}$ reflects the continuity constraint, E_{smooth} reflects the smoothness constraint, and E_{cons} imposes a potential energy to penalize VTR snakes from crossing each other.

In general, N snakes are needed to track N VTR trajectories, denoted by

$$\mathbf{Y} = \left[\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^N\right], \tag{6.4}$$

and each term of the internal energy can be represented as

$$\begin{aligned} E_{continue} &= \sum_{n=1}^{N}\left[w_1 \sum_{t=2}^{T}(y_t^n - y_{t-1}^n)^2\right] \\ &= \sum_{n=1}^{N}\left\{w_1 \left\| \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix} \mathbf{y}^n \right\|^2 \right\}, \end{aligned} \tag{6.5}$$

$$\begin{aligned} E_{smooth} &= \sum_{n=1}^{N}\left[w_2 \sum_{t=3}^{T}(y_t^n - 2y_{t-1}^n + y_{t-2}^n)^2\right] \\ &= \sum_{n=1}^{N}\left\{w_2 \left\| \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix} \mathbf{y}^n \right\|^2 \right\}, \end{aligned} \tag{6.6}$$

$$E_{cons} = \sum_{n=2}^{N}\left[w_3 \sum_{t=1}^{T}\mathcal{V}(y_t^n, y_{t-1}^n)\right]. \tag{6.7}$$

The repulsive potential energy $\mathcal{V}$ between two adjacent VTR snakes is intuitively defined by

$$\mathcal{V}(x,y) = \begin{cases} 0, & \text{when } |x-y| > 0.2 \text{ kHz} \\ \frac{1}{|x-y|}, & \text{when } \varepsilon \le |x-y| \le 0.2 \\ \frac{1}{\varepsilon}, & \text{when } |x-y| < \varepsilon \end{cases} \tag{6.8}$$

where ε is used to clamp the potential energy from going to infinity ($\varepsilon = 10^{-6}$ in the actual implementation).

The external energy represents information provided by the image itself and is simply taken to be the image intensity for this problem

$$E_{image} = -w_4 \sum_{n=1}^{N} \left[\sum_{t=1}^{T} I\left(t, y_t^n\right) \right], \tag{6.9}$$

where I is the intensity at pixel position (t, y_t^n) of the spectrogram. Notice that the weights of different energy terms w_1, w_2, w_3, w_4 need to be adjusted by experiments.

In order to use efficient optimization algorithms to minimize the energy of the snakes and do VTR tracking, analytical gradient (partial derivative with respect to $\mathbf{Y}$) of the energy function E_{snake} is highly desirable. The gradient of each term of E_{int} can be calculated directly and formulated as matrix operation for the ease of implementation, e.g.,

$$\frac{\partial E_{continue}}{\partial \mathbf{Y}} = 2w_1 \sum_{n=1}^{N} \left\{ \begin{bmatrix} 1 & -1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \mathbf{y}^n \right\}. \tag{6.10}$$

However, I as taken directly from the input image is discrete in nature (only defined for discrete pixel positions) and is troublesome for most optimization algorithms if $\mathbf{Y}$ is taken as continuous. Therefore I is first turned into a continuous function by interpolation. A piece-wise cubic spline interpolation which provides C^1 continuity is adopted here. An extra benefit of doing so is that gradient of I can also be analytically calculated from the interpolated cubic splines and used in the optimization algorithm. The optimization algorithm used is the well-established Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [67, 194], readily available from Matlab.

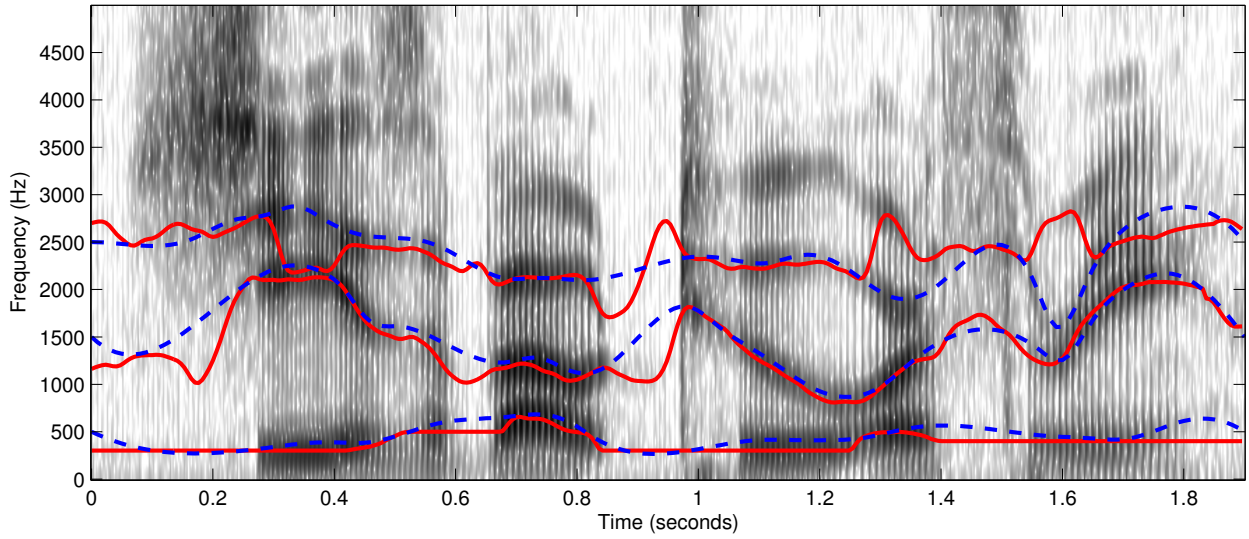

Figure 6.2: VTR tracking of a UW-XRMB sentence: *She is about two or three*. Solid red lines: VTR tracking by snakes; dash blue lines: manually tracked VTR.

6.3 VTR Tracking Experiments

Appropriate weights are first determined by trying various small scale experiments. The four weights $\{w_1, w_2, w_3, w_4\}$ are fixed to be $\{2, 15, 5, 3\}$ for examples in this section. Fig. 6.2 shows a preliminary run of applying snakes to track the first three VTR frequencies of the same sentence as in Fig. 6.1. The three VTR snakes are initialized to be at $[500\text{ Hz}, 1500\text{ Hz}, 2500\text{ Hz}]$ throughout the whole utterance respectively, and results after energy minimization are shown in the figure. It can be observed that the tracked VTR trajectories are a lot smoother comparing to Fig. 6.1, especially in the unvoiced regions, as to be expected. VTR trajectories are also correctly tracked in most vowel regions. However, some serious mistakes also occur: $F3$ is mistracked at a number of places, *e.g.*, the tracked $F3$ falsely merges with $F2$ shortly before 0.4 sec while follows $F4$ around 1.4 and 1.6 sec. This is most likely due to the crude initialization of VTR snakes and local convergence nature of the optimization algorithm used. To confirm such a guess, small scale experiments focusing on the mistracked regions with more careful initializations are carried out, and two such examples are shown in Fig. 6.3 and Fig. 6.4. In both figures,

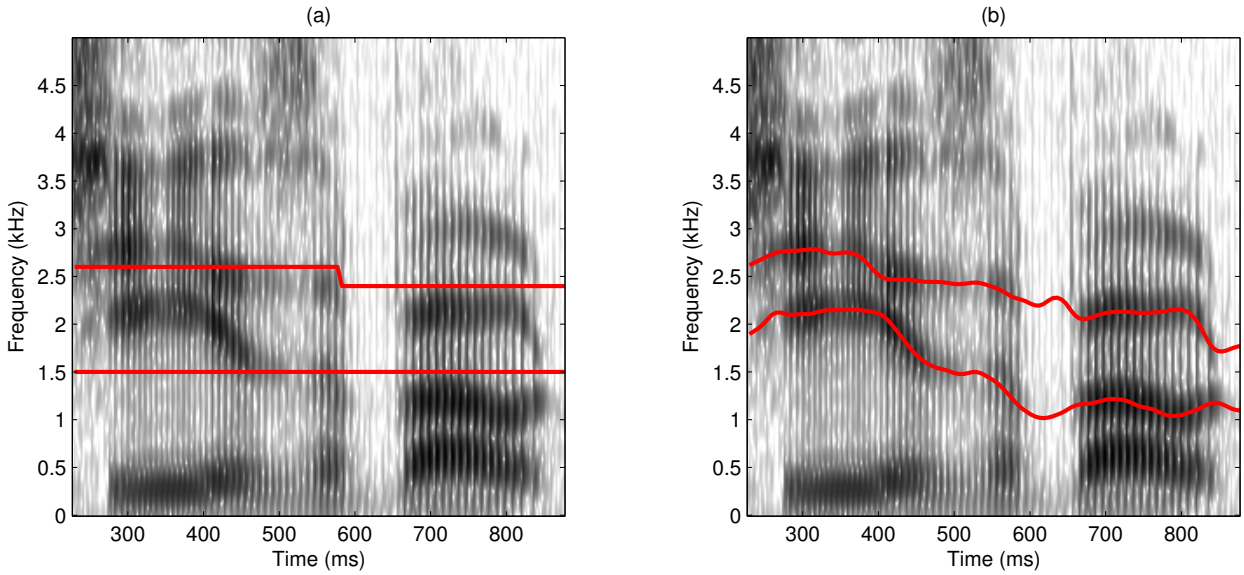

Figure 6.3: VTR ($F2$ and $F3$) tracking by snakes focusing on a small region, example 1: (a) initial values of VTR snakes; (b) after energy minimization.

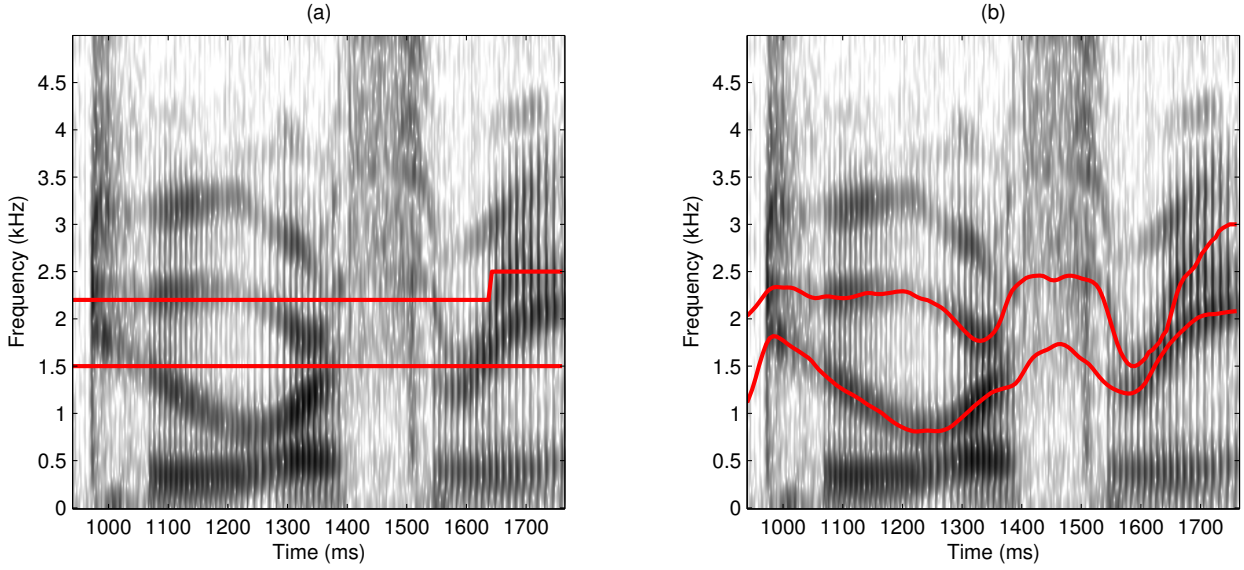

Figure 6.4: VTR ($F2$ and $F3$) tracking by snakes focusing on a small region, example 2: (a) initial values of VTR snakes; (b) after energy minimization.

correct tracking results of $F3$ are obtained by doing a slightly more careful initialization.

The simple experiments above convince us that the active image contour framework is promising at achieving robust and accurate VTR tracking for all sound classes. However, the preliminary implementation as presented here also suffers from a number of serious limitations and practical problems:

1. Adjusting the position of a snake pixel by pixel results in a very large scale optimization problem and it is computational very expensive. For example, minimizing the energy of VTR snakes for the sample sentence in Fig. 6.1 takes almost one day on a Pentium 4 2.4 GHz PC. Therefore, succinct parametrization of snake positions are required for such a method to be practical.

2. Snakes can easily get stuck into a local optimum if not initialized properly, and such local optimum points seem to be prevalent in a real spectrogram, especially for noisy speech. Therefore, good and automatic initialization schemes or global optimization methods are needed for the current method to perform robustly.

3. It is also desirable to have a region dependent (or adaptive) weighting between the internal and external energy so that image intensity can strongly influence VTR snakes in vowel regions while internal energy may prevail when the spectrogram doesn't provide enough information to guide the snakes. This may help to remove spurious movements of VTR snakes in some unvoiced regions shown in Fig. 6.2.

Some of these issues will be addressed in the next section.

6.4 Further Improvement by B-Spline Snakes and Simulated Annealing

This section offers two significant improvements over the simple active contour implementation developed in the previous sections: it provides a succinct parametrization form of VTR snakes and adopts a global optimization method.

The high complexity and slow convergence of snakes as originally proposed by Kass *et al.* have been noticed shortly after its appearance, and there have been many research

efforts to render the snakes more stable and to yield faster convergence since then. Among them one of the most successful approach is to represent the original deformable curve by a parametric B-spline form [51], which is referred to as B-spline snakes in the literature [26, 76, 166, 243]. A simple B-spline curve that fits the purpose of our VTR tracking problem is the 1-D cubic B-spline. Such a curve $s(t)$ is completely specified by a set of control points $\mathbf{y} = \{y_{-1}, y_0, \ldots, y_J, y_{J+1}\}$. Suppose these control points are uniformly spaced with interval τ, the corresponding B-spline curve $s(t)$ is constructed by:

$$s(t) = \sum_{j=-1}^{J+1} y_j \cdot \mathcal{B}^3\left(\frac{t}{\tau} - j\right), \tag{6.11}$$

where $\mathcal{B}^3(u)$ is the cubic B-spline basis function

$$\mathcal{B}^3(u) = \begin{cases} \frac{2}{3} + \frac{|u|^3}{2} - u^2, & |u| \leq 1, \\ \frac{(2-|u|)^3}{6}, & 1 < |u| \leq 2, \\ 0, & \text{otherwise.} \end{cases} \tag{6.12}$$

Fig. 6.5 plots the $\mathcal{B}^3$ basis function and a sample B-spline curve $s(t)$ generated by a set of control points. Notice that the generated spline does not pass through the control points but merely plots a smooth course among them, which is generally true for B-splines. When applied to VTR tracking, it is sufficient to place a control point every 10-20 pixels, which reduces the number of parameters by an order of magnitude comparing to the previous pixel-by-pixel implementation. Cubic B-splines are also intrinsically smooth, having continuous first and second order derivatives, which is another desirable property for VTR tracking.

The second improvement involves applying simulated annealing to solve the energy minimization problem of VTR snakes. Simulated annealing [33, 88, 135] is a powerful optimization method originated in statistical physics, inspired by the real-world phenomenon that a slow cooling process in metallurgy will alow the annealed material to approach its energy minimizing ground state. A distinctive feature of this method is that it is guaranteed to reach the *global* optimum when correctly implemented. It is most effective at solving large-scale problems with many local optima, where an energy function can be appropriately defined by some "local" interactions. All these conditions are true for our

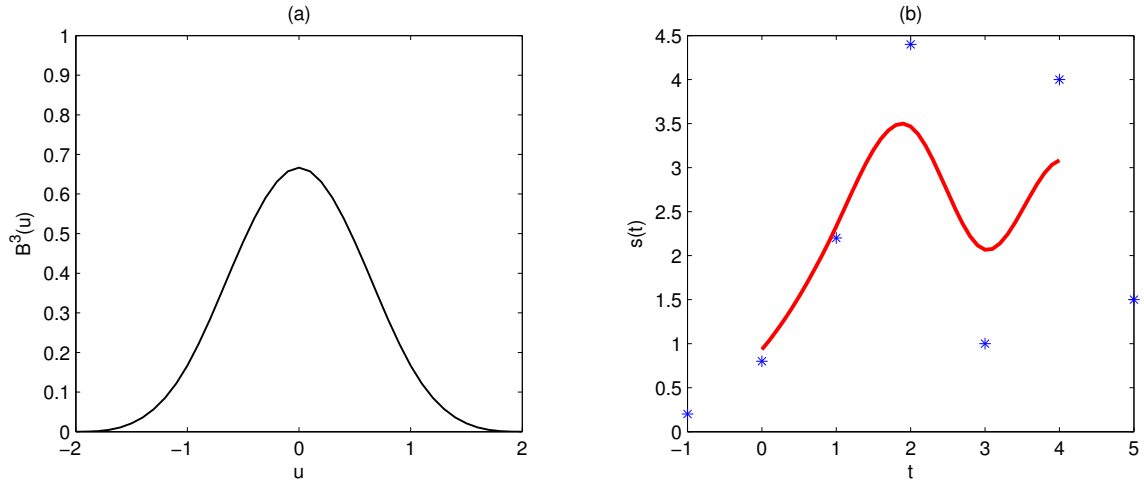

Figure 6.5: The $\mathcal{B}^3$ basis function (a) and a sample B-spline curve (b) generated by a set of control points denoted by asterisks.

VTR tracking problem: a spectrogram usually has many local optima (especially when noise is present) and is a fairly large scale problem for image processing. It is especially natural to integrate simulated annealing with snake-based VTR tracking: they both share the same notion of energy, and when B-spline snakes are used, the "locality" condition is obvious since the position of one control point only affects the shape of B-spline curve over a small neighborhood. Most of the detailed study and implementation of applying simulated annealing to the VTR tracking problem, which is under a continuous Gibbs sampling scheme, belongs to fellow group member Michael Jameison's master work [118, 119], and the details will not be repeated in this thesis. Only a number of examples and the results are presented and discussed.

Fig. 6.6 shows two examples of VTR tracking by B-spline snakes and simulated annealing. It can be observed that the VTR trajectories tracked by snakes are very smooth and agree very well with the manually labeled ones throughout the utterance. Large differences only occasionally occur at leading and trailing silences of the utterance and a few consonant regions, where the accuracy of manual tracking is also low. The average absolute distance (excluding silence periods) between the automatically and manually tracked VTR for one speaker's data in the TIMIT database is $[58\text{ Hz}, 87\text{ Hz}, 76\text{ Hz}]$ for F_1, F_2, F_3 respectively.

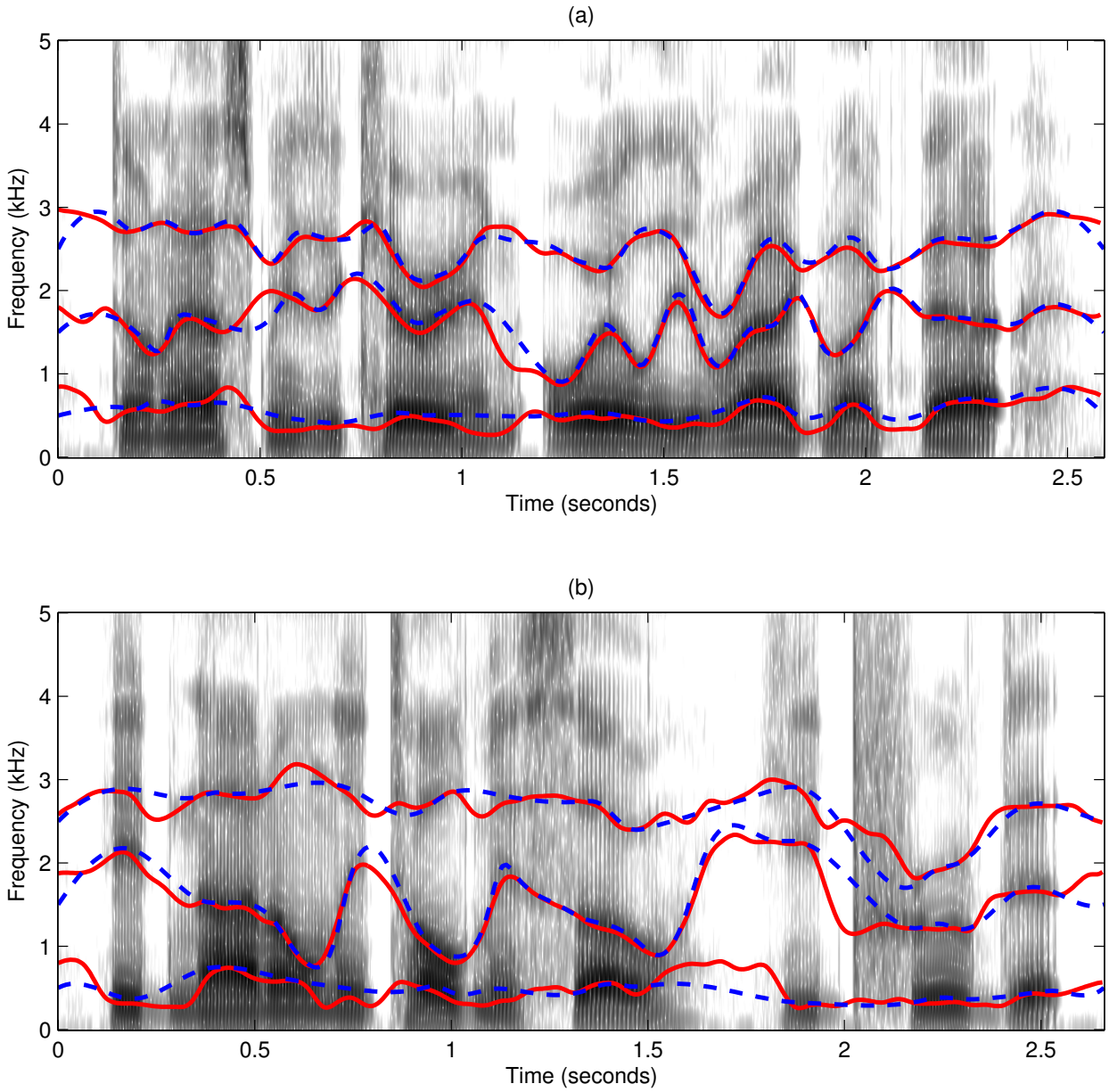

Figure 6.6: VTR tracking by B-spline snakes and simulated annealing for two typical TIMIT sentences: (a) *Don't ask me to carry an oily rag like that*; (b) *He'd not only told me so, he'd proved it.* Solid red lines: VTR tracking by B-spline snakes; dash blue lines: manually tracked VTR.

Since manual tracking can only achieve an accuracy of about ±40 Hz in voiced regions [178], the accuracy of the automatically tracked VTR frequencies by B-spline snakes is very close to human performance.

As a summary, we have developed a VTR tracker that almost rivals human performance by applying advanced active image contour estimation methods. However, our method is indeed much more complicated than the standard LPC based methods and requires a considerable amount of computation. While the computational time of tracking VTR dynamics for a typical sentence by LPC based methods is almost negligible on modern computers, it takes half an hour to an hour when B-spline snakes are used. Another limitation of such an image processing based approach is that VTR bandwidths can not be directly estimated, unlike in most LPC based methods. As will be seen shortly, VTR bandwidths are important in predicting acoustic features from VTR dynamics and thus highly desirable when building a speech model with VTR dynamics acting as the *hidden* dynamics underlying acoustic measurements. Both limitations will be addressed in the next chapter when the VTR dynamics are recovered under such a hidden dynamics modeling framework.

Chapter 7

VTR Tracking with a Hidden Dynamic Model (HDM)

As opposed to the previous chapter, where VTR dynamics are extracted based on image processing techniques, a different approach is undertaken in this chapter: to recover VTR dynamics based on formulating a hidden dynamic model (HDM) of speech. It starts by deriving an analytical formula that maps the VTR dynamics to the acoustic space, and carefully linearize the original mapping with high accuracy to facilitate further computation. Then a HDM is formulated that describes both the VTR dynamics and the mapping. The standard Kalman smoother is applied to track the underlying VTR frequencies and bandwidths accurately and efficiently. And finally as an example of application, VTR residual is used as a new acoustic feature to improve TIMIT phone recognition accuracy over a well-trained HMM baseline system.

7.1 The VTR-to-Acoustic Mapping

It is possible to learn the VTR-to-acoustic (VTA) mapping by data-driven methods as in the study of ATA mapping in Section 4.2. Indeed MLPs and mixture linear models have been applied in related previous work [157, 236], although no formal evaluation on the quality of these approximations has been carried out. However, lessons learned from using data-driven methods in Section 4.2 suggests that a knowledge-based method can be of great

advantage if it is possible to be carried out. It has the potential of being more accurate with less computation (if the knowledge itself is accurate), and also completely eliminates the need of learning extra parameters when such a mapping is included in a speech model. Even if the knowledge is incomplete, it can be supplemented by data driven methods and may still dramatically reduces the number of learnable parameters comparing to a completely data driven approach. Fortunately due to the close relationship between VTRs and acoustics, such a knowledge-based approach is possible under reasonable assumptions. The difficulty of deriving such a relationship depends on the acoustic feature desired, and a particular convenient choice is LPC-cepstra, where such a mapping can be expressed in closed-form. This derivation is carried out first.

7.1.1 VTR to LPC-cepstra Nonlinear Mapping

Here we assume an all-pole vocal tract model as introduced in Section 2.2, and represent each pole by a VTR frequency-bandwidth pair (F_n, B_n). The corresponding conjugate complex roots of the denominator in (2.1) are

$$\begin{aligned} p_n &= \exp(-\pi\frac{B_n}{F_s} + j2\pi\frac{F_n}{F_s}), \\ p_n^* &= \exp(-\pi\frac{B_n}{F_s} - j2\pi\frac{F_n}{F_s}), \end{aligned} \tag{7.1}$$

where F_s is the sampling frequency. The vocal tract transfer function with N pairs of such poles and a gain of G can be expressed as

$$H(z) = \frac{G}{\prod_{n=1}^{N}(1-p_n z^{-1})(1-p_n^* z^{-1})}. \tag{7.2}$$

Taking logarithm on both sides

$$\log H(z) = \log G - \sum_{n=1}^{N}\log(1-p_n z^{-1}) - \sum_{n=1}^{N}\log(1-p_n^* z^{-1}), \tag{7.3}$$

and further use the well-known Taylor series expansion $\log(1-x) = -\sum_{i=1}^{\infty} \frac{x^i}{i}$, we have

$$\begin{aligned}
\log H(z) &= \log G - \sum_{n=1}^{N}\sum_{m=1}^{\infty} \frac{p_n^m z^{-m}}{m} - \sum_{n=1}^{N}\sum_{m=1}^{\infty} \frac{p_n^{*m} z^{-m}}{m} \\
&= \log G - \sum_{m=1}^{\infty} \left[\sum_{n=1}^{N} \frac{p_n^m + p_n^{*m}}{m} \right] z^{-m} \\
&= \log G - \sum_{m=1}^{\infty} \left[\sum_{n=1}^{N} \frac{2}{m} \exp(-\pi m \frac{B_n}{F_s}) \cos(2\pi m \frac{F_n}{F_s}) \right] z^{-m}. \qquad (7.4)
\end{aligned}$$

LPC cepstra are the inverse z-transform of the above equation. By adopting a one-sided or unilateral definition of the z-transform, it is obvious that

$$\begin{aligned}
C_0 &= \log G, \\
C_m &= \sum_{n=1}^{N} \frac{2}{m} \exp(-\pi m \frac{B_n}{F_s}) \cos(2\pi m \frac{F_n}{F_s}), \quad m > 0.
\end{aligned} \qquad (7.5)$$

Notice that (7.5) not only gives a closed-form relationship between VTR frequency-bandwidth pairs and a m^{th} order LPC-cepstrum coefficient, but also demonstrates an important decomposition property: each VTR frequency-bandwidth pair contributes independently to a LPC-cepstrum, and the total contribution is just the summation of individual ones. Such a property greatly facilitates the efficient use of this mapping in VTR tracking and other related applications, as will become clear soon. The derivation presented here is not new, basic ideas can be found in standard speech processing textbooks [52, 197]. Under the context of VTR or formant tracking, it appeared in closely related previous work [61], and a slight variation (and extension) also appeared in [264]. However, the method of how such an approximate (under the all-pole model assumption) analytical nonlinear mapping can be used in VTR tracking and beyond is completely novel due to this thesis.

7.1.2 A Piecewise Linear Approximation

In order to effectively use the above derived nonlinear analytical mapping (7.5), a piecewise linear approximation is carefully constructed in this section. The actual process, to be

described in detail below, is relatively straightforward but the resulting piecewise linear mapping is the key leading to computational efficiency.

Let us consider the following nonlinear function

$$h(f,b) = \frac{2}{m}\exp(-\pi m\frac{b}{F_s})\cos(2\pi m\frac{f}{F_s}), \tag{7.6}$$

which represents the contribution of a specific VTR frequency-bandwidth pair (f,b) on the m^{th} order LPC-cepstrum. Notice that in order to construct a piecewise linear approximation to the analytical nonlinear mapping (7.5), we only have to construct a piecewise linear approximation for $h(f,b)$, since contributions from different VTR frequency-bandwidth pairs are already linear due to the decomposition property indicated by (7.5). However, constructing a piecewise linear approximation for $h(f,b)$ is a straightforward task, and can be completed in two steps: we first linearize f and b separately while fixing the other variable, then combine the results. $h(f,b)$ is a cosine function of f and an exponential function of b, both can be linearized to high accuracy by carefully choosing linearization points. For a cosine function, six nonuniformly-spaced linearization points are used in every half cycle, *e.g.*, $\{0, \frac{1}{8}\pi, \frac{1}{4}\pi, \frac{3}{4}\pi, \frac{7}{8}\pi, \pi\}$ are selected as the linearization points for the range of $[0,\pi]$, and such a range is divided into five linear regions by these linearization points. For the exponential function on b, the linearization points are uniformly placed every 50 Hz.

This linearization process and its effect can be clearly visualized by plotting an example, which is shown in Fig. 7.1 for $m = 6$ and $F_s = 8000$ Hz. It can be observed that the linearized functions are almost indistinguishable from the original nonlinear ones for both f and b, indicating a high accuracy fit by our choice of linearization points. When both f and b vary according to their valid range (0 to 4000 Hz for f and 0 to 500 Hz for b), the same set of linearization points are used and the original nonlinear 2D surface is approximated by a set of triangular planes defined by these linearization points. Again a picture may worth a thousand words to illustrate the linearization process and an example for $m = 5$ and $F_s = 8000$ Hz is plotted in Fig. 7.2. It can be observed that the piecewise-linearly approximated 2D surface (c) is virtually the same as the original nonlinear surface (a), and the very small approximation errors (d) are only visible by an enlarged scale.

Under such a linearization scheme, a piecewise linear approximation to the analytical nonlinear mapping (7.5) is pre-calculated and stored, including the valid range of each

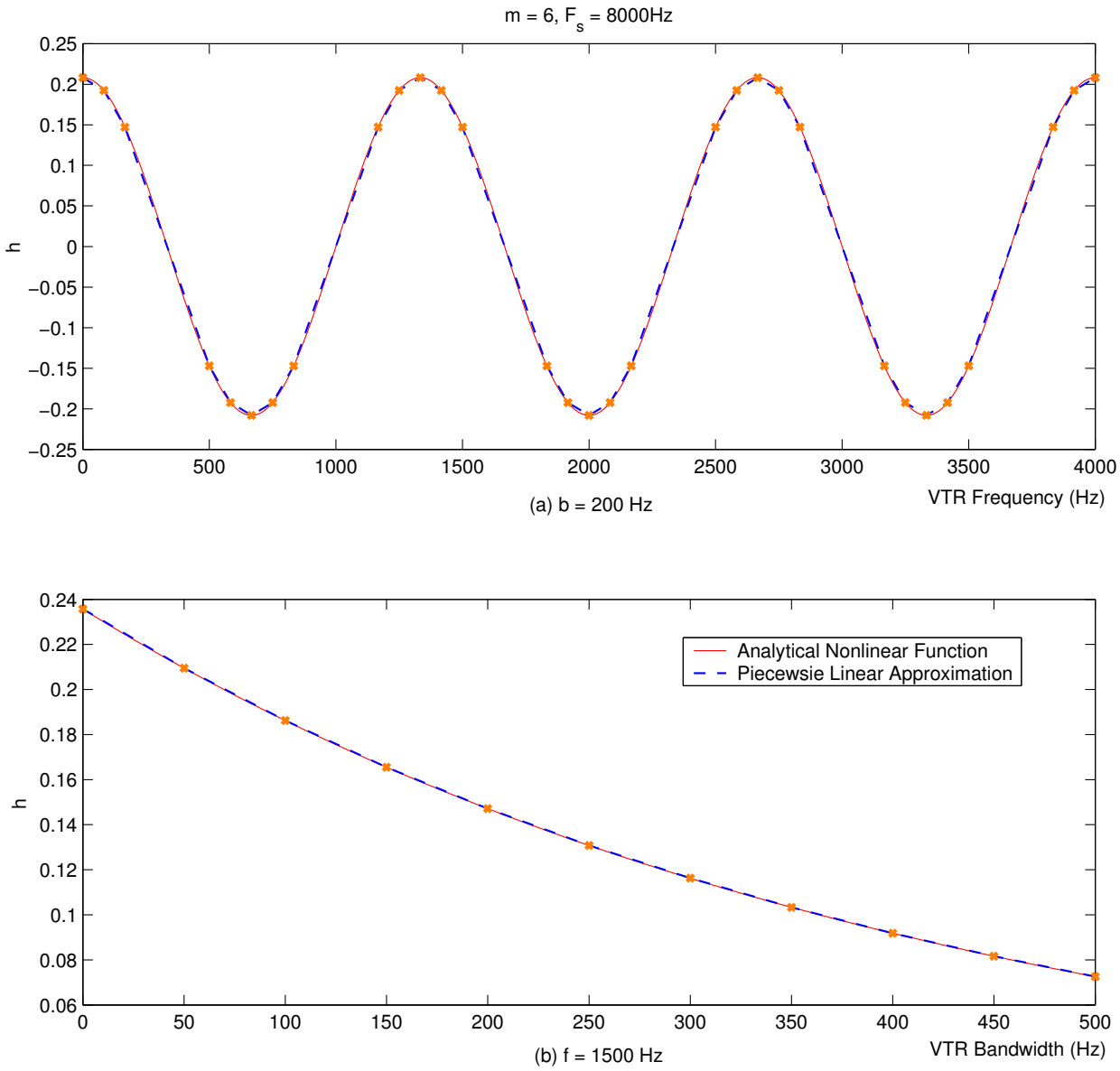

Figure 7.1: Linearization of $h(f, b)$ on f (a) and b (b) separately while fixing the other variable; the linearization points are denoted by crosses.

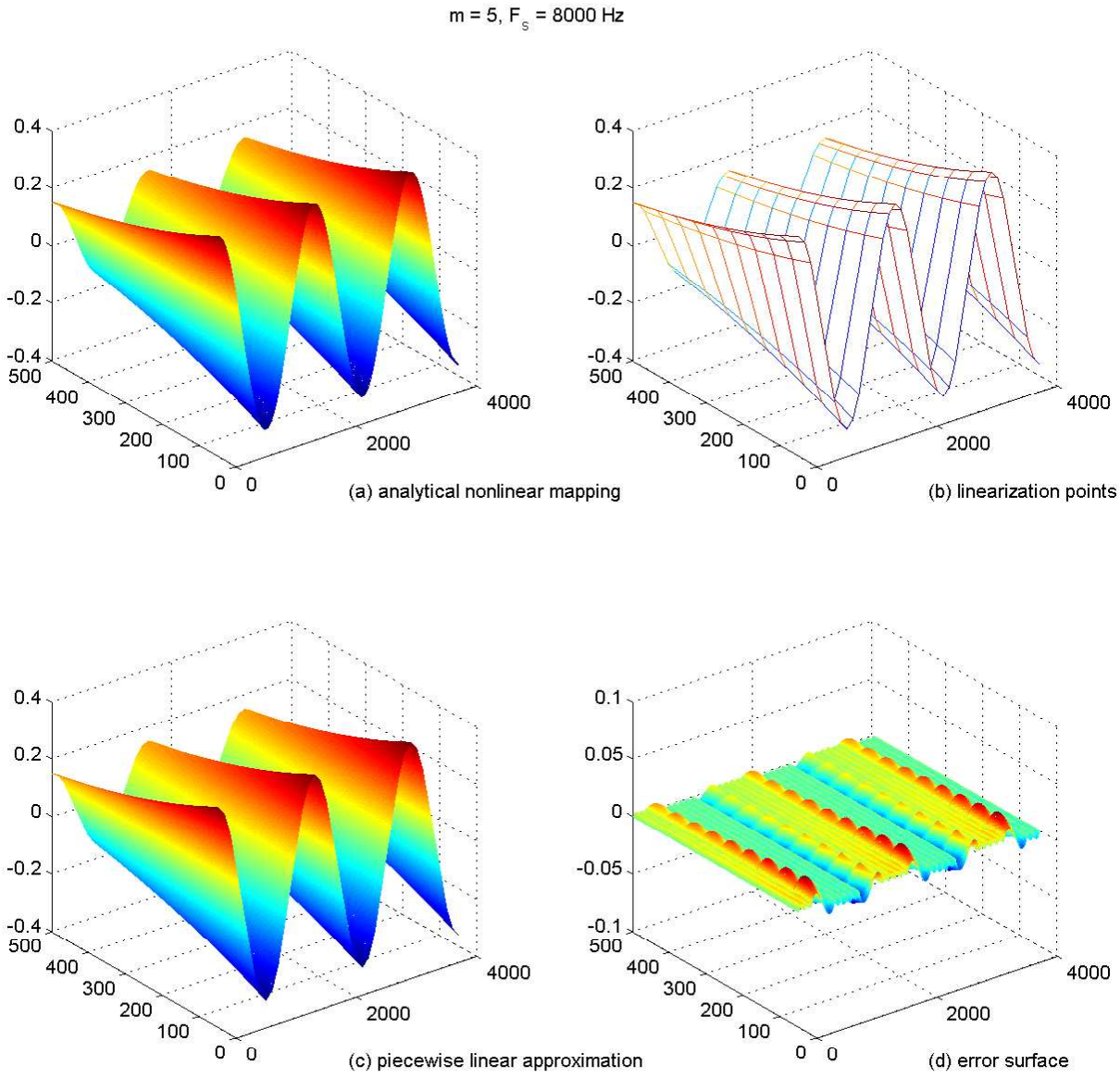

Figure 7.2: Piecewise linear approximation to an analytical 2D nonlinear function $h(f, b) = \frac{2}{m} \exp(-\pi m \frac{b}{F_s}) \cos(2\pi m \frac{f}{F_s})$.

linearization region i (of the shape of a triangular plane) and the linear function $l_i(f, b)$ associated with each region, which is of the form

$$l_i(f, b) = \alpha_i \cdot f + \beta_i \cdot b + \gamma_i. \tag{7.7}$$

When assessing such a piecewise linear mapping, it is only necessary to decide which linearization region to use based on the input value (f, b). Therefore such a carefully constructed, highly accurate piecewise linear mapping, which will be denoted as $\mathcal{P}[\cdot]$ hereafter, can also be used conveniently and efficiently.

7.1.3 How About Other Acoustic Features?

It is natural to ask whether similar high accuracy piecewise linear mappings can be constructed for other acoustic features of interest as well. Carefully examining the procedure of constructing the piecewise linear approximation in the previous section reveals that two important properties of the original VTA mapping are needed for such a procedure to be successful:

1. The acoustic features of interest must be able to be predicted from an all-pole vocal tract model.

2. It is crucial for the decomposition property, *i.e.*, each VTR frequency-bandwidth pair contributing independently, to hold so that simple linearization can be carried out.

The first property holds for almost all acoustic features, although the difficulty of obtaining such a prediction varies, and typically a closed-form solution such as (7.5) is impossible. The second property is shared by all LPC-based features, popular ones including line spectral frequencies (LSF, widely used in speech coding) and perceptual linear prediction (PLP) coefficients (popular for speech recognition). A notable exception is MFCCs, which has been used in the study of ATA mapping in Section 4.2 and is a popular acoustic feature for speech recognition. The decomposition property of VTR frequencies and bandwidths does not hold for MFCCs and a piecewise linear mapping from VTR to MFCCs can not be constructed efficiently. For simplicity, the remaining of the thesis will use LPC-cepstrum feature wherever the VTA mapping is desired.

7.2 The HDM for VTR Tracking

A general hidden dynamic modeling framework has been set up in Section 4.4, where the hidden state (or hidden dynamics) is articulatory dynamics. This section adapts this general framework to use VTR dynamics as the hidden state in speech modeling.

7.2.1 A HDM Using VTR Dynamics

Two slight modifications are made comparing to the state-space model described in Section 4.4: first we replace the articulatory dynamics by VTR dynamics; second we replace the nonlinear ATA mapping by the above constructed piecewise linear VTA mapping. After these modifications, the state-space model (or HDM) is expressed as

$$\mathbf{z}_k = \mathbf{\Phi}_s \mathbf{z}_{k-1} + (\mathbf{I} - \mathbf{\Phi}_s)\mathbf{u}_s + \mathbf{w}_k, \tag{7.8a}$$

$$\mathbf{o}_k = \mathcal{P}[\mathbf{z}_k] + \mathbf{r}_s + \mathbf{v}_k. \tag{7.8b}$$

Note that the state equation (7.8a) stays the same as before, we merely define a new state vector

$$\mathbf{z} = [F_1, F_2, F_3, F_4, B_1, B_2, B_3, B_4]^T, \tag{7.9}$$

to use VTR dynamics instead of articulatory dynamics in the state equation. The form of the observation equation (7.8b) has been slightly changed to take advantage of the phone-independent piecewise linear mapping $\mathcal{P}[\cdot]$. A learnable, phone-dependent deterministic residual vector $\mathbf{r}_s$ is also introduced to account for the inaccuracy of $\mathcal{P}[\mathbf{z}]$, in addition to the use of zero-mean white Gaussian noise $\mathbf{v}_k$. The state vector $\mathbf{z}$ itself deserves further explanations as well:

- VTR bandwidths have been included in the state vector, even though it has been mentioned previously (in Section 5.3) that they don't possess true dynamics, or in another word, they do not have physically meaningful time-correlations. When producing acoustic waveforms, VTR bandwidths can change rapidly, but there is still a well-defined bandwidth target for each phone (as listed in Tables 5.1 and 5.2). The behavior of VTR bandwidths can be adequately captured by the state equation (7.8a) as well, *e.g.*, if we set $\mathbf{\Phi}_s = 0$, then the time correlation between subsequent

state vectors will be completely removed and $\mathbf{z}_k$ will behave as a Gaussian noise with a mean at the target position, roughly corresponding to the bandwidth behavior described above. In practice, some weak time correlation between subsequent bandwidth values can still be beneficial and a small time constant (0.1, for example) may be used to reflect this.

- F_4 and B_4 have been included in the state vector even though there are no target values listed for them in Tables 5.1 and 5.2. The reason to include F_4 and B_4 is mainly to better account for the acoustic observation after some comparative studies with models only include VTR frequencies and bandwidths up to F_3 and B_3. This is equivalent to increasing the order of an all-pole model. Although phone-dependent target values of F_4 and B_4 are not revealed by previous research, their approximate range is known. Here phone-independent target values of 3500 Hz and 300 Hz are used for F_4 and B_4 respectively.

We will start with the diagonal assumption on $\mathbf{\Phi}_s$, motivated by the success of VTR trajectory fitting in Section 5.3. For simplicity, we also assert diagonal structures for the noise covariance matrices $\mathbf{Q}_s$ and $\mathbf{R}_s$. It is insightful to count the number of learnable parameters when the above proposed HDM (7.8) is used for speech modeling under such a simple setup. Suppose we use a phone table containing 50 phones and a 16×1 acoustic feature vector[1], then the total number of trainable model parameters is $(8+8+8+16+16) \times 50 = 2800$. This is in stark contrast to state-of-the-art HMM speech recognition systems which typically have a few million trainable parameters [64]. Of course the number of learnable parameters will increase if further model improvement is carried out, such as making $\mathbf{\Phi}_s$ to be a block diagonal or full matrix to better account for correlation among different VTR frequencies and bandwidths, or to split a phone into two sub-phones to model transient phones more accurately (such as /b/ which involves two distinct stages during production: a complete closure followed by a subsequent release). But the HDM as described in this section will still be able to hold a clear edge over traditional HMM based systems in terms of model parameters that need to be learned. Again, more in-depth

[1]One choice could be 15 LPC-cepstrum coefficient plus energy, where the energy will not be predicted by $\mathcal{P}[\cdot]$ but completely characterized by the trainable residue vector $\mathbf{r}_s$.

discussion of what this implies to speech recognition will be presented in Part III of the thesis.

7.2.2 A Simplified HDM for VTR Tracking

The full-fledged HDM incorporating VTR dynamics described above has most model parameters phone-dependent. In practical VTR tracking scenarios, phone transcripts are typically not known, nor do we want to recover them[2]. This section describes a simplified HDM with pre-determined model parameters to facilitate VTR tracking without extra phonetic knowledge.

The first step of setting up the simplified model is to remove model parameter dependency on the underlying speech unit. After doing so, the state-space model can be rewritten as

$$\mathbf{z}_k = \mathbf{\Phi}\mathbf{z}_{k-1} + (\mathbf{I} - \mathbf{\Phi})\mathbf{u} + \mathbf{w}_k, \tag{7.10}$$

$$\mathbf{o}_k = \mathcal{P}[\mathbf{z}_k] + \mathbf{r} + \mathbf{v}_k. \tag{7.11}$$

Next the model parameters need to be determined so that the model can be used for VTR tracking without learning its parameters on the fly. This is accomplished based on the author's experience and small scale tuning experiments, and the details are as follows.

Notice that if we set $\mathbf{\Phi} = \mathbf{I}$, the state equation (7.10) becomes

$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{w}_k, \tag{7.12}$$

which is a random walk process that provides some continuity constraint for VTR frequencies between consecutive frames while the targets play no role. This is also a common continuity constraint that has been used in a number of previously-developed VTR trackers [61, 207]. However, we believe that it is still important to use a phone-independent mean target value to limit the valid range of VTR frequencies and provide a prior nominal value for them, especially when the acoustic signal is weak. This is confirmed by experiments,

[2]This is essentially the goal of speech recognition, which is a far more difficult problem.

and after some tuning, $\mathbf{\Phi}$ and $\mathbf{u}$ are fixed to be:

$$\mathbf{\Phi} = \text{diag}([0.95, 0.95, 0.95, 0.95, 0.1, 0.1, 0.1, 0.1]), \tag{7.13}$$

$$\mathbf{u} = [0.5 \text{ kHz}, 1.5 \text{ kHz}, 2.5 \text{ kHz}, 3.5 \text{ kHz}, 0.2 \text{ kHz}, 0.3 \text{ kHz}, 0.3 \text{ kHz}, 0.3 \text{ kHz}]^T, \tag{7.14}$$

where $\mathbf{\Phi}$ is a diagonal matrix. Some weak time correlation (a time constant of 0.1) between consecutive bandwidth values is also provided. For simplicity, we set $\mathbf{r} = 0$ to totally rely on the piecewise linear mapping $\mathcal{P}[\cdot]$ to predict LPC-cepstra from VTR frequencies and bandwidths. The relative magnitude of the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ (of the state and observation noises respectively) reflects the relative importance of the state and observation equation in estimating $\hat{\mathbf{z}}_k$ and this is again set by empirical tuning. Diagonal structures are adopted for both $\mathbf{Q}$ and $\mathbf{R}$: $\mathbf{R}$ has been set to some sample variance of LPC-cepstra calculated from a small set of TIMIT sentences and $\mathbf{Q}$ is tuned to be

$$\mathbf{Q} = \text{diag}([0.04, 0.04, 0.04, 0.04, 0.01, 0.01, 0.01, 0.01]). \tag{7.15}$$

To track VTR frequencies and bandwidths of an utterance, the standard Kalman smoother [165, 220, 221] can be directly applied after the LPC-cepstra is first computed from the speech waveform. The detailed equations and implementation steps of a Kalman smoother are omitted in this section[3]. An important point to make is that the efficient use of Kalman smoother is made possible by the pre-design of the piecewise linear mapping $\mathcal{P}[\cdot]$, otherwise we will have to resort to the computationally much more expensive (and possibly also less accurate) extended Kalman smoother. The speed of this Kalman smoother based VTR tracker is comparable to standard LPC based methods: it takes less than a second to track a typical sentence on a modern PC with codes written in Matlab.

7.3 VTR Tracking Results and Analysis

VTR tracking has been performed on the whole TIMIT database, which contains a total of 6,300 sentences. For all sentences, the state vector $\mathbf{z}$ is initialized to be the same as the phone-independent target value $\mathbf{u}$.

[3]They are listed in Section 9.2, although for a different purpose.

Fig. 7.3 (a) shows the tracking result on a typical TIMIT sentence, where the four solid red lines are trajectories of F_1 to F_4, and the two dashed blue lines above and below each solid line show the $F_i + B_i$ and $F_i - B_i$ tracks. The overall trajectories are quite smooth, especially comparing to typical results of standard LPC-based methods such as the one shown in Fig. 6.1. All major resonance frequencies that can be identified by eyes have been tracked correctly, and the estimated bandwidths also appear to be reasonable. Another typical example of VTR tracking is provided in Fig. 7.3 (a), where similar tracking quality has been obtained, although careful scrutiny can reveal some minor errors, *e.g.*, the F_2 and F_3 merge around 1.5 sec hasn't been tracked accurately. Spot-checked results indicate that tracking quality similar to Fig. 7.3 (a) and Fig. 7.4 (a) has been obtained on all sentences. The Kalman smoother tracked VTR frequencies have also been compared to manually tracked ones, and the absolute distance between these two sets of VTR frequencies is $[83 \text{ Hz}, 124 \text{ Hz}, 117 \text{ Hz}]$ for F_1, F_2, F_3 respectively, for one speaker's data which has been hand-labeled. This result is not as good as the one obtained by B-spline snakes in the previous chapter on page 117, but is offset by the computational efficiency of the current method.

Since manual VTR tracking is not able to provide reliable bandwidth estimate, an alternative method is used to further analyze the quality of Kalman smoother based tracking besides visual inspection. Here we first compute predicted LPC-cepstrum features from the estimated VTR frequencies and bandwidths via equation (7.5), then calculate the difference between the predicted LPC-cepstra and the true one obtained from speech waveforms. This difference vector will be referred to as the residual feature from now on. All three features (the true LPC-cepstra, the predicted one and the residual) are calculated for the two example sentences and visually displayed as spectrograms in Fig. 7.3 and Fig. 7.4 (b)-(d). If our VTR tracker can estimate VTR dynamics accurately, we will expect to see most of the true dynamics, which can be inferred from the true spectrogram, to be reflected in the predicted spectrogram. Furthermore, if the estimated VTR frequencies and bandwidths can predict acoustic well, most of the speech energy below 4 kHz should be removed (this is roughly the frequency range F_1–F_4 and B_1–B_4 can predict) while high energy should only appear at high frequencies in the residual spectrogram. Both expectations are met in the spectrogram displays of Fig. 7.3 & 7.4, where Fig. 7.3 & 7.4 (c) show that all speech

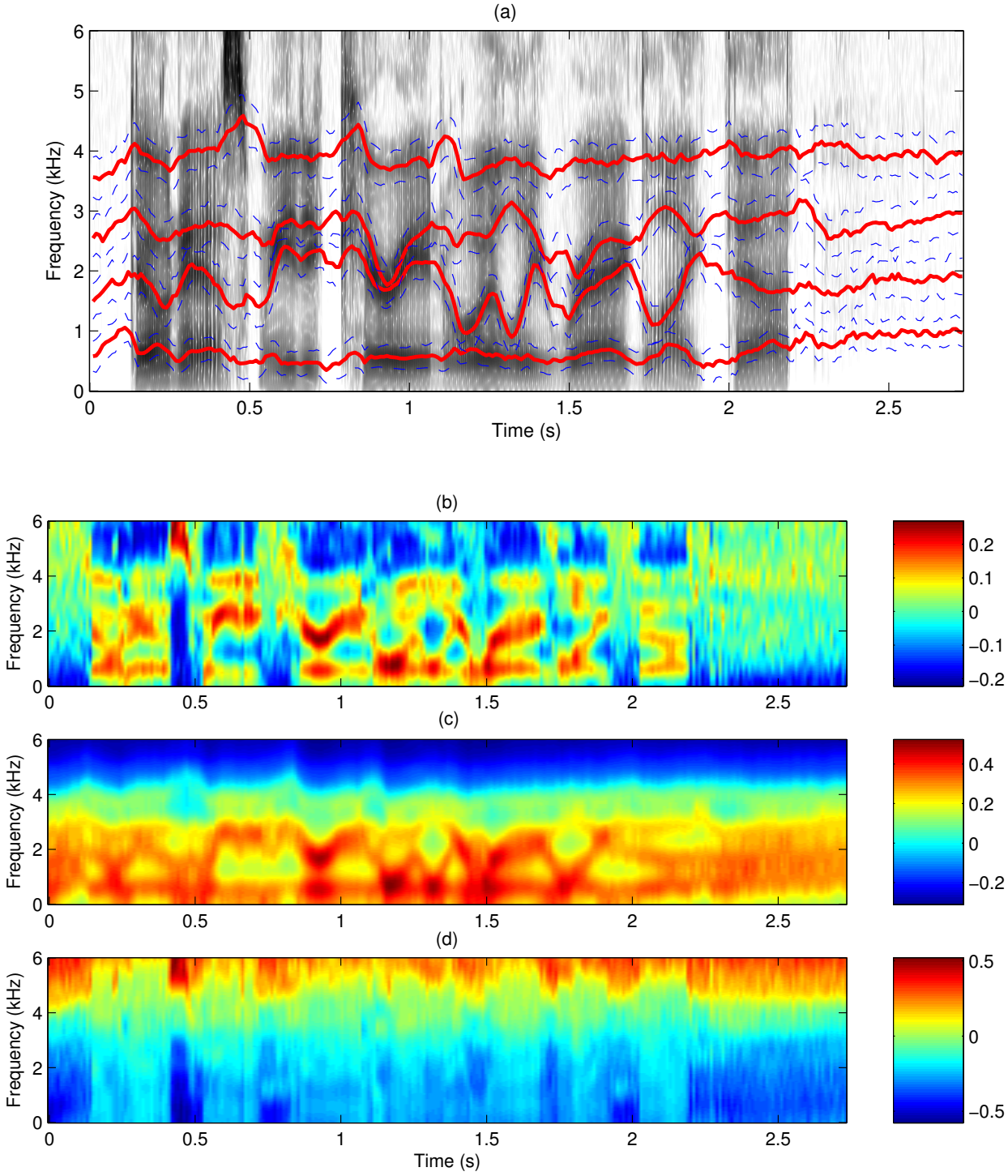

Figure 7.3: TIMIT VTR frequency and bandwidth tracking example 1: (a) tracked VTR frequency (solid red lines) ± bandwidth (dash blue lines); (b) original spectrogram with a color scale; (c) predicted spectrogram from tracked VTRs; (d) residual spectrogram.

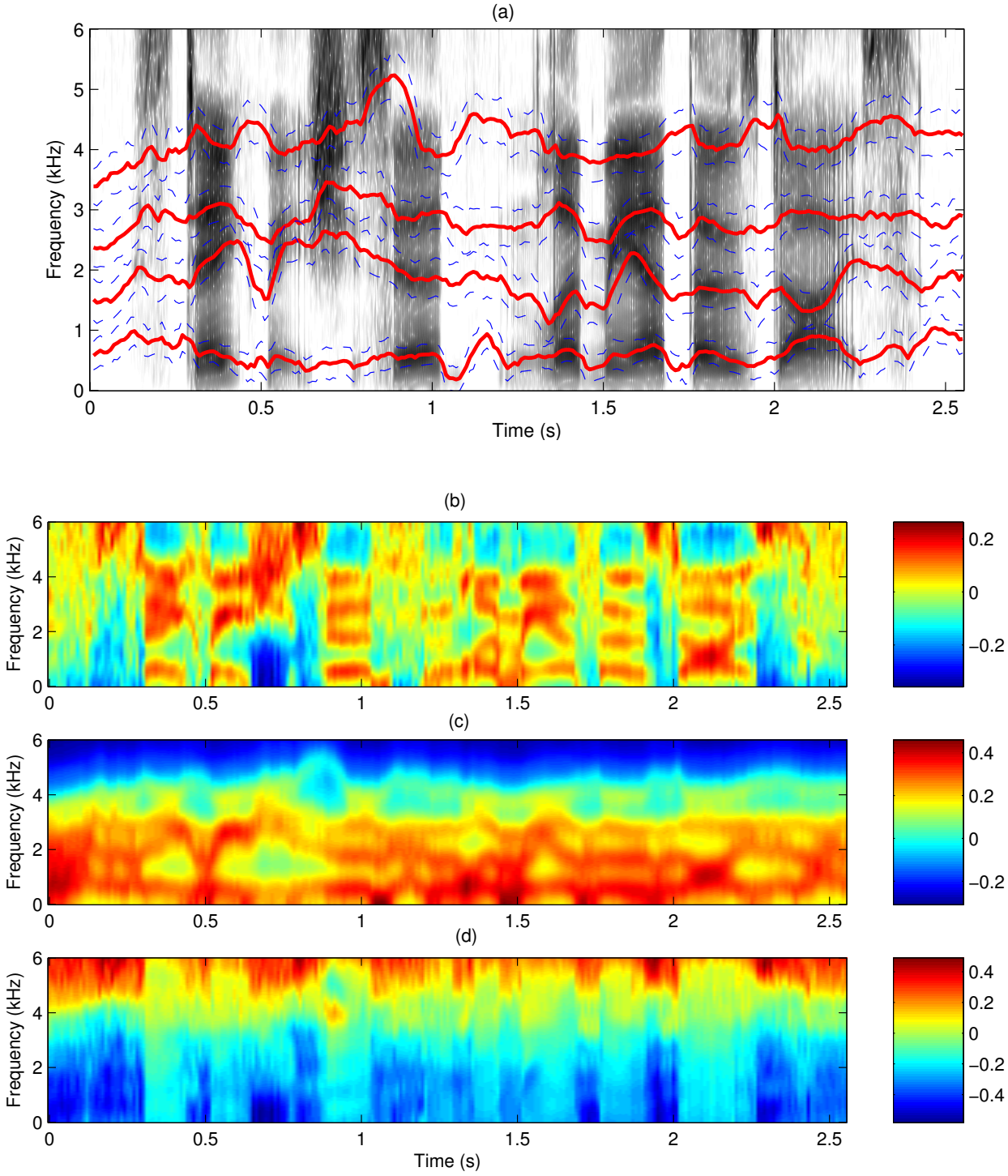

Figure 7.4: TIMIT VTR frequency and bandwidth tracking example 2: (a) tracked VTR frequency (solid red lines) ± bandwidth (dash blue lines); (b) original spectrogram with a color scale; (c) predicted spectrogram from tracked VTRs; (d) residual spectrogram.

dynamics below 4 kHz have indeed been well captured and the residual spectrogram in (d) only have substantial energy at frequencies higher than 4 kHz.

The above results and analysis confirm that the Kalman smoother based VTR tracker has performed quite well. Besides applicable to speech analysis in general, such a fast and accurate VTR tracker also plays an important role in the next part of the thesis when efforts are made to develop a full-fledged HDM based speech recognizer. It can provide much needed good initialization when variational training and decoding algorithms are applied, which will be further discussed in Chapter 10. It can be further observed that the residual spectra exhibit different characteristics for different phones or phone classes, *e.g.*, prediction is generally better in vowel regions than consonant regions (this is to be expected since VTR dynamics is mostly observable from acoustics in vowel regions). This is one reason that leads to the use of residual features in a traditional HMM-based phone recognizer, to be presented in the next section.

7.4 Using VTR Residual Feature in a HMM Speech Recognizer

This section describes the use of VTR residual to improve a high performance HMM based TIMIT phone recognizer. Readers without a background in speech recognition are encouraged to read the friendly introduction provided in Chapter 8 (the next chapter) or at least the first section of that chapter before reading this section. It should be admitted from the outset that improving a high performance HMM baseline system by merely using a different feature is very difficult[4], and typically involves lots of empirical tuning. However, our focus is not on the performance gain itself but rather to see if the residual feature can provide *any* gain at all, as a simple way to demonstrate the *discriminability* of the HDM besides its accuracy.

[4]Many similar attempts in the literature have a much worse baseline system so that performance gain may be larger.

7.4.1 Description of the Baseline System

Some technical details about the baseline HMM phone recognizer which is to be further improved are presented first.This is mostly intended as a reference for people who already know how to build a HMM based speech recognizer. Unfamiliarity with these details will in no way hinder understanding the main points of this section, nor any remaining parts of the thesis.

The goal of this particular speech recognizer is to decode the correct phoneme sequences, opposed to word sequences or sentences, on the TIMIT database. Therefore it will be more specifically referred to as a phone recognizer. The baseline system uses triphone mixture-of-Gaussian HMMs as the acoustic model, and is developed under HTK, a popular HMM toolkit freely available from the University of Cambridge [259]. A bigram phone language model is also used and trained on the TIMIT training set. The acoustic feature used as the input to the phone recognizer is 12 perceptual linear prediction (PLP) coefficients plus energy, appended by delta and acceleration coefficients, which results in a 39×1 acoustic feature vector. The phone recognizer is trained on a 48 phone table (merged from the original 64) and tested on a further merged 39 phone table, which is the standard of evaluating TIMIT phone recognition performances established after Lee and Hon [145]. It reaches a phone recognition error rate of 27.28% with 10 Gaussian mixture components. Since the best error rate ever reported by a similar system is 27.1% [143], our system qualifies as a very high performance one.

7.4.2 Why Use VTR Residual?

The use of VTR residual in a HMM phone recognizer is motivated by the following two characters of the residual vector:

1. Since most speech dynamics have been accounted for by the spectra predicted from estimated VTR frequencies and bandwidths and hence be removed in the residual feature, the HMM assumption, which assumes inter-frame independence given the underlying states, will be appropriate for the residual feature.

2. As mentioned previously, different phones or phone classes tend to have different

residual spectra. Therefore the residual feature may aid in discriminating different phones.

To compensate for the lost phonetic information contained in the predicted LPC-cepstra, the residual feature has been appended to the original full feature (PLP plus energy) used in the baseline HMM system. This so formed new "super" feature set is used as the input to train and test a HMM phone recognizer.

7.4.3 TIMIT Phone Recognition Results

It is shown that the HMM phone recognizer with the new super feature set consistently outperforms the baseline system under all conditions, although the relative improvement is small. The actual phone recognition error rates are listed in the following table. These

No. of Gaussian mixtures	8	10	12	14
PLP feature (baseline)	28.00	**27.28**	27.50	28.10
PLP + residual	27.60	**26.99**	27.45	27.80

Table 7.1: TIMIT phone recognition error rates (%) of two triphone HMM systems.

results confirm the effectiveness of the residual feature in discriminating different phones. The small improvement is likely due to the high performance already achieved by the baseline system. Further discussions on this result, under a much broader context, is provided in the next section.

7.5 Conclusions and Discussions

This section concludes the study on VTR dynamics with some insightful discussions:

- First the detailed studies carried out in this part have verified that a HDM using VTR dynamics as proposed in Section 7.2.1 is a quite accurate model of speech. The accuracy of the state equation (7.8a) is verified by the VTR trajectory fitting experiments in Section 5.3, and supported by the author's (and many others') spectrogram reading experience. The accuracy of the observation equation (7.8b) is verified by the

performance of the VTR frequency and bandwidth tracker developed in this chapter. Since the state equation (7.10) only provides some weak global constraint in VTR tracking, it is the observation equation (7.11) that provides most of the power to enable high accuracy tracking and acoustic prediction.

- Second there is a straightforward enhancement that can be made on the Kalman smoother based VTR tracker. Recall that we have set the phone-independent residual vector $\mathbf{r} = 0$ for simplicity in the current VTR tracker. However this is unnecessary and the residual vector can be trained either on a pre-selected training set or on-the-fly to further improve the VTA mapping and the quality of the VTR tracker simultaneously. The phone independent $\mathbf{r}$ may be further modeled as a random variable (thus subsumes the Gaussian noise $\mathbf{v}$) with a Gaussian mixture distribution to facilitate high-accuracy training. Such an improvement has indeed been carried out at Microsoft Research after the author's original work and appears in [63].

- Third it has been hinted a few times that the ultimate goal of this thesis is to use HDM towards speech recognition, the most difficult and comprehensive speech processing application so far. However, this also involves huge difficulty and complexity, as will become clear in the next part of the thesis dedicated to such a challenge. Such a difficulty is another factor motivating the use of VTR residual in a HMM based phone recognizer described in Section 7.4. The goal is more to resort to a relatively easy way to demonstrate *some* power of the HDM for speech recognition rather than plain uses of VTR related features to improve HMM based speech recognition systems as has been previously attempted [111, 217, 246]. Notice that the residual feature is the result of a simplified HDM used for VTR tracking. The fact that it can still provide *some* improvement over a high performance HMM baseline system makes us strongly believe that a full-fledged HDM based speech recognizer may provide considerable improvement over state-of-the-art HMM systems.

Therefore, in spite of the huge challenge, it is with great anticipation and excitement to proceed to the next and final part of the thesis where efforts are made to develop HDM based speech recognition systems.

Part III

Algorithm Development of HDM towards ASR

Chapter 8

Introduction to Automatic Speech Recognition (ASR)

This chapter introduces the speech recognition problem, reviews the hidden Markov model (HMM) which is at the core of current speech recognition systems, and motivates the use of hidden dynamic models (HDMs) as a promising new approach.

8.1 The Formulation of ASR Problem

Obviously the goal of automatic speech recognition (ASR) is to let a machine recognize what a person is saying, i.e., given the speech signal as input, we want to have the content of speech (typically sentences) as output. It may also be thought of as a machine acting as a voice-actuated "typewriter". It is necessary to formulate the problem mathematically before a computer can solve it. There could be a number of different ways to formulate the speech recognition problem, but the most successful and by far the most popular one is to treat the speech signal as a stochastic pattern and adopt a statistical pattern classification approach [69, 82].

Suppose a sequence of acoustic observations (feature vectors) $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T\}$ is available, in order to recover the underlying word string $\mathbf{W} = \{w_1, w_2, \ldots, w_N\}$, the

recognizer picks up the *most likely* word string given the observed acoustic evidence, i.e.,

$$\hat{\mathbf{W}} = \underset{\mathbf{W}\in\Omega}{\operatorname{argmax}}\, P(\mathbf{W} \mid \mathbf{O}), \tag{8.1}$$

where Ω is the set of all possible word sequences. The well-known Bayes' formula in probability theory allows us to re-write the right hand side of (8.1) as

$$P(\mathbf{W} \mid \mathbf{O}) = \frac{P(\mathbf{W})P(\mathbf{O} \mid \mathbf{W})}{P(\mathbf{O})} \propto P(\mathbf{O} \mid \mathbf{W})P(\mathbf{W}). \tag{8.2}$$

The last part of the above equation follows since $P(\mathbf{O})$ is irrelevant to the maximization procedure. After this simple manipulation, (8.1) can be rewritten as

$$\hat{\mathbf{W}} = \underset{\mathbf{W}\in\Omega}{\operatorname{argmax}}\, P(\mathbf{W})P(\mathbf{O} \mid \mathbf{W}), \tag{8.3}$$

which is sometimes referred to as the *fundamental equation* of speech recognition. This important equation also indicates what processes and components are of concern in the design of a speech recognizer. The key components of a speech recognizer is briefly described as follows.

8.1.1 Speech Preprocessing

Speech preprocessing refers to the computation of acoustic feature vectors $\mathbf{O}$, which serve as input to the statistical pattern classifier (8.3). For speech recognition, and many other speech applications, it is not a good idea to work with raw speech waveforms directly. The speech waveform simply contains too *much* (mostly redundant) information with a very high variability (even for repetitions of the same utterance spoken by the same speaker under identical acoustic environment) so that it is very difficult to extract useful information for the purpose of speech recognition (or other applications). Therefore, the goal of speech preprocessing is to keep the application-dependent, relevant information while removing as much redundant information as possible, and hopefully also achieve good robustness under different acoustic environment. Most preprocessing methods are based on some understanding of the human speech production and perception mechanism, and important ideas (such as short-time analysis) have been introduced in Chapter 2. For speech recognition in particular, LPC-derived cepstral coefficients (LPC-cep) account for early success

of speech recognizers; more recently mel-frequency cepstral coefficients (MFCCs) become the dominant choice in state-of-the-art speech recognizers, and very recently perceptual linear prediction (PLP) coefficients have also been demonstrated to have similar or slightly better performance than MFCCs in many speech recognition tasks. This area is still under active research to find more suitable acoustic features for speech recognition applications. Section 7.4, where VTR residual is used as an appended feature to improve a HMM phone recognizer, is an example of research work in this direction.

8.1.2 Acoustic Modeling

The fundamental equation of speech recognition (8.3) naturally breaks a speech recognizer into two parts. One is to calculate the prior probability of word strings $P(\mathbf{W})$, the other is to calculate the probability of a word string given the acoustic evidence $P(\mathbf{W} \mid \mathbf{O})$. The first quantity $P(\mathbf{W})$ is obtained by a language model, to be described shortly; the second quantity $P(\mathbf{W} \mid \mathbf{O})$ is obtained by an acoustic model. Arguably, acoustic modeling is the central part of a speech recognizer. The dominant choice for acoustic modeling today is hidden Markov model (HMM) and its extensions. An overview of HMM and its use as an acoustic model in speech recognition will be provided in Section 8.2.

Besides HMM, template-based dynamic time warping [198] has been used in the early days and performed well for simple applications, such as isolated-word recognition under small-vocabulary and speaker-dependent constraints. But it was abandoned later due to the difficulty of generalizing this method for large-vocabulary speech recognition tasks. Artificial neural networks (ANN) are a relatively new approach (comparing to HMM) that started catching people's interest in late 1980's [23, 241]. Such an approach has yielded good performance for small-vocabulary speech recognition tasks, and even out-performed HMM for short, isolated speech units [208], but it remains problematic to apply neural networks to large-vocabulary continuous speech recognition tasks. The main challenge is how to incorporate the dynamic behavior of speech into a (usually static) neural network structure without introducing prohibitive computational requirements. So far the most effective solution seems to be integrating neural nets with HMMs [170, 260], such as replacing the Gaussian mixture densities in standard HMMs with a neural net, and such an approach essentially becomes another extension of HMM. More recently there has been a

growing body of research [17, 59, 86, 91, 169, 267] to go beyond the conventional HMMs, and an important family of them is hidden dynamic models (HDMs) [206, 148], which is the focus of this thesis. More comprehensive review and detailed algorithm development of HDMs for speech recognition will be presented in the following sections and chapters.

8.1.3 Language Modeling

Language models are intrinsically language-specific, quite often they are also domain or application specific. There are two main routes to take in building a language model, one is knowledge or rule based and the other is data-driven.

The rule-based approach is rooted in linguistics, such as Chomsky's *formal language theory* [38] for English. The most popular one is the context free grammar (CFG) [127] due to its good balance of expressing capacity and parsing efficiency. These rule-based language models typically either accept or reject a hypothesized utterance based on the parsing result, but sometimes the output can be probabilistic as well. When used in a limited-domain with well-defined semantics, rule-based language models can be very powerful, but they are typically inadequate for large-vocabulary spontaneous speech recognition tasks due to their limited coverage.

On the other hand, data-driven language models unexceptionally adopt a probabilistic view and need to be trained from data, just like acoustic models. However, unlike training acoustic models, no speech data is needed; text (written material) alone is sufficient. To calculate $P(\mathbf{W})$, we first use Bayes' formula to decompose it in a sequential order:

$$P(\mathbf{W}) = \prod_{i=1}^{N} P(w_i \mid w_1, \ldots, w_{i-1}). \tag{8.4}$$

Of course it is unnecessary and computationally infeasible for the current word to depend on the entire past history, and the popular *n-gram* language models make the approximation that the current word only depends on the identity of its previous n words. The most popular and surprisingly powerful language model for large-vocabulary speech recognition is the *trigram* language model [120], which makes the approximation

$$P(w_i \mid w_1, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-2}, w_{i-1}). \tag{8.5}$$

Bigram and *unigram* (or *monogram*) language models are also used when there is no sufficient data to train a trigram language model or efficient hypothesis search (to be presented next) is required. The main drawback of n-gram language models is that long-span dependencies in sentence structures cannot be efficiently captured (large n typically enforces prohibitive computational cost).

There are also active research efforts to unify the knowledge-based and data-driven language models to combine their strengths while overcoming their respective limitations [35, 244].

8.1.4 Hypothesis Search

Finally we need to search over all possible word strings to find the most likely one, which is commonly known as the *decoding* problem in speech recognition. A little thought would convince the reader that brute force approach is impossible: the space of $\mathbf{W}$ is astronomically large even for a small task, and in general the possible number of word strings is infinite. Therefore, the framework under which the acoustic model and language model are constructed must allow efficient search algorithms to be carried out. This is again greatly facilitated by the state-of-the-art HMM framework, where the powerful and efficient Viterbi decoding algorithm (to be presented shortly) provides the basis for most sophisticated searching algorithms used in today's speech recognizers. An alternative approach rooted in artificial intelligence, known as *stack decoding* or *A* search* [105, 173], has also been successfully applied in some speech recognizers. Any new developments in either acoustic modeling or language modeling must also take into account the decoding problem seriously.

8.2 Overview of the Hidden Markov Model (HMM)

This section reviews the hidden Markov model (HMM) from two different aspects. The first one is a classic view that fueled the widespread adoption of HMM in speech recognition since 1980's [195, 196, 197]. The second one is under the general framework of *graphical models* [16, 18, 123], which succinctly summarizes the probabilistic features of HMM and conveniently relates it to many other statistical models, including the hidden dynamic

model (HDM) which is at the heart of this thesis. The somewhat detailed description of HMM algorithms is to facilitate later comparison with HDM algorithms, to be developed in the next chapter.

8.2.1 A Classic View of HMM

A HMM is a stochastic function of a first-order discrete-time Markov chain[1]. It is composed of two components: a Markov process on the hidden (or latent or unobservable) states s and a set of output probabilities associated with each state. A HMM is fully characterized by

- A finite alphabet $\mathcal{S} = \{1, 2, \ldots, N\}$ on which the hidden states take values. The total number of possible states is N, while there is also an unique starting state $s_0 = 0$, which is included for implementation convenience to be explained shortly.
- A time-invariant transition matrix $\mathbf{\Pi}$, where

$$\pi_{ij} = p(s_{t+1} = j \mid s_t = i). \tag{8.6}$$

- A set of output distributions

$$p(\mathbf{y} \mid s_t = s), \quad s \in \mathcal{S}, \tag{8.7}$$

  associated with each state, where $p(\cdot)$ can be any valid probability distributions, typically vector-valued. Popular choices include discrete distributions, Gaussian distribution and mixtures of Gaussian distributions.

Some authors also include an initial state distribution in specifying a HMM, but such a need is eliminated by making use of the special non-emitting state s_0 that doesn't output any observable data $\mathbf{y}$. The model always starts in s_0 and never returns. Hence, the entire first column of $\mathbf{\Pi}$ is filled with zeros while the first row stores the probabilities of starting in a "real" state s_j. Another advantage of using a special starting state is that it facilitates the concatenation of HMMs (as is commonly done in speech recognition), where the last

[1]First-order Markov chains are covered in standard statistics textbooks, such as [181, 209].

state of the previous HMM can just serve as the special starting state for the current HMM. The length T of the generated samples of time series $\mathbf{Y}_{1:T} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ can either be fixed, or more generally, be a random variable itself. For example, one way to generate a variable length observable sequence is to choose one of the possible states (usually the last one N) as the ending state. Usually the number of hidden states N is predetermined while the remaining model parameters, collectively denoted as $\mathbf{\Theta}$ which include the transition matrix $\mathbf{\Pi}$ and parameters associated with the output distributions, are learned from data.

Given the form of a HMM, there are three basic problems that have to be solved before applying it in statistical modeling. The solution to these problems leads to three elegant and well-known algorithms in speech recognition and beyond, which are first developed by Baum and colleagues in the late 1960s and early 1970s [11, 12, 13, 14]. The problems and their solutions are described as follows [196, 197]:

Problem 1: Given an observation sequence $\mathbf{Y}_{1:T} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ and all the model parameters $\mathbf{\Theta}$, how to efficiently calculate $P(\mathbf{Y}_{1:T} \mid \mathbf{\Theta})$, the probability of the observation sequence?

This is essentially to calculate the total likelihood of the data, or commonly known as the *likelihood score* or simple the *score* in speech recognition. The brute-force way to solve this problem is by first enumerating all the possible state sequences with length T (there are N^T such sequences), and then calculate the probability of the observation sequence coming from every possible state sequence. This approach can be expressed as

$$\begin{aligned} p(\mathbf{Y}_{1:T} \mid \mathbf{\Theta}) &= \sum_{\text{all } \mathbf{s}} p(\mathbf{Y}_{1:T}, \mathbf{s}_{1:T} \mid \mathbf{\Theta}) \\ &= \sum_{s_1, s_2, \cdots, s_T} p(s_1|s_0)p(\mathbf{y}_1|s_1)p(s_2|s_1)p(\mathbf{y}_2|s_2)\cdots p(s_T|s_{T-1})p(\mathbf{y}_T|s_T), \end{aligned} \tag{8.8}$$

where statistical independence among observations given the states is asserted. A little thought would reveal that such a brute-force approach will lead to a computational load exponential in T (on the order of $2T \cdot N^T$), which is infeasible in practice. However, huge computational savings can be gained by carefully making use of the conditional independence relationships suggested by the model. For example, when $T = 2$, the total likelihood

of the data can be calculated as

$$\begin{aligned} p(\mathbf{Y}_{1:2} \mid \mathbf{\Theta}) &= \sum_{s_1,s_2} p(s_1|s_0)p(\mathbf{y}_1|s_1)p(s_2|s_1)p(\mathbf{y}_2|s_2) \\ &= \sum_{s_2} p(\mathbf{y}_2|s_2) \cdot \left[\sum_{s_1} p(s_1|s_0)p(\mathbf{y}_1|s_1)p(s_2|s_1) \right]. \end{aligned} \tag{8.9}$$

This simple idea of "pushing" the sums in as far as possible so that only relevant terms get summed up lies at the heart of the *forward-backward* algorithm, which efficiently computes the total likelihood (8.8) linearly in T (on the order of N^2T). Moreover, it is also the trick behind efficient exact inference algorithms for graphical models in general, including the junction tree algorithm [45, 121, 187], the generalized distributive law [3] and the sum-product algorithm [142].

The complete forward-backward algorithm is described as follows [197]. We first define the forward variable $\alpha_t(i)$

$$\alpha_t(i) = p(\mathbf{Y}_{1:t}, s_t = i \mid \mathbf{\Theta}), \tag{8.10}$$

and compute it inductively as follows.

1. Initialization:

$$\alpha_1(i) = \pi_{0i} \cdot p(\mathbf{y}_1 \mid s_1 = i), \quad 1 \le i \le N. \tag{8.11}$$

2. Induction: for $t = 2, \ldots, T$

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\pi_{ij} \right] p(\mathbf{y}_t \mid s_t = j), \quad 1 \le j \le N. \tag{8.12}$$

3. Termination:

$$p(\mathbf{Y}_{1:T} \mid \mathbf{\Theta}) = \sum_{i=1}^{N} \alpha_T(i). \tag{8.13}$$

Notice how the idea of pushing sums in has been applied in the induction step. The forward variable $\alpha_t(i)$ also has a neat explanation: it is the probability of observing the partial sequence $\mathbf{Y}_{1:t}$ and being in state i at time t. Although the likelihood evaluation problem

is completely solved by this forward pass alone, it is convenient to define a symmetric backward pass to be used later. The backward variable $\beta_t(i)$ is defined as

$$\beta_t(i) = p(\mathbf{Y}_{t+1:T} \mid s_t = i, \mathbf{\Theta}), \tag{8.14}$$

and also computed inductively.

1. Initialization:
$$\beta_T(i) = 1, \quad 1 \le i \le N. \tag{8.15}$$

2. Induction: for $t = T-1, \ldots, 1$
$$\beta_t(i) = \sum_{j=1}^{N} \pi_{ij} p(\mathbf{y}_{t+1} \mid s_{t+1} = j) \beta_{t+1}(j), \quad 1 \le j \le N. \tag{8.16}$$

$\beta_t(i)$ has a clear meaning too: it is the probability of observing the partial sequence $\mathbf{Y}_{t+1:T}$ given the system is in state i at time t. Both the forward pass and backward pass require computation linear in T, on the order of N^2T.

Problem 2: Given an observation sequence $\mathbf{Y}_{1:T}$ and the model $\mathbf{\Theta}$, how to choose an optimal state sequence that best explains the observation?

Different criteria of optimality will lead to different solutions. For speech recognition, the most sensible criteria is to choose the state sequence that maximizes the posterior probability $p(\mathbf{s} \mid \mathbf{Y}_{1:T}, \mathbf{\Theta})$, *i.e.*, we want to find the state sequence $\hat{\mathbf{s}}$ such that

$$\begin{aligned} \hat{\mathbf{s}} &= \max_{\text{all } \mathbf{s}} p(\mathbf{s} \mid \mathbf{Y}_{1:T}, \mathbf{\Theta}) = \max_{\text{all } \mathbf{s}} p(\mathbf{Y}_{1:T}, \mathbf{s}_{1:T} \mid \mathbf{\Theta}) \\ &= \max_{s_1, s_2, \cdots, s_T} p(s_1|s_0) p(\mathbf{y}_1|s_1) p(s_2|s_1) p(\mathbf{y}_2|s_2) \cdots p(s_T|s_{T-1}) p(\mathbf{y}_T|s_T). \end{aligned} \tag{8.17}$$

Notice the similarity between equations (8.8) and (8.17): the only difference is that the summation operation in (8.8) has been replaced by the maximization operation in (8.17). Due to many properties shared by both the summation and maximization operation, conditional independence relationships can be explored similarly: instead of pushing the "sum" in, we push the "max" in this time. This results in the efficient *Viterbi* algorithm with computational cost linear in T, which is stated below.

To apply the Viterbi algorithm to find the optimal state sequence (or the best path), we first define

$$\delta_t(i) = \max_{\mathbf{s}_{1:t}} p(\mathbf{Y}_{1:t}, \mathbf{s}_{1:t-1}, s_t = i \mid \mathbf{\Theta}), \tag{8.18}$$

which is the highest likelihood ending in state i at time t. This quantity as well as the best path is computed inductively as follows.

1. Initialization:

$$\delta_1(i) = \pi_{0i} \cdot p(\mathbf{y}_1 \mid s_1 = i), \quad 1 \le i \le N, \tag{8.19a}$$

$$\phi_1(i) = 0, \quad 1 \le i \le N. \tag{8.19b}$$

2. Induction: for $t = 2, \ldots, T$

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i)\pi_{ij}\right] p(\mathbf{y}_t \mid s_t = j), \quad 1 \le j \le N, \tag{8.20a}$$

$$\phi_t(j) = \operatorname*{argmax}_{1 \le i \le N} \left[\delta_{t-1}(i)\pi_{ij}\right], \quad 1 \le j \le N. \tag{8.20b}$$

3. Termination:

$$\hat{p} = \max_{1 \le i \le N} \left[\delta_T(i)\right], \tag{8.21a}$$

$$\hat{s}_T = \operatorname*{argmax}_{1 \le i \le N} \left[\delta_T(i)\right]. \tag{8.21b}$$

4. Backtracking: for $t = T-1, \ldots, 1$

$$\hat{s}_t = \phi_{t+1}(\hat{s}_{t+1}). \tag{8.22}$$

The Viterbi algorithm may also be viewed as an instance of applying dynamic programming technique [44] to HMM. A further note is that in practice, typically both the forward-backward algorithm and the Viterbi algorithm are implemented in the log domain to avoid numerical underflow, because we need to multiply a (potentially very long) sequences of small numbers (the probabilities) in both cases.

Problem 3: Given a set of observation sequences $\mathbf{Y}_{1:T}$, how to adjust the model parameters $\mathbf{\Theta}$ to maximize $P(\mathbf{Y}_{1:T} \mid \mathbf{\Theta})$?

This is also known as the training problem, which is by far the most difficult. It has been shown that there is no optimal way of globally optimizing all the parameters given a finite observation sequence [196]. However, there exists an iterative procedure known as the *Baum-Welch* algorithm, which is a special case of the more general Expectation-Maximization (EM) algorithm [54, 164, 245], to find the local optima of HMM parameters. The rather involved formal description of the Baum-Welch algorithm will not be presented, only basic ideas are sketched here. However, a formal and insightful description of the EM algorithm, under a very general context, will be presented in the next chapter when developing algorithms for HDM.

The basic idea of Baum-Welch algorithm (as well as the more general EM algorithm) is to start with some initial guess of model parameters and use the forward-backward algorithm to calculate sufficient statistics from the data (**E step**). The model parameters are then reestimated from the calculated sufficient statistics based on the maximum likelihood criterion (**M step**). For example, the state transition probability π_{ij} is estimated by

$$\pi_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \tag{8.23}$$

where

$$\begin{aligned} \xi_t(i,j) &= p(s_t = i, s_{t+1} = j \mid \mathbf{Y}_{1:T}, \mathbf{\Theta}) \\ &= \frac{\alpha_t(i)\pi_{ij}p(\mathbf{y}_{t+1}|s_{t+1} = j)\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\pi_{ij}p(\mathbf{y}_{t+1}|s_{t+1} = j)\beta_{t+1}(j)}, \end{aligned} \tag{8.24}$$

and

$$\begin{aligned} \gamma_t(i,j) &= p(s_t = i \mid \mathbf{Y}_{1:T}, \mathbf{\Theta}) \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}, \end{aligned} \tag{8.25}$$

are both quantities that can be directly calculate based on the forward-backward algorithm. Parameter estimation formulas for the output distribution will depend on the functional form of the output distribution used in the model and the details are omitted.

Next, iterations are carried out between the E step and M step. It has been proven that both the E step and the M step guarantee to improve (or not to decrease) total likelihood of the data. Iteration stops when the total likelihood converges, or may do so after a fixed number of iterations in practice. Although the Baum-Welch algorithm can only reach a local optimum, experiences show that it is very effective at training HMMs: it usually requires only a few iterations to converge and the final result (in terms of the modeling power, or more specifically for speech recognition, in terms of word error rates) is largely insensitive to parameter initialization.

The solutions to the above three basic problems allow the highly efficient use of HMM as a generic statistical model in time series analysis, both for speech recognition and other applications, such as modeling DNA sequences in bioinformatics [72].

8.2.2 HMM as a Probabilistic Graphical Model

This section recasts HMM in the general framework of graphical models [123]. Probabilistic graphical models are graphs where nodes represent random variables and the (lack of) arcs represent conditional independence assumptions. They generally offer a compact representation of joint probability distributions, as will become clear shortly. Graphical models are the result of a merge between probability theory and graph theory: the probability theory provides the glue whereby nodes and substructures are combined, ensuring that the system as a whole is consistent; the graph theory provides both an intuitively appealing interface where humans can visualize interactions among different model components as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms. Almost all statistical models of practical interest can be put under the general graphical model formalism in a straightforward manner, and graphical models are playing an increasingly important role in the design and analysis of machine learning algorithms.

There are two main kinds of graphical models based on the use of arcs: undirected and directed. Undirected graphical models, also known as Markov random fields [36], are more popular in physics and image processing. Directed graphical models, also known as Bayesian networks [121], are more popular with the artificial intelligence and machine learning community. This thesis only concerns directed graphical models or Bayesian networks. In a Bayesian network, an arc from node A to B can be informally interpreted

as "A causing B", thus directed cycles are disallowed.

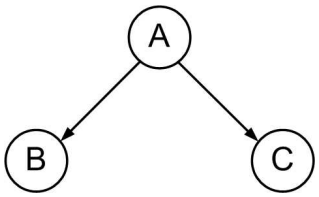

Figure 8.1: A simple Bayesian network.

Fig. 8.1 shows a simple Bayesian network. The graph encodes the conditional independence statement "B is conditionally independent of C given A" because of the lack of an arc between nodes B and C. This is usually written as $B \perp\!\!\!\perp C \mid A$. Based on this conditional independence relationship, the joint probability of the model can be expressed as

$$p(A, B, C) = p(A) \cdot p(B \mid A) \cdot p(C \mid A). \tag{8.26}$$

In general, a given graphical model is equivalent to a list of conditional independence statements, and a graph represents all distributions for which all these independence statements are true. A central task in applying statistical models is to perform *probabilistic inference*, *i.e.*, to compute the probability of a subset of random variables given values (or samples or data) of another subset. Inference is essential both to make predictions/decisions based on the model and to learn model parameters under the EM framework. It turns out that the difficulty of performing exact inference for a particular statistical model is determined by its underlying conditional independence structure, or equivalently, its representation as a graphical model.

With the above quick introduction to graphical models, it may be easily verified that a HMM can be represented as a Bayesian network in Fig. 8.2. Following conventions in graphical models, the observable random variables $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T$ have been drawn as shaded nodes. Note that Fig. 8.2 is *not* a complete representation of a HMM, *e.g.*, there is no way to represent the transition matrix $\mathbf{\Pi}$ or any special structures associated with it (such as the upper triangular structure typically used in speech recognition). In fact, Fig. 8.2 can represent more than a HMM. For example, if we allow the states $\mathbf{s}_{0:T}$ to be continuous random variables, Fig. 8.2 can also represent a state-space model (SSM),

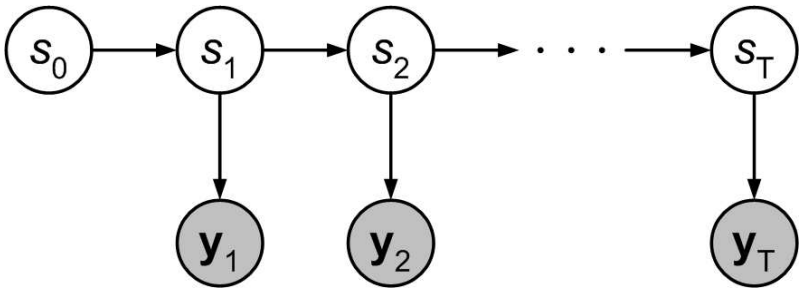

Figure 8.2: HMM represented as a Bayesian network.

also known as a stochastic linear dynamic system. Therefore, a graphical model usually represents no more than conditional independence relationships within a statistical model. However, this is not a disadvantage but an advantage instead in using graphical models since it is the conditional independence structure that largely determines the internal complexity of a statistical model. For example, the fact that efficient inference algorithms exist for both HMM and SSM (forward-backward for the former and Kalman smoother for the latter) is not a coincidence, it is because they share the same underlying graphical model representation. Therefore, we see how graphical model representations allow connections and insights to be gained when comparing different statistical models.

8.3 From HMM to HDM

HMM is a very flexible model. Without limit on the number of hidden states, the family of observation distributions, and the amount of training data from which model parameters are learned, it can account for practically any time series (or signals) arbitrarily well. However, this generality does not imply that it is a good model for a particular application. On the contrary, it suggests that by properly integrating domain-specific knowledge, alternative models may perform a lot better, achieving high accuracy with inherent parsimony and robustness. Previous chapters of the thesis establish HDM as such an example for speech modeling, and this is further illustrated in this section by comparing it to the HMM .

There is no doubt that an HMM adequately captures the discrete or symbolic nature of

speech, but it is inherently weak at capturing the continuous or dynamic nature of speech, *e.g.*, the coarticulation phenomena. This can be observed from a number of HMM modeling assumptions that are incompatible with the human speech production mechanism. For example, HMM assumes that successive observations within a state are independent and identically distributed (IID), each state has a constant mean and the states themselves only have a very short time correlation demanded by the Markov property. All these assumptions are not true for speech since the production of speech is actually driven by a highly time-correlated, continuous and smooth movement of the articulators. In fact, most recent improvements upon HMM have been dedicated to enhancing its ability to model speech dynamics, and a number of them are listed here.

- To relax the IID assumption between adjacent frames, delta and acceleration features, which are the first and second order differences of the original acoustic feature, are unexceptionally appended to the original feature in state-of-the-art HMM based ASR systems. Although this is reasonable in a sense and indeed effective at reducing recognition error rates, it is only a crude and heuristic way to model speech dynamics and also introduces additional problems [93].

- To better account for the coarticulation phenomena, current HMM systems adopt context-dependent phones as the basic speech unit, *e.g.*, the popular tri-phone unit is a phoneme conditioned on its immediate left and right phonemes [259]. However, there are some well-known long span context dependencies in speech that cannot be efficiently captured in such a manner.

- There are also a number of efforts to relax the constant mean assumption within a state, such as modeling it with a polynomial or an auto regressive function [56, 60, 203]. However, besides considerably increasing the complexity of training and test algorithms, the temporal dynamics at the surface acoustic level is also very noisy and difficult to extract.

There is a common theme underlying the above improvements of HMMs (and many others that are not listed): to improve model accuracy by increasing the number of learnable parameters. However, there are some practical limitations to this, *e.g.*, the computations

associated with HMMs grow quadratically with the number of states N (recall both the forward-back algorithm and Viterbi algorithm have time complexity on the order of N^2T), so while increasing the number of states is simple, such as moving from tri-phone to quin-phone *etc*, there is an appreciable associated computational cost, not to mention the need for more training data. In contrast, HDM takes a different route: it enriches the underlying model *structure* instead of simply increasing the number of model parameters. By modeling the internal dynamics of speech motivated by studies and insights from speech science, both short term and long term context dependencies are naturally incorporated within the continuous and smooth trajectories of the hidden dynamics. This is why we are able to achieve model accuracy and parsimony at the same time, as demonstrated by previous chapters of the thesis.

It should be further pointed out that, although model accuracy has been emphasized throughout the thesis and the above discussion, this is *not* the correct criterion to use when judging model quality for speech recognition[2]. The true criterion should be model discriminability, *i.e.*, how well a model can discriminate among different speech units, as is generally the case for any pattern recognition problems. This is also the reason why the most recent speech recognition systems have moved away from the traditional training criterion of ML or MAP (maximum a posterior) while using others that can directly improve model discriminability, such as minimum classification error (MCE) [40, 126, 125] and maximum mutual information (MMI) [7, 239, 256]. However, without a deep understanding of speech itself, it is hard to know what really constitutes the discriminable part of speech. Because most of the recent improvements on speech recognition have been a result of increasing model accuracy, and arguably HDM can provide more discriminability than HMM since HMM can be treated as a special case of HDM when used for ASR (will be illustrated in the next chapter), we strongly believe that HDM is a very viable approach for speech recognition, although such a claim needs to be further backed up by experiments. Another advantage of using a more accurate generative or internal model for speech recognition (such as HDM versus HMM) is the ability of having a better confidence measure on the recognition results, which is also an important practical issue in deploying speech recognition technologies.

[2]It is if we were to perform speech synthesis.

Having established HDM as a more accurate and parsimonious model of human speech and a promising model for speech recognition, another equally important (if not more) question is whether efficient inference and model parameter learning algorithms can be developed for HDM similar to those of HMM. The remaining chapters of the thesis present original and fruitful efforts along this line.

Chapter 9

Algorithm Development for HDM

This chapter thoroughly describes the novel and powerful variational inference and parameter learning algorithms developed for a linear switching state space model (SSSM), which is formulated from the HDM proposed for speech applications, and backs them up with extensive simulation results.

9.1 A SSSM Formulated from HDM

To facilitate algorithm development, the following linear switching state space model (SSSM),

$$\mathbf{x}_t = \mathbf{A}_s \mathbf{x}_{t-1} + \mathbf{a}_s + \mathbf{w}_s, \tag{9.1a}$$

$$\mathbf{y}_t = \mathbf{C}_s \mathbf{x}_t + \mathbf{c}_s + \mathbf{v}_s, \tag{9.1b}$$

based on the proposed HDM for speech described in Section 7.2.1, is formulated and focused in this chapter. Here, $\mathbf{A}_s$, $\mathbf{a}_s$, $\mathbf{C}_s$ and $\mathbf{c}_s$ are s-dependent deterministic parameter while $\mathbf{w}_s$ and $\mathbf{v}_s$ are Gaussian white noises whose covariance matrices are also s-dependent.

This SSSM has a couple of small differences compared to the HDM of equation (7.8). First the piecewise linear mapping $\mathcal{P}[\cdot]$ has been replaced by a pure linear mapping. This is mostly done for simplicity and the implication of using a piecewise linear mapping instead of a pure linear one with the variational algorithms to be developed will be discussed in the next chapter. Second the effect of target $\mathbf{u}_s$ is embedded within the bias term $\mathbf{a}_s$ of the

state equation. This is mathematically equivalent as long as matrix $\mathbf{A}_s$ is convergent, and we can recover $\mathbf{u}_s$ by: $\mathbf{u}_s = (\mathbf{I}-\mathbf{A}_s)^{-1}\cdot\mathbf{a}_s$. We may further simplify the appearance of (9.1) to get rid of $\mathbf{a}_s$ and $\mathbf{c}_s$ by augmenting the continuous state vector $\mathbf{x}_t$ and observation vector $\mathbf{y}_t$. However, this is not done since both $\mathbf{a}_s$ and $\mathbf{c}_s$ have clear and significant meanings in the original speech model ($\mathbf{a}_s$ embeds the target while $\mathbf{c}_s$ corresponds to the residual vector). A further note is that both s and $\mathbf{x}$ are termed "states" in this model; when we need to distinguish between them, we refer to s as the discrete state and $\mathbf{x}$ as the continuous state. The remainder of this section will first offer some further illustrations on model structure and applications of the model, then review algorithms previously developed for the model (including some closely related variants) and motivate the development of new algorithms.

9.1.1 Detailed Model Description

The most complete and accurate way of describing a statistical model is by writing down its (joint) probability distributions, which is also the most convenient form to use when deriving statistical algorithms. The probability distributions associated with the SSSM are

$$\begin{aligned} p(\mathbf{Y}_{1:T}, \mathbf{X}_{0:T}, \mathbf{s}_{0:T}) &= p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}, \mathbf{s}_{1:T}) \cdot p(\mathbf{x}_0, s_0) \\ &= \prod_{t=1}^{T} p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t) p(s_t \mid s_{t-1}) \cdot p(\mathbf{x}_0 \mid s_0) p(s_0), \end{aligned} \tag{9.2}$$

where

$$p(\mathbf{y}_t \mid \mathbf{x}_t, s_t = s) = \mathcal{N}(\mathbf{y}_t \mid \mathbf{C}_s\mathbf{x}_t + \mathbf{c}_s, \mathbf{D}_s), \tag{9.3a}$$

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t = s) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{A}_s\mathbf{x}_{t-1} + \mathbf{a}_s, \mathbf{B}_s), \tag{9.3b}$$

$$p(s_t = j \mid s_{t-1} = i) = \pi_{ij}, \tag{9.3c}$$

with initial conditions,

$$s_0 = 0, \tag{9.4a}$$

$$p(\mathbf{x}_0 \mid s_0 = 0) = \mathcal{N}(\mathbf{x}_0 \mid \mathbf{a}_0, \mathbf{B}_0), \tag{9.4b}$$

and

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\rho}, \boldsymbol{\Gamma}) = \left|\frac{\boldsymbol{\Gamma}}{2\pi}\right|^{\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\rho})^T\boldsymbol{\Gamma}(\mathbf{x}-\boldsymbol{\rho})\right], \tag{9.5}$$

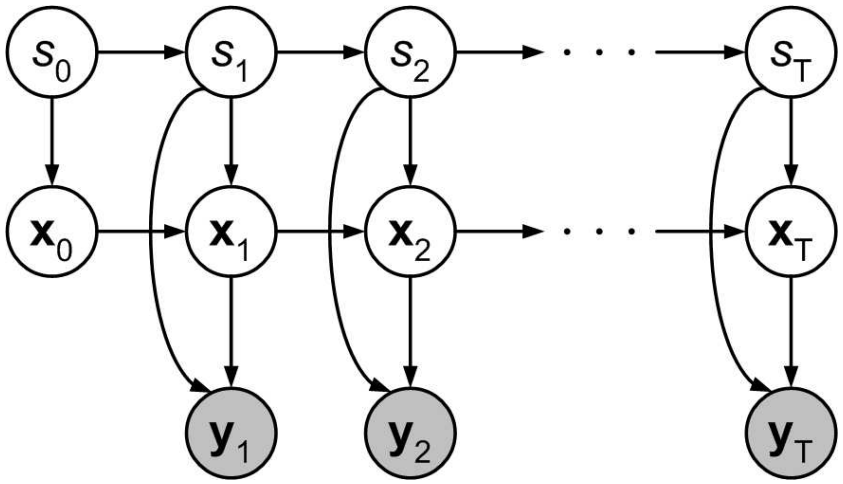

Figure 9.1: The SSSM represented as a Bayesian network.

denotes a Gaussian distribution with mean $\boldsymbol{\rho}$ and *precision* matrix $\boldsymbol{\Gamma}$, the inverse of the more commonly used covariance matrix. Note that a unique initial discrete state s_0 is used to facilitate implementation and SSSM concatenation, similar to its use in HMM. The remaining discrete states take values from a finite alphabet $\mathcal{S} = \{1, 2, \ldots, N\}$, *i.e.*, the total number of possible discrete states is N. Similarly, a special initial state $\mathbf{x}_0$ has also been defined for the continuous state. For inference purposes, both s_0 and $\mathbf{x}_0$ can be considered as known or observed. The model can also be represented as a directed graphical model or Bayesian network, shown in Fig. 9.1, where the conditional independence relationships within the model can be easily visualized.

Compared to Fig. 8.2, Fig. 9.1 also clearly demonstrates that the SSSM generalizes both the hidden Markov model (HMM) and the linear state space model (SSM); it is a probabilistic dynamic system which combines discrete and continuous dynamics. Whereas the state space model describes an observed time series in terms of a continuous hidden state vector whose dynamics is specified by the dependence of the current state on the previous one, in the SSSM those dynamics depend on additional states which are discrete. Hence, the dynamics generally vary with time, producing a structurally-rich and powerful model for time series analysis with a broad range of applications. Besides speech, the SSSM (including a number of variants that are slightly different than the one defined here)

has been applied in diverse domains such as control [8, 9], machine vision [185, 186], fault diagnosis [154], financial analysis [99, 132] and other signal processing applications [202].

In the machine learning community, the SSSM belongs to a class of models termed conditional linear Gaussian (CLG) models [153, 155], which has also been attracting research interest recently. CLG models are probabilistic models consisting of discrete and continuous variables (or nodes in the context of graphical models), where a discrete node may have both discrete and continuous children, but a continuous node may only have continuous children. Each continuous node in a CLG is a Gaussian conditioned on its parent nodes, *i.e.*, its mean and covariance matrix depend on the discrete parents, with the mean being linear in the continuous parents. The SSSM formulated here can also be viewed under the Bayesian framework even without putting prior distributions on model parameters[1] (and we will not do so in this thesis). Our model implicitly forces a continuity constraint on $\mathbf{X}_{1:T}$, and it may serve as a smoothness prior for $\mathbf{Y}_{1:T}$ [89, 136, 140].

9.1.2 Review of Previously Developed Algorithms

Whereas inference and parameter estimation in HMM and the state space model can be done exactly using the EM algorithm (known as Baum-Welch for the former and Kalman-Rauch for the latter), it is well known that inference in SSSM is computationally intractable [90]. If $\mathbf{x}_0$ starts with a single Gaussian, then the exact posterior $p(\mathbf{x}_t \mid \mathbf{Y}_{1:t})$ at time t will be a mixture of Gaussians having N^t components, *i.e.*, the number of mixtures grows exponentially with the length of the time series T. More generally, it has been shown that inference in CLG models is NP-hard; moreover, unless P=NP, approximate inference with a guaranteed nontrivial accuracy is also intractable [153, 155]. Therefore, although the use of SSSMs may seem natural in many applications, the computational challenge is huge.

Nevertheless, there have been fair a number of approximate inference and/or parameter estimation algorithms developed in the past. Most of the earlier algorithms are based on heuristically collapsing the exact posterior, which follows a mixture of Gaussians distribution, into a single Gaussian at each time step [2, 34, 132, 222], thus avoiding the exponential growth of the number of mixtures. Besides being heuristic, most of these

[1] Please be cautioned that the name "Bayesian network" has nothing to do with Bayesian inference.

algorithms also don't provide an efficient solution to recover the optimal sequence of discrete hidden states, *i.e.*, the decoding problem in speech recognition. Variational methods have been introduced to SSSMs recently [90, 184], which are also based on collapsing the Gaussian mixtures of the state posterior into a single Gaussian at each time step, but in a more principled way. The new variational approximation developed in this thesis aims to achieve the following two goals:

1. To preserve the multi-modality of the exact posterior on hidden continuous states $\mathbf{x}$, but to keep the number of mixtures from growing exponentially.

2. To develop a set of efficient algorithms which are parallel to those of HMM, including one for the highly challenging decoding problem, to facilitate the use of SSSM in speech recognition.

Since variational methods are relatively new in machine learning and may not be familiar to many readers, an introduction to these methods is first provided in the next section, before presenting our new variational approach for SSSM.

9.2 Introduction to Variational Methods

The term "variational methods" can mean different things, but in this thesis it is restricted to the scope of inference and learning for probabilistic graphical models. Under such a context, the basic idea is first introduced to machine learning by Saul *et al.* [215, 216], with a root in statistical physics [116, 182]. Variational methods offer a powerful and elegant tool to tackle otherwise intractable problems, and can significantly outperform other approximation techniques, such as sampling based methods, in some situations. However, current understanding of this method is still quite limited in general, and very often there is as much "art" as there is "science" when applying it to solve a specific problem [124, 242]. Therefore, this self-contained tutorial will mainly focus on helping the readers understand the techniques used in solving our problem at hand, rather than discussing its general use for graphical models. To achieve this goal, an insightful comparison between variational EM and exact EM algorithms is first presented, then a simple but interesting example is described to help readers develop more insight into variational approximations.

9.2.1 Variational EM versus Exact EM

The EM algorithm [54, 164, 212] provides a powerful and general framework to estimate model parameters with the presence of *incomplete* data or hidden variables, usually in a ML sense. It can be applied using both exact inference and variational inference.

For an arbitrary statistical model, let us denote the observed data by $\mathbf{Y}$, the hidden variables by $\mathbf{X}$, and the model parameters to be estimated by $\mathbf{\Theta}$. Our goal is to estimate model parameters by maximizing the likelihood of the data

$$\mathcal{L}(\mathbf{\Theta}) = \log p(\mathbf{Y} \mid \mathbf{\Theta}) = \log \int_{\mathbf{X}} p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})\, d\mathbf{X}, \tag{9.6}$$

where $\mathbf{X}$ has been assumed to be continuous random variables in general. For discrete random variables, the integral in the above equation (9.6) will be replaced by a sum.

A lower bound of $\mathcal{L}$ can be obtained using *any* distribution over the hidden variables $\mathbf{X}$:

$$\begin{aligned} \log \int_{\mathbf{X}} p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})\, d\mathbf{X} &= \log \int_{\mathbf{X}} q(\mathbf{X}) \frac{p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})}{q(\mathbf{X})}\, d\mathbf{X} \\ &\geq \int_{\mathbf{X}} q(\mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})}{q(\mathbf{X})}\, d\mathbf{X}. \end{aligned} \tag{9.7}$$

To obtain the above lower bound (9.7), the concavity of the log function as well as Jensen's inequality, widely used in information theory [160], have been applied. We define this lower bound to be a functional[2] $\mathcal{F}$:

$$\begin{aligned} \mathcal{F}(q, \mathbf{\Theta}) &= \int_{\mathbf{X}} q(\mathbf{X}) \log \frac{p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})}{q(\mathbf{X})}\, d\mathbf{X} \\ &= \int_{\mathbf{X}} q(\mathbf{X}) \log p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})\, d\mathbf{X} - \int_{\mathbf{X}} q(\mathbf{X}) \log q(\mathbf{X})\, d\mathbf{X}. \end{aligned} \tag{9.8}$$

Note that $\mathcal{F}$ is the negative of the Kullback-Leibler (KL) distance or relative entropy between q and p. In statistical physics, $\mathcal{F}$ is also the negative of a quantity known as free energy: the expected energy under q minus the entropy of q [172].

Generally speaking, the EM algorithm alternates between maximizing $\mathcal{F}$ with respect to the distribution q and model parameters $\mathbf{\Theta}$ while fixing the other. Starting with some initial parameters $\mathbf{\Theta}_0$:

[2] Since it is a "function" of the function $q(\mathbf{X})$.

E-step:

$$q_k = \underset{q}{\operatorname{argmax}}\, \mathcal{F}(q, \boldsymbol{\Theta}_{k-1}), \tag{9.9}$$

M-step:

$$\boldsymbol{\Theta}_k = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\, \mathcal{F}(q_k, \boldsymbol{\Theta}). \tag{9.10}$$

Although in general we have to resort to calculus of variations [258] to find the optimal function q in the E-step, it is easy to verify by substitution that the maximum is reached when q is the posterior distribution of $\mathbf{X}$, *i.e.*, $q_k(\mathbf{X}) = p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\Theta}_{k-1})$. In such a case the lower bound becomes an equality: $\mathcal{F}(q_k, \boldsymbol{\Theta}_{k-1}) = \mathcal{L}(\boldsymbol{\Theta}_{k-1})$[3]. Maximizing $\mathcal{F}$ in the M-step only involves maximizing the first term of (9.8), since the second term (entropy of q) does not depend on $\boldsymbol{\Theta}$, *i.e.*, the modified M-step becomes:

M-step:

$$\boldsymbol{\Theta}_k = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \int_{\mathbf{X}} p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\Theta}_{k-1}) \log p(\mathbf{X}, \mathbf{Y} \mid \boldsymbol{\Theta}_{k-1})\, d\mathbf{X}. \tag{9.11}$$

This is the expression which appears most commonly in the EM literature, and the quantity to be maximized is popularly referred to as the $\mathcal{Q}$ function. Therefore, the exact EM algorithm consists of computing the conditional distribution (or posterior) $p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\Theta}_{k-1})$ (or its moments, also know as sufficient statistics) in the E-step and maximizing the $\mathcal{Q}$ function in the M-step. As a result, the likelihood of the data will be increased (or more accurately, not decreased) at each E step and M step.

But what if the conditional distribution $p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\Theta}_{k-1})$ is intractable? This is where the variational approximation comes into the play. Even if the exact posterior is intractable, we may approximate it by a *variational* posterior $q(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\Theta}_{k-1})$ that has a pre-designed tractable structure, *i.e.*, by asserting additional conditional independence relationships among the hidden variables (but not the functional form of q). In the E-step, q needs to be maximized in a true sense of calculus of variations to determine its functional form and parameters simultaneously[4]. It also turns out that such a free form optimization of q

[3]It is nevertheless possible to show by variational principles that the exact posterior indeed maximizes $\mathcal{F}$; such a proof is provided in Appendix A.1, where it will also become clear that optimizing $\mathcal{F}$ subsumes the famous Bayes' rule.

[4]If $\mathbf{X}$ solely contains discrete random variables, regular calculus suffices.

can be successfully carried out without computing the exact posterior or the exact joint distribution $p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta}_{k-1})$ explicitly. The M-step is again to maximize $\mathcal{F}$, or equivalently, only its first term,

$$\mathcal{F}'(\mathbf{\Theta}) = \int_{\mathbf{X}} q(\mathbf{X}) \log p(\mathbf{X}, \mathbf{Y} \mid \mathbf{\Theta})\, d\mathbf{X}. \tag{9.12}$$

Therefore, the variational EM algorithm is guaranteed to increase a lower bound of the data likelihood instead of the exact likelihood itself (which is also intractable to compute) at each EM iteration. Such a property constitutes the "science" part of variational methods. The "art" part is how to choose a sensible structure of q to achieve a tight lower bound (which indicates a good approximation) yet maintain computational efficiency. This is model-dependent and may require lots of experiences and insights into the problem.

The actual use of variational methods is further demonstrated by a simple example, where exact inference is also possible so that detailed comparisons can be made. Since variational EM and exact EM share essentially the same M step (only a slight difference in the objective function), the following example will focus on the inference or E step only.

9.2.2 An Illustrative Example

Here we develop a variational inference method for the linear state-space model (SSM), where exact inference is tractable and is well-known as the Kalman smoother. Therefore, detailed comparison between the variational inference and exact inference can be made to gain valuable insights on the use and effects of variational approximation. Similar approximation will also be used when developing variational inference for the switching state space model (SSSM), where exact inference is intractable.

Derivation of the Variational Smoother

We start with the model description, and the SSM is specified by the following equations:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{a} + \mathbf{w}, \tag{9.13a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{c} + \mathbf{v}, \tag{9.13b}$$

where $\mathbf{w}$ and $\mathbf{v}$ are zero mean Gaussian white noises.

To facilitate the derivation of variational inference, the model is further represented in terms of probability distributions:

$$\begin{aligned} p(\mathbf{Y}_{1:T}, \mathbf{X}_{0:T}) &= p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}) \cdot p(\mathbf{x}_0) \\ &= \prod_{t=1}^{T} p(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \cdot p(\mathbf{x}_0), \end{aligned} \tag{9.14}$$

where

$$p(\mathbf{y}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t \mid \mathbf{C}\mathbf{x}_t + \mathbf{c}, \mathbf{D}), \tag{9.15a}$$

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{A}\mathbf{x}_{t-1} + \mathbf{a}, \mathbf{B}), \tag{9.15b}$$

$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 \mid \mathbf{a}_0, \mathbf{B}_0). \tag{9.15c}$$

Although the exact posterior $p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})$ is tractable, we may still design a variational posterior $q(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})$ to approximate it. More specifically, we adopt the completely factorized form for the variational posterior

$$q(\mathbf{X}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{x}_t). \tag{9.16}$$

For notational simplicity the dependence of q on the complete observation or data $\mathbf{Y}_{1:T}$ will be omitted from now on, but whenever we use q it is always implied. The SSM and its variational posterior used in this example are also shown graphically in Fig. 9.2. At first glance it may seem that we have made a very crude approximation since all the correlations among $\mathbf{x}$'s have been dropped. However, such a view is somewhat misleading since the correlation will be indirectly accounted for by the variational approximation: the variational posterior $q(\mathbf{x}_t)$ at each time step will be determined by *all* the data. We will come back to this point when simulation results are analyzed.

The detailed derivation of variational inference is presented first. For this model, the

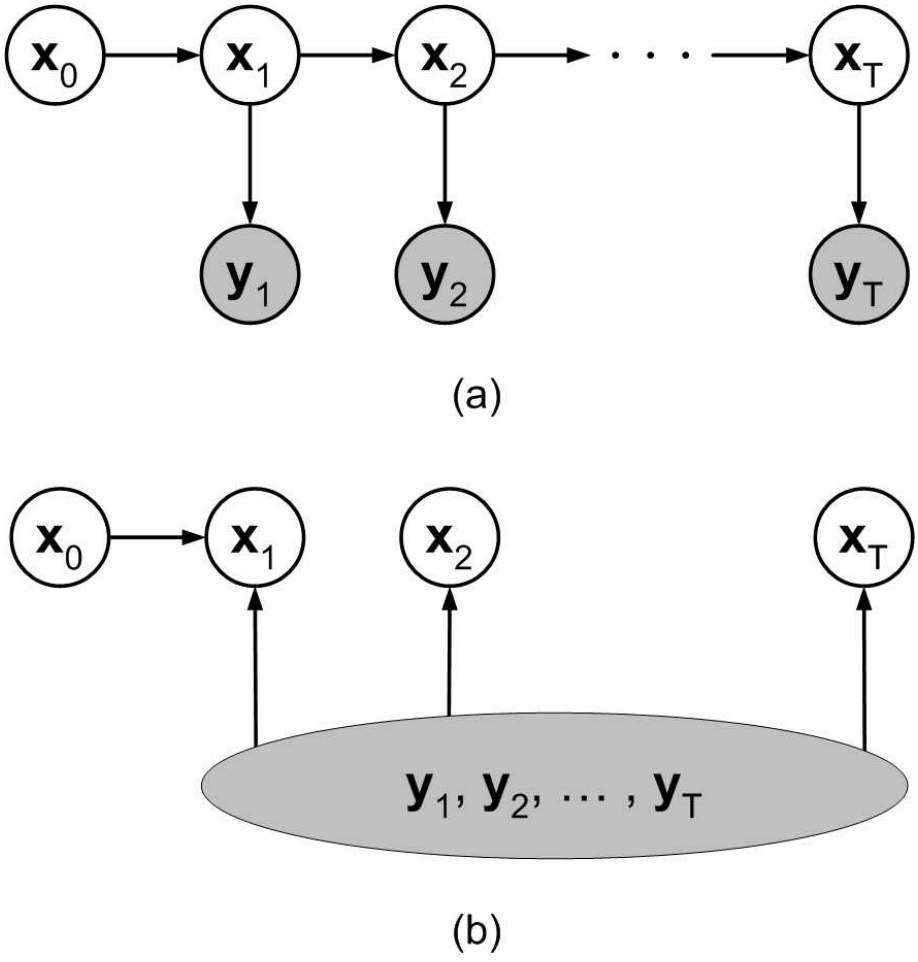

Figure 9.2: A Bayesian network representation of the SSM (a) and its variational posterior (b).

functional $\mathcal{F}$ as defined in (9.8) becomes

$$\begin{aligned}
\mathcal{F}(q) &= \int d\mathbf{X}_{1:T}\; q(\mathbf{X}_{1:T}) \left[\log p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}) - \log q(\mathbf{X}_{1:T})\right] \\
&= \int d\mathbf{X}_{1:T} \prod_{t=1}^{T} q(\mathbf{x}_t) \left[\sum_{t=1}^{T} \log p(\mathbf{y}_t \mid \mathbf{x}_t) + \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \right. \\
&\qquad\qquad \left. - \sum_{t=1}^{T} \log q(\mathbf{x}_t)\right] \\
&= \sum_{t=1}^{T} \int d\mathbf{x}_t\; q(\mathbf{x}_t) \log p(\mathbf{y}_t \mid \mathbf{x}_t) - \sum_{t=1}^{T} \int d\mathbf{x}_t\; q(\mathbf{x}_t) \log q(\mathbf{x}_t) \\
&\quad + \sum_{t=1}^{T} \int d\mathbf{x}_t\; d\mathbf{x}_{t-1}\; q(\mathbf{x}_t)\; q(\mathbf{x}_{t-1}) \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}).
\end{aligned} \tag{9.17}$$

By taking functional derivatives with respect to each $q(\mathbf{x}_t)$, we have (for $t \neq T$),

$$\begin{aligned}
\frac{\delta \mathcal{F}(q)}{\delta q(\mathbf{x}_t)} &= \log p(\mathbf{y}_t \mid \mathbf{x}_t) + \int d\mathbf{x}_{t-1}\; q(\mathbf{x}_{t-1}) \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \\
&\quad + \int d\mathbf{x}_{t+1}\; q(\mathbf{x}_{t+1}) \log p(\mathbf{x}_{t+1} \mid \mathbf{x}_t) - \log q(\mathbf{x}_t) - 1 \\
&= \log p(\mathbf{y}_t \mid \mathbf{x}_t) + E_{t-1}\left[\log p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\right] + E_{t+1}\left[\log p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)\right] \\
&\quad - \log q(\mathbf{x}_t) - 1,
\end{aligned} \tag{9.18}$$

where we define the notation

$$E_t\left[f(\cdot)\right] \triangleq \int d\mathbf{x}_t\; q(\mathbf{x}_t) f(\cdot). \tag{9.19}$$

Setting (9.18) to zero we obtain:

$$\begin{aligned}
\log q(\mathbf{x}_t) &= E_{t-1}\left[\log p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\right] + E_{t+1}\left[\log p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)\right] + \log p(\mathbf{y}_t \mid \mathbf{x}_t) - 1 \\
&= E_{t-1}\left[\frac{1}{2}\log\left|\frac{\mathbf{B}}{2\pi}\right| - \frac{1}{2}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a})^T \mathbf{B} (\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a})\right] \\
&\quad + E_{t+1}\left[\frac{1}{2}\log\left|\frac{\mathbf{B}}{2\pi}\right| - \frac{1}{2}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{a})^T \mathbf{B} (\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{a})\right] \\
&\quad + \frac{1}{2}\log\left|\frac{\mathbf{D}}{2\pi}\right| - \frac{1}{2}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c})^T \mathbf{D} (\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c}) - 1.
\end{aligned} \tag{9.20}$$

The right-hand side of (9.20) contains only first and second order terms of $\mathbf{x}_t$ plus some constants, so must be the left-hand side $\log q(\mathbf{x}_t)$, which proves that $q(\mathbf{x}_t)$ follows a Gaussian distribution. Let's assume:

$$q(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\rho}_t, \boldsymbol{\Gamma}_t), \tag{9.21a}$$

or equivalently,

$$\log q(\mathbf{x}_t) = \frac{1}{2}\log\left|\frac{\boldsymbol{\Gamma}_t}{2\pi}\right| - \frac{1}{2}(\mathbf{x}_t - \boldsymbol{\rho}_t)^T\boldsymbol{\Gamma}_t(\mathbf{x}_t - \boldsymbol{\rho}_t). \tag{9.21b}$$

From (9.20):

$$\begin{aligned}\frac{\partial \log q(\mathbf{x}_t)}{\partial \mathbf{x}_t} &= \mathbf{C}^T\mathbf{D}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c}) + E_{t-1}\left[-\mathbf{B}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a})\right] \\ &\quad + E_{t+1}\left[\mathbf{A}^T\mathbf{B}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{a})\right] \\ &= -(\mathbf{C}^T\mathbf{D}\mathbf{C} + \mathbf{B} + \mathbf{A}^T\mathbf{B}\mathbf{A})\mathbf{x}_t + \mathbf{C}^T\mathbf{D}(\mathbf{y}_t - \mathbf{c}) \\ &\quad + \mathbf{B}\mathbf{a} - \mathbf{A}^T\mathbf{B}\mathbf{a} + \mathbf{B}\mathbf{A}\boldsymbol{\rho}_{t-1} + \mathbf{A}^T\mathbf{B}\boldsymbol{\rho}_{t+1}.\end{aligned} \tag{9.22}$$

From (9.21b):

$$\frac{\partial \log q(\mathbf{x}_t)}{\partial \mathbf{x}_t} = -\boldsymbol{\Gamma}_t\mathbf{x}_t + \boldsymbol{\Gamma}_t\boldsymbol{\rho}_t. \tag{9.23}$$

Comparing (9.22) and (9.23) we have for $t \neq T$:

$$\boldsymbol{\Gamma}_t = \mathbf{C}^T\mathbf{D}\mathbf{C} + \mathbf{B} + \mathbf{A}^T\mathbf{B}\mathbf{A}, \tag{9.24}$$

$$\boldsymbol{\Gamma}_t\boldsymbol{\rho}_t = \mathbf{B}\mathbf{A}\boldsymbol{\rho}_{t-1} + \mathbf{A}^T\mathbf{B}\boldsymbol{\rho}_{t+1} + \mathbf{C}^T\mathbf{D}(\mathbf{y}_t - \mathbf{c}) + \mathbf{B}\mathbf{a} - \mathbf{A}^T\mathbf{B}\mathbf{a}, \tag{9.25}$$

and similarly for $t = T$ we can obtain:

$$\boldsymbol{\Gamma}_T = \mathbf{C}^T\mathbf{D}\mathbf{C} + \mathbf{B}, \tag{9.26}$$

$$\boldsymbol{\Gamma}_T\boldsymbol{\rho}_T = \mathbf{B}\mathbf{A}\boldsymbol{\rho}_{T-1} + \mathbf{C}^T\mathbf{D}(\mathbf{y}_T - \mathbf{c}) + \mathbf{B}\mathbf{a}. \tag{9.27}$$

This completes the derivation of variational inference on the SSM. Equations (9.24)-(9.27) are the ones need to be solved in order to obtain the time-varying mean and precision matrix of the variational posterior, and will be referred to as the variational smoother

hereafter. As a reference, the standard Kalman smoothing equations, which performs exact inference and gives true estimates of the mean $\hat{\mathbf{x}}(t \mid T)$ and covariance matrix $\mathbf{P}(t \mid T)$ of the posterior, are also listed as follows. The variational and true estimates of the posterior distribution (both are Gaussians) will be compared by simulation experiments.

Exact Inference (Kalman Smoothing) of the SSM

Initial condition:

$$\hat{\mathbf{x}}(0 \mid 0) = \mathbf{a}_0, \tag{9.28a}$$

$$\mathbf{P}(0 \mid 0) = \mathbf{B}_0^{-1}. \tag{9.28b}$$

Forward pass ($t = 1, 2, \cdots, T$):

$$\hat{\mathbf{x}}(t \mid t-1) = \mathbf{A}\hat{\mathbf{x}}(t-1 \mid t-1) + \mathbf{a}, \tag{9.29}$$

$$\mathbf{P}(t \mid t-1) = \mathbf{A}\mathbf{P}(t-1 \mid t-1)\mathbf{A}^T + \mathbf{B}^{-1}, \tag{9.30}$$

$$\tilde{\mathbf{y}}(t \mid t-1) = \mathbf{y}(t) - \mathbf{C}\hat{\mathbf{x}}(t \mid t-1) - \mathbf{c}, \tag{9.31}$$

$$\mathbf{P}_{\tilde{\mathbf{y}}}(t \mid t-1) = \mathbf{C}\mathbf{P}(t \mid t-1)\mathbf{C}^T + \mathbf{D}^{-1}, \tag{9.32}$$

$$\mathbf{K}(t) = \mathbf{P}(t \mid t-1)\mathbf{C}^T\mathbf{P}_{\tilde{\mathbf{y}}}(t \mid t-1)^{-1}, \tag{9.33}$$

$$\hat{\mathbf{x}}(t \mid t) = \hat{\mathbf{x}}(t \mid t-1) + \mathbf{K}(t)\tilde{\mathbf{y}}(t \mid t-1), \tag{9.34}$$

$$\mathbf{P}(t \mid t) = \mathbf{P}(t \mid t-1) - \mathbf{K}(t)\mathbf{P}_{\tilde{\mathbf{y}}}(t \mid t-1)\mathbf{K}(t)^T. \tag{9.35}$$

Backward pass ($t = T-1, \cdots, 1$):

$$\mathbf{G}(t) = \mathbf{P}(t \mid t)\mathbf{A}^T\mathbf{P}(t+1 \mid t)^{-1}, \tag{9.36}$$

$$\hat{\mathbf{x}}(t \mid T) = \hat{\mathbf{x}}(t \mid t) + \mathbf{G}(t)\left[\hat{\mathbf{x}}(t+1 \mid T) - \hat{\mathbf{x}}(t+1 \mid t)\right], \tag{9.37}$$

$$\mathbf{P}(t \mid T) = \mathbf{P}(t \mid t) + \mathbf{G}(t)\left[\mathbf{P}(t+1 \mid T) - \mathbf{P}(t+1 \mid t)\right]\mathbf{G}(t)^T. \tag{9.38}$$

Simulation Experiments

Now some simple simulations are performed to compare the variational smoother and the Kalman smoother. We start with the simple scalar-scalar case, *i.e.*, both the state x and observation y are scalars, and use the following model parameters:

$$a_0 = 1.5, \qquad B_0 = 10^8, \qquad A = 0.8, \qquad a = 0.4, \qquad C = 2, \qquad c = -1.5.$$

The parameters of the state equation (9.13a) are chosen to make $E[x]$ asymptotically reach a target of 2. The noise levels in the state (9.13a) and observation (9.13b) equations (represented by the precisions B and D) are varied and will be specified for each example.

Fig. 9.3 shows an example where noise levels in both the state and observation equations are moderate. It can be observed that both the Kalman smoother and the variational smoother work well: the variance estimate by the approximate variational smoother is slightly smaller than that by the Kalman smoother while the state estimate, *i.e.*, the estimated mean of the continuous hidden states, appears to be very similar for both smoothers. Fig. 9.4 shows another example where the noise level in the observation equation is further increased. As a result, the estimates should rely more on the state equation (model) than the observation equation (data). The simulation result confirms the expectation and the variational smoother again works well comparing to the Kalman smoother. The variance estimate by the variational smoother is smaller than that by the Kalman smoother but the state estimates appear to be the same. By computing the numerical difference between the state estimates by the variational and Kalman smoothers in both examples, it turns out that the largest absolute difference is only about 10^{-5}.

As a further test, $\mathbf{x}_n$ and $\mathbf{y}_n$ are both taken to be vectors with the following model parameters:

$$a_0 = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}, \qquad B_0 = \begin{bmatrix} 10^6 & 0 \\ 0 & 10^6 \end{bmatrix},$$

$$A = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.9 \end{bmatrix}, \qquad a = \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 \\ 0.8 & 0.2 \\ 0.2 & 0.8 \\ 0 & 1 \end{bmatrix}, \qquad c = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \end{bmatrix},$$

and an example is shown in Fig. 9.5. This time the variance estimate by the variational smoother is not so good: it is considerably larger than that by the Kalman smoother, but the state estimates by both smoothers still appear to be the same, with the largest numerical difference being around 10^{-5}.

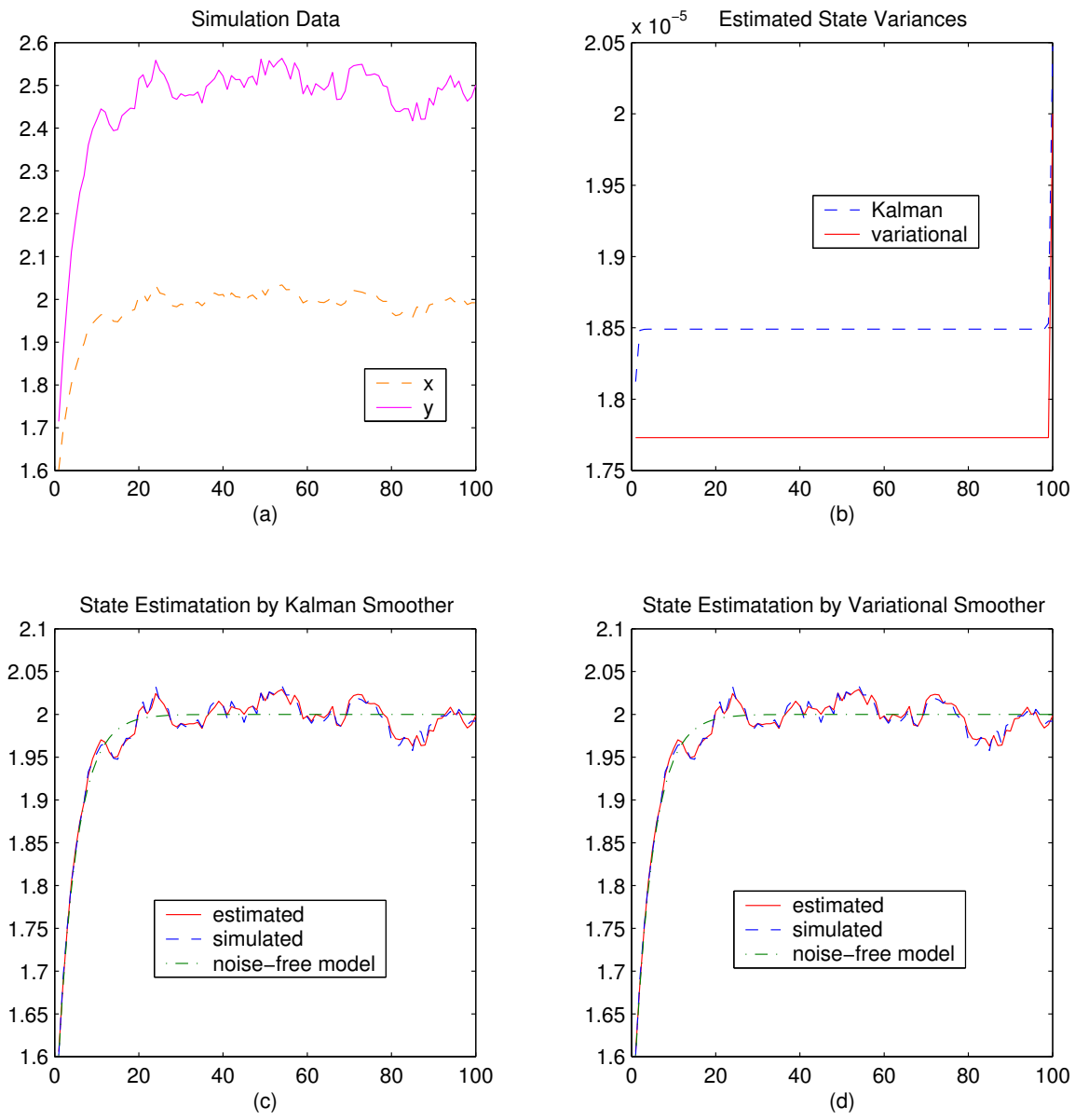

Figure 9.3: Comparison of variational and Kalman smoother in scalar-scalar case 1: $B = 10^4$, $D = 10^4$.

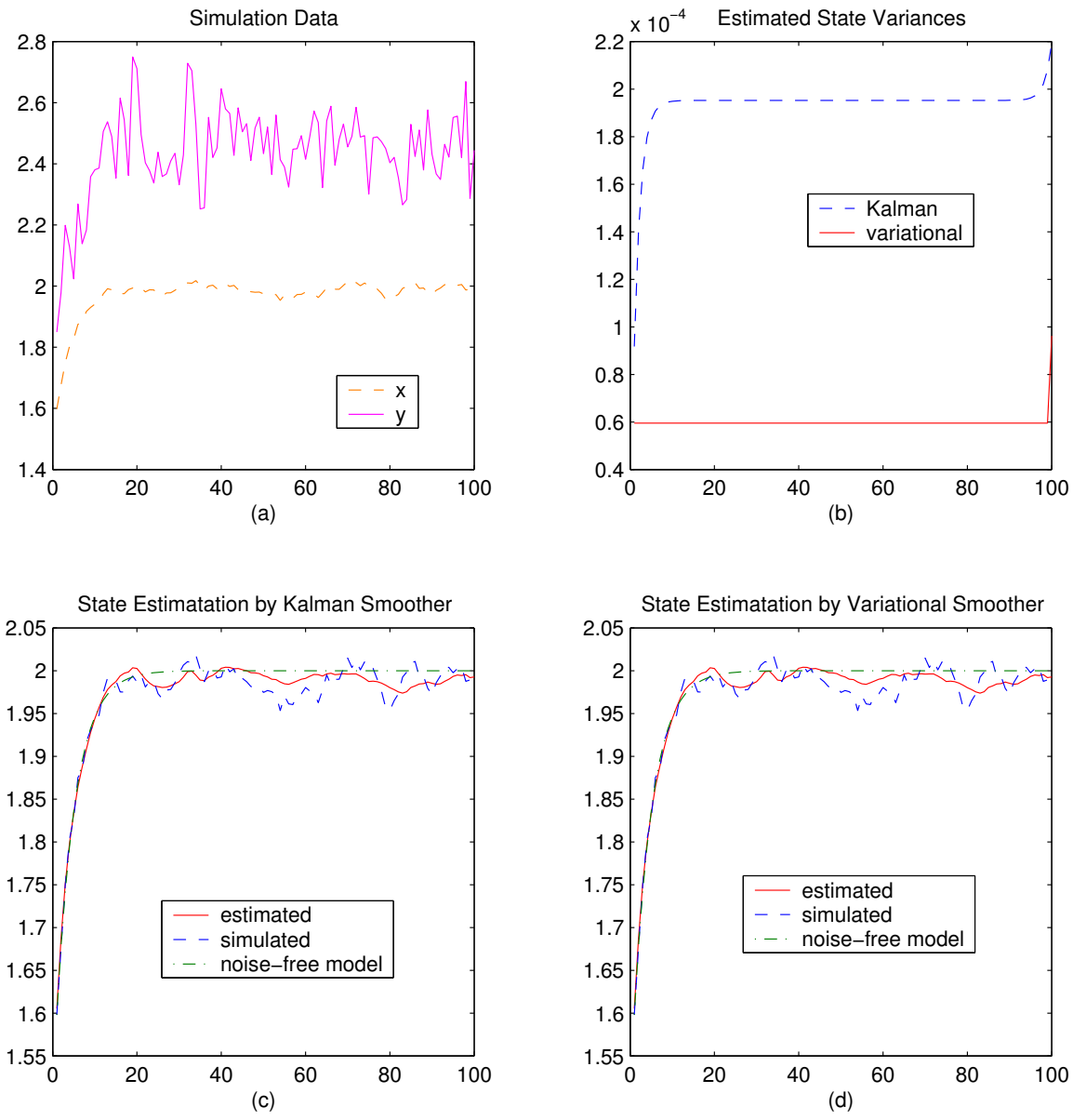

Figure 9.4: Comparison of variational and Kalman smoother in scalar-scalar case 2: $B = 10^4$, $D = 10^2$.

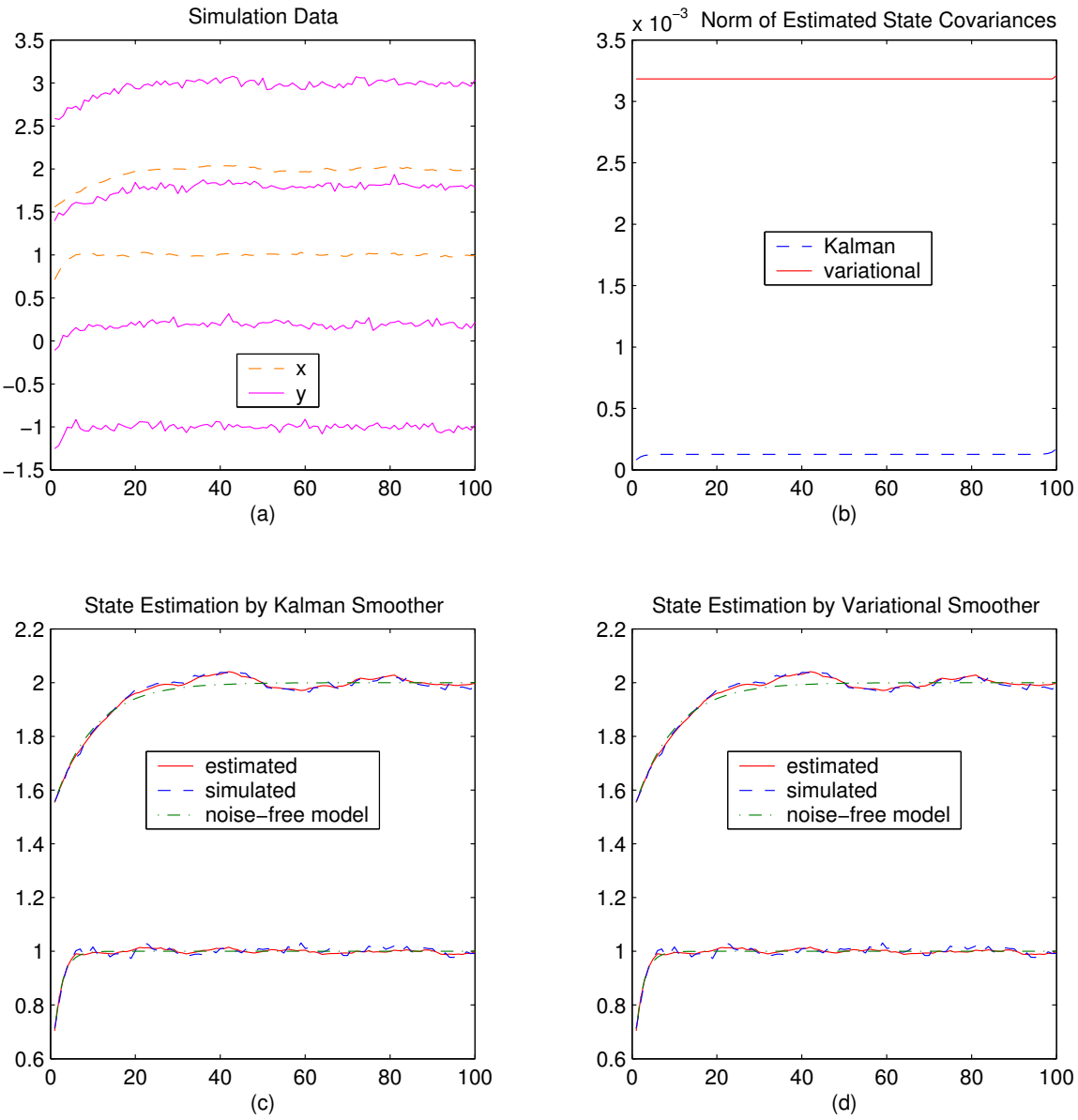

Figure 9.5: Comparison of variational and Kalman smoother in vector-vector case: $\mathbf{B} = 10^4 \cdot \mathbf{I}$, $\mathbf{D} = 10^3 \cdot \mathbf{I}$.

Analysis and Discussions

Since the exact posterior of a SSM is a Markov chain, *i.e.*,

$$p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T}) = \prod p(\mathbf{x}_1 \mid \mathbf{x}_0, \mathbf{Y}_{1:T}) \cdot p(\mathbf{x}_2 \mid \mathbf{x}_1, \mathbf{Y}_{1:T}) \cdots p(\mathbf{x}_T \mid \mathbf{x}_{T-1}, \mathbf{Y}_{1:T}), \tag{9.39}$$

we know that the completely factorized variational posterior can only be an approximation. However, the above simulation results seem to suggest that both the variational smoother and the Kalman smoother give the same estimates on the mean of the states while only the covariance matrix is different (but the variational approximation to the covariance matrix could be rather poor in some situations, such as in Fig. 9.5 (b)). The very small difference between the two state estimates is likely due to rounding errors alone. Such an empirical result can indeed be judged theoretically. If we write out the full matrix equation involving all the time steps that exact inference need to solve, where the Kalman smoother merely serves as an efficient recursive solving method, then it turns out that the equation for the state mean is exactly the same as what has been derived for the variational inference expressed in equations (9.25) and (9.27) while only equations for the covariance (or precision) matrix are different than that of the variational smoother. Details of this derivation are provided in Appendix A.2 for interested readers.

In conclusion, a completely factorized variational posterior, despite its simple structure, is able to perform quite well for SSMs: it gives *exact* estimates on the mean of hidden states $\mathbf{X}_{1:T}$ while approximate estimates on the covariance matrix due to the complete factorization assumption. It should be further noted that computational efficiency is not explicitly considered in this illustrative example. While the Kalman smoother runs in complexity linear in T, a brute force solution of equations (9.24)-(9.27) by matrix inversion has complexity $O(T^3)$, which is impractical for large scale problems. However, computational efficiency will be seriously considered when deriving variational inference and learning algorithms for SSSMs and it turns out that one of the recursive equation solving methods developed in the next section, which has complexity linear in T, will subsume solving equations (9.24)-(9.27) as a special case.

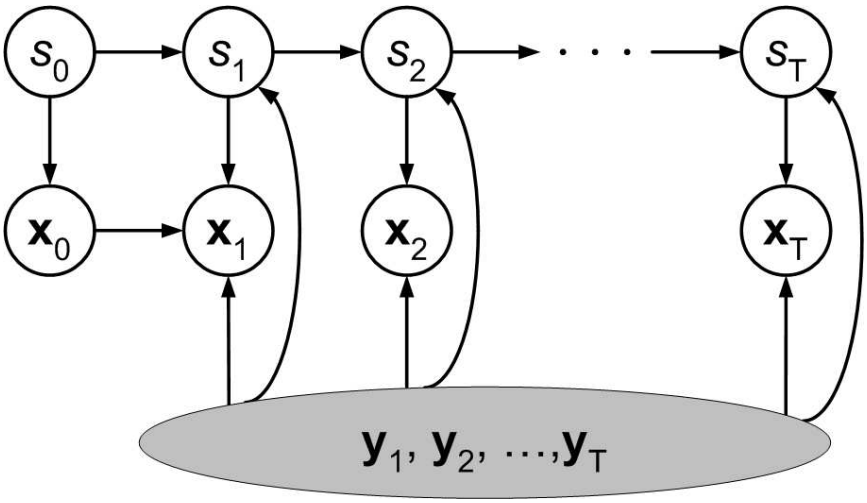

Figure 9.6: Variational posterior of the SSSM represented as a Bayesian network.

9.3 Variational Inference for SSSM

Now we are ready to develop the variational inference and parameter learning algorithms for the SSSM, where exact inference is NP hard. This section describes the variational inference or E step while the parameter estimation or M step will be described in the next section.

In variational approaches the exact posterior $p(\mathbf{X}_{1:T}, \mathbf{s}_{1:T} \mid \mathbf{Y}_{1:T})$ is approximated by a distribution with a tractable structure, denoted by q. Here we choose the following partially factorized structure shown graphically in Fig. 9.6:

$$\begin{aligned} p(\mathbf{X}_{1:T}, \mathbf{s}_{1:T} \mid \mathbf{Y}_{1:T}) &\approx q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T} \mid \mathbf{Y}_{1:T}) \\ &= q(\mathbf{X}_{1:T} \mid \mathbf{s}_{1:T}) \cdot q(\mathbf{s}_{1:T}) \\ &= \prod_{t=1}^{T} q(\mathbf{x}_t \mid s_t) q(s_t \mid s_{t-1}). \end{aligned} \tag{9.40}$$

As is customary to the notation of variational methods, the data dependence of the q's is omitted from now on but always implied.

Whereas q is an approximation, it preserves important features of the exact posterior, and in particular:

1. It incorporates temporal correlations via the Markov chain structure of $q(s_{1:T})$.

2. It incorporates correlations between the continuous and discrete states.

3. Most significantly, it incorporates multimodality of the continuous states since

$$q(\mathbf{x}_t) = \sum_s q(\mathbf{x}_t \mid s_t = s) q(s_t = s) \tag{9.41}$$

   is a mixture distribution (with N mixtures), unlike all the previously developed approximate algorithms on the same model.

On the other hand, it does not directly incorporate temporal correlations among $\mathbf{x}_t$'s; those will be introduced indirectly via the variational equations to be derived shortly. Such a variational approximation is also similar to the one used in the illustrative example of SSM in the previous section. Previous work on variational approaches to similar models [90, 184] uses the factorized form $q = q(\mathbf{x}_{1:T})q(s_{1:T})$. Whereas that form does incorporate temporal correlations among the $\mathbf{x}_t$'s, it results in a $q(\mathbf{x}_{1:T})$ which is a Gaussian, and thus unimodal. Nevertheless, such an approximation can also be applied to our model and a detailed comparison study with this alternative variational approach and other approximate methods will be carried out in the future. Our own early work [148] also used a more factorized variational posterior $q(s_{1:T}, \mathbf{x}_{1:T} \mid \mathbf{y}_{1:T}) = \prod_t q(\mathbf{x}_t \mid s_t) q(s_t)$, which preserves the multimodality of the exact posterior as well. However, that approach was later abandoned due to its lack of efficiency as well as accuracy.

To derive the functional form as well as parameters of q, we again start with the functional defined in (9.8)

$$\mathcal{F}(q) = \sum_{\mathbf{s}_{1:T}} \int d\mathbf{X}_{1:T}\, q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T}) \big[\log p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}, \mathbf{s}_{1:T}) - \log q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T})\big], \tag{9.42}$$

and optimize it w.r.t. (with respect to) q, under the restriction that q has the structure of (9.40). This is done by setting the functional derivative $\delta\mathcal{F}/\delta q(\mathbf{x}_t \mid s_t)$ and the ordinary derivative $\partial\mathcal{F}/\partial q(s_t \mid s_{t-1})$ to zero, for each t, and solving the resulting recursive equations, equivalent to minimizing the KL distance of q to the exact posterior. The derivation steps are described in detail as follows.

Simplification of $\mathcal{F}$

First $\mathcal{F}$ can be substantially simplified by utilizing the constrained structure of q

$$
\begin{aligned}
\mathcal{F}(q) &= \sum_{\mathbf{s}_{1:T}} \int d\mathbf{X}_{1:T}\; q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T}) \big[\log p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}, \mathbf{s}_{1:T}) - \log q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T})\big] \\
&= \sum_{\mathbf{s}_{1:T}} \int d\mathbf{X}_{1:T} \Big[\prod_{t=1}^{T} q(\mathbf{x}_t \mid s_t) q(s_t \mid s_{t-1})\Big] \cdot \\
&\qquad \Big\{\sum_{t=1}^{T} \log\big[p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t) p(s_t \mid s_{t-1})\big] - \sum_{t=1}^{T} \log\big[q(\mathbf{x}_t \mid s_t) q(s_t \mid s_{t-1})\big]\Big\} \\
&= \sum_{t=1}^{T} \sum_{\mathbf{s}_{1:T}} \int d\mathbf{X}_{1:T} \Big[\prod_{t=1}^{T} q(\mathbf{x}_t \mid s_t) q(s_t \mid s_{t-1})\Big] \cdot \\
&\qquad \Big\{\log\big[p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t) p(s_t \mid s_{t-1})\big] - \log\big[q(\mathbf{x}_t \mid s_t) q(s_t \mid s_{t-1})\big]\Big\} \\
&= \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \int d\mathbf{x}_{t-1}\, d\mathbf{x}_t \Big\{q(\mathbf{x}_{t-1} \mid s_{t-1}) q(\mathbf{x}_t \mid s_t) \cdot \\
&\quad \big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) + \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t) + \log p(s_t|s_{t-1}) - \log q(\mathbf{x}_t|s_t) - \log q(s_t|s_{t-1})\big]\Big\} \\
&= \sum_{t=1}^{T} \sum_{s_t} q(s_t) \int d\mathbf{x}_t\; q(\mathbf{x}_t \mid s_t) \big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) - \log q(\mathbf{x}_t|s_t)\big] \\
&+ \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \int d\mathbf{x}_{t-1}\, d\mathbf{x}_t \; \big[q(\mathbf{x}_{t-1} \mid s_{t-1}) q(\mathbf{x}_t \mid s_t) \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t)\big] \\
&+ \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \big[\log p(s_t|s_{t-1}) - \log q(s_t|s_{t-1})\big]. \qquad (9.43)
\end{aligned}
$$

In the above formulas, $q(s_{t-1})$ and $q(s_t)$ are variational marginal posterior probabilities of the discrete hidden states, *e.g.*, $q(s_t)$ is defined to be

$$q(s_t) = \sum_{\mathbf{s}_{\neg t}} \int d\mathbf{X}_{1:T} q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T}), \qquad (9.44)$$

where the notation $\mathbf{s}_{\neg t}$ represents the set of $\mathbf{s}_{1:T}$ except s_t, *i.e.*,

$$\mathbf{s}_{\neg t} \triangleq \mathbf{s}_{1:T} - \{s_t\} = \{s_1, s_2, \cdots, s_{t-1}, s_{t+1}, \cdots, s_T\}. \qquad (9.45)$$

By adopting the variational structure of (9.40), the following relationship holds:

$$q(s_t) = \sum_{s_{t-1}} q(s_{t-1}) q(s_t \mid s_{t-1}). \tag{9.46}$$

To simplify notation, we use γ to denote this discrete marginal posterior probability, *i.e.*,

$$\gamma_{s,t} \triangleq q(s_t = s), \tag{9.47}$$

and also define the associated forward and backward variational posterior transition probabilities:

$$\eta_{s's,t} \triangleq q(s_t = s \mid s_{t-1} = s'), \tag{9.48}$$

$$\bar{\eta}_{s''s,t} \triangleq q(s_t = s \mid s_{t+1} = s'') = \frac{\eta_{ss'',t+1}\gamma_{s,t}}{\gamma_{s'',t+1}} \,. \tag{9.49}$$

To further simplify notation, we also introduce $E_{s,t}$, denoting s-conditioned variational averaging, via

$$E_{s,t}\big[g(\mathbf{x}_t)\big] = \int d\mathbf{x}_t \; q(\mathbf{x}_t \mid s_t = s) g(\mathbf{x}_t), \tag{9.50}$$

where g is an arbitrary function.

Solving for $q(\mathbf{x}_t \mid s_t)$

We take functional derivative of $\mathcal{F}$ expressed in (9.43) w.r.t. every $q(\mathbf{x}_t \mid s_t = s)$ and set them to zero. When $t \neq T$,

$$\begin{aligned}
\frac{\delta \mathcal{F}}{\delta q(\mathbf{x}_t \mid s_t = s)} &= q(s_t)\big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) - \log q(\mathbf{x}_t \mid s_t) - 1\big] \\
&\quad + \sum_{s_{t-1}} q(s_{t-1}) \; q(s_t \mid s_{t-1}) \int d\mathbf{x}_{t-1} \; \big[q(\mathbf{x}_{t-1} \mid s_{t-1}) \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t)\big] \\
&\quad + \sum_{s_{t+1}} q(s_t) \; q(s_{t+1} \mid s_t) \int d\mathbf{x}_{t+1} \; \big[q(\mathbf{x}_{t+1} \mid s_{t+1}) \log p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, s_{t+1})\big] \\
&= \gamma_{s,t}\big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t = s) - \log q(\mathbf{x}_t \mid s_t = s) - 1\big] \\
&\quad + \sum_{s'} \gamma_{s',t-1} \; \eta_{s's,t} \; E_{s',t-1}\big[\log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t = s)\big] \\
&\quad + \sum_{s''} \gamma_{s,t} \; \eta_{ss'',t+1} \; E_{s'',t+1}\big[\log p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, s_{t+1} = s'')\big].
\end{aligned} \tag{9.51}$$

Setting (9.51) to zero, we have

$$\begin{aligned}\log q(\mathbf{x}_t|s_t=s) &= \log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t=s) - 1 + \sum_{s'} \frac{\gamma_{s',t-1}\cdot\eta_{s's,t}}{\gamma_{s,t}}\, E_{s',t-1}\big[\log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t=s)\big] \\ &\quad + \sum_{s''} \eta_{ss'',t+1}\, E_{s'',t+1}\big[\log p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, s_{t+1}=s'')\big] \\ &= \log \mathcal{N}(\mathbf{y}_t \mid \mathbf{C}_s\mathbf{x}_t + \mathbf{c}_s, \mathbf{D}_s) + \sum_{s'} \bar{\eta}_{ss',t-1}\, E_{s',t-1}\big[\log \mathcal{N}(\mathbf{x}_t \mid \mathbf{A}_s\mathbf{x}_{t-1} + \mathbf{a}_s, \mathbf{B}_s)\big] \\ &\quad + \sum_{s''} \eta_{ss'',t+1}\, E_{s'',t+1}\big[\log \mathcal{N}(\mathbf{x}_{t+1} \mid \mathbf{A}_{s''}\mathbf{x}_t + \mathbf{a}_{s''}, \mathbf{B}_{s''})\big] - 1. \end{aligned} \tag{9.52}$$

Since the right hand side of (9.52) is quadratic in $\mathbf{x}_t$, so must be the left hand side, which indicates that $q(\mathbf{x}_t|s_t=s)$ has to follow a Gaussian distribution. In general, we may assume

$$q(\mathbf{x}_t \mid s_t=s) = \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\rho}_{s,t}, \boldsymbol{\Gamma}_{s,t}) = \left|\frac{\boldsymbol{\Gamma}_{s,t}}{2\pi}\right|^{\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\rho}_{s,n})^T\boldsymbol{\Gamma}_{s,t}(\mathbf{x}_t - \boldsymbol{\rho}_{s,t})\right]. \tag{9.53}$$

Parameters of this time-varying, s-dependent Gaussian distribution can be obtained by differentiating both (9.52) and (9.53) w.r.t. $\mathbf{x}_t$ and matching the corresponding coefficients. From (9.52),

$$\begin{aligned}\frac{\partial \log q(\mathbf{x}_t \mid s_t=s)}{\partial\, \mathbf{x}_t} &= \mathbf{C}_s^T\mathbf{D}_s(\mathbf{y}_t - \mathbf{C}_s\mathbf{x}_t - \mathbf{c}_s) + \sum_{s'} \bar{\eta}_{ss',t-1}\, E_{s',t-1}\big[-\mathbf{B}_s(\mathbf{x}_t - \mathbf{A}_s\mathbf{x}_{t-1} - \mathbf{a}_s\big] \\ &\quad + \sum_{s''} \eta_{ss'',t+1}\, E_{s'',t+1}\big[\mathbf{A}_{s''}^T\mathbf{B}_{s''}(\mathbf{x}_{t+1} - \mathbf{A}_{s''}\mathbf{x}_t - \mathbf{a}_{s''})\big] \\ &= \mathbf{C}_s^T\mathbf{D}_s(\mathbf{y}_t - \mathbf{C}_s\mathbf{x}_t - \mathbf{c}_s) + \sum_{s'} \bar{\eta}_{ss',t-1}\big[-\mathbf{B}_s\mathbf{x}_t + \mathbf{B}_s\mathbf{A}_s\boldsymbol{\rho}_{s',t-1} + \mathbf{B}_s\mathbf{a}_s\big] \\ &\quad + \sum_{s''} \eta_{ss'',t+1}\big[\mathbf{A}_{s''}^T\mathbf{B}_{s''}(\boldsymbol{\rho}_{s'',t+1} - \mathbf{a}_{s''}) - \mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{A}_{s''}\mathbf{x}_t\big] \\ &= -(\mathbf{C}_s^T\mathbf{D}_s\mathbf{C}_s + \mathbf{B}_s + \sum_{s''} \eta_{ss'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{A}_{s''})\mathbf{x}_t \\ &\quad + \mathbf{B}_s\mathbf{A}_s\sum_{s'} \bar{\eta}_{ss',t-1}\boldsymbol{\rho}_{s',t-1} + \sum_{s''} \eta_{ss'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\boldsymbol{\rho}_{s'',t+1} \\ &\quad + \mathbf{C}_s^T\mathbf{D}_s(\mathbf{y}_t - \mathbf{c}_s) + \mathbf{B}_s\mathbf{a}_s - \sum_{s''} \eta_{ss'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{a}_{s''}\,. \end{aligned} \tag{9.54}$$

From (9.53),

$$\frac{\partial \log q(\mathbf{x}_t \mid s_t = s)}{\partial\, \mathbf{x}_t} = -\mathbf{\Gamma}_{s,t}(\mathbf{x}_t - \boldsymbol{\rho}_{s,t}) = -\mathbf{\Gamma}_{s,t}\mathbf{x}_t + \mathbf{\Gamma}_{s,t}\boldsymbol{\rho}_{s,t} \ . \tag{9.55}$$

Comparing (9.54) and (9.54), we arrive at

$$\mathbf{\Gamma}_{s,t} = \mathbf{C}_s^T \mathbf{D}_s \mathbf{C}_s + \mathbf{B}_s + \sum_{s''} \eta_{ss'',t+1} \mathbf{A}_{s''}^T \mathbf{B}_{s''} \mathbf{A}_{s''} \ , \tag{9.56}$$

$$\begin{aligned} \mathbf{\Gamma}_{s,t}\boldsymbol{\rho}_{s,t} = \mathbf{B}_s \mathbf{A}_s \sum_{s'} \bar{\eta}_{ss',t-1} \boldsymbol{\rho}_{s',t-1} + \sum_{s''} \eta_{ss'',t+1} \mathbf{A}_{s''}^T \mathbf{B}_{s''} \boldsymbol{\rho}_{s'',t+1} \\ + \mathbf{C}_s^T \mathbf{D}_s (\mathbf{y}_t - \mathbf{c}_s) + \mathbf{B}_s \mathbf{a}_s - \sum_{s''} \eta_{ss'',t+1} \mathbf{A}_{s''}^T \mathbf{B}_{s''} \mathbf{a}_{s''} \ . \end{aligned} \tag{9.57}$$

Similarly when $t = T$, it is straightforward to derive that

$$\mathbf{\Gamma}_{s,T} = \mathbf{C}_s^T \mathbf{D}_s \mathbf{C}_s + \mathbf{B}_s \ , \tag{9.58}$$

$$\mathbf{\Gamma}_{s,T}\boldsymbol{\rho}_{s,T} = \mathbf{B}_s \mathbf{A}_s \sum_{s'} \bar{\eta}_{ss',T-1} \boldsymbol{\rho}_{s',T-1} + \mathbf{C}_s^T \mathbf{D}_s (\mathbf{y}_T - \mathbf{c}_s) + \mathbf{B}_s \mathbf{a}_s \ . \tag{9.59}$$

Notice that a brute force solution of equations (9.57) and (9.59) (given model parameters and all the η's) by matrix inversion has complexity $\mathcal{O}(T^3 N^3)$, where N is the number of discrete states. Such a computational cost is too high for large scale problems. In the past we have tried to solve by simple sparse matrix techniques [148] and an approximate method with overlapping windows [149]. In this thesis an efficient and exact solving method based on probability propagation, which only has complexity $O(TN)$, is newly developed and will be presented shortly, after deriving equations to compute $q(s_t \mid s_{t-1})$.

Solving for $q(s_t \mid s_{t-1})$

We first digress a bit to point out that unconstrained optimization of $\mathcal{F}$ generally leads to unnormalized probabilities, also known as potential functions [90]. Rigorously speaking one should optimize $\mathcal{F}$ subject to a normalization constraint, as is done in Appendix A.1, or alternatively, given the factorized structure of q in (9.40), normalize it at each time step respectively. The normalization problem was not brought into attention when deriving equations for $q(\mathbf{x}_t \mid s_t)$, since for a Gaussian distribution the normalization constant is

irrelevant in obtaining its mean and precision (or covariance) matrix[5]. However, for discrete probability distributions $q(s_t \mid s_{t-1})$, proper normalization has to be done so that they can be genuinely interpreted as probabilities. This is achieved by introducing the normalization factor $z_{s',t}$ for each time t and state s', where

$$z_{s',t} = \sum_s q(s_t = s \mid s_{t-1} = s') = \sum_s \eta_{s's,t} \ , \tag{9.60}$$

and $\eta_{s's,t}$ refer to the original unnormalized probabilities.

Next we focus on partial derivatives of the form $\partial q(s_\tau)/\partial q(s_t|s_{t-1})$, which turns out to be the central piece in differentiating $\mathcal{F}(q)$ w.r.t. $q(s_t|s_{t-1})$. After working with typical examples such as

$$\frac{\partial q(s_t)}{\partial q(s_t \mid s_{t-1})} = \frac{\partial \left[\sum_{s_{t-1}} q(s_t \mid s_{t-1}) q(s_{t-1})\right]}{\partial \, q(s_t \mid s_{t-1})} = q(s_{t-1}), \tag{9.61}$$

$$\frac{\partial q(s_{t+1})}{\partial q(s_t \mid s_{t-1})} = \frac{\partial \left[\sum_{s_t} q(s_{t+1} \mid s_t) q(s_t)\right]}{\partial \, q(s_t \mid s_{t-1})} = q(s_{t+1} \mid s_t) q(s_{t-1}), \tag{9.62}$$

we can summarize in general

$$\frac{\partial q(s_\tau)}{\partial q(s_t|s_{t-1})} = \begin{cases} 0, & \text{if } \tau < t, \\ q(s_{t-1}), & \text{if } \tau = t, \\ q(s_{t+1} \mid s_t) q(s_{t-1}), & \text{if } \tau = t+1, \\ \sum\limits_{s_{t+2} \,:\, s_\tau} q(s_\tau \mid s_{\tau-1}) \cdots q(s_{t+1} \mid s_t) q(s_{t-1}), & \text{otherwise.} \end{cases} \tag{9.63}$$

Now we are ready to differentiate $\mathcal{F}(q)$ w.r.t. $q(s_t|s_{t-1})$. First the simplified form of $\mathcal{F}$ is re-presented for the convenience of reference,

$$\begin{aligned} \mathcal{F}(q) = & \sum_{t=1}^{T} \sum_{s_t} q(s_t) \int d\mathbf{x}_t \; q(\mathbf{x}_t \mid s_t) \big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t) - \log q(\mathbf{x}_t|s_t)\big] \\ & + \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \int d\mathbf{x}_{t-1} \, d\mathbf{x}_t \; \big[q(\mathbf{x}_{t-1} \mid s_{t-1}) q(\mathbf{x}_t \mid s_t) \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t)\big] \\ & + \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \big[\log p(s_t|s_{t-1}) - \log q(s_t|s_{t-1})\big]. \end{aligned} \tag{9.64}$$

[5]Notice that the normalization constant will disappear when taking $\frac{\partial \log q(\mathbf{x}_t|s_t=s)}{\partial \, \mathbf{x}_t}$.

Take partial derivatives w.r.t. every $q(s_t|s_{t-1})$,

$$\begin{aligned}\frac{\partial \mathcal{F}(q)}{\partial q(s_t|s_{t-1})} &= \sum_{\tau=t}^{T}\sum_{s_\tau}\frac{\partial q(s_\tau)}{\partial q(s_t|s_{t-1})}\int d\mathbf{x}_\tau\; q(\mathbf{x}_\tau \mid s_\tau)\big[\log p(\mathbf{y}_\tau \mid \mathbf{x}_\tau, s_\tau) - \log q(\mathbf{x}_\tau \mid s_\tau)\big] \\ &+ q(s_{t-1})\int d\mathbf{x}_{t-1}\, d\mathbf{x}_t\; \big[q(\mathbf{x}_{t-1} \mid s_{t-1})q(\mathbf{x}_t \mid s_t)\log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t)\big] \\ &+ \sum_{\tau=t+1}^{T}\sum_{s_\tau}\frac{\partial q(s_{\tau-1})}{\partial q(s_t|s_{t-1})}q(s_\tau|s_{\tau-1})\int d\mathbf{x}_{\tau-1}d\mathbf{x}_\tau\big[q(\mathbf{x}_{\tau-1}|s_{\tau-1})q(\mathbf{x}_\tau|s_\tau)\log p(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}, s_\tau)\big] \\ &+ q(s_{t-1})\big[\log p(s_t \mid s_{t-1}) - \log q(s_t \mid s_{t-1}) - 1\big] \\ &+ \sum_{\tau=t+1}^{T}\sum_{s_\tau}\frac{\partial q(s_{\tau-1})}{\partial q(s_t|s_{t-1})}q(s_\tau|s_{\tau-1})\big[\log p(s_\tau \mid s_{\tau-1}) - \log q(s_\tau \mid s_{\tau-1})\big], \end{aligned} \tag{9.65}$$

where $\partial q(s_\tau)/\partial q(s_t|s_{t-1})$ is given by (9.63) and zero terms have been removed. In spite of the highly complicated appearance of (9.65), its structure clearly suggests that we should seek a recursive formula to express $\partial\mathcal{F}/\partial q(s_t|s_{t-1})$ and setting them to zeros to solve for $q(s_t|s_{t-1})$.

Start with $t = T$ and use the more compact notations defined in (9.47)–(9.50),

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial\, \eta_{s's,T}} &= \gamma_{s',T-1}E_{s,T}\big[\log p(\mathbf{y}_T \mid \mathbf{x}_T, s_T = s) - \log q(\mathbf{x}_T \mid s_T = s)\big] \\ &+ \gamma_{s',T-1}E_{s',T-1}E_{s,T}\big[\log p(\mathbf{x}_T \mid \mathbf{x}_{T-1}, s_T = s)\big] \\ &+ \gamma_{s',T-1}\big[\log p(s_T = s \mid s_{T-1} = s') - \log \eta_{s's,T} - 1\big]. \end{aligned} \tag{9.66}$$

Setting the above equation to zero and discarding the trivial solution of $\gamma_{s',T-1} = 0$, we have

$$\begin{aligned}\log \eta_{s's,T} &= E_{s,T}\big[\log p(\mathbf{y}_T \mid \mathbf{x}_T, s_T = s) - \log q(\mathbf{x}_T \mid s_T = s)\big] \\ &+ E_{s',T-1}E_{s,T}\big[\log p(\mathbf{x}_T \mid \mathbf{x}_{T-1}, s_T = s)\big] + \log \pi_{s's} - \log z_{s',T}\;. \end{aligned} \tag{9.67}$$

Notice how we have inserted the normalization factor introduced in (9.60), absorbing the original constant "-1". To further simplify notation, we also define

$$\begin{aligned}f_{s's,t} &= E_{s,t}\big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t = s) - \log q(\mathbf{x}_t \mid s_t = s)\big] \\ &+ E_{s',t-1}E_{s,t}\big[\log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t = s)\big] + \log \pi_{s's}\;, \end{aligned} \tag{9.68}$$

and (9.67) simplifies to

$$\log \eta_{s's,T} = f_{s's,T} - \log z_{s',T} \ . \tag{9.69}$$

Subsequently, when $t = T - 1$

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial\ \eta_{s''s',T-1}} &= \gamma_{s'',T-2}\ E_{s',T-1}\big[\log p(\mathbf{y}_{T-1} \mid \mathbf{x}_{T-1}, s_{T-1} = s') - \log q(\mathbf{x}_{T-1} \mid s_{T-1} = s')\big] \\
&\quad + \gamma_{s'',T-2} \sum_s \eta_{s's,T} E_{s,T}\big[\log p(\mathbf{y}_T \mid \mathbf{x}_T, s_T = s) - \log q(\mathbf{x}_T \mid s_T = s)\big] \\
&\quad + \gamma_{s'',T-2}\ E_{s'',T-2}\ E_{s',T-1}\big[\log p(\mathbf{x}_{T-1} \mid \mathbf{x}_{T-2}, s_{T-1} = s')\big] \\
&\quad + \gamma_{s'',T-2} \sum_s \eta_{s's,T} E_{s',T-1} E_{s,T}\big[\log p(\mathbf{x}_T \mid \mathbf{x}_{T-1}, s_T = s)\big] \\
&\quad + \gamma_{s'',T-2}\big[\log p(s_{T-1} = s' \mid s_{T-2} = s'') - \log \eta_{s''s',T-1} - 1\big] \\
&\quad + \gamma_{s'',T-2} \sum_s \eta_{s's,T}\big[\log p(s_T = s \mid s_{T-1} = s') - \log \eta_{s's,T}\big].
\end{aligned} \tag{9.70}$$

Set the above equation to zero and use (9.68) and (9.69) for simplification,

$$\begin{aligned}
\log \eta_{s''s',T-1} &= E_{s',T-1}\big[\log p(\mathbf{y}_{T-1} \mid \mathbf{x}_{T-1}, s_{T-1} = s') - \log q(\mathbf{x}_{T-1} \mid s_{T-1} = s')\big] \\
&\quad + E_{s'',T-2}\ E_{s',T-1}\big[\log p(\mathbf{x}_{T-1} \mid \mathbf{x}_{T-2}, s_{T-1} = s')\big] + \pi_{s''s'} - \log z_{s'',T-1} \\
&\quad + \sum_s \eta_{s's,T}(f_{s's,T} - \log \eta_{s's,T}) \\
&= f_{s''s',T-1} - \log z_{s'',T-1} + \log z_{s',T} \ .
\end{aligned} \tag{9.71}$$

In general, it is a simple matter to show by induction that

$$\log \eta_{s's,t} = f_{s's,t} - \log z_{s',t} + \log z_{s,t+1} \ , \tag{9.72}$$

holds for all $t < T$. To extend this relationship for $t = T$, we may simply define $z_{s,T+1} = 1$ for all s.

The recursive formula (9.72) enables us to compute all $q(s_t \mid s_{t-1})$ (or η's) efficiently with the following backward pass:

1. Initialization:

$$z_{s,T+1} = 1, \ \ 1 \le s \le N. \tag{9.73}$$

2. Induction: for $t = T-1, \ldots, 1$

$$\eta_{s's,t} = \frac{1}{z_{s',t}} \exp(f_{s's,t}) z_{s,t+1} \ , \tag{9.74a}$$

$$z_{s',t} = \sum_{s} \exp(f_{s's,t}) z_{s,t+1} \ . \tag{9.74b}$$

It is necessary to compute the marginal posteriors $\gamma_{s,t} = q(s_t = s)$ as well, which are needed in computing $q(\mathbf{x}_t \mid s_t)$ and the estimation of model parameters (M-step). They are obtained by a forward pass:

1. Initialization:

$$\gamma_{s,1} = \pi_{0s}, \ \ 1 \le s \le N. \tag{9.75}$$

2. Induction: for $t = 2, \ldots, T$

$$\gamma_{s,t} = \sum_{s'} \eta_{s's,t} \ \gamma_{s',t-1} \ . \tag{9.76}$$

At each time step t, we also have to calculate $f_{s's,t}$ in order to apply the backward path. The s-conditioned averages in $f_{s's,t}$ as defined by (9.68) are straightforward to compute analytically, although the exact expression is very lengthy. To do so, we first define the following "dot product" between two matrices

$$\langle \mathbf{A}, \ \mathbf{B} \rangle = \sum_{i,j} a_{ij} \cdot b_{ij}, \tag{9.77}$$

which is a scalar summing over all elements of the element-wise product between two matrices $\mathbf{A}$ and $\mathbf{B}$ of the same dimension. Specifically, it is easy to verify that

$$\mathbf{x}^T \mathbf{A} \mathbf{y} = \langle \mathbf{A}, \ \mathbf{x}\mathbf{y}^T \rangle \tag{9.78}$$

holds for arbitrary vectors $\mathbf{x}$ and $\mathbf{y}$ of proper dimensions. After some careful algebraic

manipulations, it can be shown that

$$\begin{aligned} f_{s's,t} = & \frac{1}{2}\log\left|\frac{\mathbf{D}_s}{2\pi}\right| - \frac{1}{2}\log\left|\frac{\mathbf{\Gamma}_{s,t}}{2\pi}\right| + \frac{1}{2}\log\left|\frac{\mathbf{B}_s}{2\pi}\right| + \log\pi_{s's} \\ & - \frac{1}{2}\langle \mathbf{C}_s^T\mathbf{D}_s\mathbf{C}_s,\ \mathbf{\Gamma}_{s,t}^{-1} + \boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T\rangle + \frac{1}{2}\langle \mathbf{\Gamma}_{s,t},\ \mathbf{\Gamma}_{s,t}^{-1} + \boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T\rangle \\ & - \frac{1}{2}\langle \mathbf{B}_s,\ \mathbf{\Gamma}_{s,t}^{-1} + \boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T\rangle - \frac{1}{2}\langle \mathbf{A}_s^T\mathbf{B}_s\mathbf{A}_s,\ \mathbf{\Gamma}_{s',t-1}^{-1} + \boldsymbol{\rho}_{s',t-1}\boldsymbol{\rho}_{s',t-1}^T\rangle \\ & + (\mathbf{y}_t - \mathbf{c}_s)^T\mathbf{D}_s\mathbf{C}_s\boldsymbol{\rho}_{s,t} + \mathbf{a}_s^T\mathbf{B}_s\boldsymbol{\rho}_{s,t} + \boldsymbol{\rho}_{s',t-1}^T\mathbf{A}_s^T\mathbf{B}_s\boldsymbol{\rho}_{s,t} - \mathbf{a}_s^T\mathbf{B}_s\mathbf{A}_s\boldsymbol{\rho}_{s',t-1} \\ & - \frac{1}{2}(\mathbf{y}_t - \mathbf{c}_s)^T\mathbf{D}_s(\mathbf{y}_t - \mathbf{c}_s) - \frac{1}{2}\boldsymbol{\rho}_{s,t}^T\mathbf{\Gamma}_{s,t}\boldsymbol{\rho}_{s,t} - \frac{1}{2}\mathbf{a}_s^T\mathbf{B}_s\mathbf{a}_s. \end{aligned} \tag{9.79}$$

Finally, it is also possible to derive a mathematically equivalent forward-backward algorithm to compute all the η's and γ's, which may be more convenient under some circumstances. This can be achieved by describing the variational posterior by backward posterior transition probability $\bar{\eta}$'s instead of the forward transition probability η's.

Sparse Matrix Inversion by Probability Propagation

As mentioned previously, brute-forth solution to the coupled linear equations of (9.57) and (9.59) has complexity $\mathcal{O}(T^3N^3)$, which is impractical for large scale problems. Here we further describe a recursive method to solve equations (9.57) and (9.59) exactly with complexity only linear in (TN), which substantially improve the computational efficiency of the variational inference algorithm derived above. The key idea is to show that the solution is the mean of a particular Gaussian distribution with a Markov structure, and compute that mean by a forward-backward algorithm. Focusing on (9.57), we first rewrite it as

$$\begin{aligned} \frac{\mathbf{\Gamma}_{s,t}}{\gamma_{s,t}}(\gamma_{s,t}\boldsymbol{\rho}_{s,t}) = & \mathbf{B}_s\mathbf{A}_s\sum_{s'}\frac{\eta_{s's,t}}{\gamma_{s,t}}(\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}) + \sum_{s''}\frac{\eta_{ss'',t+1}}{\gamma_{s'',t+1}}\mathbf{A}_{s''}^T\mathbf{B}_{s''}(\gamma_{s'',t+1}\boldsymbol{\rho}_{s'',t+1}) \\ & + \mathbf{C}_s^T\mathbf{D}_s(\mathbf{y}_t - \mathbf{c}_s) + \mathbf{B}_s\mathbf{a}_s - \sum_{s''}\eta_{ss'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{a}_{s''}\ , \end{aligned} \tag{9.80}$$

where (9.49) has been used in the above manipulation. Now define

$$\boldsymbol{\chi}_t = [\gamma_{1,t}\boldsymbol{\rho}_{1,t} \quad \gamma_{2,t}\boldsymbol{\rho}_{2,t} \quad \cdots \quad \gamma_{N,t}\boldsymbol{\rho}_{N,t}]^T, \tag{9.81}$$

and collect all ρ's at a given time t, we have

$$\begin{bmatrix} \frac{\mathbf{\Gamma}_{1,t}}{\gamma_{1,t}} & & & \\ & \frac{\mathbf{\Gamma}_{2,t}}{\gamma_{2,t}} & & \\ & & \ddots & \\ & & & \frac{\mathbf{\Gamma}_{N,t}}{\gamma_{N,t}} \end{bmatrix} \boldsymbol{\chi}_t = \begin{bmatrix} \frac{1}{\gamma_{1,t}}\mathbf{B}_1\mathbf{A}_1\eta_{11,t} & \frac{1}{\gamma_{1,t}}\mathbf{B}_1\mathbf{A}_1\eta_{21,t} & \cdots & \frac{1}{\gamma_{1,t}}\mathbf{B}_1\mathbf{A}_1\eta_{N1,t} \\ \frac{1}{\gamma_{2,t}}\mathbf{B}_2\mathbf{A}_2\eta_{12,t} & \frac{1}{\gamma_{2,t}}\mathbf{B}_2\mathbf{A}_2\eta_{22,t} & \cdots & \frac{1}{\gamma_{2,t}}\mathbf{B}_2\mathbf{A}_2\eta_{N2,t} \\ \vdots & \vdots & & \vdots \\ \frac{1}{\gamma_{N,t}}\mathbf{B}_N\mathbf{A}_N\eta_{1N,t} & \frac{1}{\gamma_{N,t}}\mathbf{B}_N\mathbf{A}_N\eta_{2N,t} & \cdots & \frac{1}{\gamma_{N,t}}\mathbf{B}_N\mathbf{A}_N\eta_{NN,t} \end{bmatrix} \boldsymbol{\chi}_{t-1}$$

$$+ \begin{bmatrix} \frac{1}{\gamma_{1,t+1}}\eta_{11,t+1}\mathbf{A}_1^T\mathbf{B}_1 & \frac{1}{\gamma_{2,t+1}}\eta_{12,t+1}\mathbf{A}_2^T\mathbf{B}_2 & \cdots & \frac{1}{\gamma_{N,t+1}}\eta_{1N,t+1}\mathbf{A}_N^T\mathbf{B}_N \\ \frac{1}{\gamma_{1,t+1}}\eta_{21,t+1}\mathbf{A}_1^T\mathbf{B}_1 & \frac{1}{\gamma_{2,t+1}}\eta_{22,t+1}\mathbf{A}_2^T\mathbf{B}_2 & \cdots & \frac{1}{\gamma_{N,t+1}}\eta_{2N,t+1}\mathbf{A}_N^T\mathbf{B}_N \\ \vdots & \vdots & & \vdots \\ \frac{1}{\gamma_{1,t+1}}\eta_{N1,t+1}\mathbf{A}_1^T\mathbf{B}_1 & \frac{1}{\gamma_{2,t+1}}\eta_{N2,t+1}\mathbf{A}_2^T\mathbf{B}_2 & \cdots & \frac{1}{\gamma_{N,t+1}}\eta_{NN,t+1}\mathbf{A}_N^T\mathbf{B}_N \end{bmatrix} \boldsymbol{\chi}_{t+1}$$

$$+ \begin{bmatrix} \mathbf{C}_1^T\mathbf{D}_1(\mathbf{y}_t - \mathbf{c}_1) + \mathbf{B}_1\mathbf{a}_1 - \sum_{s''}\eta_{1s'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{a}_{s''} \\ \vdots \\ \mathbf{C}_N^T\mathbf{D}_N(\mathbf{y}_t - \mathbf{c}_N) + \mathbf{B}_N\mathbf{a}_N - \sum_{s''}\eta_{Ns'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{a}_{s''} \end{bmatrix}. \tag{9.82}$$

Therefore, by defining

$$\mathbf{U}_t = \begin{bmatrix} \frac{\mathbf{\Gamma}_{1,t}}{\gamma_{1,t}} & & & \\ & \frac{\mathbf{\Gamma}_{2,t}}{\gamma_{2,t}} & & \\ & & \ddots & \\ & & & \frac{\mathbf{\Gamma}_{N,t}}{\gamma_{N,t}} \end{bmatrix}, \tag{9.83a}$$

$$\mathbf{V}_t = \begin{bmatrix} \frac{1}{\gamma_{1,t+1}}\eta_{11,t+1}\mathbf{A}_1^T\mathbf{B}_1 & \frac{1}{\gamma_{2,t+1}}\eta_{12,t+1}\mathbf{A}_2^T\mathbf{B}_2 & \cdots & \frac{1}{\gamma_{N,t+1}}\eta_{1N,t+1}\mathbf{A}_N^T\mathbf{B}_N \\ \frac{1}{\gamma_{1,t+1}}\eta_{21,t+1}\mathbf{A}_1^T\mathbf{B}_1 & \frac{1}{\gamma_{2,t+1}}\eta_{22,t+1}\mathbf{A}_2^T\mathbf{B}_2 & \cdots & \frac{1}{\gamma_{N,t+1}}\eta_{2N,t+1}\mathbf{A}_N^T\mathbf{B}_N \\ \vdots & \vdots & & \vdots \\ \frac{1}{\gamma_{1,t+1}}\eta_{N1,t+1}\mathbf{A}_1^T\mathbf{B}_1 & \frac{1}{\gamma_{2,t+1}}\eta_{N2,t+1}\mathbf{A}_2^T\mathbf{B}_2 & \cdots & \frac{1}{\gamma_{N,t+1}}\eta_{NN,t+1}\mathbf{A}_N^T\mathbf{B}_N \end{bmatrix}, \tag{9.83b}$$

$$\mathbf{e}_t = \begin{bmatrix} \mathbf{C}_1^T\mathbf{D}_1(\mathbf{y}_t - \mathbf{c}_1) + \mathbf{B}_1\mathbf{a}_1 - \sum_{s''}\eta_{1s'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{a}_{s''} \\ \vdots \\ \mathbf{C}_N^T\mathbf{D}_N(\mathbf{y}_t - \mathbf{c}_N) + \mathbf{B}_N\mathbf{a}_N - \sum_{s''}\eta_{Ns'',t+1}\mathbf{A}_{s''}^T\mathbf{B}_{s''}\mathbf{a}_{s''} \end{bmatrix}, \tag{9.83c}$$

we can convert (9.57) to the following general form

$$\mathbf{U}_t\boldsymbol{\chi}_t = \mathbf{V}_t\boldsymbol{\chi}_{t+1} + \mathbf{V}_{t-1}^T\boldsymbol{\chi}_{t-1} + \mathbf{e}_t, \tag{9.84}$$

where $\mathbf{U}_t$ is a block diagonal matrix that is symmetric and positive semi-definite. By further defining

$$\mathbf{e}_T = \begin{bmatrix} \mathbf{C}_1^T \mathbf{D}_1(\mathbf{y}_T - \mathbf{c}_1) + \mathbf{B}_1 \mathbf{a}_1 \\ \vdots \\ \mathbf{C}_N^T \mathbf{D}_N(\mathbf{y}_T - \mathbf{c}_N) + \mathbf{B}_N \mathbf{a}_N \end{bmatrix} \tag{9.85}$$

and $\boldsymbol{\chi}_{T+1} = 0$, (9.59) can also be fit into the general form of (9.84).

Next we focus on solving linear equations of the form (9.84). Observe that the solution $\boldsymbol{\chi}_t = \boldsymbol{\mu}_t$ to the set of linear vector equations defined by (9.84) is the mean, as well as the mode, of the joint Gaussian distribution $p(\mathbf{X})$ defined by

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{t=1}^{T} \exp[\Phi_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})], \tag{9.86}$$

where

$$\Phi_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1}) = -\frac{1}{2} \boldsymbol{\chi}_t^T \mathbf{U}_t \boldsymbol{\chi}_t + \boldsymbol{\chi}_t^T \mathbf{V}_t \boldsymbol{\chi}_{t+1} + \mathbf{e}_t^T \boldsymbol{\chi}_t \tag{9.87}$$

and Z is a normalization constant. This can be easily verified since setting

$$\begin{aligned} \frac{\partial \log p(\mathbf{X})}{\partial \boldsymbol{\chi}_t} &= \frac{\partial}{\partial \boldsymbol{\chi}_t} \big(-\frac{1}{2} \boldsymbol{\chi}_t^T \mathbf{U}_t \boldsymbol{\chi}_t + \boldsymbol{\chi}_t^T \mathbf{V}_t \boldsymbol{\chi}_{t+1} + \mathbf{e}_t^T \boldsymbol{\chi}_t + \boldsymbol{\chi}_{t-1}^T \mathbf{V}_{t-1} \boldsymbol{\chi}_t \big) \\ &= -\mathbf{U}_t \boldsymbol{\chi}_t + \mathbf{V}_t \boldsymbol{\chi}_{t+1} + \mathbf{V}_{t-1}^T \boldsymbol{\chi}_{t-1} + \mathbf{e}_t = 0 \end{aligned}$$

gives equation (9.84). We further observe that the joint Gaussian distribution (9.86) may arise from the following Markov process

$$p(\mathbf{X}) = \prod_{t=1}^{T} p(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_{t+1}) \tag{9.88}$$

where

$$p(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_{t+1}) = \exp[\Phi_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})] \cdot z_{t-1}(\boldsymbol{\chi}_t) \cdot \frac{1}{z_t(\boldsymbol{\chi}_{t+1})}\ , \tag{9.89}$$

and the normalization constant z's (functions of $\boldsymbol{\chi}$'s actually) are defined by

$$z_0(\boldsymbol{\chi}_1) = 1, \tag{9.90a}$$

$$z_t(\boldsymbol{\chi}_{t+1}) = \int d\boldsymbol{\chi}_t \exp[\Phi_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})] \cdot z_{t-1}(\boldsymbol{\chi}_t). \tag{9.90b}$$

It can be shown that the mean $\boldsymbol{\mu}_t$ of the above constructed Gaussian-Markov process can be computed by an efficient forward-backward procedure. Here we present the result, with derivation details provided in Appendix A.3. The forward pass is:

1. Initialization:

$$\boldsymbol{\Omega}_0 = \boldsymbol{\omega}_0 = \mathbf{0}. \tag{9.91}$$

2. Induction: for $t = 1, \cdots, T$

$$\boldsymbol{\Upsilon}_t = \mathbf{U}_t - \boldsymbol{\Omega}_{t-1}, \tag{9.92a}$$
$$\boldsymbol{\lambda}_t = \boldsymbol{\Upsilon}_t^{-1}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1}), \tag{9.92b}$$
$$\boldsymbol{\Lambda}_t = \boldsymbol{\Upsilon}_t^{-1}\mathbf{V}_t, \tag{9.92c}$$
$$\boldsymbol{\omega}_t = \mathbf{V}_t^T\boldsymbol{\lambda}_t, \tag{9.92d}$$
$$\boldsymbol{\Omega}_t = \mathbf{V}_t^T\boldsymbol{\Lambda}_t. \tag{9.92e}$$

And the backward pass is:

1. Initialization:

$$\boldsymbol{\mu}_{T+1} = \mathbf{0}. \tag{9.93}$$

2. Induction: for $t = T, \cdots, 1$

$$\boldsymbol{\mu}_t = \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t\boldsymbol{\mu}_{t+1}. \tag{9.94}$$

The Complete E Step

As typical, the variational equations derived in this section are coupled, with the equations for $\boldsymbol{\rho}_{s,t}$, $\boldsymbol{\Gamma}_{s,t}$ depend on $\eta_{s's,t}$, $\gamma_{s,t}$ and vice versa. These equations are solved iteratively starting from a random or more suitable initialization for $\boldsymbol{\rho}_{s,t}$ and $\boldsymbol{\Gamma}_{s,t}$ or $\eta_{s's,t}$ and $\gamma_{s,t}$. The solution is the set of sufficient statistics

$$\varphi = \left\{\boldsymbol{\rho}_{s,t}, \boldsymbol{\Gamma}_{s,t}, \eta_{ss',t}, \gamma_{s,t}\right\}, \tag{9.95}$$

which are moments of the variational posterior. The appropriate initialization scheme may be strongly problem-dependent, by using all the problem-specific knowledge available. Further discussions on initialization will be presented when describing simulation experiments in Section 9.6 and discussing the use of variational EM algorithms for speech recognition in the next chapter.

9.4 Model Parameter Learning for SSSM

We denote the learnable parameters of SSSM as $\mathbf{\Theta} = \{\mathbf{A}_s, \mathbf{a}_s, \mathbf{B}_s, \mathbf{C}_s, \mathbf{c}_s, \mathbf{D}_s, \mathbf{\Pi}\}$ and estimate them in a maximum likelihood (ML) sense as in classical EM algorithms, while the only difference is that since the exact likelihood for the SSSM is intractable, the approximate likelihood $\mathcal{F}$ is maximized. The derivation is straightforward but lengthy, and we only list the final parameter update equations in this section, with derivation details provided in Appendix A.4.

$$\begin{aligned}\mathbf{A}_s = &\left[\sum_{t=1}^{T}\boldsymbol{\rho}_{s,t}\Big(\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}^T\Big) - \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_t\Big)\Big(\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}^T\Big)\right]\cdot \\ &\left[\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}(\mathbf{\Gamma}_{s',t-1}^T + \boldsymbol{\rho}_{s',t-1}\boldsymbol{\rho}_{s',t-1}^T) - \right. \\ &\left.\frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\Big(\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}\Big)\Big(\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}\Big)^T\right]^{-1},\end{aligned} \tag{9.96}$$

$$\mathbf{a}_s = \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\left[\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t} - \mathbf{A}_s\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}\right], \tag{9.97}$$

$$\begin{aligned}\mathbf{B}_s^{-1} = \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\Bigg\{&\sum_{t=1}^{T}\gamma_{s,t}(\mathbf{\Gamma}_{s,t}^{-1} + \boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T) - \left[\sum_{t=1}^{T}\boldsymbol{\rho}_{s,t}\Big(\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}^T\Big)\right]\mathbf{A}_s^T \\ &- \Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\Big)\mathbf{a}_s^T - \mathbf{A}_s\left[\sum_{t=1}^{T}\Big(\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}\Big)\boldsymbol{\rho}_{s,t}^T\right] \\ &+ \mathbf{A}_s\left[\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\big(\mathbf{\Gamma}_{s',t-1}^{-1} + \boldsymbol{\rho}_{s',t-1}\boldsymbol{\rho}_{s',t-1}^T\big)\right]\mathbf{A}_s^T \\ &+ \mathbf{A}_s\Big(\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}\Big)\mathbf{a}_s^T - \mathbf{a}_s\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_t\Big)^T \\ &+ \mathbf{a}_s\Big(\sum_{t=1}^{T}\sum_{s'}\eta_{s's,t}\gamma_{s',t-1}\boldsymbol{\rho}_{s',t-1}\Big)^T\mathbf{A}_s^T + \Big(\sum_{t=1}^{T}\gamma_{s,t}\Big)\mathbf{a}_s\mathbf{a}_s^T\Bigg\},\end{aligned} \tag{9.98}$$

$$\begin{aligned}\mathbf{C}_s = &\left[\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t\boldsymbol{\rho}_{s,t}^T - \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\Big(\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t\Big)\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_t^T\Big)\right]\cdot \\ &\left[\sum_{t=1}^{T}\gamma_{s,t}\big(\boldsymbol{\Gamma}_{s,t}^{-1}+\boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T\big) - \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\Big)\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\Big)^T\right]^{-1},\end{aligned} \tag{9.99}$$

$$\mathbf{c}_s = \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\left(\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t - \mathbf{C}_s\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\right), \tag{9.100}$$

$$\begin{aligned}\mathbf{D}_s^{-1} = \frac{1}{\sum_{t=1}^{T}\gamma_{s,t}}\Bigg\{&\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t\mathbf{y}_t^T - \Big(\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t\boldsymbol{\rho}_{s,t}^T\Big)\mathbf{C}_s^T - \Big(\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t\Big)\mathbf{c}_s^T \\ &- \mathbf{C}_s\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\mathbf{y}_t^T\Big) + \mathbf{C}_s\Big[\sum_{t=1}^{T}\gamma_{s,t}\big(\boldsymbol{\Gamma}_{s,t}^{-1}+\boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T\big)\Big]\mathbf{C}_s^T \\ &+ \mathbf{C}_s\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\Big)\mathbf{c}_s^T - \mathbf{c}_s\Big(\sum_{t=1}^{T}\gamma_{s,t}\mathbf{y}_t\Big)^T \\ &+ \mathbf{c}_s\Big(\sum_{t=1}^{T}\gamma_{s,t}\boldsymbol{\rho}_{s,t}\Big)^T\mathbf{C}_s^T + \sum_{t=1}^{T}\gamma_{s,t}\mathbf{c}_s\mathbf{c}_s^T\Bigg\},\end{aligned} \tag{9.101}$$

$$\pi_{ij} = \frac{1}{\sum_{t=1}^{T}\gamma_{i,t-1}}\cdot\sum_{t=1}^{T}\gamma_{i,t-1}\eta_{ij,t}\ . \tag{9.102}$$

Also notice that the above equations are only valid for a single training example. If multiple training examples are present (as in most practical applications), the above formulas need to be modified by adding an extra summation over the training examples in a straightforward manner, as is done in Section 4.3 on page 76. The very similar update equations are omitted.

9.5 Hidden State Recovery of SSSM

It is necessary to estimate the hidden state sequences $\hat{\mathbf{X}}_{1:T}$ and $\hat{\mathbf{s}}_{1:T}$ from the data, which is quite often the very goal of applying SSSMs in practice. For the continuous states we

use the MMSE estimator, defined w.r.t. the variational posterior, to obtain

$$\hat{\mathbf{x}}_t = \int d\mathbf{X}_{1:T}\; q(\mathbf{X}_{1:T})\mathbf{x}_t = \sum_s \gamma_{s,t}\boldsymbol{\rho}_{s,t}\;. \tag{9.103}$$

For the discrete states the variational MAP estimate is

$$\begin{aligned}\hat{\mathbf{s}}_{1:T} &= \underset{\mathbf{s}_{1:T}}{\operatorname{argmax}} \int d\mathbf{X}_{1:T}\; q(\mathbf{s}_{1:T}, \mathbf{X}_{1:T}) \\ &= \underset{\mathbf{s}_{1:T}}{\operatorname{argmax}} \prod_{t=1}^{T} q(s_t \mid s_{t-1}),\end{aligned} \tag{9.104}$$

and it can be computed efficiently by a Viterbi algorithm [115] based on the variational posterior η's. The detailed procedure is:

1. Initialization:

$$V_1(s) = \pi_{0s}, \quad 1 \le s \le N, \tag{9.105}$$

$$W_1(s) = 0, \quad 1 \le s \le N. \tag{9.106}$$

2. Induction: for $t = 2, \cdots, T$

$$V_t(s) = \max_{s'}[V_{t-1}(s')\eta_{s's,t}], \tag{9.107}$$

$$W_t(s) = \underset{s'}{\operatorname{argmax}}[V_{t-1}(s')\eta_{s's,t}]. \tag{9.108}$$

3. Termination:

$$\text{The best score} = \max_s[V_T(s)], \tag{9.109}$$

$$s_T^* = \underset{s}{\operatorname{argmax}}[V_T(s)]. \tag{9.110}$$

4. Backtracking: for $t = T-1, \cdots, 1$

$$s_t^* = W_{t+1}(s_{t+1}^*), \tag{9.111}$$

and the most likely sequence is $\mathbf{s}^* = \{s_1^*, s_2^*, \cdots, s_T^*\}$.

9.6 Simulation Experiments

Extensive simulations have been carried out to verify the correctness and effectiveness of the above derived variational EM algorithm. Here a typical example is presented, which has four discrete states with the following model parameters:

$$\begin{aligned}
&\mathbf{A}_1 = 0.7,\ \mathbf{a}_1 = 0.6,\ \mathbf{B}_1 = 1000,\ \mathbf{C}_1 = [0.8\,0.3\,0.2]^T,\ \mathbf{c}_1 = -[3\,2\,1]^T,\\
&\mathbf{A}_2 = 0.8,\ \mathbf{a}_2 = 0.5,\ \mathbf{B}_2 = 4000,\ \mathbf{C}_2 = [1.0\,0.2\,0.1]^T,\ \mathbf{c}_2 = [-1\,0\,1]^T,\\
&\mathbf{A}_3 = 0.9,\ \mathbf{a}_3 = 0.18,\ \mathbf{B}_3 = 694.4,\ \mathbf{C}_3 = [0.5\,0.4\,0.2]^T,\ \mathbf{c}_3 = [0\,1\,2]^T,\\
&\mathbf{A}_4 = 0.6,\ \mathbf{a}_4 = 0.88,\ \mathbf{B}_4 = 1563,\ \mathbf{C}_4 = [0.1\,0.7\,0.8]^T,\ \mathbf{c}_4 = [1\,2\,3]^T,
\end{aligned}$$

where $\mathbf{D}$ is 100 times the identity matrix for all four states and $\mathbf{\Pi}$ is uniform.

Since the E step is an iterative process itself, we have to initialize $\boldsymbol{\rho}$ and $\mathbf{\Gamma}$ or η and γ, and a suitable initialization scheme may be very application dependent. In this simulation experiment we choose to initialize $\boldsymbol{\rho}$ and $\mathbf{\Gamma}$ as follows:

$$\boldsymbol{\rho}_{s,t} = w \cdot (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T(\mathbf{y}_t - \mathbf{c}_s) + (1-w)\cdot(\mathbf{I} - \mathbf{A}_s)\mathbf{a}_s\ , \tag{9.112}$$

$$\mathbf{\Gamma}_{s,t} = \mathbf{C}_s^T\mathbf{D}_s\mathbf{C}_s + \mathbf{B}_s + \mathbf{A}_s^T\mathbf{B}_s\mathbf{A}_s\ . \tag{9.113}$$

That is, we initialize $\boldsymbol{\rho}_{s,t}$ to be a weighted average of the pseudo-inverse estimation from $\mathbf{y}_t$ and the target value of $\mathbf{x}$ for each discrete state[6], where the weight w is determined by the ratio of $\|\mathbf{B}_s\|$ and $\|\mathbf{D}_s\|$

Fig. 9.7 tests the sensitivity of variational inference (E step) to levels of process and observation noise, which are measured by the precision matrices $\mathbf{B}$ and $\mathbf{D}$. In all cases, the discrete state sequence (indicated by vertical lines) is recovered correctly by Viterbi decoding on η's. The continuous states, more difficult to estimate than the hidden discrete states for this example since the discrete states are relatively far apart, are recovered well under moderate noise (a), degraded as expected as the noise levels increase (b,c), and remain reasonable even in severe noise (d).

Fig. 9.8 tests both the E step and M step. Notice that SSSMs possess a high degree of degeneracy if all the model parameters are unknown, *i.e.*, different sets of model parameters

[6]Recall that $\mathbf{x}$ will reach a target as long as $\mathbf{A}$ is convergent, which is probably the only case of interest for practical applications.

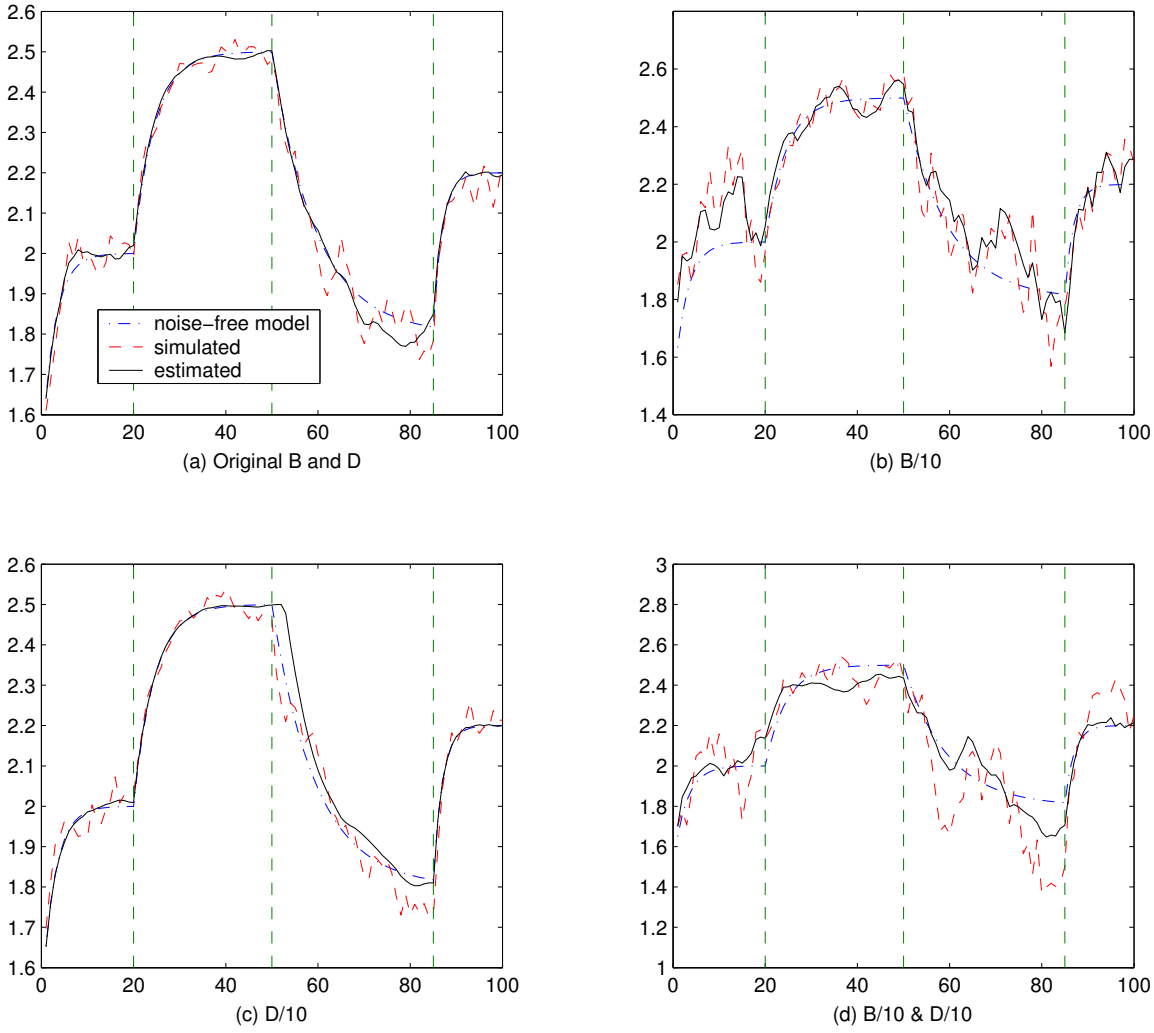

Figure 9.7: Hidden state recovery under different noise levels by variational inference.

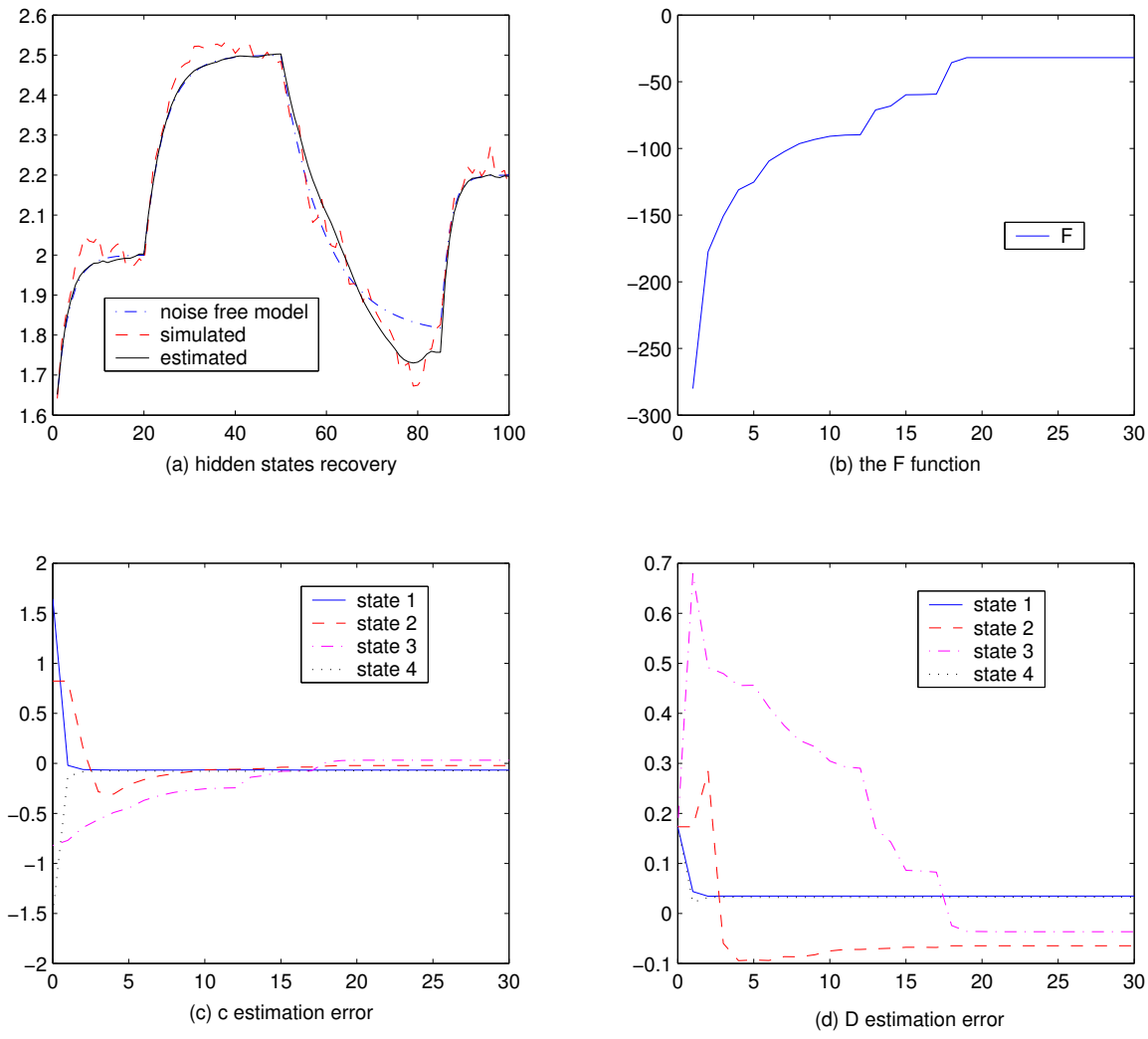

Figure 9.8: Model parameter estimation by variational EM.

can give rise to the same output distributions. To avoid such a problem, we choose to estimate $\mathbf{c}$ and $\mathbf{D}$ only while assuming other model parameters are known in this example. Given initial values $\mathbf{c}' = \mathbf{c} - 1$ and $\mathbf{D}' = \mathbf{D}/2$, it can be seen that the variational EM procedure works well: the hidden dynamics are recovered well (a) and $\mathcal{F}$ is monotonically increasing and quickly converging (b). The estimates of $\mathbf{c}$ and $\mathbf{D}$ are accurate, evidenced by the small error norms (c,d). Model degeneracies of SSSM cause little problem in applying it for practical applications, although some precautions may need to be taken. In practice, we may have enough problem-specific constraints so that the model is uniquely identifiable, or we may mainly care about the explanatory power of the model while the actual values of some or all model parameters are irrelevant. On the other hand, however, rigorous theoretical studies on system identifiability of SSSMs seems to be rare[7], and the problem has not been completely solved yet.

In summary, the typical simulation example presented above verifies the effectiveness and quick convergence of the variational EM algorithm and shows its promise for practical applications.

[7]See a recent study by Vidal *et al.* [240].

Chapter 10

Special Considerations and Experiments for ASR

This chapter discusses a number of important issues when the variational EM algorithm developed in the previous chapter is applied to speech recognition, and shows a few preliminary experiments towards developing a full-fledged speech recognizer based on HDM.

10.1 Some ASR Related Issues

Two important issues related to speech recognition are discussed in this section: one is an alternative decoding scheme of SSSM that is completely parallel to the Viterbi decoding of HMM; the other is using a piecewise linear mapping, such as the VTR-to-acoustics mapping developed in Section 7.1, to replace the pure linear mapping of the original SSSM.

10.1.1 An Alternative Decoding Scheme

When SSSMs are applied to speech recognition, the ultimate goal is to recover the underlying discrete state sequence $\mathbf{s}$ of an acoustic observation, after properly training the model. Such a decoding problem is arguably the most challenging component in a speech recognizer. In the previous chapter we have seen that decoding of SSSM can be done by a Viterbi algorithm based on the variational posterior η's. Here we derive an alternative

decoding scheme based on f's as defined in (9.68), and discuss its importance for speech recognition.

Recall that we have used the following functional $\mathcal{F}$ to lower-bound and approximate the intractable exact likelihood of SSSM:

$$\mathcal{F}[q] = \sum_{\mathbf{s}_{1:T}} \int d\mathbf{X}_{1:T}\; q(\mathbf{X}_{1:T} \mid \mathbf{s}_{1:T}) q(\mathbf{s}_{1:T}) \cdot \\ \big[\log p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}, \mathbf{s}_{1:T}) - \log q(\mathbf{X}_{1:T} \mid \mathbf{s}_{1:T}) - \log q(\mathbf{s}_{1:T})\big]. \tag{10.1}$$

Define the expectation over $\mathbf{X}_{1:T}$ given $\mathbf{s}_{1:T}$ as

$$f(\mathbf{Y}_{1:T} \mid \mathbf{s}_{1:T}) = \int d\mathbf{X}_{1:T}\; q(\mathbf{X}_{1:T} \mid \mathbf{s}_{1:T}) \big[\log p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}, \mathbf{s}_{1:T}) - \log q(\mathbf{X}_{1:T} \mid \mathbf{s}_{1:T})\big], \tag{10.2}$$

so that

$$\mathcal{F}[q] = \sum_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) \big[f(\mathbf{Y}_{1:T} \mid \mathbf{s}_{1:T}) - \log q(\mathbf{s}_{1:T})\big]. \tag{10.3}$$

Setting $\partial \mathcal{F} / \partial q(\mathbf{s}_{1:T})$ to zero to maximize $\mathcal{F}$ w.r.t. $q(\mathbf{s}_{1:T})$, we have

$$q(\mathbf{s}_{1:T}) \propto \exp\big[f(\mathbf{Y}_{1:T} \mid \mathbf{s}_{1:T})\big]. \tag{10.4}$$

It is also easy to see that $f(\mathbf{Y}_{1:T} \mid \mathbf{s}_{1:T})$ can be computed as a summation over the frames, i.e.,

$$f(\mathbf{Y}_{1:T} \mid \mathbf{s}_{1:T}) = \sum_{t=1}^{T} f_t(s_{t-1}, s_t), \tag{10.5}$$

where $f_t(s_{t-1}, s_t)$ or $f_{s's,t}$ is exactly as defined in (9.68). Therefore, the optimal discrete state sequence $\hat{\mathbf{s}}_{1:T}$ is

$$\begin{aligned} \hat{\mathbf{s}}_{1:T} &= \operatorname*{argmax}_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) = \operatorname*{argmax}_{\mathbf{s}_{1:T}} \log\big[q(\mathbf{s}_{1:T})\big] \\ &= \operatorname*{argmax}_{\mathbf{s}_{1:T}} f(\mathbf{Y}_{1:T} \mid \mathbf{s}_{1:T}) \\ &= \operatorname*{argmax}_{\mathbf{s}_{1:T}} \sum_{t=1}^{T} f_t(s_{t-1}, s_t), \end{aligned} \tag{10.6}$$

which can be computed efficiently by the following Viterbi algorithm:

1. Initialization:

$$V_1(s) = \log \pi_{0s}, \quad 1 \leq s \leq N, \tag{10.7}$$

$$W_1(s) = 0, \quad 1 \leq s \leq N. \tag{10.8}$$

2. Induction: for $t = 2, \cdots, T$

$$V_t(s) = \max_{s'}[V_{t-1}(s') + f_{s's,t}], \tag{10.9}$$

$$W_t(s) = \operatorname*{argmax}_{s'}[V_{t-1}(s') + f_{s's,t}]. \tag{10.10}$$

3. Termination:

$$\text{The best score} = \max_{s}[V_T(s)], \tag{10.11}$$

$$s_T^* = \operatorname*{argmax}_{s}[V_T(s)]. \tag{10.12}$$

4. Backtracking: for $t = T-1, \cdots, 1$

$$s_t^* = W_{t+1}(s_{t+1}^*), \tag{10.13}$$

and the most likely sequence is $\mathbf{s}^* = \{s_1^*, s_2^*, \cdots, s_T^*\}$.

Comparing to the standard HMM Viterbi decoding algorithm listed on page 150, $f_t' = f_t - \log \pi_{s's}$ plays the same role as the frame-based log likelihood of a HMM, and may be interpreted as the variational or approximate frame-based likelihood of the SSSM. Notice that this alternative Viterbi decoding algorithm is completely parallel to the Viterbi decoding of HMM since f_t, similar to the frame-based likelihood in HMM, can be computed in a frame by frame manner. However, unlike the frame-based likelihood of HMM, f_t depends on both the current and previous discrete states, not just the current state, and the whole observation sequence $\mathbf{Y}_{1:T}$, not just the current observation $\mathbf{y}_t$, due to the more complicated model structure of SSSM.

Although the Viterbi decoding algorithms on η_t and f_t are mathematically equivalent, there is an important difference between them: f_t can be broken down into a number of meaningful terms and computed on the fly while η_t cannot. Such a difference turns

out to be crucial for speech recognition. For example, the state transition matrix $\mathbf{\Pi}$ of the SSSM, corresponding to the language model when applied to speech recognition, is typically trained separately and context-dependent (thinking of the n-gram language model as an example). Such a complex and time-varying transition matrix can not be directly incorporated when computing η_t, but can be easily absorbed when calculating f_t frame by frame: we simply replace the original time-invariant prior $\pi_{s's}$ by this separately trained context-dependent language model. As a second example, a heuristic insertion penalty is often needed in practical speech recognizers to penalize frequent switchings among different discrete states, this can again be simply added as an extra term when computing f_t frame by frame, but difficult to include when computing η_t. In summary, the variational Viterbi decoding algorithm based on f_t effectively turns a SSSM into an equivalent HMM so that all the practical and efficient decoding techniques developed in the past for HMM based speech recognizers can be directly ported to SSSM or HDM based speech recognizers.

10.1.2 Effect of Piecewise Linear Mapping

The pure linear mapping from the hidden continuous state $\mathbf{x}_t$ to the measurement $\mathbf{y}_t$ ensures that all the continuous PDFs of the SSSM are Gaussians or mixtures of Gaussians. When a piecewise linear mapping $\mathcal{P}[\cdot]$ is adopted, it introduces a truncating effect, *e.g.*, the variational posterior $q(\mathbf{x}_t)$ will become mixtures of truncated Gaussians. Rigorously speaking, there is no guarantee that the E step will converge anymore due to this truncating effect. However, we may seek to minimize the truncating effect in practice. When working with real speech data, our preliminary experience shows that the convergence problem is not severe as long as the initialization is done carefully. When the underlying phoneme sequence and some preliminary segmentations are available, as is typically the case in the training phase, we choose to initialize η and γ according to the known phoneme sequence and boundaries. On the other hand, when the underlying phoneme sequence and boundaries are not known as in the testing or recognition phase, we have used the Kalman smoother based VTR tracker developed in Chapter 7 to initialize $\boldsymbol{\rho}$ and $\mathbf{\Gamma}$ and obtained desirable results in the inference step.

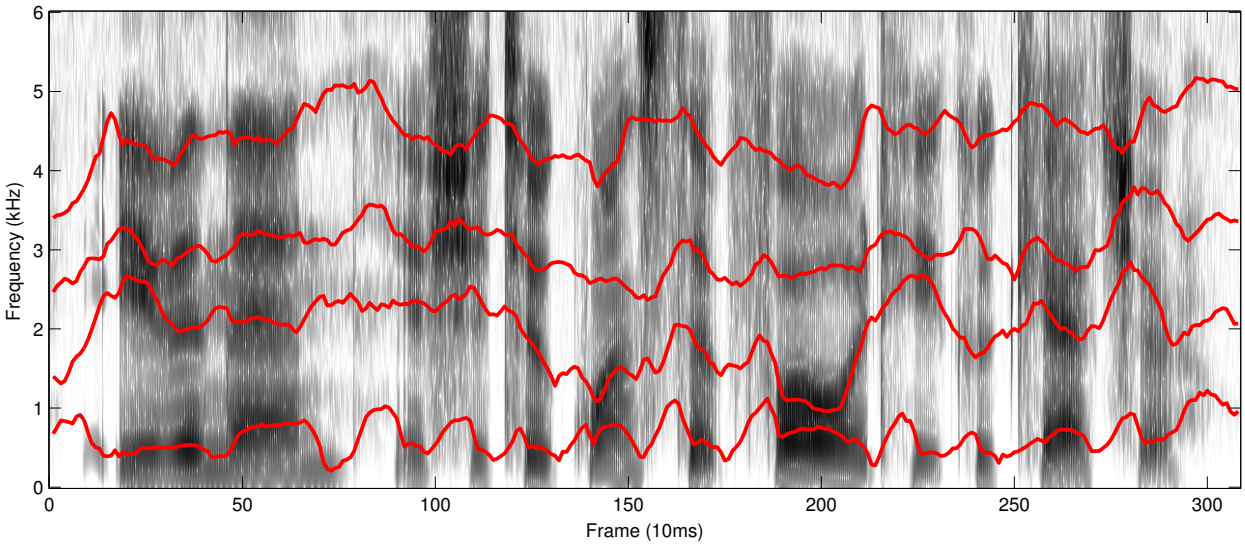

Figure 10.1: VTR tracking for a typical TIMIT sentence with variational inference.

10.2 Some Speech Examples

Various implementations towards practical speech applications have been attempted at different stages of developing the variational EM algorithms. Fig. 10.1 shows a VTR tracking example of a TIMIT sentence. This example uses the phoneme sequence and segmentation provided by the TIMIT database to initialize η and γ, and obtained good VTR tracking results not only in the clear, vocalized regions, but also in the more difficult consonant regions due to the built-in smoothness constraint of the model. It also verifies the implementation correctness and effectiveness of our algorithm when applied to real speech data.

A number of small scale speech recognition experiments have also been preformed on the Switchboard database, which contains large amount of spontaneous telephone speech and remains one of the most challenging tasks for ASR so far[1]. The next example is commonly known as an *N-best rescoring* experiment in speech recognition, which is a quick way to test novel approaches without performing the expensive decoding step. Here the model is first trained on a small subset of Switchboard data (which only contains 50

[1]This has mostly been my intern work at Microsoft Research.

Hypothesis #1:	no no it doesn't work on every issue (reference)
Hypothesis #2:	**knew knew** it doesn't work on every issue
Hypothesis #3:	no no it doesn't work **in** every issue
Hypothesis #4:	no no it doesn't work on **each** issue
Hypothesis #5:	no no it doesn't work on every **easy**

Table 10.1: Five hypotheses of a test sentence, with the correct one marked as reference.

	With initial phone boundary	With adjusted phone boundary
Hypothesis #1:	793.99	906.89
Hypothesis #2:	664.16	846.44
Hypothesis #3:	767.31	898.09
Hypothesis #4:	757.39	887.56
Hypothesis #5:	734.29	858.99

Table 10.2: Likelihood score of each hypothesis: initial phone boundaries are provided by HMM forced-alignment while adjusted phone boundaries are obtained by variational inference.

male utterances), and test on another small subset of utterances by computing the total likelihood[2] on a number of confusing hypotheses for each utterance. It confirms that our new HDM based recognizer is able to pick up the correct hypothesis for the small test set, and a typical example is shown in Tables 10.1 and 10.2. Table 10.1 lists the five hypotheses to be re-scored with the correct one marked as reference and Table 10.2 lists the likelihood score of each hypothesis under two different conditions, where it can be observed that the correct hypothesis consistently receive the highest score. A number of small-scale phone recognition experiments have also been performed on subsets of the Switchboard database but they failed to produce desirable results. Due to the complexity and limited information provided by the Switchboard Database and the large number of implementation details involved in the early-stage development of the HDM based speech recognizer, it turns out to be very difficult to carry out careful debugging and tuning to achieve better performance. Therefore, future work on the HDM based speech recognizer

[2]More accurately, the functional $\mathcal{F}$ in our variational approach.

will concentrate on the much simpler TIMIT database, and will be outlined in the next chapter.

Part IV

Overall Conclusions

Chapter 11

Summary and Future Work

This chapter concludes the thesis with summaries and discussions of future work.

11.1 Summary of Contributions

The contributions of this thesis can be summarized as follows:

- A careful study on the articulatory dynamics which reveals that they *can not* be captured by simple linear dynamic systems at a global level.
- A careful study on the VTR dynamics which verifies that they *can* be captured by simple linear dynamic systems at a global level.
- The design of a highly accurate and efficient piecewise linear mapping from VTR dynamics to LPC-cepstrum coefficients.
- The development of a novel and highly accurate VTR tracking method based on active image contours.
- The development of a fast and accurate VTR tracking method based on a simplified HDM of speech and Kalman smoothing.

- Invention of novel and powerful variational EM algorithms that facilitates the use of HDM in speech recognition applications and also contributes significantly to the machine learning community in general.

11.2 Future Research Directions

A number of interesting and highly desirable future research directions have been opened up by the original work performed in this thesis. Most of them are related to Part III of the thesis and some will be actively pursued after the thesis writing. They are briefly listed as follows.

- Evaluation of VTR tracking by B-spline snakes and simulated annealing on various types of speech data, such as those under noisy conditions. Since a global optimization method has been applied in our approach, we expect to achieve good robustness under different conditions, especially those with large number of spurious local optima in the spectrogram, either introduced by noise or other sources.

- Further study on the novel variational EM algorithms developed in this thesis. This includes detailed comparison study with other existing techniques, such as alternative variational approaches and sampling based methods, and various possible improvements to better suit practical needs.

- Variational techniques for nonlinear SSSMs. This is a very challenging problem since in general nonlinearity will introduce non-Gaussian distributions. In fact, this is a very challenging problem even for nonlinear state-space models without switching [234]. The success may lie in the integration of domain-specific knowledge to simplify the problem.

- Solid, step by step implementation and evaluation of HDM based speech recognizers. Both our past experience and other's work [210] indicate that although HDM is promising, lots of careful work must to be done in order to bring it close and possibly beat the performance of state-of-the-art HMMs. Future research work includes step

by step development on the TIMIT phone recognition task and performance comparison with increasingly complex HMM structures to gain insights and experience on how to build high-performance practical HDM based speech recognizers.

- Applying the variational EM algorithms developed in this thesis to other application domains where SSSMs have been found to be useful, such as in control, machine vision and financial analysis.

Appendix A

Derivations

A.1 Exact Inference from a Variational Principle

In this section we concentrate on the inference or E step by assuming all the model parameters are known, and use variational principles to show that the exact posterior $p(\mathbf{X} \mid \mathbf{Y})$ maximizes the functional $\mathcal{F}$. By removing dependency on model parameters, $\mathcal{F}$ becomes

$$\mathcal{F}(q) = \int_{\mathbf{X}} q(\mathbf{X}) \left[\log p(\mathbf{X}, \mathbf{Y}) - \log q(\mathbf{X})\right] \, d\mathbf{X}. \tag{A.1}$$

In order for q to truly qualify as a probability distribution, it has to further satisfy the normalization constraint, *i.e.*,

$$\int_{\mathbf{X}} q(\mathbf{X}) \, d\mathbf{X} = 1. \tag{A.2}$$

Therefore, Lagrange multiplier method is adopted to optimize $\mathcal{F}$ subject to the normalization constriant[1], and the new objective function is

$$\mathcal{O}(q, \lambda) = \int_{\mathbf{X}} q(\mathbf{X}) \left[\log p(\mathbf{X}, \mathbf{Y}) - \log q(\mathbf{X})\right] \, d\mathbf{X} + \lambda \left[\int_{\mathbf{X}} q(\mathbf{X}) \, d\mathbf{X} - 1\right]. \tag{A.3}$$

[1]More rigorously, q also has to be nonnegative everywhere; however, such a further constraint turns out to be unnecessary since the final q resulting from the free form optimization will automatically satisfy this condition.

Take the functional derivative of $\mathcal{O}$ w.r.t. $q(\mathbf{X})$ and set it to zero:

$$\frac{\delta \mathcal{O}}{\delta q(\mathbf{X})} = \log p(\mathbf{X}, \mathbf{Y}) - \log q(\mathbf{X}) - 1 + \lambda = 0, \tag{A.4}$$

$$\Rightarrow \; q(\mathbf{X}) = \frac{p(\mathbf{X}, \mathbf{Y})}{\exp(1-\lambda)}. \tag{A.5}$$

Take ordinary derivative of $\mathcal{O}$ w.r.t. λ and set it to zero

$$\frac{\partial \mathcal{O}}{\partial \lambda} = \int_{\mathbf{X}} q(\mathbf{X})\, d\mathbf{X} - 1 = 0. \tag{A.6}$$

Substitute (A.5) into (A.6)

$$\int_{\mathbf{X}} \frac{p(\mathbf{X}, \mathbf{Y})}{\exp(1-\lambda)} d\mathbf{X} - 1 = 0, \tag{A.7}$$

$$\Rightarrow \; \exp(1-\lambda) = \int_{\mathbf{X}} p(\mathbf{X}, \mathbf{Y})\, d\mathbf{X}. \tag{A.8}$$

Substituting (A.8) back into (A.5), we finally obtain

$$q(\mathbf{X}) = \frac{p(\mathbf{X}, \mathbf{Y})}{\int_{\mathbf{X}} p(\mathbf{X}, \mathbf{Y})\, d\mathbf{X}} = \frac{p(\mathbf{X}, \mathbf{Y})}{p(\mathbf{Y})} = p(\mathbf{X} \mid \mathbf{Y}). \tag{A.9}$$

Equation (A.9) not only shows that the exact posterior $p(\mathbf{X} \mid \mathbf{Y})$ maximizes $\mathcal{F}$ under the normalization constraint, but also indicates that optimizing $\mathcal{F}$ subsumes the well-known Bayes' rule.

A.2 Full Equation of Exact Inference for SSM

This section derives the full equation for the exact mean of the state posterior distribution $p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})$. First notice that the state posterior can be computed by

$$\log p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T}) = \log p(\mathbf{Y}_{1:T} \mid \mathbf{X}_{1:T}) + \log p(\mathbf{X}_{1:T}) - \log p(\mathbf{Y}_{1:T}). \tag{A.10}$$

The right hand side of the above equation is quadratic in $\mathbf{X}_{1:T}$ since both $\log p(\mathbf{Y}_{1:T} \mid \mathbf{X}_{1:T})$ and $\log p(\mathbf{X}_{1:T})$ are Gaussians while $\log p(\mathbf{Y}_{1:T})$ merely serves as a normalization constant, so must be the left hand side. Hence, $p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})$ can be nothing but a Gaussian

distribution. Second, for a Gaussian distribution, which is unimodal, its mean is the same as its mode. Therefore, the full equation of the exact mean can be obtained by finding the maximum of the state posterior $p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})$, or equivalently, its logarithm.

Ignoring the normalization constant $\log p(\mathbf{Y}_{1:T})$, which is irrelevant to maximization, the logarithm of the exact state posterior becomes

$$\begin{aligned}
\log p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T}) &= \log p(\mathbf{Y}_{1:T} \mid \mathbf{X}_{1:T}) + \log p(\mathbf{X}_{1:T}) \\
&= \log\Big[\prod_{t=1}^{T} p(\mathbf{y}_t \mid \mathbf{x}_t)\Big] + \log\Big[\prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\Big] \\
&= \sum_{t=1}^{T}\Big[\log \mathcal{N}(\mathbf{y}_t \mid \mathbf{C}\mathbf{x}_t + \mathbf{c}, \mathbf{D}) + \log \mathcal{N}(\mathbf{x}_t \mid \mathbf{A}\mathbf{x}_{t-1} + \mathbf{a}, \mathbf{B})\Big] \\
&= \sum_{t=1}^{T}\Big[\frac{1}{2}\log\Big|\frac{\mathbf{D}}{2\pi}\Big| - \frac{1}{2}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c})^T\mathbf{D}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c}) + \frac{1}{2}\log\Big|\frac{\mathbf{B}}{2\pi}\Big| \\
&\qquad - \frac{1}{2}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a})^T\mathbf{B}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a})\Big]. \qquad \text{(A.11)}
\end{aligned}$$

When $t \neq T$,

$$\begin{aligned}
\frac{\partial \log p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})}{\partial \mathbf{x}_t} &= \frac{\partial}{\partial \mathbf{x}_t}\Big[\frac{1}{2}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c})^T\mathbf{D}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c}) \\
&\qquad - \frac{1}{2}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a})^T\mathbf{B}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a}) \\
&\qquad - \frac{1}{2}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{a})^T\mathbf{B}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{a})\Big] \\
&= \mathbf{C}^T\mathbf{D}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t - \mathbf{c}) - \mathbf{B}(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{a}) \\
&\quad + \mathbf{A}^T\mathbf{B}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{a}) \\
&= \mathbf{B}\mathbf{A}\mathbf{x}_{t-1} + \mathbf{A}^T\mathbf{B}\mathbf{x}_{t+1} + \mathbf{C}^T\mathbf{D}(\mathbf{y}_t - \mathbf{c}) + \mathbf{B}\mathbf{a} - \mathbf{A}^T\mathbf{B}\mathbf{a} \\
&\quad - (\mathbf{C}^T\mathbf{D}\mathbf{C} + \mathbf{B} + \mathbf{A}^T\mathbf{B}\mathbf{A})\mathbf{x}_t. \qquad \text{(A.12)}
\end{aligned}$$

Similarly when $t = T$,

$$\frac{\partial \log p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})}{\partial \mathbf{x}_T} = \mathbf{B}\mathbf{A}\mathbf{x}_{T-1} + \mathbf{C}^T\mathbf{D}(\mathbf{y}_T - \mathbf{c}) + \mathbf{B}\mathbf{a} - (\mathbf{C}^T\mathbf{D}\mathbf{C} + \mathbf{B})\mathbf{x}_T. \qquad \text{(A.13)}$$

The exact mean of the state posterior can be obtained by setting (A.12) and (A.13) to zero for all t, and it is obvious that this results in exactly the same set of equations as the

variational smoother (9.24)-(9.27), which justifies the empirical result that the variational smoother gives the exact state mean for a SSM.

The full equations for the exact precision matrix of the state posterior can be similarly obtained by setting the second derivatives of $\log p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})$ to zero. This rather lengthy derivation is skipped, but to see why the exact precision matrix is different from the variational approximation, one only needs to note that there are nonzero cross terms in the form of $\partial \log p(\mathbf{X}_{1:T} \mid \mathbf{Y}_{1:T})/\partial \mathbf{x}_t \partial \mathbf{x}_{t-1}$ in the exact precision matrix which cannot be accounted for by a completely factorized variational posterior.

A.3 A Forward-Backward Algorithm of Probability Propagation

First it is straightforward to show by induction that $\log z_t(\boldsymbol{\chi}_{t+1})$ is quadratic in $\boldsymbol{\chi}_{t+1}$ (although it turns out to be a rather lengthy integral exercise), and we can express it generally as

$$\log z_t(\boldsymbol{\chi}_{t+1}) = \frac{1}{2}\boldsymbol{\chi}_{t+1}^T \boldsymbol{\Omega}_t \boldsymbol{\chi}_{t+1} + \boldsymbol{\omega}_t^T \boldsymbol{\chi}_{t+1} + \epsilon_t \ . \tag{A.14}$$

The initialization $z_0(\boldsymbol{\chi}_1) = 1$ implies that $\boldsymbol{\Omega}_0 = \boldsymbol{\omega}_0 = \epsilon_0 = 0$, which subsumes (9.91).

Next we define

$$g_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1}) = \Phi_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1}) + \log z_{t-1}(\boldsymbol{\chi}_t), \tag{A.15}$$

so that

$$z_t(\boldsymbol{\chi}_{t+1}) = \int d\boldsymbol{\chi}_t \ \exp\big[g_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})\big], \tag{A.16}$$

$$p(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_{t+1}) = \frac{1}{z_t(\boldsymbol{\chi}_{t+1})} \exp\big[g_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})\big]. \tag{A.17}$$

Notice that $g_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})$ is also quadratic in $\boldsymbol{\chi}_t$, and we can expand it about its maximum

$\boldsymbol{\chi}_t = \bar{\boldsymbol{\chi}}_t$ by completing a perfect square

$$\begin{aligned} g_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1}) &= \Phi_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1}) + \log z_{t-1}(\boldsymbol{\chi}_t) \\ &= -\frac{1}{2}\boldsymbol{\chi}_t^T \mathbf{U}_t \boldsymbol{\chi}_t + \boldsymbol{\chi}_t^T \mathbf{V}_t \boldsymbol{\chi}_{t+1} + \mathbf{e}_t^T \boldsymbol{\chi}_t + \frac{1}{2}\boldsymbol{\chi}_t^T \boldsymbol{\Omega}_{t-1} \boldsymbol{\chi}_t + \boldsymbol{\omega}_{t-1}^T \boldsymbol{\chi}_t + \epsilon_{t-1} \\ &= -\frac{1}{2}\boldsymbol{\chi}_t^T (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})\boldsymbol{\chi}_t + \boldsymbol{\chi}_t^T (\mathbf{V}_t \boldsymbol{\chi}_{t+1} + \mathbf{e}_t + \boldsymbol{\omega}_{t-1}) + \epsilon_{t-1} \\ &= -\frac{1}{2}\big[\boldsymbol{\chi}_t - (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1} + \mathbf{V}_t \boldsymbol{\chi}_{t+1})\big]^T (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})\big[\cdots\big] \\ &\quad + \frac{1}{2}\big[(\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1} + \mathbf{V}_t \boldsymbol{\chi}_{t+1})\big]^T (\mathbf{e}_t + \boldsymbol{\omega}_{t-1} + \mathbf{V}_t \boldsymbol{\chi}_{t+1}) + \epsilon_{t-1} \\ &= -\frac{1}{2}(\boldsymbol{\chi}_t - \bar{\boldsymbol{\chi}}_t)^T \boldsymbol{\Upsilon}_t (\boldsymbol{\chi}_t - \bar{\boldsymbol{\chi}}_t) + g_t(\bar{\boldsymbol{\chi}}_t, \boldsymbol{\chi}_{t+1}), \end{aligned} \tag{A.18}$$

where the maximum $\bar{\boldsymbol{\chi}}_t$ is

$$\begin{aligned} \bar{\boldsymbol{\chi}}_t &= (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1} + \mathbf{V}_t \boldsymbol{\chi}_{t+1}) \\ &= \boldsymbol{\lambda}_t + \boldsymbol{\Lambda}_t \boldsymbol{\chi}_{t+1}, \end{aligned} \tag{A.19}$$

and $\boldsymbol{\Upsilon}_t$, $\boldsymbol{\lambda}_t$ and $\boldsymbol{\Lambda}_t$ are as defined in (9.92a-c). Integrating (A.16) over $\boldsymbol{\chi}_t$, we obtain

$$\begin{aligned} z_t(\boldsymbol{\chi}_{t+1}) &= \int d\boldsymbol{\chi}_t \ \exp\big[-\frac{1}{2}(\boldsymbol{\chi}_t - \bar{\boldsymbol{\chi}}_t)^T \boldsymbol{\Upsilon}_t (\boldsymbol{\chi}_t - \bar{\boldsymbol{\chi}}_t) + g_t(\bar{\boldsymbol{\chi}}_t, \boldsymbol{\chi}_{t+1})\big] \\ &= \exp\big[g_t(\bar{\boldsymbol{\chi}}_t, \boldsymbol{\chi}_{t+1})\big] \cdot \int d\boldsymbol{\chi}_t \ \exp\big[-\frac{1}{2}(\boldsymbol{\chi}_t - \bar{\boldsymbol{\chi}}_t)^T \boldsymbol{\Upsilon}_t (\boldsymbol{\chi}_t - \bar{\boldsymbol{\chi}}_t)\big] \\ &= \exp\big[g_t(\bar{\boldsymbol{\chi}}_t, \boldsymbol{\chi}_{t+1})\big] \cdot \sqrt{\frac{|\boldsymbol{\Upsilon}_t|}{2\pi}}, \end{aligned} \tag{A.20}$$

$$\begin{aligned} \Rightarrow \ \log z_t(\boldsymbol{\chi}_{t+1}) &= g_t(\bar{\boldsymbol{\chi}}_t, \boldsymbol{\chi}_{t+1}) - \frac{1}{2}\log\frac{|\boldsymbol{\Upsilon}_t|}{2\pi} \\ &= \frac{1}{2}\big[(\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1} + \mathbf{V}_t \boldsymbol{\chi}_{t+1})\big]^T (\mathbf{e}_t + \boldsymbol{\omega}_{t-1} + \mathbf{V}_t \boldsymbol{\chi}_{t+1}) \\ &\quad + \epsilon_{t-1} - \frac{1}{2}\log\frac{|\boldsymbol{\Upsilon}_t|}{2\pi} \\ &= \frac{1}{2}\boldsymbol{\chi}_{t+1}^T \mathbf{V}_t^T (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1} \mathbf{V}_t \boldsymbol{\chi}_{t+1} + \big[\mathbf{V}_t^T (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1} \mathbf{e}_t + \boldsymbol{\omega}_{t-1})\big]^T \boldsymbol{\chi}_{t+1} \\ &\quad + \frac{1}{2}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1})^T (\mathbf{U}_t - \boldsymbol{\Omega}_{t-1})^{-1}(\mathbf{e}_t + \boldsymbol{\omega}_{t-1}) + \epsilon_{t-1} - \frac{1}{2}\log\frac{|\boldsymbol{\Upsilon}_t|}{2\pi}. \end{aligned} \tag{A.21}$$

Comparing (A.16) and (A.21), we obtain (9.92d-e), and this completes the derivation of the forward pass.

To derive the backward pass, notice that we have also obtained an expression for the conditionals $p(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_{t+1})$:

$$\begin{aligned} p(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_{t+1}) &= \frac{1}{z_t(\boldsymbol{\chi}_{t+1})} \exp\big[g_t(\boldsymbol{\chi}_t, \boldsymbol{\chi}_{t+1})\big] \\ &= \mathcal{N}(\boldsymbol{\chi}_t \mid \bar{\boldsymbol{\chi}}_t, \boldsymbol{\Upsilon}_t) = \mathcal{N}(\boldsymbol{\chi}_t \mid \lambda_t + \Lambda_t \boldsymbol{\chi}_{t+1}, \boldsymbol{\Upsilon}_t). \end{aligned} \tag{A.22}$$

From Bayes' rule, the corresponding marginals are

$$p(\boldsymbol{\chi}_t) = \int d\boldsymbol{\chi}_{t+1}\, p(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_{t+1}) p(\boldsymbol{\chi}_t). \tag{A.23}$$

For $t = T$, we have $p(\boldsymbol{\chi}_T) = \mathcal{N}(\boldsymbol{\chi}_T \mid \lambda_T, \boldsymbol{\Upsilon}_T)$, which follows a Gaussian distribution. Hence, the marginals $p(\boldsymbol{\chi}_t)$ at all other times are Gaussians as well. Writing them generally as

$$p(\boldsymbol{\chi}_t) = \mathcal{N}(\boldsymbol{\chi}_t \mid \boldsymbol{\mu}_t, \boldsymbol{\Gamma}_t), \tag{A.24}$$

the mean $\boldsymbol{\mu}_t$ can be computed recursively by (9.94), and the precision matrix $\boldsymbol{\Gamma}_t$ is given by

$$\boldsymbol{\Gamma}_t^{-1} = \boldsymbol{\Lambda}_t \boldsymbol{\Gamma}_{t+1}^{-1} \boldsymbol{\Lambda}_t^T + \boldsymbol{\Upsilon}_t^{-1}. \tag{A.25}$$

This completes the derivation of the backward pass.

A.4 Parameter Estimation Formulas of SSSM

This section sketches the derivation of the parameter estimation formulas (9.96)-(9.102). First we only keep the part of $\mathcal{F}$ that depends on model parameters, *i.e.*, the first term in

(9.42), define it as $\mathcal{F}'$ and simplify

$$\mathcal{F}'(\boldsymbol{\Theta}) \triangleq \sum_{\mathbf{s}_{1:T}} \int d\mathbf{X}_{1:T}\; q(\mathbf{X}_{1:T}, \mathbf{s}_{1:T}) \big[\log p(\mathbf{Y}_{1:T}, \mathbf{X}_{1:T}, \mathbf{s}_{1:T})\big] \tag{A.26}$$

$$\begin{aligned}
&= \sum_{t=1}^{T} \sum_{s_t} q(s_t) \int d\mathbf{x}_t\; q(\mathbf{x}_t \mid s_t) \big[\log p(\mathbf{y}_t \mid \mathbf{x}_t, s_t)\big] + \\
&\quad \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \int d\mathbf{x}_{t-1}\; d\mathbf{x}_t\; \big[q(\mathbf{x}_{t-1} \mid s_{t-1}) q(\mathbf{x}_t \mid s_t) \log p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, s_t)\big] \\
&\quad + \sum_{t=1}^{T} \sum_{s_{t-1},\, s_t} q(s_{t-1}) q(s_t \mid s_{t-1}) \big[\log p(s_t | s_{t-1})\big] \\
&= \sum_{t=1}^{T} \sum_{s=1}^{N} \gamma_{s,t}\; E_{s,t} \big[\log \mathcal{N}(\mathbf{y}_t \mid \mathbf{C}_s \mathbf{x}_t + \mathbf{c}_s\; , \mathbf{D}_s)\big] + \sum_{t=1}^{T} \sum_{s'=1}^{N} \sum_{s=1}^{N} \gamma_{s',t-1}\; \eta_{s's,t}\; \pi_{s's} \\
&\quad + \sum_{t=1}^{T} \sum_{s'=1}^{N} \sum_{s=1}^{N} \gamma_{s',t-1}\; \eta_{s's,t}\; E_{s',t-1} E_{s,t} \big[\log \mathcal{N}(\mathbf{x}_t \mid \mathbf{A}_s \mathbf{x}_{t-1} + \mathbf{a}_s\; , \mathbf{B}_s)\big] \\
&= \sum_{t=1}^{T} \sum_{s=1}^{N} \gamma_{s,t}\; E_{s,t} \Big[\frac{1}{2} \log \Big|\frac{\mathbf{D}_s}{2\pi}\Big| - \frac{1}{2} (\mathbf{y}_t - \mathbf{C}_s \mathbf{x}_t - \mathbf{c}_s)^T \mathbf{D}_s (\mathbf{y}_t - \mathbf{C}_s \mathbf{x}_t - \mathbf{c}_s)\Big] + \\
&\sum_{t=1}^{T} \sum_{s'=1}^{N} \sum_{s=1}^{N} \gamma_{s',t-1}\, \eta_{s's,t} E_{s',t-1} E_{s,t} \Big[\frac{1}{2} \log \Big|\frac{\mathbf{B}_s}{2\pi}\Big| - \frac{1}{2} (\mathbf{x}_t - \mathbf{A}_s \mathbf{x}_{t-1} - \mathbf{a}_s)^T \mathbf{B}_s (\mathbf{x}_t - \mathbf{A}_s \mathbf{x}_{t-1} - \mathbf{a}_s)\Big] \\
&\quad + \sum_{s'=1}^{N} \sum_{s=1}^{N} \Big(\sum_{t=1}^{T} \gamma_{s',t-1}\; \eta_{s's,t}\Big) \cdot \pi_{s's}\; .
\end{aligned} \tag{A.27}$$

The functional form of (A.27) clearly shows that we can estimate the model parameters separately in three groups: $\{\mathbf{C}_s, \mathbf{c}_s, \mathbf{D}_s\}$, $\{\mathbf{A}_s, \mathbf{a_s}, \mathbf{B}_s\}$ and $\{\pi_{ij}\}$. To estimate $\{\mathbf{C}_s, \mathbf{c}_s, \mathbf{D}_s\}$, we set the partial derivatives of $\mathcal{F}'$ w.r.t. these parameters to zero and solve the resulting equations.

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{C}_s} = \sum_{t=1}^{T} \gamma_{s,t}\; E_{s,t} \big[\mathbf{D}_s (\mathbf{y}_t - \mathbf{C}_s \mathbf{x}_t - \mathbf{c}_s) \mathbf{x}_t^T\big] = 0, \tag{A.28}$$

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{c}_s} = \sum_{t=1}^{T} \gamma_{s,t}\; E_{s,t} \big[\mathbf{D}_s (\mathbf{y}_t - \mathbf{C}_s \mathbf{x}_t - \mathbf{c}_s)\big] = 0, \tag{A.29}$$

$$\frac{\partial \mathcal{F}'}{\partial \mathbf{D}_s} = \sum_{t=1}^{T} \gamma_{s,t}\, E_{s,t}\big[\frac{1}{2}\mathbf{D}_s^{-1} - \frac{1}{2}(\mathbf{y}_t - \mathbf{C}_s\mathbf{x}_t - \mathbf{c}_s)(\mathbf{y}_t - \mathbf{C}_s\mathbf{x}_t - \mathbf{c}_s)^T\big] = 0. \tag{A.30}$$

From (A.28),

$$\mathbf{D}_s \sum_{t=1}^{T} \gamma_{s,t} \left[\mathbf{y}_t \boldsymbol{\rho}_{s,t}^T - \mathbf{C}_s(\boldsymbol{\Gamma}_{s,t}^{-1} + \boldsymbol{\rho}_{s,t}\boldsymbol{\rho}_{s,t}^T) - \mathbf{c}_s\boldsymbol{\rho}_{s,t}^T\right] = 0. \tag{A.31}$$

From (A.29),

$$\mathbf{D}_s \sum_{t=1}^{T} \gamma_{s,t} \left(\mathbf{y}_t - \mathbf{C}_s\boldsymbol{\rho}_{s,t} - \mathbf{c}_s\right) = 0. \tag{A.32}$$

Solving for $\mathbf{C}_s$ and $\mathbf{c}_s$ from (A.31) and (A.32) (discarding the trivial solution of $\mathbf{D}_s = 0$), we arrive at the parameter estimation formulas of $\{\mathbf{C}_s, \mathbf{c}_s\}$ as in (9.99) and (9.100). From (A.30), we obtain

$$\mathbf{D}_s^{-1} = \frac{1}{\sum_{t=1}^{T} \gamma_{s,t}} E_{s,t}\big[(\mathbf{y}_t - \mathbf{C}_s\mathbf{x}_t - \mathbf{c}_s)(\mathbf{y}_t - \mathbf{C}_s\mathbf{x}_t - \mathbf{c}_s)^T\big], \tag{A.33}$$

which simplifies to (9.101) after working out the expectation analytically. Parameter estimation formulas of $\{\mathbf{A}_s, \mathbf{a_s}, \mathbf{B}_s\}$ as in (9.96)-(9.98) are obtained analogously and the details are omitted.

To obtain the parameter estimation formula of $\pi_{s's}$, we use the following result without detailed proof.

Suppose $a_i > 0$, $x_i > 0$ for $i = 1, \ldots, N$, then the maximum of

$$\sum_{i=1}^{N} a_i \cdot x_i, \tag{A.34}$$

subject to the normalization constraint

$$\sum_{i=1}^{N} x_i = 1, \tag{A.35}$$

is obtained by setting

$$x_i = \frac{a_i}{\sum_{i=1}^{N} a_i}\ . \tag{A.36}$$

The above result can be readily shown by solving a standard Lagrange optimization problem with Lagrange multipliers. Identifying a_i with $\sum_{t=1}^{T} \gamma_{s',t-1}\, \eta_{s's,t}$, we obtain the ML estimate of $\pi_{s's}$ as

$$\begin{aligned} \pi_{s's} &= \frac{1}{\sum_{s=1}^{N} \sum_{t=1}^{T} \gamma_{s',t-1}\, \eta_{s's,t}} \cdot \sum_{t=1}^{T} \gamma_{s',t-1}\, \eta_{s's,t} \\ &= \frac{1}{\sum_{t=1}^{T} \gamma_{s',t-1}} \cdot \sum_{t=1}^{T} \gamma_{s',t-1}\, \eta_{s's,t}, \end{aligned} \tag{A.37}$$

which is the same as (9.102). Such a formula may also be interpreted based on the concept of counting event occurrences as in traditional HMMs [196, 197].

Bibliography

[1] **Acero**, A. (**1999**), "Formant analysis and synthesis using hidden Markov models", in *Eurospeech*, 1047–1050, Budapest, Hungary.

[2] **Ackerson**, G. A. and **Fu**, K. S. (**1970**), "On state estimation in switching enviroments", *IEEE Trans. Automatic Control*, 15(1): 10–17.

[3] **Aji**, S. M. and **McEliece**, R. J. (**2000**), "The generalized distributive law", *IEEE Trans. Information Theory*, 46(2): 325–343.

[4] **Allen**, J., **Hunnicutt**, M. S., and **Klatt**, D. H. (**1987**), *From Text to Speech: The MITalk System*, Cambridge University Press, Cambridge, UK.

[5] **Atal**, B. S. and **Hanauer**, L. (**1971**), "Speech analysis and synthesis by linear prediction of speech wave", *J. Acoust. Soc. Am.*, 50: 637–655.

[6] **Baer**, T., **Gore**, J. C., **Gracco**, L. C., and **Nye**, P. W. (**1991**), "Analysis of vocal tract shape and dimension using magnetic resonance imaging: Vowels", *J. Acoust. Soc. Am.*, 90(2): 799–828.

[7] **Bahl**, L. R., **Brown**, P. E., **de Souza**, P. V., and **Mercer**, R. L. (**1986**), "Maximum mutual information estimation of hidden Markov model parameters for speech recognition", in *Proc. ICASSP*, vol. 1, 49–52, Tokyo, Japan.

[8] **Baltzakis**, H. and **Trahanias**, P. (**2002**), "Hybrid mobile robot localization using switching state-space models", in *Proc. IEEE International Conference on Robotics & Automation*, 366–372, Washington, DC.

[9] **Bar-Shalom**, Y. and **Li**, X.-R. (**1993**), *Estimation and Tracking*, Artech House, Boston.

[10] **Barron**, A. R. (**1993**), "Universal approximation bounds for superpositions of a sigmoidal function", *IEEE Trans. Information Theory*, 39(3): 930–945.

[11] **Baum**, L. E. (**1972**), "An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes", *Inequalities*, 3: 1–8.

[12] **Baum**, L. E. and **Eagon**, J. A. (**1967**), "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology", *Bulletin of American Mathematical Society*, 73: 360–363.

[13] **Baum**, L. E. and **Petrie**, T. (**1966**), "Statistical inference for probabilistic functions of finite state Markov chains", *Annals of Mathematical Statistics*, 37: 1554–1563.

[14] **Baum**, L. E., **Petrie**, T., **Soules**, G., and **Weiss**, N. (**1970**), "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains", *Annals of Mathematical Statistics*, 41(1): 164–171.

[15] **Bell**, A. G. (**1906**), *The Mechanism of Speech*, Funk & Wagnalls, New York, reprinted from the proceedings of the first summer meeting of the American association to promote the teaching of speech to the deaf.

[16] **Bilmes**, J. A. (**2002**), "What HMMs can do", *Technical Report UWEETR-2002-0003*, Dept of EE, University of Washington, Seattle, WA.

[17] **Bilmes**, J. A. (**2003**), "Buried Markov models: A graphical-modeling approach to automatic speech recognition", *Computer Speech & Language*, 17(2-3): 213–231.

[18] **Bilmes**, J. A. (**2004**), "Graphical models and automatic speech recognition", in *Mathematical Foundations of Speech and Language Processing*, edited by M. Johnson, S. P. Khudanpur, M. Ostendorf, and R. Rosenfeld, 191–245, Springer-Verlag, New York, NY.

[19] **Bishop**, C. M. (**1995**), *Neural Networks for Pattern Recognition*, Oxford University Press, New York, NY.

[20] **Blake**, A. and **Isard**, M. (**1998**), *Active Contours*, Springer-Verlag, New York, NY.

[21] **Blake**, A. and **Yuille**, A. (editors) (**1992**), *Active Vision*, MIT Press, Cambridge, MA.

[22] **Borden**, G. J. and **Harris**, K. S. (**1984**), *Speech Science Primer: Physiology, Acoustics, and Perception of Speech*, 2nd edition, Williams & Wilkins, Baltimore, MD.

[23] **Bourlard**, H. and **Morgan**, N. (**1994**), *Connectionist speech recognition : a hybrid approach*, Kluwer Academic Publishers, Boston, MA.

[24] **Breiman**, L. (**1996**), "Bagging predictors", *Machine Learning*, 24(2): 123–140.

[25] **Bridle**, J. S., **Deng**, L., **Picone**, J., **Richards**, H. B., **Ma**, J., **Kamm**, T., **Shuster**, M., **Pike**, S., and **Regan**, R. (**1998**), "An investigation of segmental hidden dynamic models of speech coarticulation for automatic speech recognition", in *Final Report for the 1998 Workshop on Language Engineering*, 1–61, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD.

[26] **Brigger**, P., **Hoeg**, J., and **Unser**, M. (**2000**), "B-spline snakes: A flexible tool for parametric contour detection", *IEEE Trans. Image Process.*, 9(9): 1484–1496.

[27] **Browman**, C. P. and **Goldstein**, L. M. (**1986**), "Towards an articulatory phonology", *Phonology Yearbook*, 3: 219–252.

[28] **Browman**, C. P. and **Goldstein**, L. M. (**1989**), "Articulatory gestures as phonological units", *Phonology*, 6: 201–251.

[29] **Browman**, C. P. and **Goldstein**, L. M. (**1992**), "Articulatory phonology: An overview", *Phonology Yearbook*, 49: 155–180.

[30] **Bruce**, I. C., **Karkhanis**, N. V., **Young**, E. D., and **Sachs**, M. B. (**2002**), "Robust formant tracking in noise", in *Proc. ICASSP*, vol. 1, 281–284, Orlando, FL.

[31] **Campbell**, J. P. (**1997**), "Speaker rcognition: A tutorial", *Proc. of the IEEE*, 85(9): 1437–1462.

[32] **Carlson**, R., **Granstrom**, B., and **Karlsson**, I. (**1991**), "Experiments with voice modeling in speech synthesis", *Speech Communication*, 10: 481–489.

[33] **Cerny**, V. (**1985**), "Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Applications*, 45: 1985.

[34] **Chang**, C. B. and **Athans**, M. (**1978**), "State estimation for discrete systems with switching parameters", *IEEE Trans. Aerospace and Electronic Systems*, 14(3): 418–425.

[35] **Chelba**, C. and **Jelinek**, F. (**2000**), "Structured language modeling", *Computer Speech & Language*, 14(4): 283–332.

[36] **Chellappa**, R. and **Jain**, A. (editors) (**1993**), *Markov Random Fields: Theory and Application*, Academic Press, Boston, MA.

[37] **Chen**, S., **Cowan**, C. F. N., and **Grant**, P. M. (**1991**), "Orthogonal least squares learning algorithm for radial basis funtion networks", *IEEE Trans. Neural Networks*, 2(2): 302–309.

[38] **Chomsky**, N. (**1965**), *Aspects of the Theory of Syntax*, MIT Press, Cambridge, MA.

[39] **Chomsky**, N. and **Halle**, M. (**1968**), *The Sound Pattern of English*, Harper & Row, New York.

[40] **Chou**, W., **Juang**, B.-H., and **Lee**, C.-H. (**1992**), "Segmental GPD training of HMM based speech recognizer", in *Proc. ICASSP*, vol. 1, 473–476, San Francisco, CA.

[41] **Clements**, G. N. and **Hume**, E. V. (**1995**), "The internal organization of speech sounds", in *The Handbook of Phonological Theory*, edited by J. A. Goldsmith, 245–306, Blackwell, Cambridge, UK.

[42] **Cohen**, L. (**1995**), *Time-Frequency Analysis*, Prentice Hall, Englewood Cliffs, NJ.

[43] **Cohen**, L. D. and **Cohen**, I. (**1993**), "Finite-element methods for active contour models and balloons for 2-D and 3-D images", *IEEE Trans. PAMI*, 15(11): 1131–1147.

[44] **Cormen**, T. H., **Leiserson**, C. E., **Rivest**, R. L., and **Stein**, C. (**2001**), *Introduction to Algorithms*, 2nd edition, The MIT Press, Cambridge, MA.

[45] **Cowell**, R. (**1998**), "Introduction to inference for Bayesian networks", in *Learning in Graphical Models*, edited by M. I. Jordan, 9–26, Kluwer Academic Publishers, Norwell, MA.

[46] **Cybenko**, G. (**1989**), "Approximation by superpositions of a sigmoid function", *Mathematics of Control, Signals, and Systems*, 2(4): 303–314.

[47] **Dang**, J. and **Honda**, K. (**2001**), "A physiological model of a dynamic vocal tract for speech production", *Acoustical Science and Technology*, 22(6): 415–425.

[48] **Dang**, J. and **Honda**, K. (**2004**), "Construction and control of a physiological articulatory model", *J. Acoust. Soc. Am.*, 115(2): 853–870.

[49] **Dang**, J., **Sun**, J., **Deng**, L., and **Honda**, K. (**1999**), "Speech synthesis using a physiological articulatory model with feature-based rules", in *Proc. ICPhS*, 2267–2270, San Francisco, CA.

[50] **Das**, S. K. and **Picheny**, M. A. (**1996**), "Issues in practical large vocabulary isolated word recognition: the IBM Tangora system", in *Automatic Speech and Speaker Recognition: Advanced Topics*, edited by C.-H. Lee, F. K. Soong, and K. K. Paliwal, 457–479, Kluwer Academic Publishers, Norwell, MA.

[51] **De Boor**, C. (**2001**), *A Practical Guide to Splines*, 2nd edition, Springer-Verlag, New York, NY.

[52] **Deller**, J. R., **Proakis**, J. G., and **Hansen**, J. H. (**1993**), *Discrete-Time Processing of Speech Signals*, Prentice Hall, Upper Saddle River, NJ.

[53] **Dembowski**, J. and **Westbury**, J. R. (**1999**), "Contextual influences on stop consonant articulatory postures in connected speech", in *Proc. ICPhS*, 2419–2422, San Francisco, CA.

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

[55] **Demuth**, H. and **Beale**, M. (**2001**), *Neural Network Toolbox User's Guide (Version 4)*, The Mathworks, Natick, MA.

[56] **Deng**, L. (**1992**), "A generalized hidden Markov model with state-conditioned trend functions of time for the speech signal", *Signal Processing*, 27: 65–78.

[57] **Deng**, L. (**1996**), "Transiems as dynamically defined, sub-phonemic units of speech: A computational model", *Signal Processing*, 49: 25–35.

[58] **Deng**, L. (**1997**), "Autosegmental representation of phonological units of speech and its phonetic interface", *Speech Communication*, 22(2): 211–222.

[59] **Deng**, L. (**1998**), "A dynamic, feature-based approach to the interface between phonology and phonetics for speech modeling and recognition", *Speech Communication*, 24(4): 299–323.

[60] **Deng**, L., **Aksmanovic**, M., **Sun**, D., and **Wu**, J. (**1992**), "Speech recognition using hidden Markov model with polynomial regression functions as nonstationary states", *Signal Processing*, 27: 65–78.

[61] **Deng**, L., **Bazzi**, I., and **Acero**, A. (**2003**), "Tracking vocal tract resonances using an analytical nonlinear predictor and a target-guided temporal constraint", in *Proc. Eurospeech*, vol. 1, 73–76, Geneva, Switzerland.

[62] **Deng**, L. and **Huang**, X. (**2004**), "Challenges in adopting speech recognition", *Communications of the ACM*, 47(1): 69–75.

[63] **Deng**, L., **Lee**, L. J., **Attias**, H., and **Acero**, A. (**2004**), "A structured speech model with continuous hidden dynamics and prediction-residual training for tracking vocal tract resonances", in *Proc. ICASSP*, vol. 1, 557–560, Montreal, QC.

[64] **Deng**, L. and **Ma**, J. (**2001**), "Spontaneous speech recognition using a statistical coarticulatory model for the vocal-tract-resonance dynamics", *J. Acoust. Soc. Am.*, 108(6): 3036–3048.

[65] **Deng**, L. and **O'Shaughnessy**, D. (**2003**), *Speech Processing: A Dynamic and Optimization-Oriented Approach*, Marcel Dekker Inc., New York.

[66] **Deng**, L. and **Sun**, D. X. (**1994**), "A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features", *J. Acoust. Soc. Am.*, 95(5): 2702–2719.

[67] **Dennis**, J. E. and **Schnabel**, R. B. (**1996**), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Soc for Industrial & Applied Math, Philadelphia, PA.

[68] **Dietterich**, T. G. (**2000**), "Ensemble methods in machine learning", *Lecture Notes in Computer Science*, 1857: 1–15.

[69] **Duda**, R. O., **Hart**, P. E., and **Stork**, D. G. (**2001**), *Pattern classification*, 2nd edition, John Wiley & Sons, New York, NY.

[70] **Dudley**, H. (**1939**), "Remaking speech", *J. Acoust. Soc. Am.*, 11: 169–177.

[71] **Dudley**, H., **Riesz**, R. R., and **Watkins**, S. A. (**1939**), "A synthetic speaker", *J. Franklin Inst.*, 227: 739–764.

[72] **Durbin**, R., **Eddy**, S., **Krogh**, A., and **Mitchison**, G. (**1998**), *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, New York, NY.

[73] **Dusan**, S. V. (**2000**), *Statistical Estimation of Articulatory Trajectories from the Speech Signal Using Dynamical and Phonological Constraints*, Ph.D. thesis, University of Waterloo, Waterloo, ON, Canada.

[74] **Eviatar**, H. and **Somorjai**, R. (**1996**), "A fast, simple active contour algorithm for biomedical images", *Pattern Recognition Letters*, 17(9): 969–974.

[75] **Fant**, G. (**1970**), *Acoustic Theory of Speech Production*, Mouton, The Hague.

[76] **Figueiredo**, M., **Leitão**, J., and **Jain**, A. (**2000**), "Unsupervised contour representation and estimation using B-splines and a minimum description length criterion", *IEEE Trans. Image Process.*, 9(6): 1075–1087.

[77] **Flanagan**, J. L. (**1972**), *Speech Synthesis, Analysis and Perception*, 2nd edition, Springer-Verlag.

[78] **Foresee**, F. D. and **Hagan**, M. T. (**1997**), "Gauss-Newton approximation to Bayesian regularization", in *Proc. ICNN*, 1930–1935, Houston, TX.

[79] **Fowler**, C. A. (**1995**), "Speech production", in *Speech, Language and Communication*, edited by J. L. Miller and P. D. Eimas, chapter 2, 29–61, Academic Press, San Diego, CA.

[80] **Freund**, Y. and **Schapire**, R. E. (**1997**), "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer and System Sciences*, 55(1): 119–139.

[81] **Fujimura**, O., **Kiritani**, S., and **Ishida**, H. (**1973**), "Computer-controlled radiography for observation of the movements of articulatory and other human organs", *Computers in Biology and Medicine*, 3: 371–384.

[82] **Fukunaga**, K. (**1990**), *Introduction to statistical pattern recognition*, 2nd edition, Academic Press, Boston, MA.

[83] **Funahasi**, K.-I. (**1989**), "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, 2(3): 183–192.

[84] **Furui**, S. (**1997**), "Recent advances in speaker recognition", *Pattern Recognition Letters*, 18: 859–872.

[85] **Furui**, S. (**2002**), "Recent progress in spontaneous speech recognition and understanding", in *Proc. IEEE Workshop on Multimedia Signal Processing*, 253–258, St. Thomas, US Virgin Islands.

[86] **Gao**, Y., **Bakis**, R., **Huang**, J., and **Xiang**, B. (**2000**), "Multistage coarticulation model combining articulatory, formant and cepstral features", in *Proc. ICSLP*, 25–28, Beijing, China.

[87] **Geiger**, D., **Gupta**, A., **Costa**, L. A., and **Vlontzos**, J. (**1995**), "Dynamic programming for detecting, tracking, and matching deformable contours", *IEEE Trans. PAMI*, 17(3): 294–302.

[88] **Geman**, S. and **Geman**, D. (**1984**), "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images", *IEEE Trans. PAMI*, 6: 721–741.

[89] **Gersch**, W. (**1992**), "Smoothness priors", in *New Directions in Time Series Analysis*, edited by D. R. Brillinger, 111–146, Springer-Verlag, New York, NY.

[90] **Ghahramani**, Z. and **Hinton**, G. E. (**2000**), "Variational learning for switching state-space models", *Neural Computation*, 12: 831–864.

[91] **Glass**, J. R. (**2003**), "A probabilistic framework for segment-based speech recognition", *Computer Speech & Language*, 17(2-3): 137–152.

[92] **Gold**, B. and **Morgan**, N. (**2000**), *Speech and Audio Processing: Processing and Perception of Speech and Music*, John Wiley & Sons, New York.

[93] **Goldberger**, J., **Burshtein**, D., and **Franco**, H. (**1999**), "Segmental modeling using a continuous mixture of nonparametric models", *IEEE Trans. Speech Audio Process.*, 7(3): 262–271.

[94] **Goldsmith**, J. A. (**1990**), *Autosegmental and metrical Phonology*, Blackwell, Oxford, UK.

[95] **Gonzalez**, R. C. and **Woods**, R. E. (**2002**), *Digital Image Processing*, 2nd edition, Prentice Hall, Upper Saddle River, NJ.

[96] **Gorin**, A. L., **Riccardi**, G., and **Wright**, J. H. (**1999**), "How may I help you?", in *Computational Models of Speech Pattern Processing*, edited by K. M. Ponting, 328–349, Springer-Verlag, Berlin, Germany.

[97] **Hagan**, M. T., **Demuth**, H. B., and **Beale**, M. H. (**1996**), *Neural Network Design*, PWS Publishing, Boston, MA.

[98] **Hagan**, M. T. and **Menhaj**, M. B. (**1994**), "Training feedforward networks with the Marquardt algorithm", *IEEE Trans. Neural Networks*, 5(6): 989–993.

[99] **Hamilton**, J. D. (**1994**), "A new approach to the economic analysis of nonstationary time series and the business cycle", *Econometrica*, 57: 357–384.

[100] **Hanselman**, D. and **Littlefield**, B. (**2001**), *Mastering Matlab 6: A Comprehensive Titorial and Reference*, Prentice Hall, Upper Saddle River, NJ.

[101] **Hanson**, H. M., **Maragos**, P., and **Potamianos**, A. (**1994**), "A system for finding speech formants and modulation via energy seperation", *IEEE Trans. Speech Audio Process.*, 2(3): 436–443.

[102] **Hardcastle**, W. J. (**1972**), "The use of electropalatography in phonetic research", *Phonetica*, 25: 1977–215.

[103] **Hardcastle**, W. J. and **Hewlett**, N. (editors) (**1999**), *Coarticulation: Theory, Data and Techniques*, Cambridge University Press, Cambridge, UK.

[104] **Harshman**, R., **Ladefoged**, P., and **Goldstein**, L. (**1977**), "Factor analysis of tongue shapes", *J. Acoust. Soc. Am.*, 62(3): 693–707.

[105] **Hart**, P. E., **Nilsson**, N. J., and **Raphael**, B. (**1968**), "A formal basis for the heuristic determination of minimum cost paths", *IEEE Trans. Systems Science & Cybernetics*, 4(2): 100–107.

[106] **Hashi**, M., **Honda**, K., and **Westbury**, J. R. (**2003**), "Time-varying acoustic and articulatory characteristics of American English /r/: A cross-speaker study", *Journal of Phonetics*, 31(1): 3–22.

[107] **Hashi**, M., **Westbury**, J. R., and **Honda**, K. (**1998**), "Vowel posture normalization", *J. Acous. Soc. Am.*, 104(4): 2426–2437.

[108] **Haykin**, S. (**1999**), *Neural Networks: A Comprehensive Foundation*, 2nd edition, Prentice-Hall, Upper Saddle River, NJ.

[109] **Heinz**, J. M. and **Stevens**, K. N. (**1964**), "On the derivation of aera functions and acoustic spectra from cineradiographic film of speech", *J. Acoust. Soc. Am.*, 36: 1037.

[110] **Helmholtz**, H. L. F. (**1954**), *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, 2nd edition, Dover Publications, New York, translated from the fourth (and last) German edition of 1877.

[111] **Holmes**, J. N., **Holmes**, W. J., and **Garner**, P. N. (**1997**), "Using formant frequencies in speech recognition", in *Proc. Eurospeech*, 2083–2086, Rhodes, Greece.

[112] **Hopkins**, W. G. (**2000**), *A New View of Statistics*, Internet Society for Sport Science, URL `http://www.sportsci.org/resource/stats/`.

[113] **Hornik**, K., **Stinchcombe**, M., and **White**, H. (**1989**), "Multilayer feedforward networks are universal approximators", *Neural Networks*, 2(5): 359–366.

[114] **Huang**, X., **Acero**, A., **Alleva**, F., **Hwang**, M., **Jiang**, L., and **Mahajan**, M. (**1996**), "From Sphinx-II to Whisper — making speech recognition usable", in *Automatic Speech and Speaker Recognition: Advanced Topics*, edited by C.-H. Lee, F. K. Soong, and K. K. Paliwal, 481–508, Kluwer Academic Publishers, Norwell, MA.

[115] **Huang**, X., **Acero**, A., and **Hon**, H.-W. (**2001**), *Spoken Language Processing: A Guide to Theory, Algorithm , and System Development*, Prentice Hall, Upper Saddle River, NJ.

[116] **Itzykson**, C. and **Drouffe**, J. M. (**1991**), *Statistical Field Theory*, Cambridge University Press, Cambridge, UK.

[117] **Jakobson**, R., **Gunnar**, C., **Fant**, M., and **Halle**, M. (**1967**), *Preliminaries to Speech Analysis: the Distinctive Features and Their Correlates*, 2nd edition, MIT Press, Cambridge, MA.

[118] **Jamieson**, M. (**2002**), *A Continuous Gibbs Annealer for Contour Estimation*, Master's thesis, University of Waterloo, Waterloo, ON, Canada.

[119] **Jamieson**, M., **Fieguth**, P., and **Lee**, L. J. (**2003**), "Parametric contour estimation by simulated annealing", in *Proc. ICIP*, vol. 3, 449–452, Barcelona, Spain.

[120] **Jelinek**, F. (**1997**), *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, MA.

[121] **Jensen**, F. V. (**1996**), *An Introduction to Bayesian Networks*, UCL Press, London, UK.

[122] **Jolliffe**, I. T. (**1986**), *Principal Component Analysis*, Springer-Verlag, New York, NY.

[123] **Jordan**, M. I. (editor) (**1998**), *Learning in Graphical Models*, Kluwer Academic Publishers, Norwell, MA.

[124] **Jordan**, M. I., **Ghahramani**, Z., **Jaakkola**, T. S., and **Saul**, L. K. (**1999**), "An introduction to variational methods for graphical models", *Machine Learning*, 37(2): 183–233.

[125] **Juang**, B.-H., **Chou**, W., and **Lee**, C.-H. (**1997**), "Minimum classification error rate methods for speech recognition", *IEEE Trans. Speech Audio Process.*, 5(3): 257–265.

[126] **Juang**, B.-H. and **Katagiri**, S. (**1992**), "Discriminative learning for minimum error classification", *IEEE Trans. Signal Process.*, 40(12): 3043–3054.

[127] **Jurafsky**, D. and **Martin**, J. H. (**2000**), *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, Upper Saddle River, NJ.

[128] **Kass**, M., **Witkin**, A., and **Terzopoulos**, D. (**1987**), "Snakes: Active contour models", *International Journal of Computer Vision*, 1(4): 321–331.

[129] **Katamba**, F. (**1989**), *An Introduction to Phonology*, Longman, New York.

[130] **Kelso**, J. A. S., **Saltzman**, E. L., and **Tuller**, B. (**1986**), "The dynamical perspective on speech production: Data and theory", *Journal of Phonetics*, 14: 29–59.

[131] **Kent**, R. D., **Adams**, S. G., and **Turner**, G. S. (**1996**), "Models of speech production", in *Principles of Experimental Phonetics*, edited by N. J. Lass, 3–45, Mosby.

[132] **Kim**, C.-J. and **Nelson**, C. R. (**1999**), *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*, MIT Press, Cambridge, MA.

[133] **King**, S. and **Wrench**, A. (**1999**), "Dynamical system modelling for articulator movement", in *Proc. ICPhS*, 2259–2262, San Francisco, CA.

[134] **Kiritani**, S. (**1986**), "X-ray microbeam method for measurement of articulatory dynamics: Techniques and results", *Speech Communication*, 5(2): 119–140.

[135] **Kirkpatrick**, S., **Gellat**, S. D., and **Vecchi**, M. P. (**1983**), "Optimization by simulated annealing", *Science*, 220(5): 671–680.

[136] **Kitagawa**, G. and **Gersch**, W. (**1996**), *Smoothness Priors Analysis of Time Series*, Springer-Verlag, New York.

[137] **Klatt**, D. H. (**1980**), "Software for a cascade/parallel formant synthesizer", *J. Acoust. Soc. Am.*, 67(3): 971–995.

[138] **Klatt**, D. H. and **Klatt**, L. C. (**1990**), "Analysis, synthesis and perception of voice quality variations among female and male talkers", *J. Acoust. Soc. Am.*, 87(2): 820–857.

[139] **Koenig**, R., **Dunn**, H. K., and **Lacy**, L. Y. (**1946**), "The sound spectrograph", *J. Acoust. Soc. Am.*, 18: 19–49.

[140] **Kohn**, R. and **Ansley**, C. F. (**1988**), "Equivalence between Bayesian smoothness priors and optimal smoothing for function estimation", in *Bayesian Analysis of Time Series and Dynamic Models*, edited by J. C. Spall, 393–430, Marcel Dekker, New York, NY.

[141] **Kopec**, G. E. (**1986**), "Formant tracking using hidden Markov models and vector quantization", *IEEE Trans. Acoust., Speech, Signal Process.*, ASSP-34(4): 709–729.

[142] **Kschischang**, F. R., **Frey**, B. J., and **Loeliger**, H.-A. (**2001**), "Factor graphs and the sum-product algorithm", *IEEE Trans. Information Theory*, 47(2): 498–549.

[143] **Lamel**, L. and **Gauvain**, J. L. (**1993**), "High performance speaker-independent phone recognition using CDHMM", in *Proc. Eurospeech*, 121–124, Berlin, Germany.

[144] **Laprie**, Y. and **Berger**, M.-O. (**1996**), "Cooperation of regularization and speech heuristics to control automatic formant tracking", *Speech Communication*, 19(4): 255–269.

[145] **Lee**, K.-F. and **Hon**, H.-W. (**1989**), "Speaker-independent phone recognition using hidden Markov models", *IEEE Trans. Acoust., Speech, Signal Process.*, 37(11): 1641–1648.

[146] **Lee**, L. J. (**1998**), "Fast training algorithms for multilayer perceptrons", SD 675 Project Report, University of Waterloo.

[147] **Lee**, L. J. (**1999**), *A Two-Dimensional Computational Model for Articulatory Motions in Speech Production*, Master's thesis, University of Waterloo, Waterloo, ON, Canada.

[148] **Lee**, L. J., **Attias**, H., and **Deng**, L. (**2003**), "Variational inference and learning for segmental switching state space models of hidden speech dynamics", in *Proc. ICASSP*, vol. 1, 872–875, Hongkong.

[149] **Lee**, L. J., **Attias**, H., **Deng**, L., and **Fieguth**, P. (**2004**), "A multimodal variational approach to learning and inference in switching state space models", in *Proc. ICASSP*, vol. 5, 505–508, Montreal, QC.

[150] **Lee**, L. J., **Dang**, J., and **Deng**, L. (**1999**), "A computational model for 2D articulation: Speech production with potential use in recognition", in *Proc. ICPhS*, 2529–2532, San Francisco, CA.

[151] **Lee**, L. J., **Fieguth**, P., and **Deng**, L. (**2001**), "A functional articulatory dynamic model for speech production", in *Proc. ICASSP*, vol. 1, 797–800, Salt Lake City, UT.

[152] **Lee**, M., **van Santen**, J., **Mőbius**, B., and **Olive**, J. (**1999**), "Formant tracking using segmental phonemic information", in *Eurospeech*, Budapest, Hungary.

[153] **Lerner**, U. and **Parr**, R. (**2001**), "Inference in hybrid networks: theoretical limits and practical algorithms", in *Proc. UAI*, 310–318, Seattle, WA.

[154] **Lerner**, U., **Parr**, R., **Koller**, D., and **Biswas**, G. (**2000**), "Bayesian fault detection and diagnosis in dynamic systems", in *Proc. UAI*, 531–537, Austin, TX.

[155] **Lerner**, U. N. (**2002**), *Hybrid Bayesian Networks for Reasoning about Complex Systems*, Ph.D. thesis, Stanford University, Stanford, CA.

[156] **Lippmann**, R. P. (**1997**), "Speech recognition by machines and humans", *Speech Communication*, 22(1): 1–15.

[157] **Ma**, J. Z. and **Deng**, L. (**2004**), "Target-directed mixture dynamic models for spontaneous speech recognition", *IEEE Trans. Speech Audio Process.*, 12(1): 47–58.

[158] **Ma**, Z. J. (**2000**), *Spontaneous Speech Recognition Using Statistical Dynamic Models for the Vocal-Tract-Resonance Dynamics*, Ph.D. thesis, University of Waterloo, Waterloo, ON, Canada.

[159] **MacKay**, D. J. C. (**1992**), "Bayesian interpolation", *Neural Computation*, 4(3): 415–447.

[160] **MacKay**, D. J. C. (**2003**), *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, Cambridge, UK.

[161] **Maeda**, S. (**1996**), "Phonemes as concatenable units: VCV synthesis using a vocal-tract synthesizer", in *Sound Patterns of Connected Speech: Description, Models and Explanation*, edited by A. P. Simpson and M. Pätzod, 145–164, Kiel University, Germany.

[162] **Marquardt**, D. W. (**1963**), "An algorithm for least squares estimation of non-linear parameters", *Journal of the Society for Industrial and Applied Mathematics*, 11: 431–441.

[163] **McCandless**, S. S. (**1974**), "An algorithm for automatic formant extraction using LPC spectra", *IEEE Trans. Acoust., Speech, Signal Process.*, 22(2): 135–141.

[164] **McLachlan**, G. J. and **Krishnan**, T. (**1997**), *The EM Algorithm and Extensions*, John Wiley & Sons, New York, NY.

[165] **Mendel**, J. M. (**1995**), *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Prentice Hall, Englewood Cliffs, NJ.

[166] **Menet**, S., **Saint-Marc**, P., and **Medioni**, G. (**1990**), "B-snakes: Implementation and application to stereo", in *Proc. DARPA Image Understanding Workshop*, 720–726.

[167] **Møller**, M. F. (**1993**), "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks*, 6: 525–533.

[168] **Moody**, J. and **Darken**, C. J. (**1989**), "Fast learning in networks of locally-tuned processing units", *Neural Computation*, 1: 281–294.

[169] **Moore**, R. (**1999**), "Speech pattern processing", in *Computational Models of Speech Pattern Processing*, edited by K. M. Ponting, 1–9, Springer-Verlag, Berlin, Germany.

[170] **Morgan**, N. and **Bourlard**, H. (**1995**), "Continuous speech recognition: An introduction to hybrid HMM/connectionist approach", *IEEE Signal Processing Magazine*, 12(3): 25–42.

[171] **Munhall**, K. G., **Vatikiotis-Bateson**, E., and **Tohkura**, Y. (**1995**), "X-ray film database for speech research", *J. Acoust. Soc. Am.*, 98: 1222–1224, URL `http://psyc.queensu.ca/~munhallk/05_database.htm`.

[172] **Neal**, R. M. and **Hinton**, G. E. (**1998**), "A view of the EM algorithm that justifies incremental, sparse and other variants", in *Learning in Graphical Models*, edited by M. I. Jordan, 355–368, Kluwer Academic Publishers, Norwell, MA.

[173] **Nilsson**, N. J. (**1982**), *Principles of Artificial Intelligence*, Springer-Verlag, Berlin, Germany.

[174] **Niyogi**, P. and **Girosi**, F. (**1996**), "On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions", *Neural Computation*, 8(4): 819–842.

[175] **Nygaard**, L. C. and **Pisoni**, D. B. (**1995**), "Speech perception: New directions in research and theory", in *Speech, Language and Communication*, edited by J. L. Miller and P. D. Eimas, chapter 3, 63–96, Academic Press, San Diego, CA.

[176] **Opitz**, D. and **Maclin**, R. (**1999**), "Popular ensemble methods: An empirical study", *Journal of Artificial Intelligence Research*, 11: 169–198.

[177] **Oppenheim**, A. V., **Willsky**, A. S., and **Nawab**, S. H. (**1997**), *Signals & Systems*, 2nd edition, Prentice Hall, Upper Saddle River, NJ.

[178] **O'Shaughnessy**, D. (**2000**), *Speech Communications: Human and Machine*, 2nd edition, IEEE Press, New York.

[179] **Ostendorf**, M., **Digalakis**, V. V., and **Kimball**, I. A. (**1996**), "From HMM's to segment models: A unified view of stochastic modeling for speech recognition", *IEEE Trans. Speech Audio Process.*, 4(5): 360–278.

[180] **Ott**, R. L. (**1993**), *An Introduction to Statistical Methods and Data Analysis*, 4th edition, Duxbury Press, Belmont, CA.

[181] **Papoulis**, A. (**1991**), *Probability, Random Variables, and Stochastic Processes*, 3rd edition, McGraw-Hill, New York, NY.

[182] **Parisi**, G. (**1988**), *Statistical Field Theory*, Addison-Wesley, Redwood City, CA.

[183] **Park**, J. and **Sandberg**, I. W. (**1991**), "Universal approximation using radial-basis-function networks", *Neural Computation*, 3: 246–257.

[184] **Pavlović**, V., **Frey**, B. J., and **Huang**, T. S. (**1999**), "Variational learning in mixed-state dynamic graphical models", in *Proc. UAI*, 522–530, Stockholm, Sweden.

[185] **Pavlović**, V., **Rehg**, J. M., and **Cham**, T.-J. (**2000**), "A dynamic Bayesian network approach to tracking using learned dynamic models", in *Proc. Hybrid System Computation and Control*, 366–380, Pittsburgh, PA.

[186] **Pavlović**, V., **Rehg**, J. M., **Cham**, T.-J., and **Murphy**, K. P. (**1999**), "A dynamic Bayesian network approach to figure tracking using learned dynamic models", in *Proc. ICCV*, 94–101, Kerkyra, Greece.

[187] **Pearl**, J. (**1988**), *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA.

[188] **Percival**, D. B. and **Walden**, A. T. (**2000**), *Wavelet Methods for Time Series Analysis*, Cambridge University Press, Cambridge, UK.

[189] **Perkell**, J. S., **Cohen**, M. H., **Svirsky**, M. A., **Matthies**, M. L., **Garabieta**, I., and **Jackson**, M. T. (**1992**), "Electromagnetic midsaggital articulometer systems for transducing speech articulatory movements", *J. Acoust. Soc. Am.*, 92(6): 3078–3096.

[190] **Peterson**, G. E. and **Barney**, H. L. (**1952**), "Control methods used in a study of the vowels", *J. Acoust. Soc. Am.*, 24(2): 175–184.

[191] **Poggio**, T. and **Girosi**, F. (**1990**), "Networks for approximation and learning", *Proc. IEEE*, 78(9): 1481–1497.

[192] **Pols**, L. C. (**1997**), "Flexible, robust, and efficient human speeh recognition", *Proc. of the Institute of Phonetic Sciences*, 21: 1–10, University of Amsterdam.

[193] **Ponting**, K. M. (**1999**), "Forward", in *Computational Models of Speech Pattern Processing*, edited by K. M. Ponting, Springer-Verlag, Berlin.

[194] **Press**, W. H., **Teukolsky**, S. A., **Vetterling**, W. T., and **Flannery**, B. P. (**1992**), *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge University Press, New York, NY.

[195] **Protize**, A. B. (**1988**), "Hidden Markov models: A guided tour", in *Proc. ICASSP*, vol. 1, 7–13, New York, NY.

[196] **Rabiner**, L. R. (**1989**), "A tutorial on hidden Markov models and selected applications in speech recognition", *IEEE Proceedings*, 77(2): 257–286.

[197] **Rabiner**, L. R. and **Juang**, B.-H. (**1993**), *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ.

[198] **Rabiner**, L. R. and **Levinson**, S. E. (**1981**), "Isolated and connected word recognition — theory and selected applications", *IEEE Trans. Communications*, 29(5): 621–669.

[199] **Rabiner**, L. R. and **Schafer**, R. W. (**1978**), *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ.

[200] **Ranganath**, S. (**2000**), "Contour extraction from cardiac MRI studies using snakes", *IEEE Trans. Medical Imaging*, 14(2): 328–338.

[201] **Rao**, P. and **Barman**, A. D. (**2000**), "Speech formant frequency estimation: Evaluating a nonstationary analysis method", *Signal Processing*, 80(8): 1655–1667.

[202] **Raphael**, C. (**2002**), "A hybrid graphical model for rhythmic parsing", *Artificial Intelligence*, 137(1-2): 217–238.

[203] **Reinhard**, K. and **Niranjan**, M. (**2002**), "Diphone subspace mixture trajectory models for HMM complementation", *Speech Communication*, 38(3-4): 237–265.

[204] **Reynolds**, D. A. (**1994**), "Experimental evaluation of features for robust speaker identification", *IEEE Trans. Speech Audio Process.*, 2(4): 639–643.

[205] **Reynolds**, D. A. (**1995**), "Speaker identification and verification using Gaussian mixture speaker models", *Speech Communication*, 17(1): 91–108.

[206] **Richards**, H. B. and **Bridle**, J. S. (**1999**), "The HDM: A segmental hidden dynamic model of coarticulation", in *Proc. ICASSP*, vol. 1, 357–360, Phoenix, AZ.

[207] **Rigoll**, G. (**1986**), "A new algorithm for estimation of formant trajectories directly from the speech signal based on an extended Kalman-filter", in *Proc. ICASSP*, 1229–1232, Tokyo, Japan.

[208] **Robinson**, T., **Hochberg**, M., and **Renals**, S. (**1994**), "Ipa: Improved phone modelling with recurrent neural networks", in *Proc. ICASSP*, vol. 1, 37–40.

[209] **Ross**, S. M. (**2003**), *Introduction to Probability Models*, 8th edition, Academic Press, San Diego, CA.

[210] **Rosti**, A.-V. I. and **Gales**, M. J. F. (**2004**), "Rao-Blackwellised Gibbs sampling for switching linear dynamical systems", in *Proc. ICASSP*, vol. 1, 809–812, Montreal, QC.

[211] **Roweis**, S. T. (**1999**), *Data Driven Production Models for Speech Processing*, Ph.D. thesis, California Institue of Technology.

[212] **Roweis**, S. T. and **Ghahramani**, Z. (**1999**), "A unifying review of linear Gaussian models", *Neural Computation*, 11(2): 305–345.

[213] **Saltzman**, E. L. and **Munhall**, K. G. (**1989**), "A dynamical approach to gestural patterning in speech production", *Ecological Psychology*, 1(4): 333–382.

[214] **Sanguineti**, V., **Laboissière**, R., and **Ostry**, D. J. (**1998**), "A dynamic biomechanical model for neural control of speech production", *J. Acoust. Soc. Am.*, 103(3): 1615–1627.

[215] **Saul**, L. K., **Jaakkola**, T. S., and **Jordan**, M. I. (**1996**), "Mean field theory for sigmoid belief networks", *Journal of Artificial Intelligence Research*, 4: 61–76.

[216] **Saul**, L. K. and **Jordan**, M. I. (**1996**), "Exploiting tractable substructures in intractable networks", in *Advances in Neural Information Processing Systems 8*, edited by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, MIT Press, Cambridge, MA.

[217] **Schmid**, P. and **Barnard**, E. (**1997**), "Explicit, N-best formant features for vowel classification", in *Proc. ICASSP*, 991–994.

[218] **Schroeter**, J. and **Sondi**, M. M. (**1994**), "Techniques for estimating vocal-tract shapes from the speech signal", *IEEE Trans. Acoust., Speech, Signal Process.*, 2(1): 133–150.

[219] **Shafer**, R. W. and **Rabiner**, L. R. (**1970**), "Systems for automatic formant analysis of speech", *J. Acoust. Soc. Am.*, 47(2): 634–648.

[220] **Shanmugan**, K. S. and **Breipohl**, A. M. (**1988**), *Random Signals: Detection, Estimation, and Data Analysis*, John Wiley & Sons, New York, NY.

[221] **Shumway**, R. H. (**1988**), *Applied Statistical Time Series Analysis*, Prentice Hall, Englewood Cliffs, NJ.

[222] **Shumway**, R. H. and **Stoffer**, D. S. (**1991**), "Dynamic linear systems with switching", *Journal of the American Statistical Association*, 86(415): 763–769.

[223] **Snell**, R. C. (**1993**), "Formant location from LPC analysis data", *IEEE Trans. Speech Audio Process.*, 1(2): 129–134.

[224] **Stevens**, K. N. (**2000**), *Acoustic Phonetics*, MIT Press, Cambridge, MA.

[225] **Stevens**, K. N. and **Bickley**, C. A. (**1991**), "Constraints among parameters simplify control of Klatt formant synthesizer", *Journal of Phonetics*, 19: 161–174.

[226] **Stone**, M. and **Davis**, E. P. (**1995**), "A head and transducer support system for making ultrasound images of tongue jaw movement", *J. Acoust. Soc. Am*, 98(6): 3107–3112.

[227] **Stone**, M., **Jr**, M. H. G., and **Zhang**, Y. (**1997**), "Principal component analysis of cross sections of tongue shapes in vowel production", *Speech Communication*, 22(2-3): 173–184.

[228] **Stone**, M. and **Lundberg**, A. (**1996**), "Three-dimensional tongue surface shapes of English consonants and vowels", *J. Acoust. Soc. Am.*, 99(6): 3728–3737.

[229] **Story**, B. H., **Titze**, I. R., and **Hoffman**, E. A. (**1996**), "Vocal tract area functions from magnetic resonance imaging", *J. Acoust. Soc. Am.*, 100(1): 537–544.

[230] **Sun**, J. and **Deng**, L. (**2001**), "An overlapping-feature based phonological model incorporating linguistic constraints: Applications to speech recognition", *J. Acoust. Soc. Am.*, 111(2): 1086–1101.

[231] **Sussman**, H. M., **McCaffrey**, H. A., and **Matthews**, S. A. (**1991**), "An investigation of locus equations as a source of relational invariance for stop place categorization", *J. Acoust. Soc. Am.*, 90(3): 1309–1325.

[232] **Sweet**, H. (**1877**), *Handbook of Phonetics*, Clarendon Press, Oxford, UK, reprinted by McGrath Publishing Co. in 1970.

[233] **Sweet**, H. (**1890**), *A Primer of Phonetics*, Clarendon Press, Oxford, UK.

[234] **Tanizaki**, H. (**1996**), *Nonlinear Filters*, 2nd edition, Springer-Verlag, Berlin, Germany.

[235] **Thompson**, M. A. and **Robl**, P. E. (**1982**), "X-ray microbeam for speech research", *Nuclear Instruments & Methods in Physics Research*, 193: 257–259.

[236] **Togneri**, R. and **Deng**, L. (**2001**), "An EKF-based algorithm for learning statistical hidden dynamic model parameters for phonetic recognition", in *Proc. ICASSP*, vol. 1, 465–468, Salt Lake City, UT.

[237] **Togneri**, R., **Ma**, J., and **Deng**, L. (**2001**), "Parameter estimation of a target-directed dynamic system model with switching states", *Signal Processing*, 81(5): 975–987.

[238] **Unser**, M. and **Stone**, M. (**1992**), "Automated detection of the tongue surface in sequences of ultrasound images", *J. Acoust. Soc. Am.*, 91(5): 3001–3007.

[239] **Valtchev**, V., **Odell**, J. J., **Woodland**, P. C., and **Young**, S. J. (**1997**), "MMIE training of large vocabulary recognition systems", *Speech Communication*, 22(4): 303–314.

[240] **Vidal**, R., **Chiuso**, A., and **Soatto**, S. (**2002**), "Observability and identifiability of jump linear systems", in *Proc. IEEE Conference on Decision and Control*, vol. 4, 3614–3619, Las Vegas, NV.

[241] **Waibel**, A., **Hanazawa**, T., **Hinton**, G., **Shikano**, K., and **Lang**, K. J. (**1989**), "Phoneme recognition using time-delayed neural networks", *IEEE Trans. Acoust., Speech, Signal Process.*, 37(12): 1888–1898.

[242] **Wainwright**, M. J. and **Jordan**, M. I. (**2003**), "Graphical models, exponential families, and variational inference", *Technical Report 649*, Department of Statistics, University of California, Berkeley, Berkeley, CA.

[243] **Wang**, M., **Evans**, J., **Hassebrook**, L., and **Knapp**, C. (**1996**), "A multistage, optimal active contour model", *IEEE Trans. Image Process.*, 5(11): 1586–1591.

[244] **Wang**, Y., **Mahajan**, M., and **Huang**, X. (**2000**), "A unified context-free grammar and n-gram model for spoken language processing", in *Proc. ICASSP*, vol. 2, 1639–1942, Istanbul, Turkey.

[245] **Watanabe**, M. and **Yamaguchi**, K. (editors) (**2004**), *The EM Algorithm and Related Statistical Models*, Marcel Dekker, New York, NY.

[246] **Welling**, L. and **Ney**, H. (**1998**), "Formant estimation for speech recognition", *IEEE Trans. Speech Audio Process.*, 6(1): 36–48.

[247] **Westbury**, J. R. (**1991**), "The significance and measurement of head position during speech production experiments using the X-ray microbeam system", *J. Acoust. Soc. Am.*, 89: 1782–1791.

[248] **Westbury**, J. R. (**1994**), *X-Ray Microbeam Speech Production Database User's Handbook*, Waisman Center on Mental Retardation & Human Development, University of Wiscosin, Madison, WI.

[249] **Westbury**, J. R. and **Hashi**, M. (**1997**), "Lip-pellet positions during vowels and labial consonants", *Journal of Phonetics*, 25: 405–419.

[250] **Westbury**, J. R., **Hashi**, M., and **Lindstorm**, M. J. (**1998**), "Differences among speakers in lingual articulation for American English /r/", *Speech Communication*, 26(3): 203–226.

[251] **Westbury**, J. R., **Lindstorm**, M. J., and **McClean**, M. D. (**2002**), "Tongues and lips without jaws: A comparison of methods for decoupling speech movements", *Journal of Speech, Language and Hearing Research*, 45: 651–662.

[252] **Westbury**, J. R. and **Lindstrom**, M. J. (**2000**), "Two-dimensional shape functions applied to speech kinematic data", in *Proc. 5th Seminar on Speech Production: Models and Data*, 65–68, Kloster Seeon, Bavaria.

[253] **Westbury**, J. R., **Severson**, E. J., and **Hashi**, M. (**1999**), "Lip laws for consonants", in *Proc. ICPhS*, 2025–2028, San Francisco, CA.

[254] **Westbury**, J. R., **Severson**, E. J., and **Lindstorm**, M. J. (**2000**), "Kinematic event patterns in speech: Special problems", *Language and Speech*, 43: 403–428.

[255] **Wilhelms-Tricarico**, R. (**1997**), "A biomechanical and physiologically-based vocal tract model and its control", *Journal of Phonetics*, 24: 23–28.

[256] **Woodland**, P. C. and **Povey**, D. (**2002**), "Large scale discriminative training of hidden Markov models for speech recognition", *Computer Speech & Language*, 16(1): 25–47.

[257] **Yegnanarayana**, B. and **Veldhuis**, R. N. J. (**1998**), "Extraction of vocal-tract system characteristics from speech signals", *IEEE Trans. Speech Audio Process.*, 6(4): 313–327.

[258] **Young**, L. C. (**1980**), *Lectures on the Calculus of Variations and Optimal Control Theory*, 2nd edition, Chelsea, New York, NY.

[259] **Young**, S., **Evermann**, G., **Kershaw**, D., **Moore**, G., **Odell**, J., **Ollason**, D., **Povey**, D., **Valtchev**, V., and **Woodland**, P. (**December 2002**), *The HTK Book (for HTK Version 3.2)*, Cambridge University.

[260] **Zavaliagkos**, G., **Zhao**, Y., **Schwartz**, R., and **Makhoul**, J. (**1994**), "A hybrid segmental neural net/hidden Markov model system for continuous speech recognition", *IEEE Trans. Speech Audio Process.*, 2(1): 151–160.

[261] **Zemel**, R. S. and **Pitassi**, T. (**2001**), "A gradient-based boosting algorithm for regression problems", in *Advances in Neural Information Processing Systems 13*, edited by T. K. Leen, T. G. Dietterich, and V. Tresp, 696–702, MIT Press, Cambridge, MA.

[262] **Zheng**, Y. and **Hasegawa-Johnson**, M. (**2003**), "Analysis of the three-dimensional tongue shape using a three-index factor analysis model", *J. Acoust. Soc. Am.*, 113(1): 478–486.

[263] **Zheng**, Y. and **Hasegawa-Johnson**, M. (**2003**), "Particle filtering approach to Bayesian formant tracking", in *Proc. IEEE Workshop on Statistical Signal Processing*, 601–604, St. Louis, MO.

[264] **Zheng**, Y. and **Hasegawa-Johnson**, M. (**2004**), "Formant tracking by mixture state particle filter", in *Proc. ICASSP*, vol. 1, 565–568, Montreal, QC.

[265] **Zue**, V. W. (**1976**), *Acoustic Characteristics of Stop Consonants: A Controlled Study*, Ph.D. thesis, MIT, Cambridge, MA.

[266] **Zue**, V. W. (**1991**), "Notes on speech spectrogram reading", Course Note, MIT, Cambridge, MA.

[267] **Zweig**, G. (**2003**), "Bayesian network structures and inference techniques for automatic speech recognition", *Computer Speech & Language*, 17(2-3): 173–193.